is the Schur complement. The modelling error $\delta_{i+1} = /r_{i+1}/_2^2$ corresponding to the addition of a basis function $F$ can be computed as:

$$\|r_{i+1}\|_2^2 = \|y - X_{i+1}a_{i+1}\|_2^2 = \|r_i\|_2^2 - \|(X_i g_{i+1} + F)b\|_2^2 = \|r_i\|_2^2 - b^T s_{i+1} b, 0 \leq i \leq n-1 \tag{6}$$

with $/r_0/_2^2 = /y/_2^2$. The quantity $e_{i+1} = b^T s_{i+1} b$ denotes the error reduction corresponding to the addition of a basis function $F$ [12]. In order to ensure positive definiteness of the dot product matrix $X_{i+1}^T X_{i+1}$ the quantity $e_{i+1}$ should be bounded by $0 \in e_{i+1} \in /r_i/_2^2$. It should be noted that $\delta_{i+1} = /r_{i+1}/_2^2 = T \cdot MSE$ is the Squared Error, $T$ is the number of samples in time series $y$, and MSE denotes the Mean Squared Error. Detailed description and additional discussions regarding the method are given in [11–13].

### 2.1 Assessment and Selection of Basis Functions

The explicit expression of error reduction can be used to select a subset of basis functions to form a model from a candidate set $U$ retaining $N$ basis functions. Trigonometric, exponential and linear functions have been considered for modelling in [12,13], while a Non-Negative Adaptive Auto - Regression approach was followed in [11]. The procedure of selecting an appropriate basis requires computation of the potential error reduction for each member of the set $U$. This procedure is algorithmically described in Algorithm 1. The procedure, described by Algorithm 1, proceeds through all candidate basis in $U$ sequentially, storing the respective error reductions to vector $u$. Then, the algorithm proceeds by selecting the index of the basis function that lead to maximum error reduction under the constraints that ensure positive definiteness.

**Algorithm 1.** Basis Search
$(k = bs(y, G_i, D_i^{-1}, X_i, U, \delta_i))$

**Let** $N$ denote the number of candidate basis functions in $U$.
$e_i = 0, 1 \leq i \leq N$
**for** $i \in [1, N] \subset \mathbb{N}$ **do**
    $F = U_i$
    $g_{i+1} = -G_i D_i^{-1} G_i^T X_i^T F$
    $s_{i+1} = F^T(F + X_i g_{i+1})$
    $b = s_{i+1}^{-1}(F^T + g_{i+1}^T X_i^T)y$
    $e_i = b^T s_{i+1} b$
**end for**
$k = \arg\max_{i \in [1,N]} e_i$ under the constraint $0 \leq e_i \leq \tau_i$

The set $U$ can host any type of basis functions even Machine Learning models such as Support Vector Machines [8]. In the current manuscript we focus on lagged basis function of the form:

$$U = [y_{-1} \;\; y_{-2} \;\; y_{-3} \;\; y_{-4} \;\; \dots \;\; y_{-N}], \tag{7}$$

with $y_0 = y$. In practice, the number of samples in lagged time series $y_{-i}$ is $n-N$, since the latest sample is removed (retained in the responses $y$), along with the first $N-1$ samples from each candidate lagged basis to ensure that there are no missing data. In case multiplicative interactions between basis functions are allowed [14], e.g. $y_i y_j \ldots$ the number of basis functions into the candidate set are:

$$\binom{N}{k} + N\ k,\ k > 1 \tag{8}$$

where $k$ is the order of allowed interactions. In the case of $k = 1$, then the number of candidate basis functions is equal to $N$.

Additional constraints can be imposed during the selection of a basis function that leads to the maximum error reduction. These constraints can be imposed during step 10 of Algorithm 1. Examples of constraints include non-negativity of the coefficients [11] or imposing a threshold on their magnitude.

After selection of an appropriate basis function or lag, addition of this basis function has to be performed and the corresponding matrices to be updated. Several basis functions can be fitted by executing the process described by Algorithm 1 followed by the process of Algorithm 2, iteratively. The fitting process is terminated based on criteria regarding the fitting error, e.g. [12]:

$$\sqrt{\delta_{i+1}} < \Delta\sqrt{\delta_0}. \tag{9}$$

Another approach is to terminate the fitting process based on the magnitude of the coefficients:

$$|b| < \Delta|a_1|. \tag{10}$$

where $b$ is the coefficient corresponding to the $i+1$ added basis function, while $a_1$ is the coefficient corresponding to the basis function added first. In both termination criteria $\Delta$is the prescribed tolerance. It should be noted that the second criterion is more appropriate in the case of lagged basis.

**Algorithm 2.** Add Basis Function
$([G_{i+1}, D_{i+1}^{-1}, X_{i+1}, a_{i+1}, \delta_{i+1}] = addbasis(y, G_i, D_i^{-1}, X_i, F, a_i, \delta_i))$

1: $g_{i+1} = -G_i D_i^{-1} G_i^T X_i^T F$
2: $s_{i+1} = F^T(F + X_i g_{i+1})$
3: $b = s_{i+1}^{-1}(F^T + g_{i+1}^T X_i^T)y$
4: Check termination criterion and terminate if met
5: $a_{i+1} = \begin{bmatrix} a_i + g_{i+1} b \\ b \end{bmatrix}$
6: $\tau_{i+1} = \tau_i - b^T s_{i+1} b$
7: $G_{i+1} = \begin{bmatrix} G_i & g_{i+1} \\ 0 & 1 \end{bmatrix}$; $D_{i+1} = \begin{bmatrix} D_i^{-1} & 0 \\ 0 & s_{i+1}^{-1} \end{bmatrix}$; $X_{i+1} = \begin{bmatrix} X_i & F \end{bmatrix}$

## 3 Performance Optimization and Parallelization

The procedure, described by Algorithm 1, proceeds through all candidate basis in $U$ sequentially. It can be performed in parallel by assigning a portion of basis functions to each thread of a multicore processor. However, the most computationally intensive operations are matrix by vector and vector by vector, which are BLAS2 (Basic Linear Algebra Subroutines - Type 2) and BLAS1 operations, respectively, [5]. These operations lead to decreased performance compared to operations between matrices which are BLAS3 operations, since they require increased memory transfers and cache tiling and data re-use is limited [7]. Thus, to increase performance the most computationally intensive part, which is the basis search described by Algorithm 1, has to be redesigned in order to compute the corresponding error for all candidate basis functions.

Let us consider a set of $N$ candidate basis functions, represented as vectors of length $N-n$, stored in the columns of matrix $U$ ($(n-N)\times N$). The columns $g_{i+1}^{(j)}, 1 \in j \in N$ corresponding to each basis can be computed by the following matrix multiplication operations:

$$\mathfrak{g}_{i+1} = [g_{i+1}^{(1)} \;\; g_{i+1}^{(2)} \;\; \cdots \;\; g_{i+1}^{(N)}] = -G_i D_i^{-1} G_i^T X_i^T U. \tag{11}$$

The matrix $\mathfrak{g}_{i+1}$ $(i \times N)$ is formed by four dense matrix multiplications, however matrix $D_i^{-1}$ is diagonal matrix, thus it can be retained as a vector. Multiplying matrix $D_i^{-1}$ by another matrix is equivalent to multiplying the elements of each row $j$ with the corresponding element $d_{j,j}^{-1}$ of the vector retaining the elements of the diagonal matrix. For the remainder of the manuscript we will denote this operation as $(\phi)$. This operation can be used in the process described in Algorithm 2. This reduces operations required, as well as storage requirements, since a matrix multiplication is avoided and the computation can be performed in place in memory.

Following computation of the columns stored in matrix $\mathfrak{g}_{i+1}$, the Schur Complements $s_{i+1}^{(j)}, 1 \in j \in N$ corresponding to the candidate basis functions are computed as follows:

$$\mathfrak{s}_{i+1} = diag(\tilde{\mathfrak{s}}_{i+1}) = diag(U^T(U + X_i \mathfrak{g}_{i+1})). \tag{12}$$

The formula $U^T(U + X_i \mathfrak{g}_{i+1})$ leads to the computation of Schur complement of the block incorporation of basis and not the individual Schur complements corresponding to the candidate basis function, which are stored in the diagonal of the result. The Schur complement $\tilde{\mathfrak{s}}_{i+1}$ is a dense matrix of dimensions $N \times N$ and requires substantial computational effort. In order to avoid unnecessary operations each diagonal element can be computed as follows:

$$(\mathfrak{s}_{i+1})_j = (U^T)_{j,:}((U)_{:,j} + X_i(\mathfrak{g}_{i+1})_{:,j}), \tag{13}$$

where $(\,.\,)_{i,j}$ denotes an element at position (i,j) of a matrix and (:) denotes all elements of a row or column of a matrix. In order to compute all elements of the diagonal concurrently, Eq. (13) can be reformed as follows:

$$\mathfrak{s}_{i+1} = (U^T \odot (U + X_i \mathfrak{g}_{i+1})^T)v, \tag{14}$$

where $\odot$ denotes the Hadamard product of two matrices and $v$ is a vector of the form $[1 \quad 1 \quad \ldots \quad 1]^T$. The vector $\mathfrak{s}_{i+1}$ $(N \times 1)$ retains the Schur complements corresponding to the candidate basis functions in $U$. Dedicated (Optimized) Hadamard product is not included in the standard BLAS collection, however it is included in vendor versions or CUDA (Compute Unified Device Architecture). Following the same notation the coefficients corresponding to the basis functions in set $U$ can be computed as:

$$\mathfrak{b} = \mathfrak{s}_{i+1}^{-1} \odot (U^T + \mathfrak{g}_{i+1}^T X_i^T) y \tag{15}$$

and the corresponding error reductions as:

$$\mathfrak{e} = \mathfrak{b} \odot \mathfrak{s}_{i+1} \odot \mathfrak{b}. \tag{16}$$

The most appropriate basis function is selected by finding the maximum error reduction in vector $\mathfrak{e}$. The matrix based basis selection procedure is algorithmically described in Algorithm 3.

**Algorithm 3.** Matrix Based Basis Search
$(k = mbbs(y, G_i, D_i^{-1}, X_i, U, \delta_i))$

1: **Let** $N$ denote the number of candidate basis functions in $U$.
2: $v_i = 1, 1 \leq i \leq N$
3: $\mathfrak{g}_{i+1} = -G_i(D_i^{-1} \star (G_i^T X_i^T U))$
4: $\tilde{U} = U + X_i \mathfrak{g}_{i+1}$
5: $\mathfrak{s}_{i+1} = (U^T \odot \tilde{U})v$
6: $\mathfrak{b} = \mathfrak{s}_{i+1}^{-1} \odot \tilde{U}^T y$
7: $\mathfrak{e} = \mathfrak{b} \odot \mathfrak{s}_{i+1} \odot \mathfrak{b}$
8: $k = \arg\max_{i \in [1,N]} \mathfrak{e}_i$ under the constraint $0 \leq \mathfrak{e}_i \leq \tau_i$

A block variant of Algorithm 3 can be utilized to process batches of candidate functions. This can be performed by splitting matrix $U$ into groups, processing them and accumulating the corresponding error reductions in vector $\mathfrak{e}$ before computing the index of the most effective basis function. Despite the advantages in terms of performance, the matrix and block based matrix approaches have increased memory requirements. The memory requirements are analogous to the number of candidate basis functions, since they have to be evaluated before assessment, while in the original approach each candidate basis is evaluated only before its assessment. Thus, the matrix approach requires $\mathcal{O}(N(n-N))$, the block approach $\mathcal{O}(max(\xi(n-N)), b_s(n-N)))$ and the original approach $\mathcal{O}(\xi(n-N))$ 64-bit words, where $\xi$ denotes the number of basis functions included in the model and $b_s$ the number of basis in each block.

### 3.1 Graphics Processing Unit Acceleration

The operations involved in Matrix Based Basis Search, given in Algorithm 3, can be efficiently accelerated in a Graphics Processing Unit (GPU). However, most

GPU units suffer from substantial reduction of the double precision performance, e.g. 32× in the case of double precision arithmetic (Geforce RTX 20 series). In order to mitigate this issue, 32-bit floating point operations and 16-bit half precision floating point operations are utilized. This gives rise to mixed precision computations, where GPU related operations are performed in reduced precision, while CPU related ones are performed in double precision arithmetic. This approach enables acceleration while minimizing round off errors from reduced precision computation.

The proposed scheme utilized a similar approach in order to accelerate the most computationally intensive part of the process, which is the basis search. Computations in the GPU require data movement from the main memory to the GPU memory, which is a time consuming operation, thus should be limited. For the case of the Matrix Based Basis Search algorithm, data should be transferred in the GPU before processing. This includes the time series $y$, the matrix $G_i$ and vector $D_i$, the matrix of included basis $X_i$ and the previous modelling error $\delta_i$ in order to mark basis that could hinder positive definiteness. Before copying these arrays to the GPU memory, they should be cast to single precision arithmetic to ensure increased performance during computations. The matrix $U$, retaining the candidate basis, and time series $y$ can be transferred in the GPU before starting the fitting process, since they are "a priori" known, while all the other matrices should be updated after addition of new basis function to the model. However, the update process includes only a small number of values to be transferred at each iteration.

**Algorithm 4.** GPU accelerated modelling

1: **Let** $y$ denote the time series to be modelled, $N$ the maximum lag, $n$ the number of samples in $y$.
2: $v_i = 1, 1 \leq i \leq n - N$
3: $\mathbf{y} = cpu2gpu(single(y))$
4: $\mathbf{U} = cpu2gpu(single([y_{-1} \;\; y_{-2} \;\; \ldots \;\; y_{-N}])$
5: $\tau_0 = \|y\|_2^2$; $\boldsymbol{\alpha} = cpu2gpu(single(\tau_0))$
6: $[G_1, D_1^{-1}, X_1, a_1, \tau_1] = addbasis(y, [\,], [\,], [\,], v, [\,], \tau_0)$
7: $\boldsymbol{G} = cpu2gpu(single(G_1))$; $\boldsymbol{D}^{-1} = cpu2gpu(single(D_1))$
8: $\boldsymbol{X} = cpu2gpu(single(X_1))$; $\boldsymbol{\alpha} = cpu2gpu(single(\tau_1))$
9: **for** $i \in [1, N]$ **do**
10: $\quad \mathbf{k} = mbbs(\mathbf{y}, \boldsymbol{G}, \boldsymbol{D}^{-1}, \boldsymbol{X}, \boldsymbol{U}, \boldsymbol{\alpha})$ ▷ GPU
11: $\quad k = gpu2cpu(\boldsymbol{k})$; $F = y_{-k}$
12: $\quad [G_{i+1}, D_{i+1}^{-1}, X_{i+1}, a_{i+1}, \tau_{i+1}] = addbasis(y, G_i, D_i^{-1}, X_i, F, a_i, \tau_i)$
13: $\quad \boldsymbol{G} = update(single(G_{i+1}))$; $\boldsymbol{D}^{-1} = update(single(D_{i+1}^{-1}))$
14: $\quad \boldsymbol{X} = update(single(X_{i+1}))$; $\boldsymbol{\alpha} = cpu2gpu(single(\tau_{i+1}))$
15: **end for**

The process is described in Algorithm 4. The matrices and vectors stored in the GPU memory are given in bold. The process terminates if the termination criterion of Eq. (10) is met during the basis addition process or line 12 of

Algorithm 4. It should be noted that the first basis included removes the mean value of the time series $y$. This is performed in line 6 where the *addbasis* function is invoked. In practice, addition of the first basis function is performed using the equations: $s_1 = F^T F, a_1 = s_1^{-1} F^T y, D_1^{-1} = s_1^{-1}, G_1 = 1$, where $F$ is substituted with a vector $((n - N) \times 1)$ with all its components set to unity.

The functions *cpu2gpu* and *gpu2cpu* are used to transfer data from CPU memory to GPU memory and from GPU memory to CPU memory, respectively. Conversion of matrices, vectors and variables from double precision to single precision arithmetic is performed with function *single* and the function *update* is utilized to update matrices and vectors involved in the Matrix Based Basis Search performed in the GPU. At each iteration $2 + i + (n - N), 1 \in i \in \xi$ single precision floating point numbers need to be transferred to GPU memory, with $\xi$ denoting the number of basis functions included in the model.

A full GPU version of Algorithm 4 can be also used, by forming and updating all matrices directly to the GPU memory. In this approach, the matrix of candidate matrices $U$ and the time-series $y$ need to be transferred to the GPU memory, before computation commences. The value of the first coefficient should also be transferred to the CPU since it is required by the termination criterion. Moreover, in every iteration the new coefficient has to be transferred to CPU in order to assess model formation through the termination criterion. This approach uses solely single precision arithmetic and is expected to yield slightly different results due to rounding errors.

## 4 Numerical Results

In this section the applicability, performance and accuracy of the proposed scheme is examined by applying the proposed technique to two time series. The first time series is composed of large number of samples and lagged basis functions without multiplicative interactions are used as the candidate set. The scalability of different approaches is assessed with respect to single precision, mixed precision and double precision arithmetic executed either on CPU or GPU. The second time series has a reduced number of samples, however an extended set of lagged candidate basis functions, which include second order interactions, are included. The characteristics of the time series are given in Table 1. The two time series were extracted from R studio. The error measures used to assess the forecasting error was Mean Absolute Percentage Error (MAPE), Mean Absolute Deviation (MAE) and Root Mean Squared Error (RMSE):

$$MAPE = \frac{100}{T} \sum_{i=1}^{T} \frac{|y_i - \hat{y}_i|}{|y_i|}, \quad MAE = \frac{1}{T} \sum_{i=1}^{T} |y_i - \hat{y}_i|, \quad RMSE = \sqrt{\frac{1}{T} \sum_{i=1}^{T} (y_i - \hat{y}_i)^2} \tag{17}$$

where $y_i$ are the actual values, $\hat{y}_i$ the forecasted values and $T$ the length of the test set. All forecasts are performed out-of-sample using a multi-step approach without retraining. All experiments were executed on a system with an Intel Core i7 9700K 3.6–4.9 GHz CPU (8 cores) with 16 GBytes of RAM memory and

an NVIDIA Geforce 2070 RTX (2304 CUDA Cores) with 8 GBytes of memory. All CPU computations were carried out in parallel using Intel MKL, while the GPU computations were carried out using NVIDIA CUDA libraries.

**Table 1.** Model time series with description and selected splitting.

| # | Name | Frequency | Train | Test | Description |
|---|---|---|---|---|---|
| 1 | **Call volume for a large North American bank** | 5-min | 22325 | 5391 | Volume of calls, per five minute intervals, spanning 164 days starting from 3 March 2003 |
| 2 | **Daily female births in California** | Daily | 304 | 61 | Daily observations in 1959 |

Different notation is used for the variants of the proposed scheme:

**CPU-DP**: Matrix based CPU implementation using double precision arithmetic. This is the baseline implementation.
**CPU-SP**: Matrix based CPU implementation using single precision arithmetic.
**CPU-DP-GPU**: Matrix based CPU/GPU implementation using mixed precision arithmetic. The basis search is performed in the GPU using single precision arithmetic, while incorporation of the basis function is performed in double precision arithmetic.
**CPU-DP-GPU-block($n_b$)**: Block matrix based CPU/GPU implementation using mixed precision arithmetic. The basis search is performed in the GPU using single precision arithmetic, while incorporation of the basis function is performed in double precision arithmetic.
**GPU**: Pure matrix based GPU implementation using single precision arithmetic.

The parameter $n_b$ denotes the number of blocks. The block approach requires less GPU memory.

### 4.1 Time Series 1 - Scalability and Accuracy

The average value of the dataset is 192.079, the minimum value is 11 and the maximum value is 465. For this model the lagged candidate basis has been utilized, while higher degree interactions are not allowed, resulting in an additive model. The performance in seconds for all variants is given in Fig. 1, while speedups are presented in Fig. 2. The pure GPU and CPU-DP-GPU implementations have the best performance overall leading to the best speedups. The pure GPU implementation has a speedup greater than 20× for more than 50 basis functions with a maximum of approximately 27×, with respect to the baseline implementation. With respect to the CPU single precision arithmetic implementation the pure GPU approach has a speedup of approximately 10× for more than 50 basis functions. The CPU-DP-GPU has a maximum speedup of

approximately 22× attained for 134 basis functions. After that point the speedup decreases because the double precision operations in the CPU do not scale with same rate, reducing the overall speedup. It should be noted that even for low number of basis functions, e.g. 6, the speedup of the pure GPU implementation is approximately 6× with respect to CPU-DP and approximately 4× with respect to CPU-SP implementation.

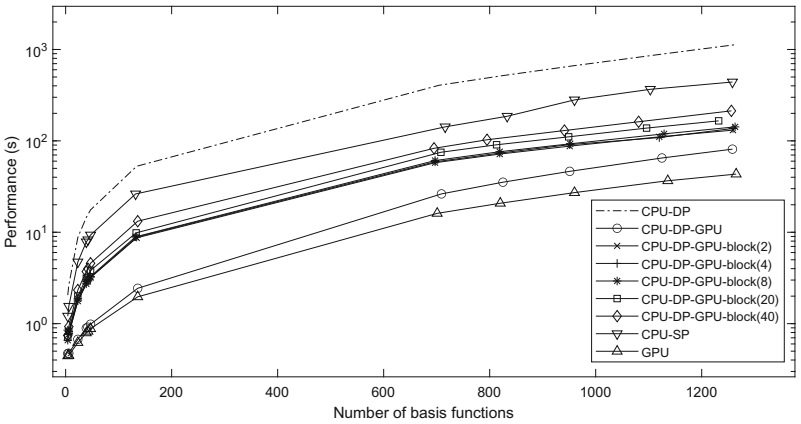



**Fig. 1.** Performance for all variants for different number of basis functions.

The performance of the block variants degrades when increasing the number of blocks retaining the candidate basis functions, since they require more data transfers between CPU and GPU. The speedups for the block variants range from 2.5×–8.6× over CPU-DP implementation and 1.5×–3.4× over the CPU-SP implementation.

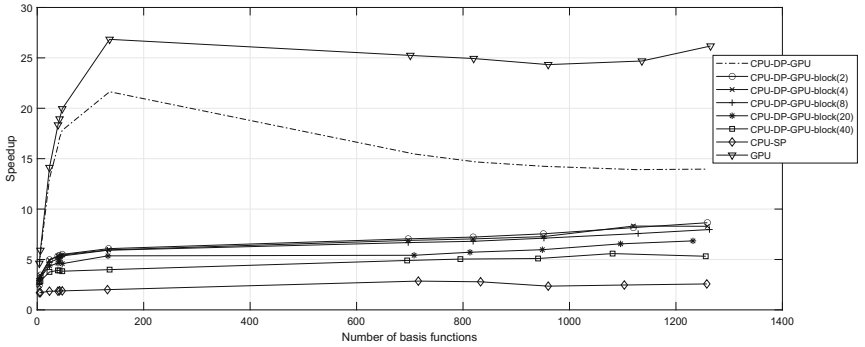



**Fig. 2.** Speedup for all variants for different number of basis functions.

**Table 2.** Minimum, maximum and average number of basis functions, RMSE, MAE and MAPE for all variants for the first time series.

| | # Basis | | | RMSE | | | MAE | | | MAPE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\Delta$ | Min | Max | Avg | Min | Max | Avg | Min | Max | Avg | Min | Max | Avg |
| 0.1 | 4 | 4 | 4.0 | 41.96 | 41.96 | 41.96 | 34.25 | 34.25 | 34.25 | 25.80 | 25.80 | 25.80 |
| 0.07 | 6 | 6 | 6.0 | 44.86 | 44.86 | 44.86 | 36.87 | 36.87 | 36.87 | 29.29 | 29.29 | 29.29 |
| 0.04 | 23 | 23 | 23.0 | 26.32 | 26.32 | 26.32 | 19.61 | 19.61 | 19.61 | 11.76 | 11.76 | 11.76 |
| 0.02 | 39 | 39 | 39.0 | 24.07 | 24.07 | 24.07 | 18.07 | 18.07 | 18.07 | 11.23 | 11.23 | 11.23 |
| 0.01 | 42 | 42 | 42.0 | 24.13 | 24.14 | 24.14 | 18.12 | 18.12 | 18.12 | 11.22 | 11.22 | 11.22 |
| 0.009 | 47 | 47 | 47.0 | 23.86 | 23.86 | 23.86 | 17.96 | 17.96 | 17.96 | 11.45 | 11.45 | 11.45 |
| 0.006 | 132 | 136 | 134.1 | 23.31 | 23.64 | 23.47 | 17.65 | 17.96 | 17.81 | 11.66 | 11.92 | 11.83 |
| 0.005 | 695 | 716 | 702.8 | 23.16 | 23.64 | 23.22 | 17.50 | 17.96 | 17.56 | 11.49 | 11.92 | 11.55 |
| 0.004 | 795 | 833 | 818.7 | 23.15 | 23.64 | 23.21 | 17.49 | 17.96 | 17.55 | 11.48 | 11.92 | 11.53 |
| 0.003 | 941 | 960 | 952.3 | 23.16 | 23.17 | 23.17 | 17.50 | 17.52 | 17.51 | 11.50 | 11.52 | 11.51 |
| 0.002 | 1081 | 1136 | 1116.0 | 23.16 | 23.17 | 23.16 | 17.49 | 17.50 | 17.50 | 11.46 | 11.47 | 11.47 |
| 0.001 | 1232 | 1266 | 1257.3 | 23.16 | 23.16 | 23.16 | 17.49 | 17.50 | 17.50 | 11.46 | 11.47 | 11.47 |

The number of basis functions included in the model along with forecasting errors are given in Table 2. The number of basis functions as well as the errors are not substantially affected by the use of mixed or single precision arithmetic. More specifically, up to approximately 50 basis functions all variants produce almost identical results. However, above 50 basis functions there is a minor difference in the number of basis functions included in the model which in turn slightly affects the error measures. The difference in the number of included basis functions is caused by rounding errors in the computation of error reduction $\delta$. This is caused by the rounding errors in the formation of the column vectors $\mathfrak{g}_{i+1}$, involved in the computation of respective Schur complements and potential basis coefficients.

An important observation is that the error measures regarding forecasts do not significantly reduce after the incorporation of approximately 130 basis functions. Thus, additional basis functions increase the complexity of the model. In order to ensure sparsity of the underlying model, a different termination criterion can be used, since the termination criterion of Eq. (10) depends on the magnitude of the entries of the basis functions and is more susceptible to numerical rounding errors. The new termination criterion based on error reduction percentage is as follows:

$$\sqrt{\delta_i} - \sqrt{\delta_i - e_{i+1}} < \Delta\sqrt{\delta_i}, \tag{18}$$

where $e_{i+1}$ denotes the potential error reduction that will be caused by the incorporation of the $i+1$-th basis. $\Delta \in [0, 1] \subset \mathbb{R}$ denotes the acceptable percentage of error reduction to include a basis function. This criterion will be used to model the second time series along with higher level interactions.

### 4.2 Time Series 2 - Flexibility and Higher Order Interactions

The average value of the dataset is 41.9808, the minimum value is 23 and the maximum value is 73.

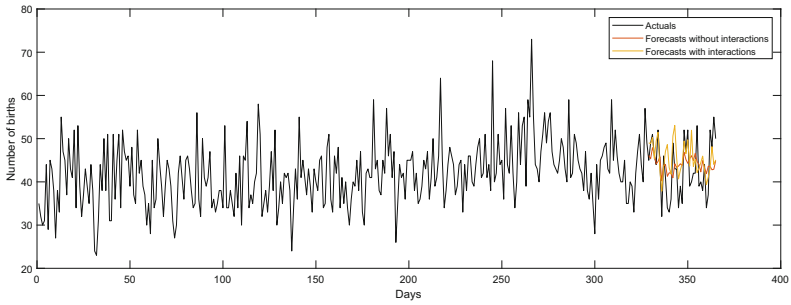



**Fig. 3.** Actual along with forecasted values with and without interactions.

The candidate set is composed of lagged basis functions and second order interactions of the form $y_j y_k$. The termination criterion of Eq. (18) was used with $\Delta = 0.002$, with maximum lags equal to 50. In Fig. 3 the actuals along with the forecasted values computed with and without interactions are given. The inclusion of second order interactions results in capturing the nonlinear behavior of the time series in the forecasted values. The error measures without interactions were: $RMSE = 5.96$, $MAE = 5.11$ and $MAPE = 12.33$, while with interactions the error measures were: $RMSE = 6.18$, $MAE = 4.93$ and $MAPE = 12.20$. The inclusion of higher order interactions led to reduction of the error measures and showcases the flexibility of the approach allowing for the inclusion of arbitrary order interactions in the candidate basis functions.

The execution time for CPU-SP, CPU-DP and GPU were 1.1621, 2.2076 and 0.3932, respectively. Thus, the speedup of the pure GPU variant was approximately 3× over the CPU-SP variant and 5.6× over the CPU-DP version. The pure GPU version is efficient even for time series with small number of samples, under a sufficiently sized space of candidate basis functions.

## 5 Conclusion

A matrix based parallel adaptive auto-regressive modelling technique has been proposed. The technique has been parallelized in multicore CPUs and GPUs and a block variant has been also proposed, based on a matrix (BLAS3) recast of the required operations. The pure GPU variant presented speedup up to 27× over the double precision arithmetic parallel CPU version and 10× over the parallel single precision CPU version for time series with large number of samples. The use of single and mixed precision did not affect substantially the forecasting error,

rendering the technique suitable for modelling and forecasting large time series. Implementation details and discussions on higher order interactions between basis functions have been also given. The applicability and effectiveness of the method were also discussed and new termination criterion based on potential error reduction of basis functions, which is invariant to scaling, has been given.

Future work is directed towards the design of an improved basis search that will reduce the search space based on a tree approach. Moreover, backfitting procedures will be considered.

**Acknowledgement.** This research has emanated from research conducted with the financial support of Science Foundation Ireland under Grant 18/SPP/3459.

## References

1. Abadi, M., et al.: Tensorflow: a system for large-scale machine learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. OSDI 2016, pp. 265–283. USENIX Association, USA (2016)
2. Ben-Nun, T., Hoefler, T.: Demystifying parallel and distributed deep learning: an in-depth concurrency analysis. ACM Comput. Surv. **52**(4) (2019). https://doi.org/10.1145/3320060
3. Box, G.E.P., Jenkins, G.M.: Time Series Analysis Forecasting and Control. Holden Day, San Francisco (1976)
4. Brown, R.G.: Smoothing, Forecasting and Prediction of Discrete Time Series. Prentice Hall, Englewood Cliffs (1963)
5. Choi, J., Dongarra, J., Ostrouchov, S., Petitet, A., Walker, D., Whaley, R.C.: A proposal for a set of parallel basic linear algebra subprograms. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 1995. LNCS, vol. 1041, pp. 107–114. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-60902-4_13
6. Dai, Y., He, D., Fang, Y., Yang, L.: Accelerating 2D orthogonal matching pursuit algorithm on GPU. J. Supercomput. **69**(3), 1363–1381 (2014). https://doi.org/10.1007/s11227-014-1188-8
7. Dongarra, J., et al.: The Sourcebook of Parallel Computing. Morgan Kaufmann, San Francisco (2002)
8. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: Mozer, M.C., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9. MIT Press, Cambridge (1997)
9. Fang, Y., Chen, L., Wu, J., Huang, B.: GPU implementation of orthogonal matching pursuit for compressive sensing. In: 2011 IEEE 17th International Conference on Parallel and Distributed Systems, pp. 1044–1047 (2011). https://doi.org/10.1109/ICPADS.2011.158
10. Filelis-Papadopoulos, C.K.: Incomplete inverse matrices. Numer. Lin. Algebra Appl. **28**(5), e2380 (2021). https://doi.org/10.1002/nla.2380
11. Filelis-Papadopoulos, C.K., Kirschner, S., O'Reilly, P.: Forecasting with limited data: predicting aircraft co2 emissions following covid-19. Submitted (2021)
12. Filelis-Papadopoulos, C.K., Kyziropoulos, P.E., Morrison, J.P., O'Reilly, P.: Modelling and forecasting based on recurrent pseudoinverse matrices. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12745, pp. 229–242. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77970-2_18

13. Filelis-Papadopoulos, C.K., Kyziropoulos, P.E., Morrison, J.P., O'Reilly, P.: Modelling and forecasting based on recursive incomplete pseudoinverse matrices. Mathematics and Computers in Simulation, accepted (2022)
14. Friedman, J.H.: Multivariate adaptive regression splines. Ann. Stat. **19**(1), 1–67 (1991). https://doi.org/10.1214/aos/1176347963
15. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Comput. **9**(8), 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
16. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. Int. J. Forecast. **20**, 5–13 (2004)
17. Korenberg, M.J., Paarmann, L.D.: Orthogonal approaches to time-series analysis and system identification. IEEE Signal Process. Mag. **8**(3), 29–43 (1991)
18. Li, B., et al.: Large scale recurrent neural network on GPU. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 4062–4069 (2014). https://doi.org/10.1109/IJCNN.2014.6889433
19. Li, Q., Salman, R., Test, E., Strack, R., Kecman, V.: GPUSVM: a comprehensive CUDA based support vector machine package. Open Comput. Sci. **1**(4), 387–405 (2011). https://doi.org/10.2478/s13537-011-0028-7
20. Lu, Y., Zhu, Y., Han, M., He, J.S., Zhang, Y.: A survey of GPU accelerated SVM. In: Proceedings of the 2014 ACM Southeast Regional Conference. ACM SE 2014, Association for Computing Machinery, New York, NY, USA (2014)
21. Makridakis, S., Spiliotis, E., Assimakopoulos, V.: The m4 competition: 100,000 time series and 61 forecasting methods. Int. J. Forecast. **36**(1), 54–74 (2020). https://doi.org/10.1016/j.ijforecast.2019.04.014. m4 Competition
22. Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Trans. Signal Process. **41**(12), 3397–3415 (1993). https://doi.org/10.1109/78.258082
23. Oh, K.S., Jung, K.: GPU implementation of neural networks. Pattern Recogn. **37**(6), 1311–1314 (2004). https://doi.org/10.1016/j.patcog.2004.01.013
24. Paoletti, M.E., Haut, J.M., Tao, X., Miguel, J.P., Plaza, A.: A new GPU implementation of support vector machines for fast hyperspectral image classification. Remote Sens. **12**(8) (2020). https://doi.org/10.3390/rs12081257
25. Tan, K., Zhang, J., Du, Q., Wang, X.: GPU parallel implementation of support vector machines for hyperspectral image classification. IEEE J. Sel. Top. Appl. Earth Obser. Remote Sens. **8**(10), 4647–4656 (2015)

# Intersection Representation of Big Data Networks and Triangle Enumeration

Wali Mohammad Abdullah, David Awosoga(✉), and Shahadat Hossain

University of Lethbridge, Lethbridge, AB, Canada
{w.abdullah,shahadat.hossain}@uleth.ca, odo.awosoga@gmail.com

**Abstract.** Triangles are an essential part of network analysis, representing metrics such as transitivity and clustering coefficient. Using the correspondence between sparse adjacency matrices and graphs, linear algebraic methods have been developed for triangle counting and enumeration, where the main computational kernel is sparse matrix-matrix multiplication. In this paper, we use an intersection representation of graph data implemented as a sparse matrix, and engineer an algorithm to compute the "$k$-count" distribution of the triangles of the graph. The main computational task of computing sparse matrix-vector products is carefully crafted by employing compressed vectors as accumulators. Our method avoids redundant work by counting and enumerating each triangle exactly once. We present results from extensive computational experiments on large-scale real-world and synthetic graph instances that demonstrate good scalability of our method. In terms of run-time performance, our algorithm has been found to be orders of magnitude faster than the reference implementations of the **miniTri** data analytics application [18].

**Keywords:** Intersection matrix · Local triangle count · Forward degree cumulative · Forward neighbours · Sparse graph · $k$-count

## 1 Introduction

The presence of triangles in network data has led to the creation of many metrics to aid in the analysis of graph characteristics. As such, the ability to count and enumerate these triangles is crucial to applying these metrics and gaining further insights into the underlying composition and distribution of these graphs. Generalizations aside, the applications of triangle counting are as ubiquitous as the triangles themselves, including transitivity ratio - the ratio between the number of triangles and the paths of length two in a graph - and clustering coefficient - the fraction of neighbours for a vertex $i$ of a graph who are neighbours themselves. Other real-life applications of triangle counting include spam detection [4], network motifs in biological pathways [12], and community discovery [13]. However, before any network analysis can be undertaken, the underlying data structure of a graph must be critically examined and understood. An efficient


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 413–424, 2022.
https://doi.org/10.1007/978-3-031-08757-8_34

representation of network data will dictate analysis capabilities and improve algorithm performance and data visualization potential [5]. Note that large real-life networks are typically sparse in nature, so efficient computations of these graphs must be able to account for their sparsity and skewed degree distribution [3]. A consistent structure makes linear algebra-based triangle counting methods appealing, and most methods use direct or modified matrix-matrix multiplication, with a notable exception being the implementation of Low et al. [11]. This paper expands upon the preliminary ideas of a poster presentation from the 2021 IEEE Big Data Conference (Big Data) [2], and here we propose an "intersection" representation of network data obtained as a list of edges [17] and based on sparse matrix data structures [8]. Our triangle enumeration algorithm derives its simplicity and efficiency by employing matrix-vector product calculations as its main computational kernel. The local triangle count and edge support information are then acquired from the enumerated triangles obtained as the result of this matrix-vector multiplication.

### 1.1 *k*-count Distribution

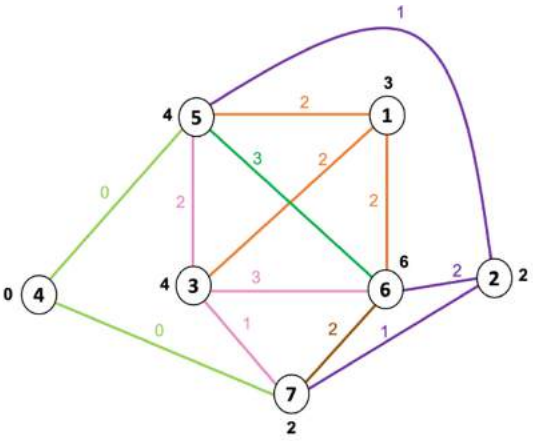
Application proxies provide a simple yet realistic way to assess the performance of real-life applications' architecture and design. Below, we outline the main components of the miniTri data analytics proxy [18], which we use to demonstrate the effectiveness of our intersection-based graph representation and computation.

Let $G = (V, E)$ be a connected and undirected graph without multiple edges and self loops, where $V$ denotes the set of vertices and $E$ denotes the set of edges. For $v \in V$ a path of length 2 through $v$ is a sequence of vertices $u-v-w$ such that $e_1 = \{u, v\} \in E$ and $e_2 = \{v, w\} \in E$. Such a length-2 path is termed a *wedge* at vertex $v$. Let $d(v)$ denote the number of edges *incident* on $v$, also defined as the number of vertices $x$ such that $\{v, x\} \in E$. The number of wedges in $G$ is then given by $\sum_{v \in V} \binom{d(v)}{2}$. A wedge $u - v - w$ is a *closed wedge* or a *triangle* if $e_3 = \{v, w\} \in E$. Let $\delta(v)$ and $\delta(e)$ denote the number of triangles incident on vertex $v$ and edge $e = \{u, v\}$ respectively. In the literature $\delta(v)$ is known as the *local triangle count* or *triangle degree* of vertex $v$ and $\delta(e)$ is known as the *support* or *triangle degree* of edge $e = \{u, v\}$. We denote by $\Delta(G)$ the number of triangles contained in graph $G$. Since a triangle is counted at each of its three vertices, we have $\Delta(G) = \frac{1}{3} \sum_{v \in V} \delta(v)$. Let $H = (V', E')$ be a subgraph of $G$ where $|V'| = k$ and each pair of vertices are connected by an edge ($H$ is a $k$-clique). Then $H$ contains $\binom{k}{3}$ triangles and $\delta(v) = \binom{(k-1)}{2}$ and $\delta(e) = (k-2)$ for $v \in V'$ and $e \in E'$. Let $t$ be a triangle in $G$ and let $\delta(t_x) = \min_x \delta(x)$, where $x$ is a vertex of $t$ and $\delta(t_e) = \min_e \delta(e)$ where $e$ is an edge of $t$. The *k-count* of triangle $t$ is defined to be the largest $k$ such that

1. $\delta(t_x) \geq \binom{(k-1)}{2}$ and
2. $\delta(t_e) \geq (k-2)$

The main computational task of miniTri is to compute the $k$-count distribution of the triangles of an input graph. Figure 1 displays an example input graph with 7

vertices and 13 edges. Each vertex $i$ is circled and contains a label that represents its identity. Beside each vertex $i$ is an integer denoting its local triangle count $\delta(i)$, and there is an integer beside each edge $e = \{i, j\}$ denoting its support $\delta(e)$. The graph contains 7 triangles. The table of Fig. 1 enumerates the triangles in the graph and displays the local triangle count and support of the vertices and edges together with the $k$-count of the triangles. Each row of the table lists the vertex labels of a triangle followed by the local triangle count, support, and $k$-count. There are 4 triangles with $k$-count value 4 and 3 triangles with $k$-count value 3. Let $\phi$ be the size of the largest clique in $G$. Then the graph contains at least $\binom{\gamma}{3}$ triangles with $k$-count value of at least $\phi$. Therefore, the $k$-count distribution can be used to obtain a bound on the size of the largest clique of a graph.




| u | v | w | $\delta(u)$ | $\delta(v)$ | $\delta(w)$ | $\delta(e_1)$ | $\delta(e_2)$ | $\delta(e_3)$ | K-Count |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 5 | 3 | 4 | 4 | 2 | 2 | 2 | 4 |
| 1 | 3 | 6 | 3 | 4 | 6 | 2 | 2 | 3 | 4 |
| 1 | 5 | 6 | 3 | 4 | 6 | 2 | 2 | 3 | 4 |
| 2 | 5 | 6 | 2 | 4 | 6 | 1 | 2 | 3 | 3 |
| 2 | 5 | 7 | 2 | 6 | 2 | 2 | 1 | 2 | 3 |
| 3 | 5 | 6 | 4 | 4 | 6 | 2 | 3 | 3 | 4 |
| 3 | 6 | 7 | 4 | 6 | 2 | 3 | 1 | 2 | 3 |

**Fig. 1.** $k$-count table for the example input graph

The remainder of the paper is organized as follows. In Sect. 2, we introduce the notion of the intersection representation of network data and our data structure, followed by an illustrative example describing the main ideas in our intersection matrix-based triangle enumeration method. Section 3 outlines the computing environment employed to perform numerical experiments and presents triangle enumeration results on three sets of representative network data. miniTri1 [18] and its successor, which we call miniTri2, are the reference implementations by which we present comparative running times and demonstrate that our method scales very well on massive network data, and can be very flexible in its extensions to the analysis of network characteristics such as truss decomposition [7] and triangle ranking [6]. The paper is summarized in Section 4 with a discussion on future research directions.

## 2 Intersection Representation of Network Data

Let the vertices in $V$ be labelled $1, 2, \ldots, |V| = n$. Using the labels on the vertices, a unique label can be assigned to each edge $e_k = \{v_i, v_j\}, i < j, k = 1, 2, \ldots, |E| = m$.

The *intersection representation* of graph $G$ is a matrix $X \in \{0,1\}^{m \times n}$ in which for each column $j$ of $X$ there is a vertex $v_j \in V$ and $\{v_i, v_j\} \in E$ whenever there is a row $k$ for which $X(k,i) = 1$ and $X(k,j) = 1$. The rows of $X$ represent the edge list sorted by vertex labels. Therefore, matrix $X$ can be viewed as an assignment to each vertex a subset of $m$ labels such that there is an edge between vertices $i$ and $j$ if and only if the inner product of the columns $i$ and $j$ is 1. Since the input graph is unweighted, the edges are simply ordered pairs, and can be sorted in $O(m)$ time. Unlike the adjacency matrix which is unique (up to a fixed labelling of the vertices) for graph $G$, there can be more than one *intersection matrix* representation associated with graph $G$ [1]. We exploit this flexibility to store a graph in a structured and space-efficient form.

### 2.1 Adjacency Matrix-Based Triangle Counting

Many existing triangle counting methods use the sparse representation of adjacency matrices in their calculations. The adjacency matrix $A(G) \equiv A \in \{0,1\}^{|V| \times |V|}$ associated with graph $G$ is defined as,

$$A(i,j) = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

It is well known in the literature that the number of closed walks of length $k \geq 0$ are obtained in the diagonal entries of $k^{th}$ power $A^k$. Therefore, the total number of triangles in a graph $G, \Delta(G)$, is given by the trace of $A^3$,

$$\Delta(G) = \frac{1}{6} Tr(A^3).$$

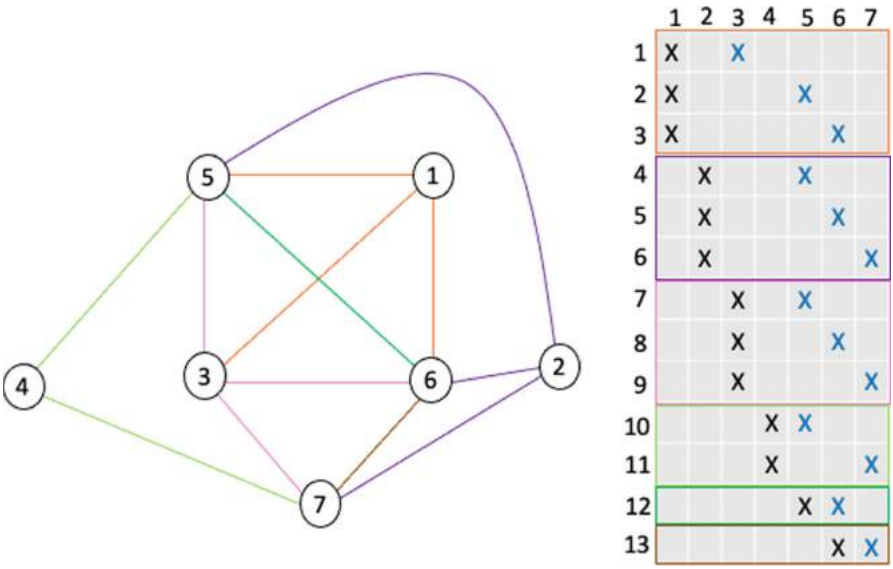



**Fig. 2.** Intersection matrix representation of the example input graph

The factor of $\frac{1}{6}$ accounts for the multiple counting of a triangle (the number of ways closed walks of length 3 can be obtained is $3! = 6$). There is a large body of literature on sparse linear algebraic triangle counting methods based on adjacency matrix representation of the data [5]. miniTri's triangle counting implementation takes the adjacency matrix $A$ of the input graph and creates an incidence matrix $B$ from it [18]. The enumeration and counting of the triangles occur in the overloaded matrix multiplication $C = AB$, where entries in the resultant matrix $C$ with a value of 2 correspond to a completed triangle. This method triple-counts each triangle, once for each vertex, so the final result is divided by 3 to give the total number of triangles in the graph. Since the multiplication of two sparse matrices usually results in a dense matrix, this is a memory intensive process.

### 2.2 Intersection Matrix-Based Triangle Counting

Graph algorithms can be effectively expressed in terms of linear algebra operations [9], and we combine this knowledge with our proposed data representation to count the triangles in a structured three-step method. For vertex $i$ we first find its neighbours $j > i$ such that $\{i, j\} \in E$ by multiplying the submatrix of $X$ consisting of rows corresponding to edges incident on $i$ (let us call them $(i - j)-$rows) by the transpose of the vector of ones of compatible length. A value of 1 in the vector-matrix product indicates that the corresponding vertex $j$ is a neighbour of vertex $i$.

Next, we multiply the submatrix of $X$ consisting of columns $j$ identified in the previous step and the rows below the $(i-j)-$rows by a vector of ones of compatible length. A value of 2 in the matrix-vector product indicates a triangle of the form $(i, j, j')$ where $j$ and $j'$ are neighbours of vertex $i$ with $j < j'$. Let $k$ be the row index in matrix $X$ for which the matrix-vector product contains a 2. Then it must be that $X(k, j) = 1$ and $X(k, j') = 1$. Since each row of $X$ contains exactly 2 nonzero entries that are 1, it follows that $\{j, j'\} \in E$. This operation is identical to performing a set intersection on the forward neighbours of vertices $j$ and $j'$.

The number of triangles in the graph is given by the sum of the number of triangles associated with each vertex as described. Since the edges are represented in sorted order in our algorithm, unlike many other triangle counting methods [18], each triangle is counted exactly once. Figure 2 displays the intersection matrix representation of the input graph $X$. The triangles of the form $(1, j, j')$ where $j, j' \in \{3, 5, 6\}$ are obtained from the product $X(7 : 13, [3\ 5\ 6]) * \mathbf{1}$, where $\mathbf{1}$ denotes the vector of ones. The product has a 2 at locations corresponding to rows 7, 8, and 12 of $X$ and the associated triangles are $(1, 3, 5), (1, 3, 6)$, and $(1, 5, 6)$. Therefore, there are three triangles incident on vertex 1, and it can be easily verified that the graph contains a total of 7 triangles across all of the vertices.

### 2.3 Data Structure

In our preliminary implementation, we use two arrays to store useful information that can be computed after we sort the edges. `FDC` (Forward Degree Cumulative) is an array of size $n$, with elements corresponding to the total number of "forward neighbours" across the vertices of a graph. Forward neighbours are defined as the neighbours of a vertex that have a higher label than the vertex of interest. With the vertices of the graph labelled, finding the forward degree of a vertex $j$ can be calculated as `fd(j) = FDC[j+1] - FDC[j]`. `FN` is an array of size $m$ that stores *which* vertices are the forward neighbours of a vertex $j$. Using `FN` we can find these forward neighbours of $j$ as `fn(j) = FN[k]`, where $k$ ranges from `FDC[j]` to `FDC[j+1]-1`. The arrays `FDC` and `FN` thus save the vector-matrix products needed to find the forward neighbours. Figure 3 displays the arrays `FDC` and `FN` for the graph of Fig. 2.

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FN = | 3 | 5 | 6 | 5 | 6 | 7 | 5 | 6 | 7 | 5 | 7 | 6 | 7 |
| FDC = | 1 | | 4 | | 7 | | 10 | | 12 | | 13 | | 14 |

**Fig. 3.** FN and FDC for the example graph.

### 2.4 Local Triangle Count and Edge Support

As discussed in Sect. 1, there are many other metrics related to triangle computation that can be found using our intersection matrix data structure. The bases for these metrics are the triangle degrees, which are the number of triangles incident on an edge (edge support) or vertex (local triangle count) of a graph. This is illustrated in Fig. 4 as **edgeDeg** and **vertDeg**, respectively, derived from Fig. 1.


**Fig. 4.** vertDeg and edgeDeg for the example graph.

Let $j$ be the column (vertex) of matrix $X$ (graph $G$) currently being processed in the **fullCount** algorithm. For each pair of its forward neighbours $j'$ and $j''$ there is an edge between them if and only if both of the corresponding columns contain a 1 in some row $k$ identifying the triangle $(j, j', j'')$. In terms of the matrix-vector multiplication in line 7 of algorithm **fullCount**, vector $T$ will get updated as $T(k) \leftarrow 2$. Thus the triangle $(j, j', j'')$ can be enumerated and stored instantly. The vertex triangle degrees of each triangle are dynamically updated with this same information, and stored in an array. The edge triangle degrees are stored in a separate array and updated by exploiting the structure of the `FN` and `FDC` arrays in tandem. The entries of the `FDC` array, while primarily used to store the forward degree of a vertex, also contain the edge number (edge id) that the forward neighbourhood of the vertex of interest begins and ends at. Since the sub-arrays in `FN` that correspond to the forward neighbourhood of the vertices are in the same order as the listed edges of the intersection matrix, any edge between two vertices can be identified by first finding the distance between the higher labelled vertex and the beginning of the forward neighbourhood in which it is found (using `FN`), and then adding this distance to the entry in `FDC` that corresponds to the edge of the lower numbered vertex. Finally, the $k$-count distribution of the triangles is used to give a bound on the maximum clique of a graph [18], and with the triangles enumerated and the edge and vertex triangle degrees computed and stored as shown in Fig. 4, the $k$-count calculations can be quickly computed using the method described Sect. 1. The algorithm in its entirety is given in the next section.

### 2.5 Algorithm

**fullCount** $(X)$
Input: Intersection matrix $X$
1: Calculate FDC ξ Forward degree cumulative
2: Calculate FN ξ Forward neighbour
3: $count \leftarrow 0$ ξ Number of triangles
4: **for** $j = 1$ to $n - 1$ **do** ξ $j \in V$, where $V$ is the set of vertices
5: $fd \leftarrow FDC[j + 1] - FDC[j]$ ξ $fd$ is the forward degree of $j$
6: **if** $fd > 1$ **then** ξ $j$ has more than one forward neighbour
7: $T = X([FDC(j + 1) : m], fn_j) * \mathbf{1}$
8: $S = \{t \mid T[t] = 2\}$
9: **if** $S \neq \emptyset$ **then**
10: $count \leftarrow count + |S|$
11: **for** $t \in S$ **do**
12: update edgeDeg ξ Array of triangle edge degrees
13: update vertDeg ξ Array of triangle vertex degrees
14: $Triangles \leftarrow Triangles \cup t$ ξ Array that stores enumerated triangles
15: $kCountTable \leftarrow$ computeKCounts($count$, $vertDeg$, $edgeDeg$, $Triangles$)
16: **return** $count$, $vertDeg$, $edgeDeg$, $kCountTable$, and $Triangles$

## 3 Numerical Results

This section contains experimental results from selected test instances. The first set comprises real-world social networks from the Stanford Network Analysis Project `(SNAP)`, obtained from the Graph Challenge website [15]. SNAP is a collection of more than 50 large network datasets containing a large number of nodes and edges, including social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks [10]. The first set of experiments were performed using a Dell Precision T1700 MT PC with a 4th Gen Intel Core I7-4770 Processor (Quad Core HT, with 3.4GHz Turbo and 8GB RAM), running Centos Linux v7.9. The implementation language was *C++* and the code was compiled using $-O3$ optimization flag with a *g++* version 4.4.7 compiler. Performance times are reported in seconds and were averaged over three runs where possible, using the following implementation abbreviations: *mt1* for miniTri1, *mt2* for miniTri2, and *int* for our intersection algorithm.

Figure 5 shows the speedups of our algorithm versus the two reference miniTri implementations on these real-world instances. The speedups are a unitless measurement defined as the ratio of the miniTri counting time divided by that of our algorithm. For the triangle counting algorithms, our speedups ranged from 22× to an impressive 1383× over miniTri1, and from 16× to 642× over miniTri2, with two instances ("flickrEdges" and "Cit-Patents") failing to compute with miniTri2. Instances with an "*" had speedups greater than 650× against miniTri1 and were cut off from the figure for ease of viewing.

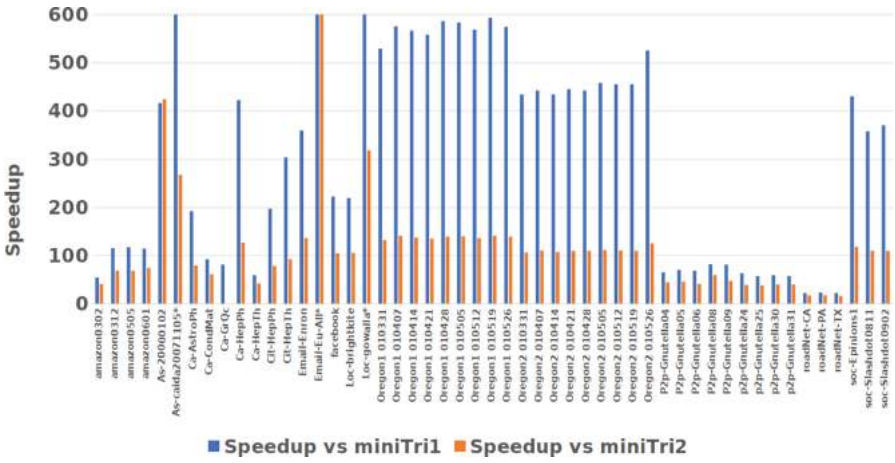



**Fig. 5.** Comparing our intersection algorithm with both miniTri implementations on large real world networks

**Table 1.** Comparing our intersection algorithm with miniTri on large synthetic networks.

| Graph characteristics | | | | Time in seconds | |
|---|---|---|---|---|---|
| Name | $\|V\|$ | $\|E\|$ | $\Delta(G)$ | mt1 | int |
| graph500-scale18-ef16 | 262144 | 4194304 | 82287285 | 17440 | 9.357 |
| graph500-scale19-ef16 | 524288 | 8388608 | 186288972 | 49211.8 | 25.21 |
| graph500-scale20-ef16 | 1048576 | 16777216 | 419349784 | 197456 | 72.34 |
| graph500-scale21-ef16 | 2097152 | 33554432 | 935100883 | N/A | 171.2 |
| graph500-scale22-ef16 | 4194304 | 67108864 | 2067392370 | N/A | 481.43 |
| graph500-scale23-ef16 | 8388608 | 134217728 | 4549133002 | N/A | 1340.05 |
| graph500-scale24-ef16 | 16777216 | 268435456 | 9936161560 | N/A | 3317.15 |
| graph500-scale25-ef16 | 33554432 | 536870912 | 21575375802 | N/A | 7959.39 |

Table 1 compares our algorithm performance on large synthetic test instances from GraphChallenge to miniTri1 (miniTri2 was only able to compute the first instance and thus omitted). "N/A" denotes instances where miniTri1 timed out after four days of computation. Due to the large sizes of this second set of instances, they were run on the large High Performance Computing system (Graham cluster) at Compute Canada. On the first 3 instances, our method is over 1800 times faster than miniTri1, and the relative performance improves with increasing instance size, further demonstrating the scalability of our triangle counting algorithm.

Figure 6 demonstrates our algorithm's performance on relatively dense brain networks from the Network Repository [14], back in the Linux environment. These graphs have between 15 and 268 million edges and up to 42 trillion triangles, and neither miniTri implementation was able to provide results for any

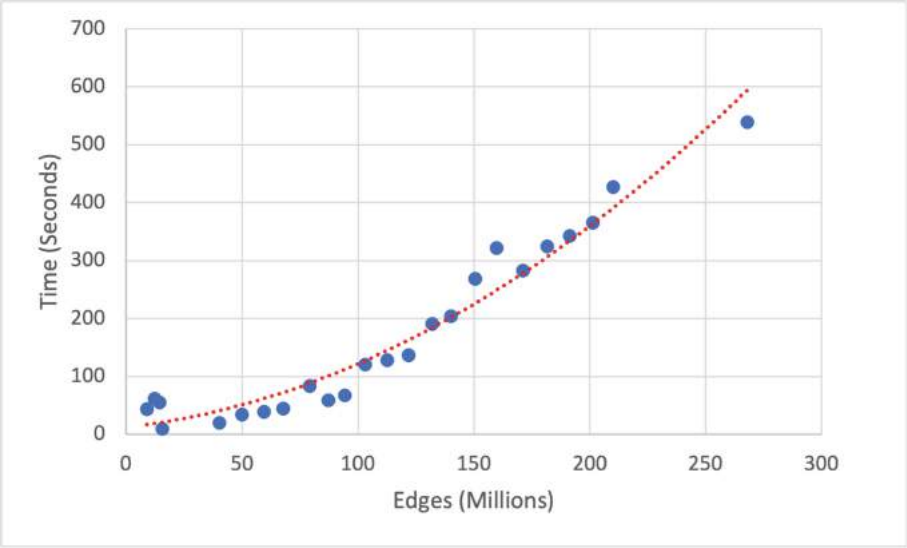



**Fig. 6.** Testing our intersection algorithm on networks with billions of triangles.

of the instances. The line of best fit is a polynomial of degree 2 and shows that our algorithm scales very well with graphs with massive amounts of triangles.

Our intersection-based implementation also produces competitive results when compared to the state-of-the art triangle counting algorithms [16]. Algorithms were analyzed and compared by fitting a model of graph counting times, $T_{tri}$, as a function of the number of edges $N_e = |E|$. This data was then used to estimate the parameters $N_1$ (the number of edges that can be processed in one second) and $\alpha$:

$$T_{tri} = (N_e/N_1)^{\sigma}$$

to compare different counting implementations. Implementations with a larger $N_1$ and smaller $\alpha$ perform the best, and the top entries from the 2019 review had $N_1$ values ranging from $5 \times 10^5$ to $5 \times 10^8$, and $\alpha$ values ranging from $\frac{1}{2}$ to $\frac{4}{3}$. For reference, our algorithm had $\alpha = \frac{3}{4}$ and $N_1 = 1 \times 10^7$.

After examining the comparative performance of our triangle counting algorithm, we proceeded to expand the implementation to include the metrics described in Sect. 2.4 - triangle counting, triangle vertex degree, triangle edge degree, and k-count calculations. Similar to the basic counting experimental results, our intersection method of this "full count" was faster than miniTri1 and miniTri2 on every instance, with speedups ranging between 2× and 177× on the ten largest instances, displayed in Table 2. One noteworthy observation about these results is that due to the data structure that stored the enumerated triangles, the k-count calculation of our algorithm ran much faster than those of miniTri, even though the code implementation was nearly identical. This demonstrates the versatility of `FDC` and `FN` in their ability to perform a wide range of network analytics.

**Table 2.** Comparing our full count intersection algorithm with miniTri1 and miniTri2 on large real world networks.

| Graph characteristics | | | | Time in seconds | | | Speedup | |
|---|---|---|---|---|---|---|---|---|
| Name | $\|V\|$ | $\|E\|$ | $\Delta(G)$ | mt1 | mt2 | int | mt1/int | mt2/int |
| Loc-gowalla | 196591 | 950327 | 2273138 | 156 | 106.4 | 0.882 | 177 | 121 |
| roadNet-PA | 1090920 | 1541898 | 67150 | 2.067 | 1.792 | 0.076 | 27 | 24 |
| roadNet-TX | 1393383 | 1921660 | 82869 | 2.544 | 2.207 | 0.070 | 36 | 32 |
| flickrEdges | 105938 | 2316948 | 107987357 | 1112 | N/A | 553.1 | 2 | $\infty$ |
| amazon0312 | 400727 | 2349869 | 3686467 | 26.8 | 22.29 | 0.932 | 29 | 24 |
| amazon0505 | 410236 | 3356824 | 3951063 | 28.71 | 23.39 | 0.997 | 29 | 23 |
| amazon0601 | 403394 | 3387388 | 3986507 | 28.24 | 25.09 | 0.998 | 28 | 25 |
| roadNet-CA | 1965206 | 5533214 | 120676 | 3.706 | 3.212 | 0.134 | 28 | 24 |
| Cit-Patents | 3774768 | 33037894 | 7515023 | 157.21 | N/A | 3.502 | 45 | $\infty$ |

## 4 Conclusion

Network data is usually input as a list of edges which can be preprocessed into a representation such as an adjacency matrix or adjacency list, suitable for algorithmic processing. We have presented a simple, yet flexible scheme based on intersecting edge labels, the intersection matrix, for the representation of and calculation with network data. A new linear algebra-based method exploits this intersection representation for triangle computation – a kernel operation in big data analytics. By using sparse matrix-vector products instead of the memory-intensive matrix-matrix multiplication, our implementation has the capacity to enumerate and extend triangle analysis in graphs so that important information such as triangle vertex and edge degree can be gleaned in a fraction of the time of reference implementation of miniTri on large benchmark instances. The computational results from a set of large-scale synthetic and real-world network instances clearly demonstrate that our basic implementation is efficient and scales well. The two arrays `FDC` and `FN` together constitute a compact representation of the sparsity pattern of network data, requiring only $n + m$ units of storage. This is incredibly useful in the exchange of network data, with the potential to allow for the computation of many additional intersection matrix-based network analytics such as rank and triangle centrality [6]. A shared memory parallel implementation of this method using OpenMP is being developed, with very optimistic preliminary results. This algorithm can still be tuned, and cache efficiency is being studied for additional optimizations, exploring temporal and spatial locality to analyze the memory footprint and provide further improvements. A natural extension of the research presented in this paper is to use the intersection representation in *graphlet* counting methods. Similar to the $k$-count distribution, graphlet frequency distribution (a vector of the frequency of different graphlets in a graph) provides local topological properties of graphs [17].

**Acknowledgments.** This research was supported in part by NSERC Discovery Grant (Individual), NSERC Undergraduate Student Research Award, and the AITF Graduate Student Scholarship. A part of our computations were performed on Compute Canada HPC system (http://www.computecanada.ca), and we gratefully acknowledge their support.

## References

1. Abdullah, W.M., Hossain, S., Khan, M.A.: Covering large complex networks by cliques—a sparse matrix approach. In: Kilgour, D.M., Kunze, H., Makarov, R., Melnik, R., Wang, X. (eds.) AMMCS 2019. SPMS, vol. 343, pp. 117–127. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-63591-6_11
2. Abdullah, W.M., Awosoga, D., Hossain, S.: Intersection representation of big data networks and triangle counting. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 5836–5838 (2021). https://doi.org/10.1109/BigData52589.2021.9671349
3. Al Hasan, M., Dave, V.S.: Triangle counting in large networks: a review. Wiley Interdiscipl. Rev. Data Min. Knowl. Disc. **8**(2), e1226 (2018)
4. Becchetti, L., Boldi, P., Castillo, C.: Efficient algorithms for large-scale local triangle counting. ACM Trans. Knowl. Discovery Data **4**, 1–28 (2010)
5. Burkhardt, P.: Graphing trillions of triangles. Inf. Vis. **16**(3), 157–166 (2017)
6. Burkhardt, P.: Triangle centrality. arXiv:abs/2105.00110 (2021)
7. Cohen, J.: Trusses: Cohesive subgraphs for social network analysis. Natl. Secur. Agency Tech. Rep. **16**(3.1) (2008)
8. Hasan, M., Hossain, S., Khan, A.I., Mithila, N.H., Suny, A.H.: DSJM: a software toolkit for direct determination of sparse Jacobian matrices. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) ICMS 2016. LNCS, vol. 9725, pp. 275–283. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_34
9. Kepner, J., Gilbert, J.: Graph algorithms in the language of linear algebra. SIAM (2011)
10. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data. Accessed 02 Oct 2019
11. Low, T.M., Rao, V.N., Lee, M., Popovici, D., Franchetti, F., McMillan, S.: First look: linear algebra-based triangle counting without matrix multiplication. In: 2017 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6 (2017). https://doi.org/10.1109/HPEC.2017.8091046
12. Milo, R., Shen-Orr, S., Itzkovitz, S.: Network motifs: simple building blocks of complex network. Science **298**, 824–827 (2002)
13. Palla, G., Dereny, I., Frakas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435**, 814–818 (2005)
14. Rossi, R.A., Ahmed, N.K.: The network data repository with interactive graph analytics and visualization. In: AAAI (2015). https://networkrepository.com
15. Samsi, S., et al.: Static graph challenge: Subgraph isomorphism (2017). http://graphchallenge.mit.edu/data-sets. Accessed 09 July 2021
16. Samsi, S., et al.: Graphchallenge.org triangle counting performance (2020)
17. Szpilrajn-Marczewski, E.: A translation of sur deux propriétés des classes d'ensembles by. Fund. Math **33**, 303–307 (1945)
18. Wolf, M.M., Berry, J.W., Stark, D.T.: A task-based linear algebra building blocks approach for scalable graph analytics. In: 2015 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–6. IEEE (2015)

# Global Surrogate Modeling by Neural Network-Based Model Uncertainty

Leifur Leifsson[1(✉)], Jethro Nagawkar[2], Laurel Barnet[3], Kenneth Bryden[3], Slawomir Koziel[4,5], and Anna Pietrenko-Dabrowska[5]

[1] School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907, USA
leifur@purdue.edu

[2] Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA
jethro@iastate.edu

[3] Department of Mechanical Engineering, Iowa State University, Ames, IA 50011, USA
{labarnet,kmbryden}@iastate.edu

[4] Engineering Optimization and Modeling Center, Department of Engineering, Reykjavík University, Menntavegur 1, 102, Reykjavík, Iceland
koziel@ru.is

[5] Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
anna.dabrowska@pg.edu.pl

**Abstract.** This work proposes a novel adaptive global surrogate modeling algorithm which uses two neural networks, one for prediction and the other for the model uncertainty. Specifically, the algorithm proceeds in cycles and adaptively enhances the neural network-based surrogate model by selecting the next sampling points guided by an auxiliary neural network approximation of the spatial error. The proposed algorithm is tested numerically on the one-dimensional Forrester function and the two-dimensional Branin function. The results demonstrate that global surrogate modeling using neural network-based function prediction can be guided efficiently and adaptively using a neural network approximation of the model uncertainty.

**Keywords:** Global surrogate modeling · Neural networks · Model uncertainty · Error based exploration

## 1 Introduction

There is often a need in engineering to assess the performance of a process (e.g., through physical or computer experiments) with a limited number of evaluations. In such cases, surrogate models are often used to approximate the output response of the process over a given data [3,17,19]. The surrogates are fast to


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 425–434, 2022.
https://doi.org/10.1007/978-3-031-08757-8_35

evaluate and can be used to either explore the output response or exploit them to determine a set of parameter values that yield optimal performance.

Modern surrogate modeling strategies start by constructing a surrogate of an initial data set and then progress in cycles using prediction and uncertainty estimates (if available) to select the next sampling point [26]. Several such approaches have been proposed, including the efficient global optimization (EGO) algorithm [9] and Bayesian optimization (BO) [8,16,20,24]. The EGO algorithm [9] follows this strategy by modeling the output response as a random variable and selects the next point to be sampled by maximizing the expected improvement over the best current solution. BO follows the same idea as EGO, but the approach is formalized rigorously through Bayesian theory [16,20,24].

Gaussian process regression (GPR) (or Kriging) [3,7,11] is widely used with EGO and BO because of its unique feature of providing a prediction of the mean of the underlying data and a prediction of its uncertainty. In particular, GPR provides the mean squared error of the predictor using the same data for constructing the predictor. EGO and BO utilize the predictor and its error estimate to compute a criterion to guide the algorithm to adaptively enhance the predictor. Both EGO and BO typically use ther expected improvement as the criterion [3,27]. The major disadvantages of GPR modeling, however, are that the computational cost scales cubically with the number of observations, and does not scale well to higher dimensions [13]. This issue can be partially relieved by using graphical processing units (GPUs) and parallel computing [14].

Neural network (NN) regression modeling [6], on the other hand, scales much more efficiently for the optimization of complex and large data sets [13,22]. It should be noted that the training cost of NNs depends on various factors, such as sample size, number of epochs, and architecture complexity. A major limitation of NN regression modeling is that uncertainty estimates are, in general, not readily available for a single prediction [13]. Rather, it is necessary to make use of an ensemble of NNs with a range of predictions. Bayesian neural networks (BNNs) are an example of such class of algorithms [5,12,25]. Current BNN approaches, however, are approximation methods because exact NN-based Bayesian inference is computationally intractable. Using dropout as a Bayesian approximation to represent model uncertainty in deep NNs (DNNs) is an example of one such approach [4]. Current BNN algorithms are, however, computationally intensive.

There is recent interest in creating surrogate modeling algorithms that combine the predictive capabilities of NNs and the uncertainty estimates of GPR. Renganathan et al. [18] use DNNs in place of a polynomial to model the global trend function in GPR modeling. This approach improves the prediction capabilities while still retaining the model uncertainty of GPR. Nevertheless, that approach is still limited in the same way as the original GPR modeling approach. Zhang et al. [28] propose an algorithm that creates and adaptively enhances a multifidelity DNN by exploiting information from low-fidelity data sets. This approach is limited to exploitation only and cannot perform exploration or search a criterion that balances exploration and exploitation.

In this paper, a novel adaptive global surrogate modeling algorithm is proposed that follows the EGO strategy but uses NNs in place of GPR. Specifically, the proposed algorithm iteratively constructs two NN models, one for the prediction of a given process output and the other for the model uncertainty. The proposed algorithm uses separate data sets to construct each NN model. In each cycle, the model uncertainty is used to select the next sampling point and then update the NN prediction model. The algorithm terminates once the uncertainty measure has reached a specified tolerance or the maximum number of samples is reached. In this work, the spatial error in the prediction is used as the uncertainty measure, and it is maximized in each cycle to select the next sampling point. The proposed algorithm is tested on two low-dimensional analytical problems. The results demonstrate that global modeling using NN-based function prediction can be guided efficiently and adaptively by an NN approximation of the model uncertainty.

The next section introduces the proposed algorithm. The following section presents results of numerical experiments using one- and two-dimensional analytical functions. Finally, concluding remarks are presented.

## 2 Methods

The proposed approach is summarized in Algorithm 1. The algorithm requires two initial data sets that are used to fit separate neural networks. One neural network models the process output in terms of the input parameters, and the other models the spatial error in the first neural network. Let $(\mathbf{X}, \mathbf{Y})_f$ be the set of sample points used to fit the neural network to the process output, and let $(\mathbf{X}, \mathbf{Y})_u$ be the set of sample points used to fit the neural network to the spatial uncertainty. Here, $\mathbf{X}_f = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(p)}\}^T$ is the set of the input parameter sample points and $\mathbf{Y}_f = (y^{(1)}(\mathbf{x}^{(1)}), ..., y^{(p)}(\mathbf{x}^{(p)}))^T$ the corresponding set of model outputs. Furthermore, $\mathbf{X}_u = \{\mathbf{x}^{(1)}, ..., \mathbf{x}^{(q)}\}^T$ is the set of input parameter sample points and $\mathbf{Y}_u = (y^{(1)}(\mathbf{x}^{(1)}), ..., y^{(q)}(\mathbf{x}^{(q)}))^T$ the corresponding set of model outputs. In this work, it is assumed that the data sets $(\mathbf{X}, \mathbf{Y})_f$ and $(\mathbf{X}, \mathbf{Y})_u$ are distinctly different. Both sets are created using Latin hypercube sampling (LHS) [15].

To fit the neural networks within the proposed algorithm, the mean squared error (MSE) loss function is minimized:

$$\mathcal{L} = \frac{\sum_{l=1}^{N} (\hat{y}^{(l)} - y^{(l)})^2}{N}, \tag{1}$$

where $N$ is the number of samples in the training data. The loss function minimizes the mismatch between the training data, $y$, and the predicted values, $\hat{y}$, of the neural network [6,21]. To minimize the loss function, the adaptive moments (ADAM) optimization algorithm is used [10] along with the backpropagation algorithm [2] to compute the gradients. The neural network setup used in this work, in particular the number of hidden layers and the number of neurons per hidden layer, is case dependent and is described in the numerical experiments.

**Algorithm 1.** Adaptive global surrogate modeling algorithm with neural network-based prediction and uncertainty

**Require:** initial data sets $(\mathbf{X}, \mathbf{Y})_f$ and $(\mathbf{X}, \mathbf{Y})_u$
  **repeat**
    fit neural network to function with available data $(\mathbf{X}, \mathbf{Y})_f$
    compute uncertainty with available data $(\mathbf{X}, \mathbf{Y})_u$
    fit neural network to uncertainty with available data $(\mathbf{X}, \mathbf{Y})_u$
    $\mathbf{P} \leftarrow \arg\max_{\mathbf{x}} \hat{s}^2(\mathbf{x})$
    $\mathbf{X}_f \leftarrow \mathbf{X}_f \cup \mathbf{P}$
    $\mathbf{Y}_f \leftarrow \mathbf{Y}_f \cup y(\mathbf{P})$
  **until** convergence

Other hyperparameters are common between the cases. Specifically, the tangent hyperbolic is used as the activation function, the learning rate is set to 0.001, and the number of epochs is fixed with a value of 3,000. The neural network algorithm is implemented using TensorFlow [1].

The neural network algorithm is used in each cycle to construct a surrogate model, $\hat{y}_f$, of the process output, $y$, using $(\mathbf{X}, \mathbf{Y})_f$. In this work, the uncertainty measure of $\hat{y}_f$ is estimated by the square of the spatial error and is written as

$$s^2(\mathbf{x}) = (\hat{y}_f(\mathbf{x}) - y_f(\mathbf{x}))^2. \tag{2}$$

In the proposed algorithm, $s(\mathbf{x})^2$ is computed in each cycle using the data set $(\mathbf{X}, \mathbf{Y})_u$, and the neural network algorithm is used to construct the surrogate model $\hat{s}(\mathbf{x})^2$.

To select the next sampling point in each cycle of the proposed algorithm, the uncertainty measure $\hat{s}(\mathbf{x})^2$ is maximized using differential evolution [23]. The algorithm is terminated if $\hat{s}(\mathbf{x})^2$ is lower than a specified tolerance or the number of cycles exceeds a specified maximum value.

## 3 Numerical Experiments

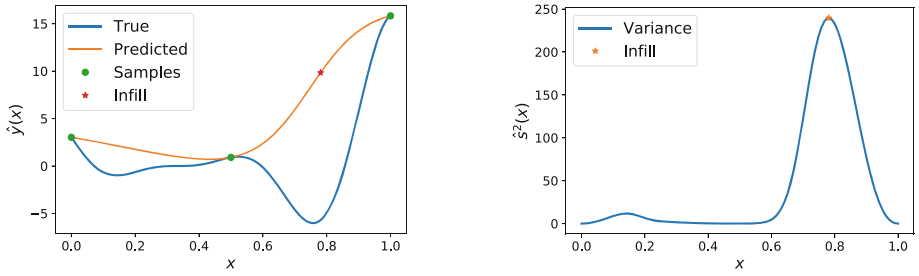
The results of numerical experiments with the proposed algorithm are presented in this section. Two analytical cases are considered, the first case has one input parameter and the second has two.

### 3.1 One-Dimensional Forrester Function

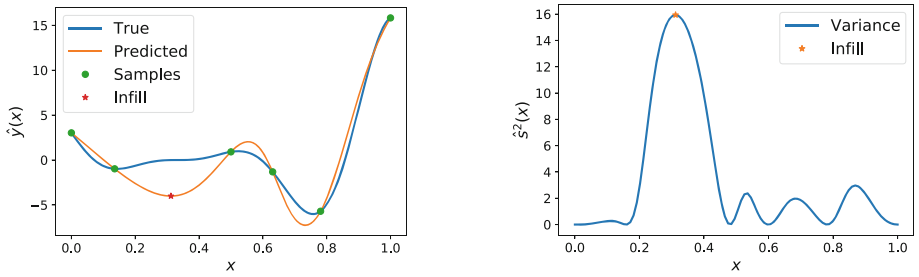
The one-dimensional analytical function developed by Forrester et al. is written as

$$y(x) = (6x - 2)^2 \sin(12x - 4), \tag{3}$$

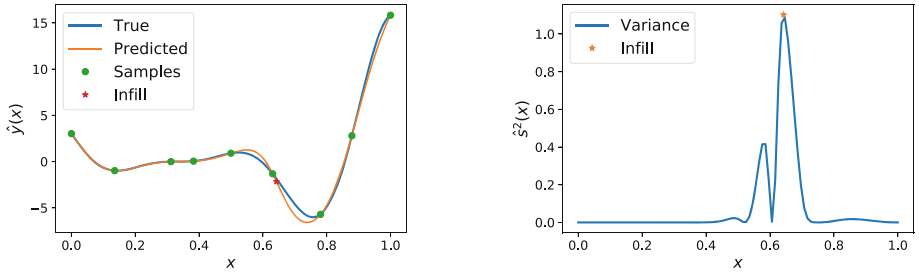
where $x \in [0,1]$. The proposed algorithm is applied to the modeling of this function using three uniformly distributed initial samples and ten infill points. The total number of samples for modeling the function is, therefore, 13. Ten



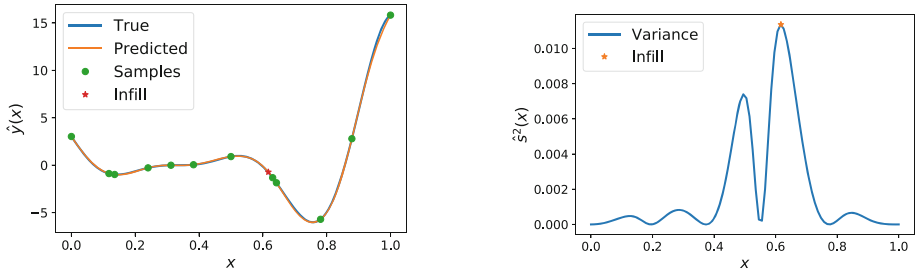

(a)




(b)




(c)




(d)

**Fig. 1.** Forrester function prediction (left) and uncertainty (right) at iterations: (a) 0, (b) 3, (c) 6, (d) 9.

uniformly distributed samples are used for modeling the uncertainty. The number of hidden layers is set to three and the number of neurons in each hidden layer is set to 50.

Figure 1 shows the modeling progression at iterations 0 (3 initial samples and the first infill point), 3 (3 initial and 3 prior infills and the new infill point), 6 (3 initial and 6 prior infills and the new infill), and 9 (with all the initial and infill samples), respectively. Specifically, each subfigure shows the neural network prediction $\hat{y}(x)$ along with the true function $y(x)$, sample points and the next sampling point, as well as the neural network model of the uncertainty $\hat{s}^2(x)$. The location of the maximum uncertainty in the interval from 0 to 1 guides the sampling so that the prediction quickly aligns with the true function through exploration.

The global accuracy of the surrogate models is measured using the root mean squared error (RMSE), which is evaluated using a separate testing data set. Figure 2(a) shows how the RMSE changes over the iterations and is reduced to around 0.1. Figure 2(b) shows how the maximum model uncertainty reduces over the iterations from around 250 to 0.01, or by four orders of magnitude.

### 3.2 Two-Dimensional Branin Function

The two-dimensional Branin function is written as

$$y(x_1, x_2) = \left( x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos(x_1) + 10, \qquad (4)$$

where $x_1, x_2 \in [0, 10]$. The proposed algorithm models this function with ten initial samples selected using LHS and fifty additional infill points for a total of sixty points at the end of fifty iterations. One hundred points, selected through LHS, are used for the uncertainty model. For this case, the number of hidden layers was set to two, with fifty neurons in each hidden layer.

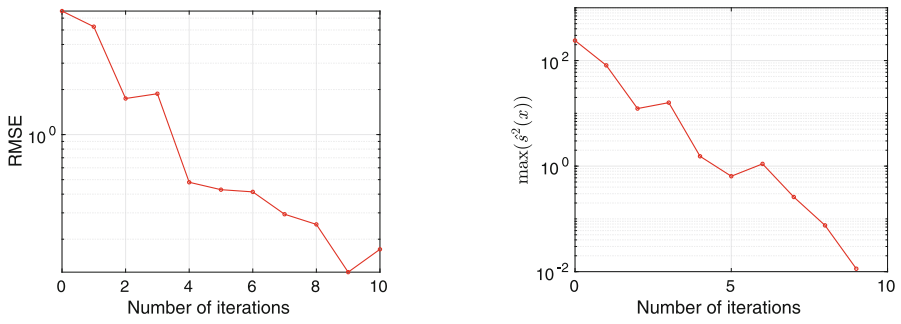



**Fig. 2.** Forrester function modeling error evolution: (a) root mean squared error of the prediction model, (b) maximum variance of the uncertainty model.

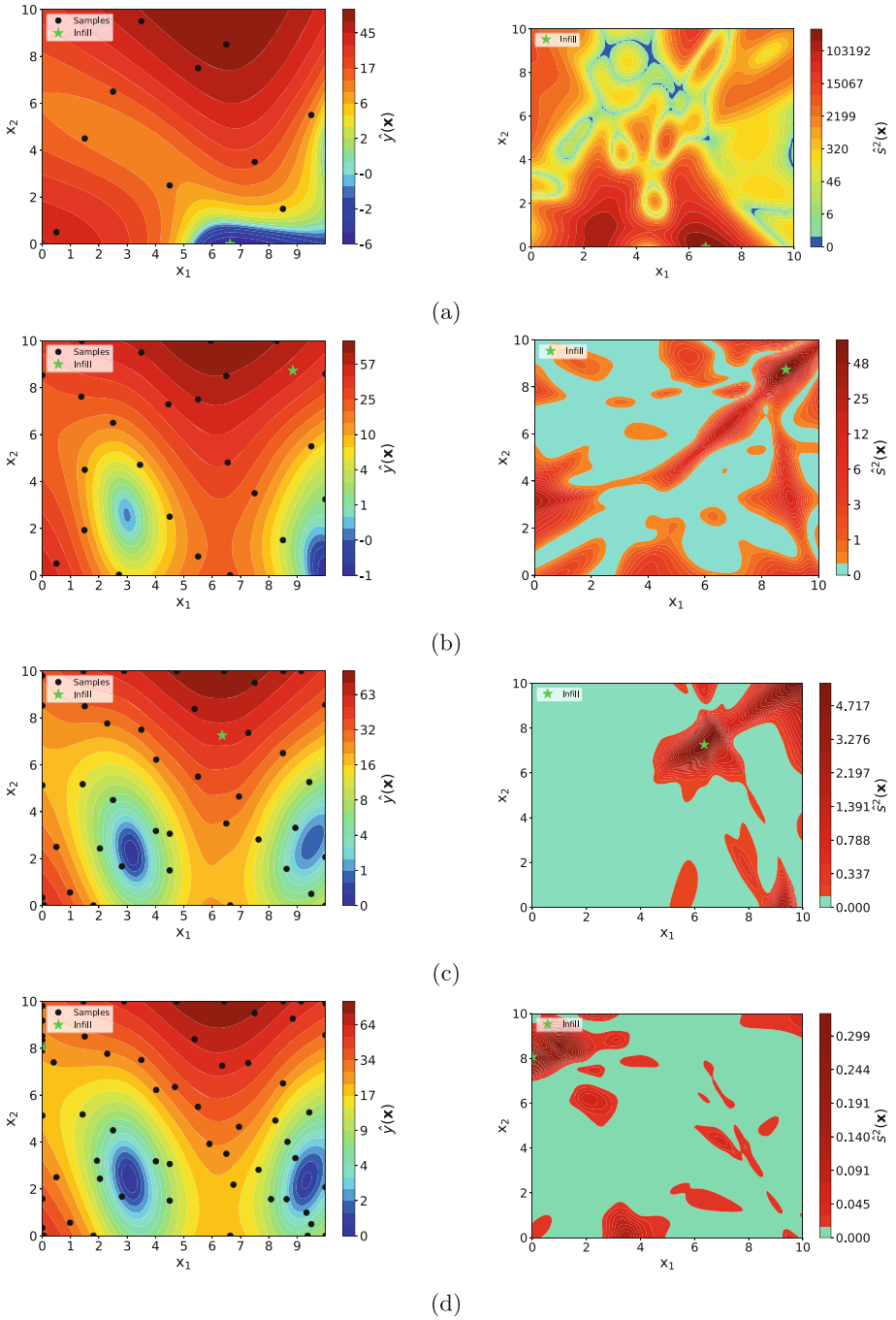



**Fig. 3.** Branin function prediction (left) and uncertainty (right) at iterations: (a) 0, (b) 14, (c) 31, (d) 49.

The modeling progression, at iterations 0 (initial samples only), 14, 31, and 49, is shown in Fig. 3. The left plot in each subfigure shows the neural network model of the function $\hat{y}(x_1, x_2)$ with the sample points used and the next selected infill point, while the right plot shows the neural network model of the uncertainty $\hat{s}^2(x_1, x_2)$ that is being used to select that infill point.

Figure 4 illustrates the global improvement of the surrogate model as the algorithm progresses through the iterations with Fig. 4(a) showing how the RMSE for the surrogate model reduces down to 0.2, and Fig. 4(b) showing how the maximum predicted model uncertainty $\hat{s}^2(x_1, x_2)$ reduces by six orders of magnitude (from roughly $3.7 \cdot 10^5$ to 0.1). The close comparison between the final neural network model and the true function can be seen in Fig. 5 with (a) the contour plot of the true function and (b) the final predicted model with all of the sample points indicated.

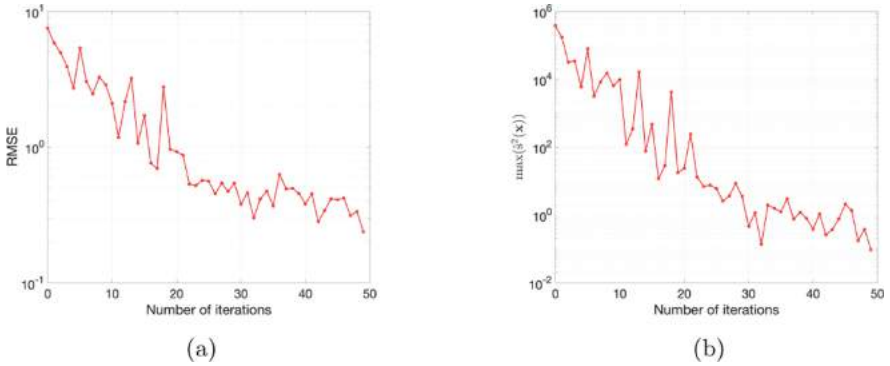



**Fig. 4.** Branin function global modeling error evolution: (a) root mean squared error of the prediction model, (b) maximum variance of the uncertainty model.

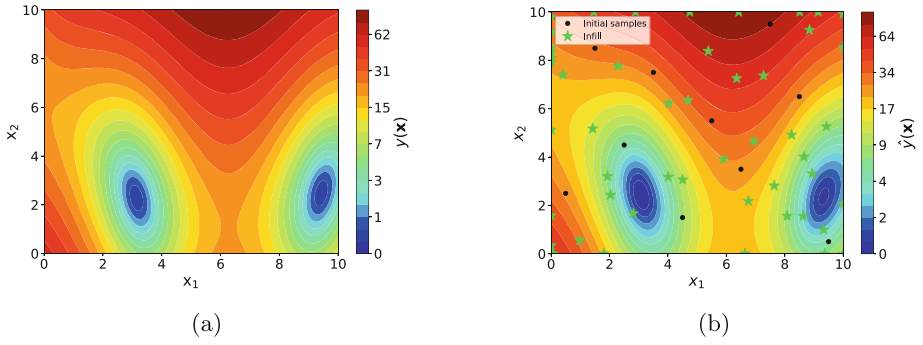



**Fig. 5.** Branin function: (a) true, (b) predicted.

## 4 Conclusion

Global modeling of large data sets is important for decision-making in experimentally and computationally-driven discoveries in engineering and science. The proposed approach of combining efficient global optimization strategies and neural network modeling directly tackles this important problem. Specifically, this paper demonstrates that global modeling using neural network-based function prediction can be guided by an auxiliary neural network approximation of the prediction spatial error that enables efficient adaptive surrogate modeling of large data sets. This capability will support scientists and engineers to make decisions on whether and where in the parameter space to do a physical experiment or computational simulation.

Future work will focus on improving the proposed algorithm to permit adaptive sampling of the uncertainty model as well as using data from multiple levels of fidelity. Furthermore, the process of updating the neural network fit in each cycle of the algorithm needs to be accelerated. Other uncertainty metrics than the prediction variance also need to be explored. An important step will be to compare the proposed approach against current state-of-the-art methods and to characterize the computational costs of each approach. Performing numerical experiments on high-dimensional problems involving physical and computational data is of current interest.

**Acknowledgements.** This material is based upon work supported in part by the Department of Energy under a Laboratory Directed Research and Development grant at Ames Laboratory.

## References

1. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems (2015). http://tensorflow.org/. software available from tensorflow.org
2. Chauvin, Y., Rumelhart, D.E.: Backpropagation: Theory, Architectures, and Applications. Psychology press, Hillsdale (1995)
3. Forrester, A.I.J., Keane, A.J.: Recent advances in surrogate-based optimization. Prog. Aerosp. Sci. **45**(1–3), 50–79 (2009)
4. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In: Proceedings of the 33rd International Conference on Machine Learning, pp. 1050–1059 (2016)
5. Goan, E., Fookes, C.: Bayesian neural networks: an introduction and survey. In: Mengersen, K.L., Pudlo, P., Robert, C.P. (eds.) Case Studies in Applied Bayesian Data Science. LNM, vol. 2259, pp. 45–87. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-42553-1_3
6. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)
7. Huang, D., Allen, T., Notz, W., Zeng, N.: Global optimization of stochastic black-box systems via sequential kriging meta-models. J. Global Optim. **34**(3), 441–466 (2006)
8. Jones, D.R.: A non-myopic utility function for statistical global optimization. J. Global Optim. **21**, 345–383 (2001)

9. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455–492 (1998)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
11. Krige, D.G.: Statistical approach to some basic mine valuation problems on the witwatersrand. J. Chem. Metallurgical Min. Eng. Soc. South Africa **52**(6), 119–139 (1951)
12. Lampinen, J., Vehtari, A.: Bayesian approach for neural networks - review and case studies. Neural Netw. **14**(3), 257–274 (2001)
13. Lim, Y.F., Ng, C.K., Vaitesswar, U.S., Hippalgaonkar, K.: Extrapolative Bayesian optimization with Gaussain process and neural network ensemble surrogate models. Adv. Intell. Syst. **3**, 2100101 (2021)
14. Liu, H., Ong, Y.S., Shen, X., Cai, J.: When Gaussian process meets big data: a review of scalable GPs. IEEE Trans. Neural Networks Learn. Syst. **31**(11), 4405–4423 (2020)
15. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979)
16. Mockus, J.: Application of Bayesian approach to numerical methods of global and stochastic optimization. J. Global Optim. **4**(4), 347–365 (1994)
17. Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidyanathan, R., Tucker, P.K.: Surrogate-based analysis and optimization. Prog. Aerosp. Sci. **21**(1), 1–28 (2005)
18. Renganathan, S.A., Maulik, R., Ahuja, J.: Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. Aerosp. Sci. Technol. **111**, 106522 (2021)
19. Sacks, J., Welch, W., Michell, J.T., Wynn, P.H.: Design and analysis of computer experiments. Stat. Sci. **4**, 409–423 (1989). https://doi.org/10.1214/ss/1177012413
20. Sasena, M.J.: Flexibility and Efficiency Enhancement for Constrained Global Design Optimization with Kriging Approximations. Ph.D. thesis, University of Michigan, USA (2002)
21. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
22. Snoek, J., et al.: Scalable Bayesian optimization using deep neural networks. In: Proceedings of the 32nd International Conference on Machine Learning, pp. 2171–2180 (2015)
23. Storn, R., Price, K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J. Global Optim. **11**, 341–359 (1997)
24. Streltsov, S., Vakili, P.: A non-myopic utility function for statistical global optimization. J. Global Optim. **14**(3), 283–298 (1999)
25. Titterington, D.M.: Bayeisan methods for neural networks and related models. Stat. Sci. **19**(1), 128–139 (2004)
26. Viana, F.A., Haftka, R.T., Watson, L.T.: Efficient global optimization algorithm assisted by multiple surrogate techniques. J. Global Optim. **56**, 669–689 (2013)
27. Zhan, D., Xing, H.: Expected improvement for expensive optimization: a review. J. Global Optim. **78**(3), 507–544 (2020). https://doi.org/10.1007/s10898-020-00923-x
28. Zhang, X., Xie, F., Ji, T., Zhu, Z., Zheng, Y.: Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization. Comput. Methods Appl. Mech. Eng. **373**(1), 113485 (2021)

# Analysis of Agricultural and Engineering Systems Using Simulation Decomposition

Yen-Chen Liu[1], Leifur Leifsson[2(✉)], Anna Pietrenko-Dabrowska[3], and Slawomir Koziel[3,4]

[1] Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA
clarkliu@iastate.edu
[2] School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907, USA
leifur@purdue.edu
[3] Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
anna.dabrowska@pg.edu.pl
[4] Engineering Optimization and Modeling Center, Department of Engineering, Reykjavík University, Menntavegur 1, 102 Reykjavík, Iceland
koziel@ru.is

**Abstract.** This paper focuses on the analysis of agricultural and engineering processes using simulation decomposition (SD). SD is a technique that utilizes Monte Carlo simulations and distribution decomposition to visually evaluate the source and the outcome of different portions of data. Here, SD is applied to three distinct processes: a model problem, a nondestructive evaluation testing system, and an agricultural food-water-energy system. The results demonstrate successful implementations of SD for the different systems, and the illustrate the potential of SD to support new understanding of cause and effect relationships in complex systems.

**Keywords:** Simulation decomposition · Food-water-energy systems · Nondestructive evaluation · Physics-based simulations · Parameter variability

## 1 Introduction

Simulation decomposition (SD) [1] is not only a great visualizing technique for engineering processes, but also an efficient application to utilize the resulting data set created by the Monte Carlo [2] sampling process. An illustration, such as a stacked bar chart, can express the decomposed variability of simulation inputs and outputs to understand the results. Furthermore, SD has the potential to understand cause and effect relationships between the input and output parameters of a given system [3].

In this work, a recent SD method [3] is applied to problems in the area of agriculture and engineering system analysis. The first problem is intended


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 435–444, 2022.
https://doi.org/10.1007/978-3-031-08757-8_36

to illustrate the SD method using is a model problem with three inputs and one output. The second problem has relevance to nondestructive evaluation and involves an ultrasonic testing (UT) system with a pillbox void. The last problem involves an agricultural model that computes the nitrogen export of the state of Iowa. The commonality of the second and the third problems is that they both involve processes that utilize physics-based computational simulations and have input parameters with variability. The SD analysis provides a graphical representation of the effect of input variability on the simulation outputs.

The next section gives the formulation of the problem and a description of the SD analysis technique. The following section presents the results of the three numerical cases. The last section summarizes this work and discusses the potential future work.

## 2 Methods

This section describes the general problem formulation and gives the details of the SD method.

### 2.1 Problem Statement

System analysis can be represented as a black box model as

$$y = f(\mathbf{x}), \tag{1}$$

where the left-side of the equation represents the model output $y$ and the right-side of the equation represents the model $f$ with input parameters $\mathbf{x}$. It is important to understand the effects of uncertainties of the input parameters on the output response when making design decisions. In this work, the effects of the input parameters on the output parameters are visualized using simulation decomposition (SD).

### 2.2 Simulation Decomposition

SD [1,3] is an approach to visualize the effects of variability on models. Furthermore, SD analysis is used to distinguish the influences of different cases of inputs affecting the model output. Figure 1 shows a flowchart of the SD workflow. The process starts by generating random samples as the input data set. First, specify the statistical distributions for each input parameter and choose the states of interest for each input parameter individually. Then, divide the distributions into sub-distributions according to the chosen states. Next, generate every possible combination of the input data from the sub-distribution using Monte Carlo sampling and categorize them into different cases. These cases would determine the model output that is to-be-decomposed. Then, conduct the simulations with the input samples and obtain the output data set. Meanwhile, register the output of each simulation according to their cases. Lastly, construct the outcome probability for each case and decompose the full outcome probability into different cases

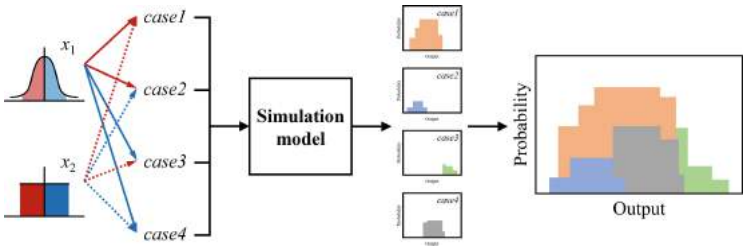



**Fig. 1.** A schematic of the simulation decomposition workflow.

accordingly. The importance of each case can be observed by the occurrence in the decomposed output visually, and cross-checking important cases provide a sense of importance for the input parameters.

## 3 Numerical Examples

This section presents the results of SD analysis of a model problem, an ultrasonic nondestructive testing system, and the Iowa food-energy-water system analyzed.

### 3.1 Model Problem

The simple analytical function is written as [4]

$$y(\boldsymbol{x}) = x_1 + x_2 x_3^2, \tag{2}$$

where $x_1$, $x_2$, and $x_3$ are the input parameters with the variabilities given in Table 1, and $y$ is the function output parameter. A data set of a total $10^6$ points is created using Latin hypercube sampling (LHS) [5].

Figure 2 shows how each input parameter is decomposed into two states that define eight cases for the parameter space. All the parameters are considered to be uniformly distributed. For $x_1$, 500 is set to divide the complete distribution into sub-distribution, and 50 and 5 are used to divide $x_2$ and $x_3$, respectively.

Figure 3 shows the decomposed distribution of the output $y$. In general, a low value of $y$ is contributed by almost all combinations of the input parameters. However, a high value of $y$ is mainly obtained by the cases with high values of

**Table 1.** Input parameters and their statistical distributions for the model problem.

| Parameter | Distribution |
|---|---|
| $x_1$ | $U(0, 1000)$ |
| $x_2$ | $U(0, 100)$ |
| $x_3$ | $U(0, 10)$ |

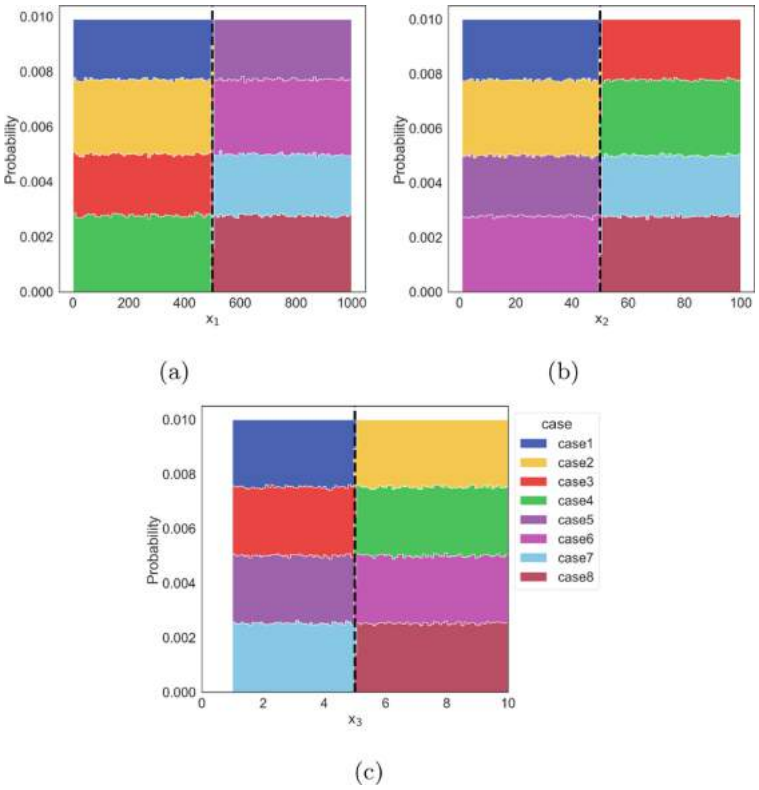



**Fig. 2.** Decomposed distribution of input parameters from SD for the model problem.

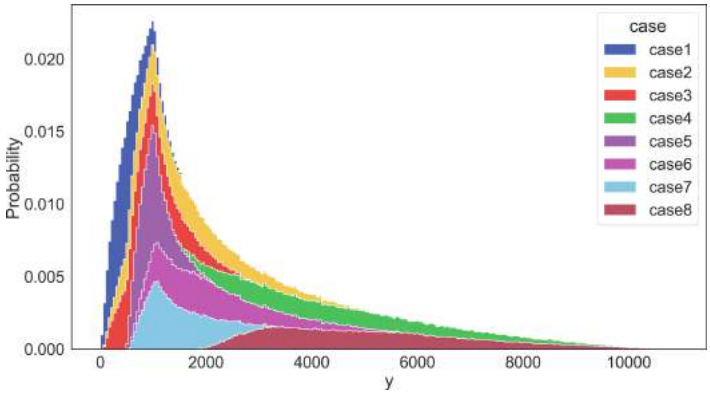



**Fig. 3.** Decomposed distribution of the model problem output.

$x_2$ and $x_3$. In particular, in the cases where $x_1$ and $x_2$ are both low, the simple function can still yield high $y$ with high $x_3$, and vice versa. In this case, the SD analysis shows that the output of the analytical function is dominated by $x_3$, which is as expected when inspecting the function directly.

### 3.2 Ultrasonic Testing (UT) System with a Pillbox Void

Ultrasound testing (UT) is a widely used nondestructive testing (NDT) technique for flaw detection. In this problem, a pillbox-inclusion-defect under planar UT transducer is considered [6]. Figure 4 shows the setup of the problem. The planar transducer is placed in water and the probe angle ($\theta$) and the probe coordinates ($x$ and $y$) are varied. A fused quartz block with a pillbox-like void is inspected by the transducer where the distance between the transducer and the surface of the block ($z_1$) and the distance between the surface of the block and the defect ($z_2$) can vary based on the setup. The variability distributions for this problem are given in Table 2. The output response is the reflected pulse ($v$) received by the transducer. A data set of $10^5$ data points is generated by LHS [5] for this problem.

Figure 5 shows the sub-distributions of sampled input parameters in two states. The variability of probe angle ($\theta$) is considered normal distribution and the rest of input parameters are considered uniform distributions in this problem. For $\theta$, the complete distribution is divided by the statistical mean of $0^*$. The sub-distributions of the probe coordinates $x$ and $y$ are both divided by 0.5 mm. $z_1$ and $z_2$ are consider using full distribution in this work for simplicity. The total of eight cases are shown in Fig. 5.

Figure 6 shows the decomposed distribution of output response ($v$). A diagonal trend can be observed. In particular, the SD analysis shows that high values of the inputs yield low values of response. Furthemore, the results show that the response is dominated by $\theta$, followed by $x$, and then $y$.

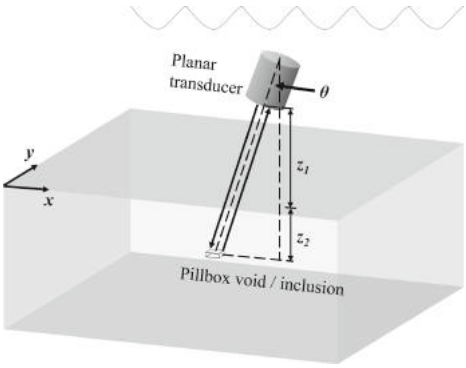



**Fig. 4.** A schematic showing the setup of the ultrasonic testing system.

**Table 2.** Ultrasonic testing system input parameters statistical distributions.

| Parameter | Distribution |
|---|---|
| $\alpha$ (deg) | $N(0, 0.5^2)$ |
| $x$ (mm) | $U(0, 1)$ |
| $y$ (mm) | $U(0, 1)$ |
| $z_1$ (mm) | $U(24.9, 25.9)$ |
| $z_2$ (mm) | $U(12.5, 13.5)$ |

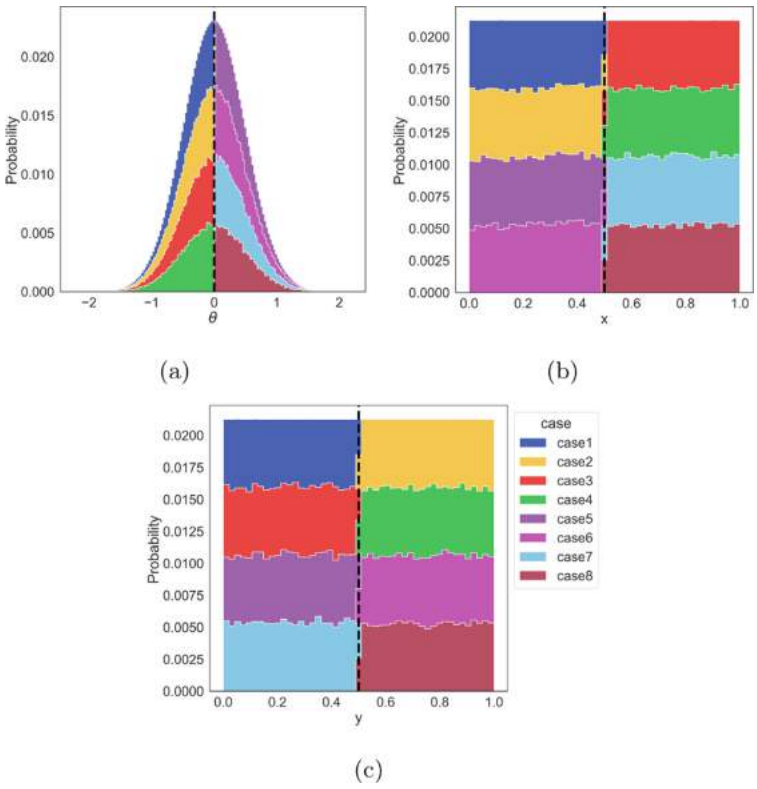



**Fig. 5.** Decomposed distribution of input parameters from SD for UT system.

### 3.3 Iowa Food-Energy-Water System

A simulation-based model of the Iowa food-water-energy (IFEW) system computes the surplus nitrogen ($N_s$) considering the weather, agriculture, and animal agriculture domains in the state of Iowa [7]. Figure 7 show an extended design structure matrix (XDSM) diagram of the simulation model. The input parameters, intermediate parameters and output parameters are listed in Table 3, and the details of the computation for different domains are described in [7]. This

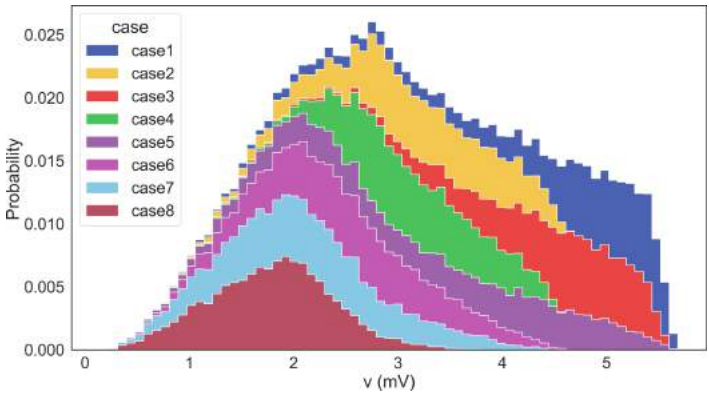



**Fig. 6.** Decomposed distribution of ultrasonic testing system output parameter.

work focuses on the SD analysis of the variability input parameters given in Table 4 and assumes other input parameters are fixed. A data set of $10^5$ points is created for the SD analysis using LHS [5].

Figure 8 shows the distribution of sampled input parameters in two states and the four corresponding cases. The variability distributions of the July temperature ($w_1$) and precipitation ($w_2$) are considered to be normal and log-normal, respectively. A temperature of 76 °F and a precipitation 2.5 in. are used to divide the distributions into sub-distributions of $w_1$ and $w_2$, respectively. The states can be identified as regular temperature (below 76 °F) and high temperature (above 76 °F). Similarly, the states of precipitation are low precipitation (below 25 in.) and regular precipitation (above 25 in.).

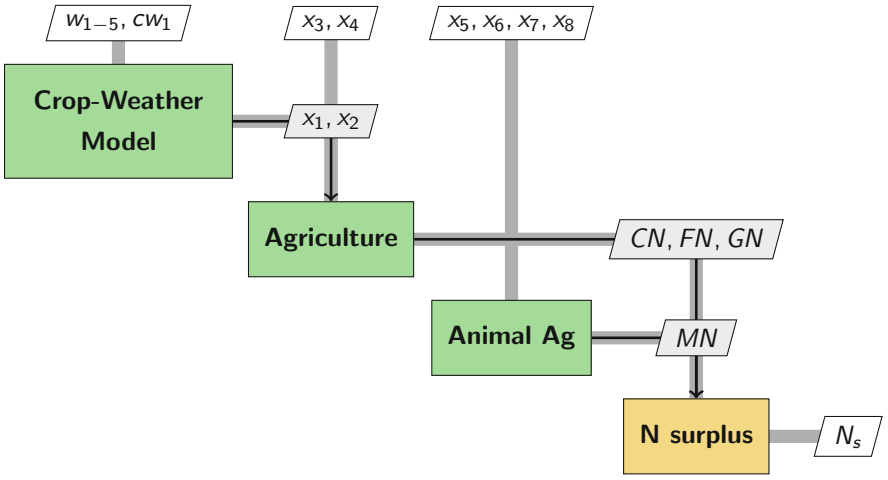



**Fig. 7.** Extended design structure matrix of Iowa food-water-energy system model.

**Table 3.** Parameters of the IFEW simulation system model.

| Parameter | Description |
|---|---|
| $w_1$ | July temperature |
| $w_2$ | July precipitation |
| $w_3$ | June precipitation |
| $x_4$ | July-August average temperature |
| $x_5$ | July-August average precipitation |
| $cw_1$ | May planting progress |
| $x_1$ | Corn yield |
| $x_2$ | Soybean yield |
| $x_3$ | Rate of commercial nitrogen for corn |
| $x_4$ | Rate of commercial nitrogen for soybean |
| $x_5$ | Hog/pigs population |
| $x_6$ | Beef cattle population |
| $x_7$ | Milk cows population |
| $x_8$ | Other cattle population (heifers + slaughter cattle) |
| $CN$ | Commercial nitrogen (nitrogen in commercial fertilizers) |
| $FN$ | Biological fixation nitrogen of soybean crop |
| $GN$ | Grain nitrogen (Nitrogen harvested in grain) |
| $MN$ | Manure nitrogen (Nitrogen in animal manure) |
| $N_s$ | Surplus nitrogen in soil |

**Table 4.** Input parameters of the IFEWS simulation model with variability.

| Parameter | Distribution |
|---|---|
| $w_1$ (°F) | $N(74,\ 2^2)$ |
| $w_2$ (in.) | $LogN(0.4,\ 0.4^2)$ |

Figure 6 shows the decomposed distribution of nitrogen surplus ($N_s$), i.e., the output parameter of the IFEW system model. The results suggests that the major contribution of nitrogen surplus is coming from regular temperature and regular precipitation of July. Other cases are the combinations of less common weather conditions which lead to extreme amounts of nitrogen surplus but are rare (Fig. 9).

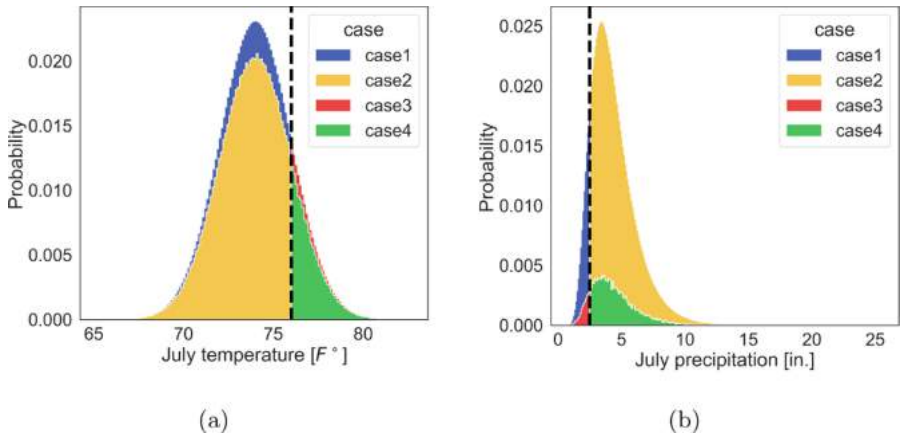



**Fig. 8.** Decomposed distribution of input parameters from SD for the IFEW system.

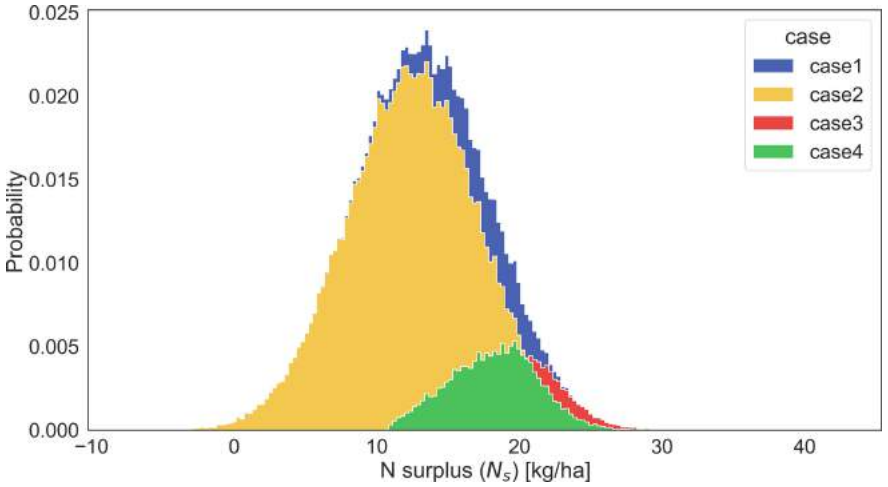



**Fig. 9.** Decomposed distribution of the nitrogen surplus output of the IFEW system simulation model.

## 4 Conclusion

This work demonstrates that simulation decomposition (SD) can support the analysis of agricultural and engineering systems involving input parameters of variability. In particular, SD can provide new insights into the effects of the model input ranges on its output. This insight can be useful in understanding cause and effect relations in complex systems. Future work will explore the potential of combining global sensitivity analysis (GSA) with the SD technique. Furthermore, the use of surrogate modeling in conjunction with those analyses will be investigated to create computationally efficient algorithms for analysis with SD and GSA.

**Acknowledgements.** This material is based upon work supported by the United States National Science Foundation under grant no. 1739551.

## References

1. Mariia Kozlova, M.C., Luukka, P.: Simulation decomposition: new approach for better simulation analysis of multi-variable investment projects. Fuzzy Econ. Rev. **21**(1), 3–8 (2016)
2. Kroese, D.P., Brereton, T., Taimre, T., Botev, Z.I.: Why the Monte Carlo method is so important today. WIREs Comput. Stat. **6**(6), 386–392 (2014)
3. Kozlova, M., Yeomans, J.: Multi-variable simulation decomposition in environmental planning: an application to carbon capture and storage. J. Environ. Inform. Lett. **1**(1), 20–26 (2019)
4. Bilal, N.: Implementation of Sobol's method of global sensitivity analysis to a compressor simulation model. In: 22nd International Compressor Engineering Conference, p. 2385, July 2014
5. Forrester, A., Sóbester, A., Keane, A.: Engineering Design via Surrogate Modelling. Wiley, Germany (2008)
6. Gurrala, P., Chen, K., Song, J., Roberts, R.: Full wave modeling of ultrasonic NDE benchmark problems using Nystrom method. Rev. Progress Quant. Nondestruct. Eval. **36**(1), 1–8 (2017)
7. Raul, V., Liu, Y.C., Leifsson, L., Kaleita, A.: Effects of weather on Iowa nitrogen export estimated by simulation-based decomposition. Sustainability **14**(3), 1060 (2022)

# Neural Network-Based Sequential Global Sensitivity Analysis Algorithm

Yen-Chen Liu[1], Leifur Leifsson[2(✉)], Slawomir Koziel[3,4], and Anna Pietrenko-Dabrowska[4]

[1] Department of Aerospace Engineering, Iowa State University, Ames, IA 50011, USA
clarkliu@iastate.edu
[2] School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907, USA
leifur@purdue.edu
[3] Engineering Optimization and Modeling Center, Department of Engineering, Reykjavík University, Menntavegur 1, 102 Reykjavík, Iceland
koziel@ru.is
[4] Faculty of Electronics Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland
anna.dabrowska@pg.edu.pl

**Abstract.** Performing global sensitivity analysis (GSA) can be challenging due to the combined effect of the high computational cost, but it is also essential for engineering decision making. To reduce this cost, surrogate modeling such as neural networks (NNs) are used to replace the expensive simulation model in the GSA process, which introduces the additional challenge of finding the minimum number of training data samples required to train the NNs accurately. In this work, a recently proposed NN-based GSA algorithm to accurately quantify the sensitivities is improved. The algorithm iterates over the number of samples required to train the NNs and terminates using an outer-loop sensitivity convergence criteria. The iterative surrogate-based GSA yields converged values for the Sobol' indices and, at the same time, alleviates the specification of arbitrary accuracy metrics for the NN-based approximation model. In this paper, the algorithm is improved by enhanced NN modeling, which lead to an overall acceleration of the GSA process. The improved algorithm is tested numerically on problems involving an analytical function with three input parameters, and a simulation-based nondestructive evaluation problem with three input parameters.

**Keywords:** Global sensitivity analysis · Surrogate modeling · Neural networks · Sobol' indices · Termination criteria

## 1 Introduction

The study of sensitivity analysis (SA) [1,2] is important in many engineering and science applications. Individual effects and interactions of the input parameters


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 445–454, 2022.
https://doi.org/10.1007/978-3-031-08757-8_37

on the output model response can be quantified by SA [3,4]. Engineers and scientists can use SA to understand the importance of parameters in experimental or computational investigations. There are two types of SA, local [5] and global [6] SA. Local SA usually refers to using the the local output model response to quantify the effect of small local perturbations in the inputs. In global SA, it utilizes the variance of output model response to quantify the effect due to the input variability in the entire parameter space. This work focuses on the use of global variance-based SA with Sobol' indices [3,4] for simulation-based problems.

In this paper, a recently developed algorithm for surrogate-based GSA [7] is improved and applied to new testing problems. In the NN-based sequential algorithm, the number of samples is iteratively increased with the goals of obtaining the converged Sobol' indices with the training cost as minimum as possible. This approach not only alleviates the needs to specify arbitrary surrogate modeling accuracy metrics, but also reliefs from the fact that accuracy metrics for surrogate models do not guarantee that converged Sobol' indices are obtained. In this work, the implementation of the NN training has been improved significantly, which leads to improved convergence of the GSA algorithm. The algorithm is tested numerically on two problems; an analytical function with three parameters and a simulation-based problem with five parameters.

The next section describes the problem statement and gives the details of the GSA algorithm. The following section presents results of numerical experiments. Finally, conclusions are presented and remarks on future steps are given.

## 2 Methods

This work proposes a sequential algorithm to quantify the global sensitivities of each input variability parameter to the simulation-based model outputs. Figure 1 shows the flowchart of the proposed algorithm. The algorithm starts from an initial sample plan, $\mathbf{x}$, which takes a small number of samples from the input parameters with their variability. Latin hypercube sampling (LHS) [8] is used in this work to randomly select sample data points from each probability distribution of the inputs. The corresponding outputs or observations, $y$, are then generated from the simulation model. A surrogate model, $\hat{y}(\mathbf{x})$, is constructed using these inputs and outputs as training data. The input-output behavior of the simulation model is imitated by the surrogate. Next, GSA is performed with this surrogate model using Sobol' sensitivity indices. The calculation of the Sobol' indices is a Monte Carlo process, therefore the convergence of these indices are checked within an inner-loop. In the inner-loop, the Sobol' indices are computed by sampling the current surrogate model, and the number of samples is increased during each iteration (e.g., one order of magnitude for this work) until it achieves the convergence of the inner-loop. Then, the converged inner-loop indices are checked by the outer-loop. The above process is resampled with an increasing number of sample plan from the simulation model until the outer-loop convergence criteria are met. The number of sample plan affects training an

accurate surrogate model, and the outputs of the surrogate affect the precision of the GSA. The outcome of the proposed algorithm yields the corresponding surrogate model and the converged Sobol' indices of GSA.

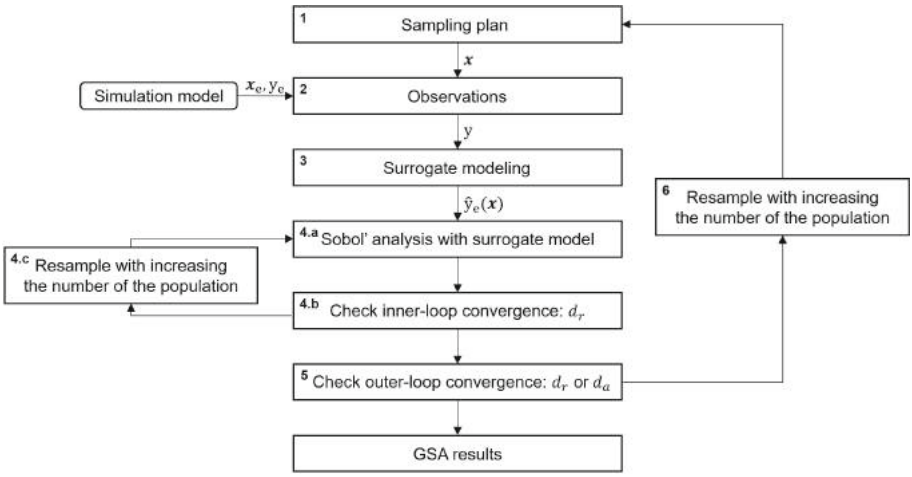



**Fig. 1.** A flowchart of the sequential global sensitivity analysis algorithm with neural network-based prediction.

Neural networks (NNs) are used in a variety of applications in the world. In this work, NN is used to be the surrogate and mimic the behavior of the simulation model. The general structure of NN is a hierarchy of features [9] with three parts: input layer, output layer, and hidden layers [10,11]. All the layers are composed by "neurons" which are the fundamental units of computation [9]. The number of neurons in the input and output layers are the same as the number of input and output variables of the simulation model. Hidden layers are the layers in-between the input and output layers. There could be zero or more hidden layers in a neural network. The number of hidden layers and the number of neurons in each hidden layer usually varies from case to case.

This work uses Sobol' indices [3,4] for the global sensitivity analysis. It is a variance-based method that quantifies the single effects of individual inputs and the interactions of combination of inputs on the simulation model output. The first-order Sobol' indices [4] that quantify the effect of individual inputs are given by

$$S_i = \frac{V_i}{\mathrm{Var}(y)} = \frac{\mathrm{Var}_{x_i}(E_{\mathbf{x}_{\sim i}}(y|x_i))}{\mathrm{Var}(y)}, \tag{1}$$

where $S_i$ is the contribution of individual $x_i$ on the output variance of the simulation model. The total-order or total-effect Sobol' indices [4] that quantify the interactions of combined inputs are given by

$$S_{T,i} = 1 - \frac{\mathrm{Var}_{\mathbf{x}_{\sim i}}(E_{x_i}(y|\mathbf{x}_{\sim i}))}{\mathrm{Var}(y)}, \tag{2}$$

where $S_{T,i}$ measures the contribution of both individual $x_i$ and the interactions between $x_i$ and other input parameters on the output variance of the simulation model.

The proposed sequential algorithm includes an outer-loop that samples the simulation model to generate the NN-based surrogate models and an inner-loop that samples the trained NN-based surrogate model to computes the Sobol' sensitivity indices. The converged NN-based surrogate model and Sobol' indices are obtained by the termination of the outer- and inner-loop based on two measurements of the Sobol' indices between successive iterations. The first measurement is computed by the absolute relative change of Sobol' indices defined as

$$d_r[s_i] = \left| \frac{s_i^{(\mathrm{n})} - s_i^{(\mathrm{n}-1)}}{s_i^{(1)}} \right|, \tag{3}$$

where $s$ is the value of the Sobol' indices and is calculated separately for first- and total-order indices, $i$ is the index of input parameter, and $n$ is the current iteration index. The loop is terminated when $d_r[s_i] \leq \epsilon_r$ for all $s_i$. In this work, $\epsilon_r$ is set to 0.1. The second measurement is computed by the absolute change of Sobol' indices, given by

$$d_a[s_i] = \left| s_i^{(\mathrm{n})} - s_i^{(\mathrm{n}-1)} \right|, \tag{4}$$

where $s$ is the value of the Sobol' indices and is calculated separately for first- and total-order indices, $i$ is the index of input parameter, and $n$ is the current iteration index. The loop is terminated when $d_a[s_i] \leq \epsilon_a$ for all $s_i$. In this work, $\epsilon_a$ is set to 0.01. Both outer- and inner-loop can be terminated by either $d_r[s_i] \leq \epsilon_r$ or $d_a[s_i] \leq \epsilon_a$ being true.

## 3 Numerical Experiments

### 3.1 Case 1: Analytical Function

An analytical function with three input variables and one single output function is used to demonstrate the algorithm. The function is written as

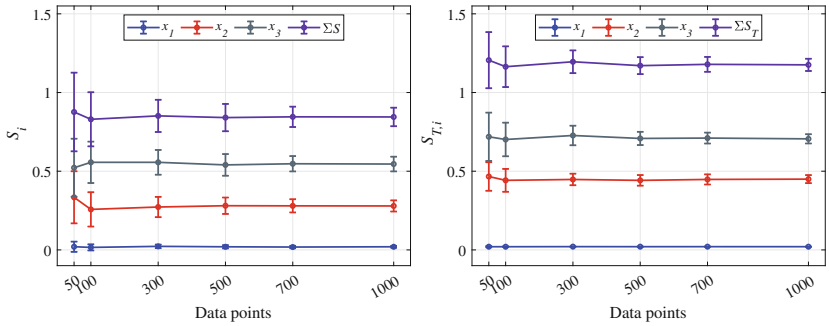
$$f(\boldsymbol{x}) = x_1 + x_2 x_3^2, \tag{5}$$

where $x_1 \in U(0, 1000)$, $x_2 \in U(0, 100)$, and $x_3 \in U(0, 10)$ are the input parameters and their associated PDFs, and $y$ is the function output.

Figure 2 shows the convergence of the direct GSA of the true model. Direct GSA converged at 1000 sampling points. Figure 3 shows the inner-loop GSA convergence using the NN-based algorithm and Fig. 4 shows the outer-loop convergence. The algorithm terminates at 200 samples. The NN models are trained using two hidden layers, with twenty neurons, and the tangent hyperbolic activation function. The learning rate is set to 0.001 and the batch size is set to 16.

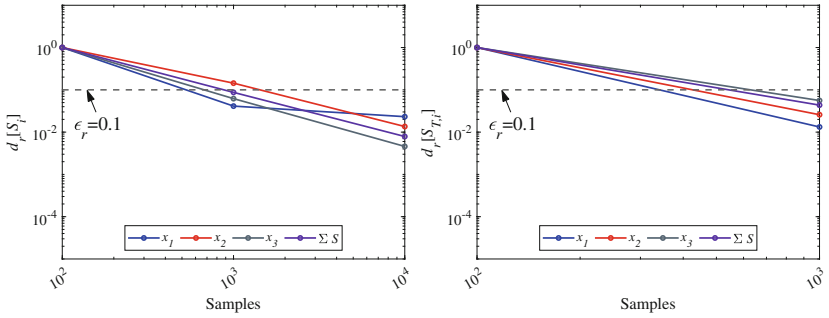
The maximum number of epochs and $\beta_1$ are set to 2,000 and 0.9, respectively. L2 regularization is used with $\lambda = 0.1$ while training the models.

Table 1 shows a comparison of the global sensitivities obtained from direct and NN-based numerical experiments. It can be seen that the NN-based yields global sensitivities within 1.7% of the true values while using only a fraction of the cost of the direct approach.




(a) (b)

**Fig. 2.** Case 1 convergence of GSA directly on the true model: (a) first-order indices, and (b) total-order indices.




(a) (b)

**Fig. 3.** Case 1 inner-loop convergence of $s_i$ for the NN trained with 100 LHS samples: (a) first-order indices, and (b) total-order indices.

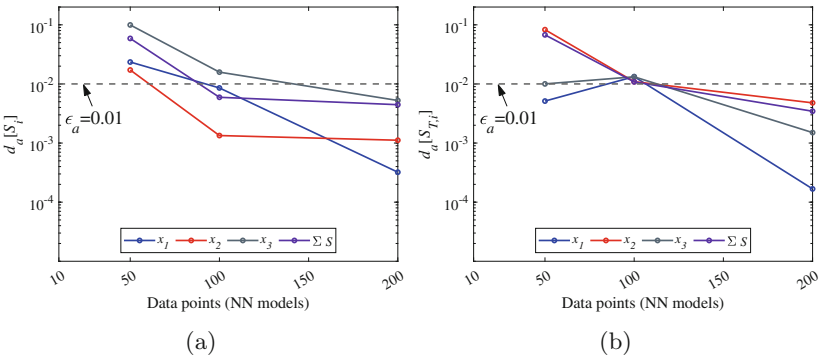



**Fig. 4.** Case 1 outer-loop convergence of $s_i$ terminated on $d_a \leq \epsilon_a$ criteria: (a) first-order indices, and (b) total-order indices.

**Table 1.** Case 1 comparison of Sobol' index values between the true model and the converged NN model.

| x | $S_i$ | | | $S_{T,i}$ | | |
|---|---|---|---|---|---|---|
| | True function | Seq. GSA | % error | True function | Seq. GSA | % error |
| $x_1$ | 0.0199 | 0.0201 | 1% | 0.0204 | 0.0201 | 1.5% |
| $x_2$ | 0.2793 | 0.2745 | 1.7% | 0.4497 | 0.4453 | 1% |
| $x_3$ | 0.5456 | 0.5347 | 2% | 0.7065 | 0.7053 | 0.2% |

### 3.2 Case 2: Ultrasonic Testing (UT) of a Pillbox-Defect

In this numerical experiment the pillbox-inclusion-defect under planar transducer ultrasonic (UT) nondestructive testing (NDT) benchmark case is used [12,13]. Figure 5 shows the setup of the problem. The planar transducer is placed in water and the probe angle ($\theta$) and the probe coordinates ($x$ and $y$) are varied. A fused quartz block with a pillbox-like void is inspected by the transducer where the distance between the transducer and the surface of the block ($z_1$) and the distance between the surface of the block and the defect ($z_2$) can vary based on the setup. The variability parameters with their associated PDFs are $\theta \in N(0, 0.5^2)$ $deg$, $x \in U(0,1)$ mm, $y \in U(0,1)$ mm, $z_1 \in U(24.9, 25.9)$ mm, and $z_2 \in U(12.5, 13.5)$ mm. The output response is the reflected pulse ($v$) received by the transducer.

The simulation model for this case uses the Kirchhoff approximation (KA) to simulate the voltage wave receives by the transducer. The center frequency of the planar transducer is set to 10 MHz. The longitudinal wave speed is set to 6,200 m/s, and the shear wave speed is set to 3,180 $m/s$. The density of the fused quartz block is set to 4,420 kg/m$^3$. The NN models in this case are trained using two hidden layers, with thirty neurons. Similar to the previous case, the tangent hyperbolic activation function is used. The learning rate and $\beta_1$ of the ADAM optimizer are set to 0.001 and 0.9, respectively. The batch size is set to

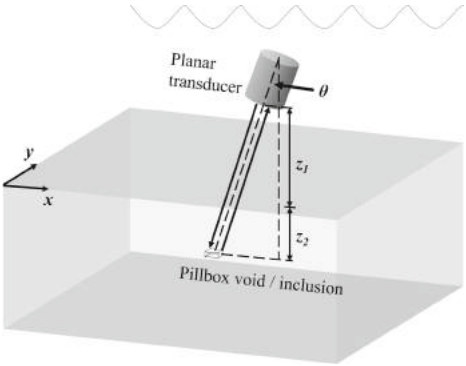



**Fig. 5.** A schematic of setup for the ultrasonic testing case.

16 and the maximum number of epochs is set to 2,000. L2 regularization is used with $\lambda = 0.1$ while training the models.

Figure 6 shows the direct GSA converged at 3000 sampling points. The convergence criteria of sequential GSA in this case were the same as in the previous case. The outer-loop sequentially iterated from 100 to 400 LHS samples. Figures 7(a) and 7(b) show the inner-loop GSA convergence using the NN-based algorithm for the first- and total-order indices require $10^6$ and $10^3$ samples, respectively. Figure 8 shows the outer-loop convergence plots for the first- and total-order indices both terminated at 400 samples. Figure 9 shows that the distance from the transducer to the defect has a negligible effect on the output response, while the probe angle has the highest effect follows by $y$ coordinate then $x$ coordinate. Table 2 compares the Sobol' indices values from the proposed method to those from the true function. It shows a good match of the the Sobol' indices values while the cost of sequential GSA is an order of magnitude less than the direct GSA.

**Table 2.** Case 2 comparison of Sobol' index values between the true model and the converged NN model.

| x | $S_i$ | | | $S_{T,i}$ | | |
|---|---|---|---|---|---|---|
| | True model | Seq. GSA | % error | true model | Seq. GSA | % error |
| $\theta$ | 0.5599 | 0.56 | 0.02% | 0.7424 | 0.7456 | 0.4% |
| $x$ | 0.0623 | 0.0571 | 8.3% | 0.2086 | 0.2082 | 0.2% |
| $y$ | 0.1841 | 0.1919 | 4.2% | 0.2449 | 0.2467 | 0.7% |
| $z_1$ | $5.2 \times 10^{-5}$ | $6.5 \times 10^{-5}$ | – | $5.3 \times 10^{-5}$ | $1.0 \times 10^{-4}$ | – |
| $z_2$ | $6.6 \times 10^{-4}$ | $5.2 \times 10^{-4}$ | – | $8.4 \times 10^{-4}$ | $7.6 \times 10^{-4}$ | – |

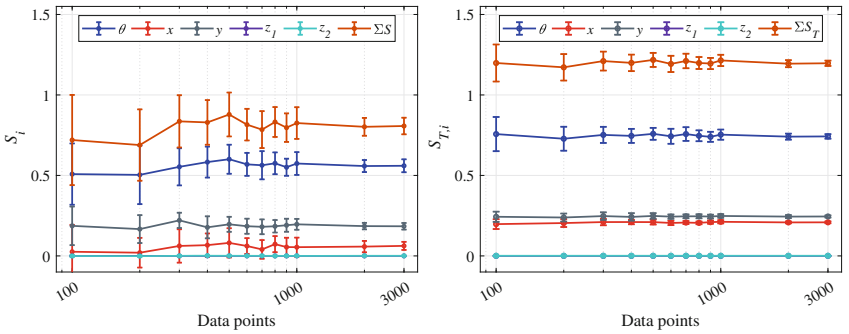



**Fig. 6.** Case 2 convergence of GSA directly on the physics model: (a) first-order indices, and (b) total-order indices.

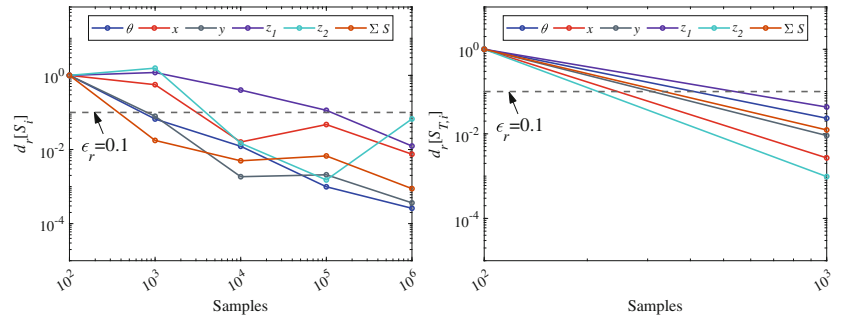



**Fig. 7.** Case 2 inner-loop convergence of $s_i$ for the NN trained with 400 LHS samples: (a) first-order indices, and (b) total-order indices.

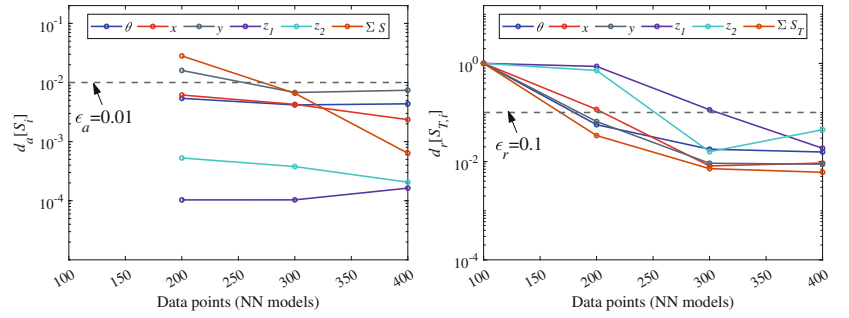



**Fig. 8.** Case 2 outer-loop convergence of $s_i$ terminated on (a) $d_a \leq \epsilon_a$ criteria: first-order indices, and on (b) $d_r \leq \epsilon_r$ criteria: total-order indices.

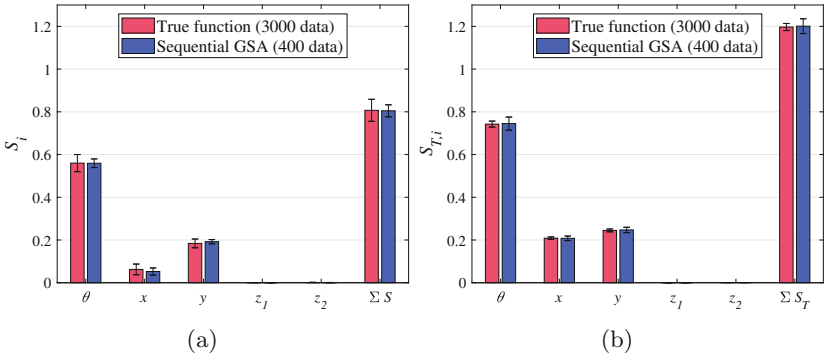



**Fig. 9.** Case 2 Sobol' index values of input parameters computed by the converged NN model: (a) first-order indices, and (b) total-order indices.

## 4 Conclusion

Global sensitivity analysis (GSA) of large-scale data sets is an important problem in engineering and science decision-making. The algorithm presented in this paper directly tackles this important task in the context of simulation-based problems. In particular, this work demonstrates that simulation-based GSA using neural network-based function prediction can be iteratively improved and terminated once accurate sensitivities are obtained, thereby enabling efficient adaptive GSA for large-scale problems.

Future steps in this work will focus on how to adaptively sample the NN model using model uncertainty, which will alleviate the resampling stage of the algorithm. An important step in this work will be to characterize the computational cost and benchmark it against current state-of-the-art methods, as well as to perform numerical experiments on high-dimensional problems involving physical and computational data.

**Acknowledgements.** This material is based upon work supported by the United States National Science Foundation under grant no. 1739551.

## References

1. Ferretti, F., Saltelli, A., Tarantola, S.: Trends in sensitivity analysis practice in the last decades. Sci. Total Environ. **568**, 666–670 (2016). https://doi.org/10.1016/j.scitotenv.2016.02.133
2. Iooss, B., Saltelli, A.: Introduction to Sensitivity Analysis. Springer International Publishing, Switzerland (2015). https://doi.org/10.1007/978-3-319-12385-1_31
3. Sobol', I., Kucherekoand, S.: Sensitivity estimates for nonlinear mathematical models. Math. Model. Comput. Exp. **1**, 407–414 (1993)
4. Sobol', I.: Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. Math. Comput. Simul. **55**, 271–280 (2001)

5. Zhou, X., Lin, H.: Local sensitivity analysis. In: Encyclopedia of GIS, pp. 1116–1119 (2017)
6. Homma, T., Saltelli, A.: Importance measures in global sensitivity analysis of nonlinear models. Reliabil. Eng. Syst. Saf. **52**, 1–17 (1996)
7. Liu, Y.-C., Nagawkar, J., Leifsson, L., Koziel, S., Pietrenko-Dabrowska, A.: Iterative global sensitivity analysis algorithm with neural network surrogate modeling. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12745, pp. 298–311. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77970-2_23
8. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2), 239–245 (1979). http://www.jstor.org/stable/1268522
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)
10. Haykin, S.S.: Neural Networks and Learning Machines, 3rd edn. Pearson Education, Upper Saddle River (2009)
11. Schmidhuber, J.: Deep learning in neural networks: an overview. Neural Netw. **61**, 85–117 (2015). https://doi.org/10.1016/j.neunet.2014.09.003
12. Gurrala, P., Chen, K., Song, J., Roberts, R.: Full wave modeling of ultrasonic NDE benchmark problems using Nystrom method. Rev. Progr. Quant. Nondestruct. Eval. **36**(1), 1–8 (2017)
13. Du, X., Leifsson, L., Meeker, W., Gurrala, P., Song, J., Roberts, R.: Efficient model-assisted probability of detection and sensitivity analysis for ultrasonic testing simulations using stochastic metamodeling. J. Nondestruct. Eval. Diagnost. Prognost. Eng. Syst. **2**(4), 041002(4) (2019). https://doi.org/10.1115/1.4044446

# Development of an Event-Driven System Architecture for Smart Manufacturing

Maksymilian Piechota[1], Mikołaj Nowak[1], and Dariusz Król[2(✉)]

[1] Faculty of Information and Communication Technology, Wrocław University of Science and Technology, Wrocław, Poland

[2] Department of Applied Informatics, Wrocław University of Science and Technology, Wrocław, Poland

dariusz.krol@pwr.edu.pl

**Abstract.** This paper describes the automated production data acquisition and integration process in the architectural pattern Tweeting Factory. This concept allows the use of existing production equipment with PLCs and the use of industrial IoT prepared for Industry 4.0. The main purpose of the work is to propose an event-driven system architecture and to prove its correctness and efficiency. The proposed architecture is able to perform transformation operations on the collected data. The simulation tests were carried out using real data from the factory shop-floor, services prepared for production monitoring, allowing the calculation of KPIs. The correctness of the solution is confirmed on 20 production units by comparing its results with the blackboard architecture using SQL queries. Finally, the response time for calculating ISO 22400 performance indicators is examined and it was verified that the presented solution can be considered as a real-time system.

**Keywords:** Application case studies · Data integration · Engineering optimization and design · Event-driven system architecture · Tweeting factory

## 1 Introduction

The paper focuses on a concept called Tweeting Factory [3,8]. Currently, in the manufacturing industry for data acquisition, the PLC controllers are mostly used [5]. However together with the development of the Industry 4.0 concept, intelligent sensors that are capable to communicate with various other systems [10] are getting more recognition. Additionally they can do some initial data processing. In the case of intelligent sensors, the whole process can be done by the sensor itself and the output values can be sent further over the protocols like AMQP or MQTT. Those new features open a variety of new capabilities to the production line systems. The Tweeting Factory is an architecture proposal for utilizing new features and providing the capabilities. It is an architecture pattern


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 455–468, 2022.
https://doi.org/10.1007/978-3-031-08757-8_38

providing a communication between production units, data processing services, and output data subscribers [12]. The machines are responsible for messages dispatched that are then processed by the services and the services send new messages. All messages in the system can be utilized by the subscribers. Every tenant in the system has access to all messages - the architecture implements the Publish-Subscribe pattern [6]. There are no constraints for the messages exchanged in the system. Only a few recommendations are made for the messages' metadata, such as the origin, subject, and publishing time. The services can read or send new messages, communicate with external systems and other data sources in order to leverage external information, data processing outside the system, or data recording.

The paper aims to validate the described architecture. Furthermore, the experiment designed for the verification purpose is described: based on the data acquisition process, the production environment project was designed that was the basis for the simulation environment. The simulation experiment was performed with the real production data provided by the cooperating company. The simulation results were compared with the results calculated by the blackboard style data acquisition process. Additionally, the Tweeting Factory performance was measured and analyzed. Finally, in the last section, the conclusions and possibilities for further research are discussed.

Before describing the methods in detail, we introduce the related work that influenced our research. Tweeting Factory term was used for the initial literature review. However, only four papers [3,8,11,12] were found using the term. Additionally, an alternative name for the architectural model was discovered - LISA, i.e., Line Information System Architecture [11]. Taking the alternative name into account and the low number of papers found in initial survey, there was a reasonable doubt that the Tweeting Factory concept exists, but the different name is used. Another term that was found during the analysis is called Digital Twin. It seems that Digital Twin is the higher abstraction concept and Tweeting Factory might be one of its implementations.

During the analysis, no comparison with the blackboard style SQL data acquisition method was found. The authors find this subject very important to eventually prove the advantages of the Tweeting Factory over the existing solution. Therefore, following research gaps were determined, that aims to be the authors contribution:

- RQ1: Is the Tweeting Factory concept correct?
- RQ2: Does the Tweeting Factory concept fulfill the real-time system constraints?
- RQ3: How the Tweeting Factory concept can improve the data acquisition process?
- RQ4: Is the Tweeting Factory a correct replacement of the SQL method?

## 2 Overview of the Tweeting Factory

### 2.1 Data Acquisition

Data acquisition is a process of measuring the physical values and storing them in a digital form [7]. The following phases of the process can be highlighted:

1. Physical value extraction
2. The value measurement by a sensor
3. The measured values transmission to a registration device
4. The values conversion to a digital form by ADC
5. Storing of the digital values

Data can serve different purposes, for example, KPIs computation, historical data analysis. event reporting such as machines' condition monitoring, environment parameters control, failure reporting, and material fatigue. Evolution of the Industry 4.0 concept is strictly coupled to intelligent sensors. In comparison with the previous generation sensors, intelligent sensors are capable to communicate with the other environment's participants, follow precise production, leverage production ontology, and use machine learning algorithms for independent conversation [10]. Industry 4.0 compatible sensors are capable to address all phases of data acquisition, besides the last one - the data storage. For this purpose, we use the cloud computing infrastructure. The paper's authors consider Tweeting Factory concept as a good candidate solution to address this requirement.

### 2.2 From Tweets to Decisions

Tweeting Factory concept provides additional benefits for the manufacturing environments. Additional features of the architecture were found in the papers [8,12]: KPI indexes calculation, hybrid systems optimization, energy consumption optimization, production process planning, quality assurance, production machines management, and monitoring. It is possible to use Tweeting Factory concept for data storing only [3], however additional services provided by the concept create its value. For example, for KPIs calculation, the stored data from a requested period can be used instead of single data coming directly from PLCs. All above-mentioned data are calculated in real-time, so it is possible to immediately generate a notification event when the specific index reaches a predefined threshold. A significant advantage of Tweeting Factory allowing to add services operating on available data is the ability to connect to external services. Based on that, the systems using the architecture are capable to easily connect and take part in the Shared Industry concept [14] or to be used in other applications outside the manufacturing industry [16]. Considering the Tweeting Factory as a data stream, it is also possible to apply machine learning algorithms.

### 2.3 Tweeting Factory Architecture

The base sources of the data are production units. Temperature or humidity are examples of the working environment measured values. Machine's work parameters such as state (idle/active), specific state time, or smaller component state

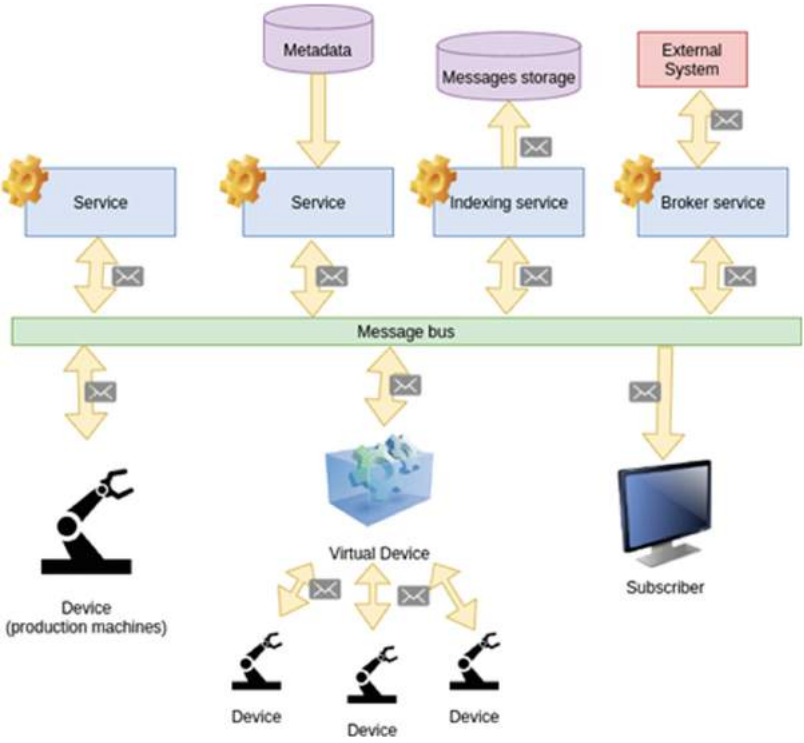



**Fig. 1.** The Tweeting Factory architecture model

are also measured. The machines can send that information themselves or to the connected PLCs. Devices other than production units can be found on the production line, for instance, a barcode reader. They can also send data themselves or to the PLC controllers. Another category of data sources are services. Services can also subscribe by receiving various types of messages available on Tweeting Factory's bus. In most cases, the data published by the service is just a transformation of the data received. Tweeting Factory provides support to connect and leverage external systems. Metadata models proposed in [11] allow to decorate the messages with the context of the external systems. However, the metadata is not the only advantage. Tweeting Factory provides a way to emit events registered in external systems. It allows to gather context data, not limited to data generated by production machines, for example, information entered manually by an employee or imported from HR systems, such as work hours, holidays, employee availability, or workplace assignments.

The Tweeting Factory architecture is depicted in Fig. 1. Production units, including virtual ones, can measure and convert the physical values to a digital representation itself. However, they are not capable to store data. This responsibility belongs to indexing services and subscribers.

When dealing with messages from different sources, it is highly possible to get incompatible messages' formats. The incompatibility might be a reason for incorrect data handling. This problem is called the interoperability issue and is considered on a semantic level [13]. Production machines and external systems produce information with incompatible labels, what leads the subscriber service's data parsing issues. A well-known solution of the interoperability issue is an application of a domain ontology or an application of an existing information exchange standard, such as ISO 10303 [4]. Standardization of the system's message layout solves the interoperability issue on the semantic level [13]. The literature provides different solutions [1], however, considering a variety of manufacturing industry applications, it is impossible to establish one unified standard. Different ontologies might be leveraged in providing a unified standard for the specific application. For example, in the paper, the authors decided to replace labels created by specific machines with URL labels. The interoperability issue resolved can be seen in Fig. 2. The external system's messages need to be translated by the broker service. For the production machines, if it is not possible to configure their messages' labels, translation services [15] should be introduced.

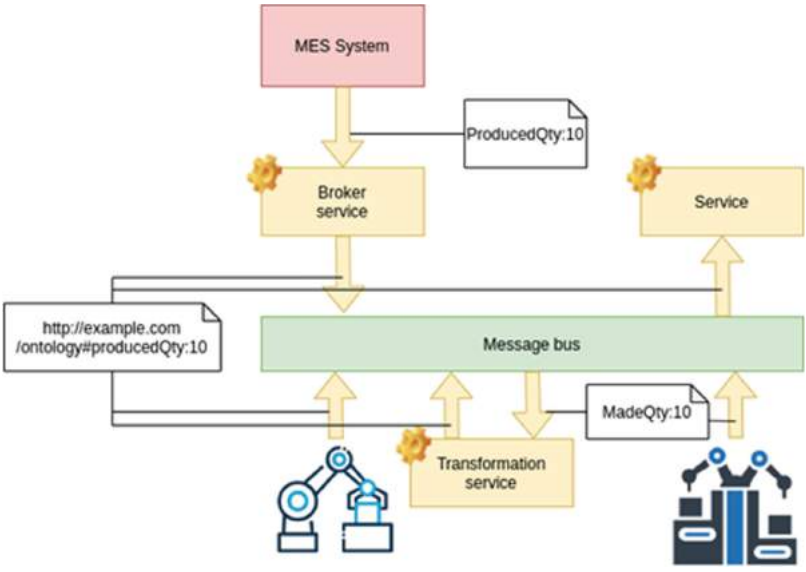



**Fig. 2.** Ontology based solution of the interoperability issue in the Tweeting Factory environment

## 3 Experiments

In this section, an experiment validating the Tweeting Factory concept and its results are presented. In the experiment, KPI metrics were calculated by Tweeting Factory simulation environment and by the blackboard style method. The

analysis of the experiment's results allowed the authors to answer the presented paper's research questions. To prove the Tweeting Factory correctness (RQ1), the calculated metrics were compared with the metrics calculated by standard methods. The comparison proved also if the Tweeting Factory is a good replacement for the legacy SQL method (RQ4). Additionally, the performance of the system is measured and analyzed (RQ2).

The experiment was divided into the following steps:

1. Analyze source data.
2. Calculate KPI indexes by the standard method.
3. Design the new data acquisition method.
4. Complete the implementation.
5. Conduct a simulation run.
6. Analyze the results.

Source data analysis (1) aims to check if the provided production data is correct for the experiment, what quality it is, and what time range and units should be chosen as an input to the simulation. The initial KPI calculation (2) was done by the legacy SQL method. The OEE-based metrics were chosen for the experiment, considering their universality [2]. The data acquisition method design (3) requires the specific production infrastructure analysis and leads to the Tweeting Factory architecture preparation for the specific environment. Simulation implementation based on the previous step's results. A set of programming tools was prepared that played the service role in the Tweeting Factory environment. Another tools were verifying the resulting data. The simulation run (5) was a simple run of the prepared tools. The results analysis (6) aimed to compare the resulting KPI metrics with the standard method's initial calculation from the previous step (2). Additionally, calculation times were analyzed for performance metrics.

Real-world data provided by the heating devices factory was used in the experiment. It provides such features as: production order reporting, machines' production data automated acquisition, gathered data analysis. In the experiment, the automatically collected data as well as the data provided by the employees was used. The provided database served as a source of the production data.

After data analysis, the following data was determined to be required for the experiment:

1. Products data: the data source production machine's name, the product completion time, the amount of the successfully produced and rejected products, estimated reference unit time of work. Described data was available in various tables in provided database.
2. Production machine's work time: the machine's code, the event's start timestamp, the event's end timestamp, the event's type, the event's value.
   The event's value corresponding to work time is either Work or PW. This type of event can take value 0 (stopped working) or 1 (started working). These records were emitted every time a machine started or stopped to work and also at the top of every hour.

Based on determined event types it is possible to determine KPI metrics to calculate:

- $PQ$ - Produced Quantity - based on products' completion events,
- $RQ$ - Rejected Quantity - based on products' completion events,
- $PRI$ - Planned Run Time - based on products' completion events,
- $APT$ - Actual Production Time - based on a machine's work time events,
- $PBT$ - Planned Busy Time - based on a machine's work time events.

In order to calculate the OEE index [2], the following component metrics are required:

$$A = \frac{APT}{PBT},\ E = \frac{PRI * PQ}{APT},\ Q = \frac{PQ - RQ}{PQ} \tag{1}$$

None of the above indexes can't reach a value greater than 1, however it can happen as a result of false PRI estimation. Then the performance index is limited to 1. The OEE index is a product of three indexes:

$$OEE = A * E * Q = \frac{PRI * (PQ - RQ)}{BPT} \tag{2}$$

Assuming the available experiment's data is sufficient for the OEE index calculation.

The last step of the source data analysis is to determine the event data subset for the simulation. Every event data contains the code of a machine and its timestamp. It allows to connect the machine's event data with a product's events. The number of products' completion events versus production unit is presented in Fig. 3(a). To achieve better readability, the machines with the number of such events less than 20 were filtered out. From the plot, it can be noticed that four machines emit more events: 0005, 5309-1, 5309-2, 8105.

The number of a production machine's work time events is presented in Fig. 3(b). It can be noticed that the machines highlighted above, except for 0005, have also a lot of work time events.

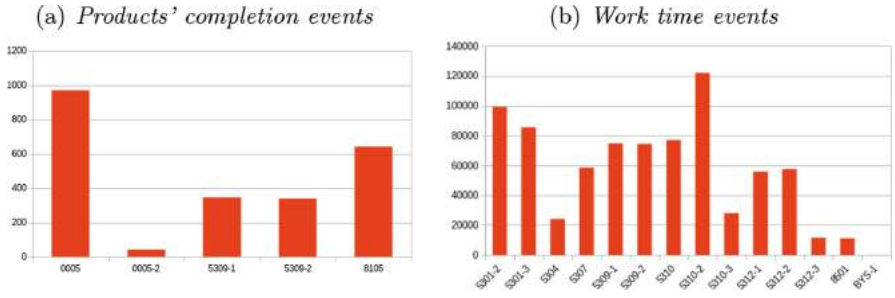



**Fig. 3.** Distribution versus production unit

Assuming the three machines highlighted above (5309-1, 5309-2, 8105) were chosen for the simulation, as those were the only ones that had a high number

of events of both types. To determine the time range of events for the simulation, the number of both types of events generated by the chosen machines was analyzed versus months. The results are presented in Fig. 4(a) and Fig. 4(b).

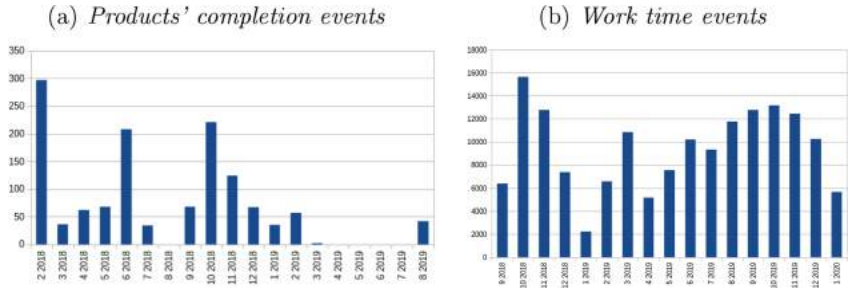



**Fig. 4.** Distribution versus months

To use the largest unit of data, the following time range was selected from 2018-09-17 to 2019-09-16. It can be noticed that there are months when no products' completions were registered. It may be because employees refrain from registering the events in the system terminals. In such cases, it is impossible to calculate the quality, performance and the OEE indexes, but the availability index is still possible to calculate. Additionally, considering a low number of products' completion events in general, the indexes will be calculated versus days.

### 3.1 KPI Indexes Calculation by the Legacy Method

To verify the Tweeting Factory concept, the KPI metrics were calculated offline using SQL query. The well-known considerations and limitations were applied to the legacy method.

### 3.2 A New Data Acquisition Method

Considering that the available data set does already contain the production information of units and entered by employees, only OEE index acquisition is considered. The available production data is further considered as production events (employee or machine). Following Tweeting Factory architecture, the events are transmitted by the message bus between the machines and the services that subscribe and publish the events. For the needs of the simulation, the assumption was made that the system is publishing the data on the message bus instead of storing it. The following services were introduced to the designed environment:

- a service transforming the machines' events,
- a service calculating the KPI indexes in real time,
- a service storing the KPI indexes into a database.

The designed architecture is presented in Fig. 5.

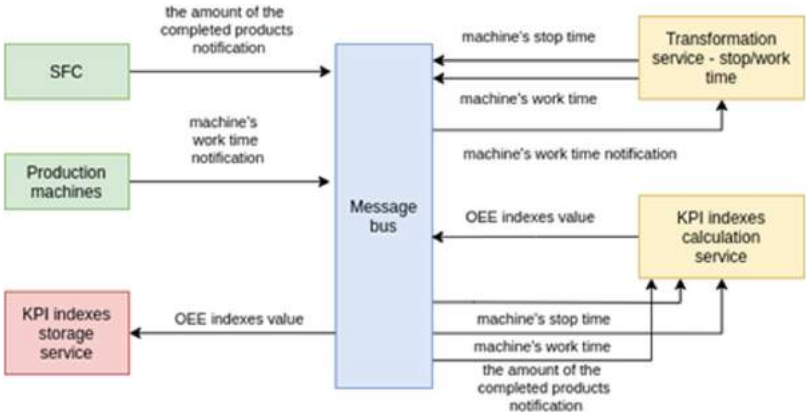



**Fig. 5.** The Tweeting Factory architecture proposal

Having the Tweeting Factory not specifying the message formats [12], JSON format is applied in the architecture. The base message format includes: start, end, machine, e_type, and payload. The e_type and payload field values are specific to the registered event. The e_type field is the key value of the registered event. The payload field contains the data specific for the event:

- the number of completed product notifications: QtyAll - an amount of produced products, QtyRejected - an amount of rejected products, RefTime - an estimated reference unit time for a product,
- work time notification: work - 0 for stopped, 1 for running
- work time: time - work time in seconds
- stop time: time - stop time in seconds
- OEE indexes value: pq - an amount of all produced products, rq - an amount of all rejected products, pri_pq - the product of of manufactured products and planned working time per unit, in seconds, apt - machine's work time in seconds, pbt - machine's estimated work time in seconds, quality - quality index value, - efficiency - efficiency index value, availability - availability index value, oee - OEE index value.

### 3.3 Simulation Implementation

To adjust the design to the simulation conditions, the following extensions were made:

- the source events generated originally by the system and the machines are emitted by the simulation program,

– in order to shorten the simulation time (365 days of the chosen time range), the simulation clock event is introduced,
– start and end events indicating the whole simulation start and end time are introduced,
– in order to measure a KPI calculation time, the message's metadata were extended by a new parameter trace_id, being an unique message's id and a new service calculating the time.

The extensions implemented on the architecture are presented in Fig. 6.

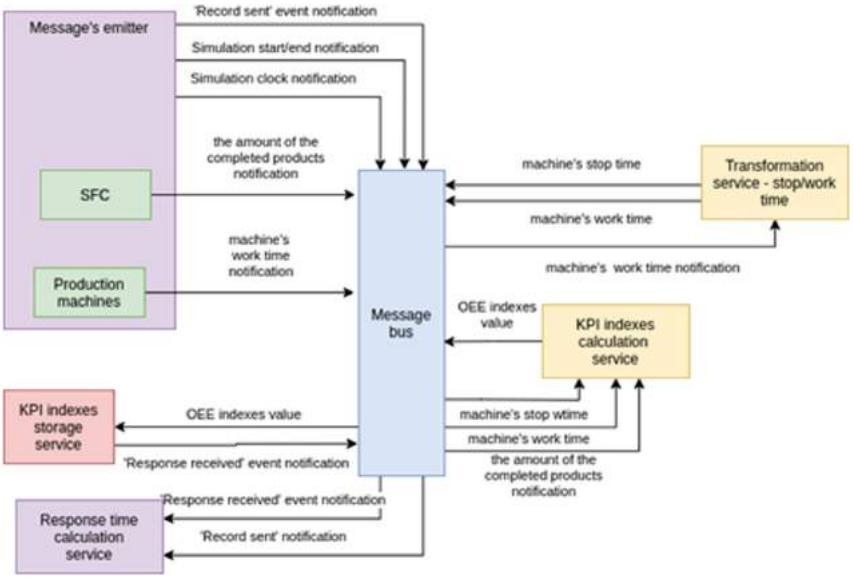



**Fig. 6.** Simulation environment's extended architecture

The Tweeting Factory concept does not specify any software for the message bus [12] implementation. The authors decided to use RabbitMQ[1]. The main advantage of the software over the Apache ActiveMQ[2] is ease of use and an extensive documentation. All services are implemented using Python3 and the Pika library[3] that enables a communication with the RabbitMQ server over the AMQP 0.9.1 protocol. Additionally, a CSV file with the input data for the simulation messages emitter was prepared. The CSV contains an array of source events, sorted by the publication time. A new type of messages were added in the extended simulation architecture:

[1] https://www.rabbitmq.com/.
[2] https://activemq.apache.org/.
[3] https://pypi.org/project/pika/.

– 'record sent' event notification: real_time - the event's timestamp, id - id for time measurement,
– 'response received' event notification: real_time - the event's timestamp, id - id for time measurement,
– simulation clock notification: clock - simulation timestamp,
– simulation start notification,
– simulation end notification.

### 3.4 Simulation Run

The run was performed on a PC equipped with AMD Ryzen 5 3600 processor unit and 16GB 3200MHz RAM memory. The message bus component was implemented in the Podman container[4] and docker.io/library/rabbitmq:3.8.4-management container image was used.

The average KPI indexes for the machines are presented in Table 1.

**Table 1.** The average KPI indexes for the production units

| Unit | Quality | Efficiency | Availability | OEE |
|---|---|---|---|---|
| (a) | (b) | (c) | (d) | (e) |
| 5309-1 | 0.9988 | 0.4625 | 0.4644 | 0.2657 |
| 5309-1 | 0.9949 | 0.5431 | 0.4087 | 0.2543 |
| 8105 | 0.9884 | 0.7952 | 0.6920 | 0.5857 |

Column (a) shows the symbolic codes of the selected production units. In column (b) the quality value is placed, while in the next two columns there are efficiency (c) and availability values (d). The last column (e) lists the total value of OEE.

The correctness verification was performed by the comparison of the Tweeting Factory approach results with the results of the legacy method. The following metrics were compared: PQ, RQ, PRI*PQ, APT, PBT, Quality, Efficiency, Availability, OEE. The comparisons have occurred 1095 times (365 days, 3 machines). No significant differences between the output results were found. The performance verification was done by the analysis of the data generated by the simulation indicating the waiting time for the KPI index calculation. The time was measured 113542 times. Average time was 0.012520 s, i.e., 12.52 ms. In the case of about 99% of the results, the calculation time was less than 25 ms. Figure 7(a) presents the box plot of the data. Large number of outliers can be generated by the temporary high load of the KPI calculation service or the temporary spike of the processor demand of the operating system.

Anderson-Darling test [9] was performed to determine whether a normal distribution adequately describes a set of data. Significance level 0.05 was assumed. The following hypotheses were tested:

[4] https://podman.io/.

– H0: the results are normally distributed,
– H1: the results are not normally distributed.

The output p-value was $3.7 \times 10^{-24}$, so the alternative hypothesis was accepted. Shapiro-Wilk test [17] was also performed to confirm the previous test output. The input data for the test was limited to 2000 records to keep the test's power. Significance level 0.05 was assumed. The following hypotheses were tested:

– H0: the results are normally distributed.
– H1: the results are not normally distributed.

The output p-value was $6.923531 \times 10^{-57}$, so the alternative hypothesis was accepted. Quantile distribution of the KPI indexed calculation time is presented in Fig. 7(b). The distribution is not linear, which means the distribution is not normal.

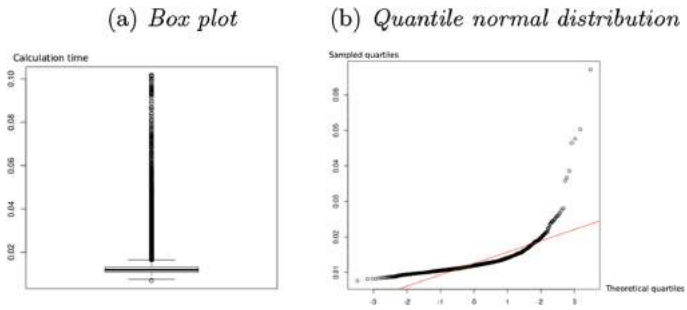



**Fig. 7.** KPI indexes calculation

To check if the calculated average value is a good representation of the random variable, the Wilcoxon test [37] was performed for one attempt. Significance level 0.05 was assumed. The following hypotheses were tested: - H0 - random variable's value's distribution is symmetric around = 0.012520. - H1 - random variable's value's distribution is not symmetric around = 0.012520. The output p-value was 0, so the alternative hypothesis was accepted. The median and percentile values are considered in further analysis. Statistical analysis showed no coincidence with the normal distribution of the computation time KPI indicators and the asymmetry of the time distribution with respect to the mean.

## 4 Conclusions

No differences between KPI index values calculated by the Tweeting Factory and the blackboard style method prove the Tweeting Factory concept correctness (RQ1). The response time of the proposed system that was shorter than 17ms in more than 95% cases puts the system among the real-time systems (RQ2).

Considering the above two conclusions, the Tweeting Factory becomes a great candidate for a replacement of the SQL legacy method (RQ4), providing a lot of advantages (RQ3):

- ease of new production units and service integration,
- capability to add and remove new components without a need for reconfiguration,
- control over the messages flow,
- ESB controls data exchange security,
- complex metrics calculation in real-time,
- support to integrate the system with external systems.

There are also cons that need to be taken into account when considering the concept:

- data transformations required for the components using unsupported protocols,
- single point of failure - ESB,
- security of data exchange under the sole control of the message broker.

The literature analysis demonstrated a low number of concept applications in the industry. Taking the experiment's promising results into account the authors state that the lack of real life application studies is a major gap that may be a subject for the further research. Considering the literature survey conclusion that the Tweeting Factory concept can be used as a Digital Twin core component, the Digital Twin with Tweeting Factory application in Shared Industry [14]. Leveraging the real-time feature of the concept, another interesting research subject could be managed and optimized using machine learning algorithms.

**Acknowledgment.** Part of the work presented in this paper was received financial support from the statutory funds at the Wrocław University of Science and Technology and DSR Ltd.

## References

1. Çetiner, G., Ismail, A., Hassan, A.: Ontology of manufacturing engineering. In: 5th International Advanced Technologies Symposium, p. 6 (2009)
2. De Ron, A., Rooda, J.: Equipment effectiveness: OEE revisited. IEEE Trans. Semicond. Manuf. **18**(1), 190–196 (2005)
3. Dressler, N.: Towards The Tweeting Factory. Master's thesis, KTH Industrial Engineering and Management, SE-100 44 Stockholm (2015)
4. Feeney, A.: The step modular architecture. J. Comput. Inf. Sci. Eng. **2**(2), 132–135 (2002)
5. Gittler, T., Gontarz, A., Weiss, L., Wegener, K.: A fundamental approach for data acquisition on machine tools as enabler for analytical industrie 4.0 applications. Procedia CIRP **79**, 586–591 (2019)
6. Hoffmann, M.: Smart Agents for the Industry 4.0, 1st edn. Springer Vieweg, Heidelberg (2019). https://doi.org/10.1007/978-3-658-27742-0

7. Kos, T., Kosar, T., Mernik, M.: Development of data acquisition systems by using a domain-specific modeling language. Comput. Ind. **63**(3), 181–192 (2012)
8. Lennartson, B., Bengtsson, K., Wigström, O., Riazi, S.: Modeling and optimization of hybrid systems for the tweeting factory. IEEE Trans. Autom. Sci. Eng. **13**(1), 191–205 (2016)
9. Nelson, L.: The Anderson-Darling test for normality. J. Qual. Technol. **30**(3), 298–299 (1998)
10. Schütze, A., Helwig, N., Schneider, T.: Sensors 4.0 - smart sensors and measurement technology enable industry 4.0. J. Sensors Sensor Syst. **7**(1), 359–371 (2018)
11. Theorin, A., et al.: An event-driven manufacturing information system architecture. IFAC-PapersOnLine **48**(3), 547–554 (2015)
12. Theorin, A., et al.: An event-driven manufacturing information system architecture for industry 4.0. Int. J. Prod. Res. **55**(5), 1297–1311 (2017)
13. Tursi, A.: Ontology-Based approach for Product-Driven interoperability of enterprise production systems. Phd thesis, Université Henri Poincaré - Nancy 1, Politecnico di Bari (2009)
14. Wang, G., Zhang, G., Guo, X., Zhang, Y.: Digital twin-driven service model and optimal allocation of manufacturing resources in shared manufacturing. J. Manuf. Syst. **59**, 165–179 (2021)
15. Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging, 1st edn. Addison-Wesley, Boston (2003)
16. Yan, X.: Knowledge Acquisition from Streaming Data through a Novel Dynamic Clustering Algorithm. Phd thesis, North Carolina Agricultural and Technical State University (2018)
17. Zhao, L., Chuang, Z., Ke-Fu, X., Meng-Meng, C.: A computing model for real-time stream processing. In: 2014 International Conference on Cloud Computing and Big Data, pp. 134–137 (2014)

# Boundary Geometry Fitting with Bézier Curves in PIES Based on Automatic Differentiation

Krzysztof Szerszeń(✉) and Eugeniusz Zieniuk

Institute of Computer Science, University of Bialystok, Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland
{kszerszen,ezieniuk}@ii.uwb.edu.pl

**Abstract.** This paper presents an algorithm for fitting the boundary geometry with Bézier curves in the parametric integral equation system (PIES). The algorithm determines the coordinates of control points by minimizing the distance between the constructed curves and contour points on the boundary. The minimization is done with the Adam optimizer that uses the gradient of the objective function calculated by automatic differentiation (AD). Automatic differentiation eliminates error-prone manual routines to evaluate symbolic derivatives. The algorithm automatically adjusts to the actual number of curves and their degrees. The presented tests show high accuracy and scalability of the proposed approach. Finally, we demonstrate that the resulting boundary may be directly used by the PIES to solve the boundary value problem in 2D governed by the Laplace equation.

**Keywords:** Automatic differentiation · Parametric curve fitting · Bézier curves · Parametric integral equation system (PIES) · Boundary value problems

## 1 Introduction

One of the main difficulties during computer simulation of boundary value problems (BVP) is the appropriate definition of the computational domain. Typically it is done by dividing the problem domain into finite elements (FEM) [1] or only the boundary of that domain into boundary elements (BEM) [2]. However, such discretization is extremely laborious as it requires hundreds or thousands of elements and even more nodes are necessary to declare them. The alternative is to introduce the mathematical and geometric tools used in computer graphics and CAD/CAM systems. This is used in the parametric integral equation system (PIES) where the boundary of the computational domain is bounded by parametric curves [3] and surfaces [4]. Moreover, due to the analytical integration of the boundary geometry directly in the PIES formula, there is a lot of freedom in choosing the appropriate structure of the boundary representation. Our previous studies have shown that it is particularly effective to declare such boundary geometries by employing Bézier parametric curves defined by a small set of control points. This allows for a continuous representation of the boundary and the number of the curves is significantly smaller than finite or boundary elements required to solve the same problem.


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 469–475, 2022.
https://doi.org/10.1007/978-3-031-08757-8_39

Despite these advantages, parametric curves also have some implementation difficulties since its shape is determined by control points that generally do not lie on the curve. Curve fitting is a fundamental problem in computer graphics and has become a very active scientific area. Mathematically, it can be formulated as optimization problem to minimize the distance between the points describing the approximated shape and the points on the parametric curve. There is a rich literature to solve such optimization problem with several linear [5] and non-linear [6] least-squares techniques. Another approaches are based on biologically inspired solutions: genetic algorithms [7], simulated annealing [8], particle swarm optimization [9], evolutionary algorithms [10], artificial immune systems [11]. In [12] neural network-based curve fitting technique is presented. However, most existing algorithms are based on gradient descent minimization [13]. On the other hand, analytical evaluation of gradients is a labor-intensive task and sensitive to human errors, especially in the case of real problems defined by many design variables related to the coordinates of control points.

This paper attempts a new look at the parametric curve fitting applied to the boundary approximation in PIES. This is done by minimizing the distance between the constructed curves and the contour points on the boundary with the Adam optimizer [14], where the required derivatives of the objective function are computed by automatic differentiation (AD). The idea of AD is based on a decomposition of an input function into elementary operations, whose local derivatives are easy to compute. While AD has a long history and is supported by many publications [15, 16], the recent revolution in deep learning and artificial intelligence has brought a new stage in the development of advanced AD tools and new optimization algorithms. Popular libraries such as TensorFlow [17] and PyTorch [18] provide efficient AD and optimization algorithms for large models with thousands or even millions of parameters. The results of the presented tests illustrate good fitting properties of the proposed algorithm for various shapes of the boundary and high scalability. The fitted boundaries are directly used in PIES to solve BVP governed by the Laplace equation.

## 2 Defining the Boundary by Bézier Curves in PIES

We consider a boundary value problem governed by the Laplace equation defined in a domain $\Omega$ with a boundary $\Gamma$, as shown in Fig. 1a.

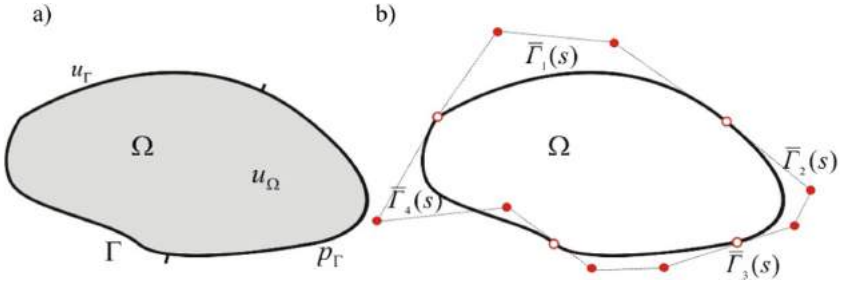



**Fig. 1.** BVP defined in the domain $\Omega$ with the boundary $\Gamma$(a), boundary description in PIES with 4 Bézier curves and 12 control points (b).

The boundary of that domain is described in PIES by a set of Bézier curves. A single Bézier curve of degree $m$ is defined by the location of the $m+1$ control points $P_i$ and is mathematically described as

$$\overline{\Gamma}(s) = \sum_{i=0}^{m} \binom{m}{i} s^i (1-s)^{m-i} P_i, \tag{1}$$

where $s$ is a parametric coordinate along the curve ($0 \leq s \leq 1$). Figure 1b shows a practical definition of the boundary formed by joining 4 cubic Bézier curves. The formula of PIES for 2D BVP governed by the Laplace equation is presented below

$$0.5u_l(\overline{s}) = \sum_{j=1}^{n} \int_{s_{j-1}}^{s_j} \left\{ \overline{U}^*_{lj}(\overline{s}, s) p_j(s) - \overline{P}^*_{lj}(\overline{s}, s) u_j(s) \right\} J_j(s) ds, \tag{2}$$

where $s_{j-1} \leq s \leq s_j$, $s_{l-1} \leq \overline{s} \leq s_l$, $l = 1, 2, ..., n$.

The Bézier curves $\overline{\Gamma}(s)$ are included analytically in the kernels $\overline{U}^*_{lj}(\overline{s}, s)$ and $\overline{P}^*_{lj}(\overline{s}, s)$ written as

$$\overline{U}^*_{lj}(\overline{s}, s) = \ln \frac{1}{[\eta_1^2 + \eta_2^2]^{0.5}}, \tag{3}$$

$$\overline{P}^*_{lj}(\overline{s}, s) = \frac{\eta_1 n_1^{(j)}(s) + \eta_2 n_2^{(j)}(s)}{\eta_1^2 + \eta_2^2}, \tag{4}$$

$$\text{where } \eta_1 = \overline{\Gamma}_l^{(1)}(\overline{s}) - \overline{\Gamma}_j^{(1)}(s),\ \eta_2 = \overline{\Gamma}_l^{(2)}(\overline{s}) - \overline{\Gamma}_j^{(2)}(s)\,. \tag{5}$$

Moreover, $n_1^{(j)}(s)$, $n_2^{(j)}(s)$ denote the components of the normal vector to the boundary and $J_j(s)$ is the Jacobian. The reader can find details of the PIES mathematical formulation in several previous papers, for example in [3].

## 3 Proposed Algorithm

Let $G_\Gamma$ be a set of data points sampled along the boundary and $G_{\overline{\Gamma}}$ is a set of points lying on Bézier curves. The objective function $loss_{L2}(G_\Gamma, G_{\overline{\Gamma}})$ of our optimization problem is the $L_2$ distance between $G_\Gamma$ and $G_{\overline{\Gamma}}$. The gradient of the objective function with respect to the corresponding control points $P$ of the curves is written as

$$\frac{\partial loss(P)}{\partial P} = \frac{\partial loss_{L2}(G_\Gamma, G_{\overline{\Gamma}})}{\partial \overline{\Gamma}} \frac{\partial \overline{\Gamma}}{\partial P}. \tag{6}$$

The gradient (6) can be derived analytically, however, is laborious and prone to human errors. Therefore, we compute it employing AD which decomposes the derivative of the objective function into those of elementary operations based on the chain rule. This decomposition can be described by a computational graph that shows the relations between individual differentials. For demonstrating the procedure, an elementary case

with one cubic Bézier curve defined by 4 control points is analyzed below. In this case, the formula (6) reduces to the following form

$$\frac{\partial loss(P)}{\partial P} = \frac{\partial}{\partial P}(G_\Gamma - CAP)^2, \tag{7}$$

where $C, A, P$ are referred to a matrix representation for the cubic Bézier curve

$$C = \begin{bmatrix} s^3 & s^2 & s & 1 \end{bmatrix}, A = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, P = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}. \tag{8}$$

Finally, the computational graph for formula (7) is presented below.

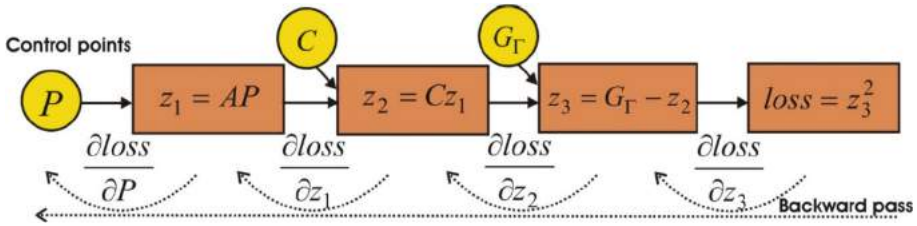



**Fig. 2.** Structure of the computational graph for a single cubic Bézier curve.

Table 1 contains the vertices $z_i$ corresponding to the intermediate computed variables for elemental operations and their derivatives for forward and backward data flow.

**Table 1.** Forward and backward propagation of derived values for the graph shown in Fig. 2.

| | Forward pass | Backward pass |
|---|---|---|
| $z_1 = AP$ | $\frac{\partial z_1}{\partial P} = A$ | $\frac{\partial loss}{\partial P} = \frac{\partial loss}{\partial z_1}\frac{\partial z_1}{\partial P} = -2CA$ |
| $z_2 = Cz_1$ | $\frac{\partial z_2}{\partial P} = \frac{\partial z_2}{\partial z_1}\frac{\partial z_1}{\partial P} = CA$ | $\frac{\partial loss}{\partial z_1} = \frac{\partial loss}{\partial z_2}\frac{\partial z_2}{\partial z_1} = -2C$ |
| $z_3 = G_\Gamma - z_2$ | $\frac{\partial z_3}{\partial P} = \frac{\partial z_3}{\partial z_2}\frac{\partial z_2}{\partial P} = -CA$ | $\frac{\partial loss}{\partial z_2} = \frac{\partial loss}{\partial z_3}\frac{\partial z_3}{\partial z_2} = -2$ |
| $loss = z_3^2$ | $\frac{\partial loss}{\partial P} = \frac{\partial loss}{\partial z_3}\frac{\partial z_3}{\partial P} = -2CA$ | $\frac{\partial loss}{\partial z_3} = 2$ |

The computational graph can dynamically update its structure for more Bézier curves and control points. The full diagram of the procedure with backward mode AD as best suited for our problem is shown in Fig. 3.

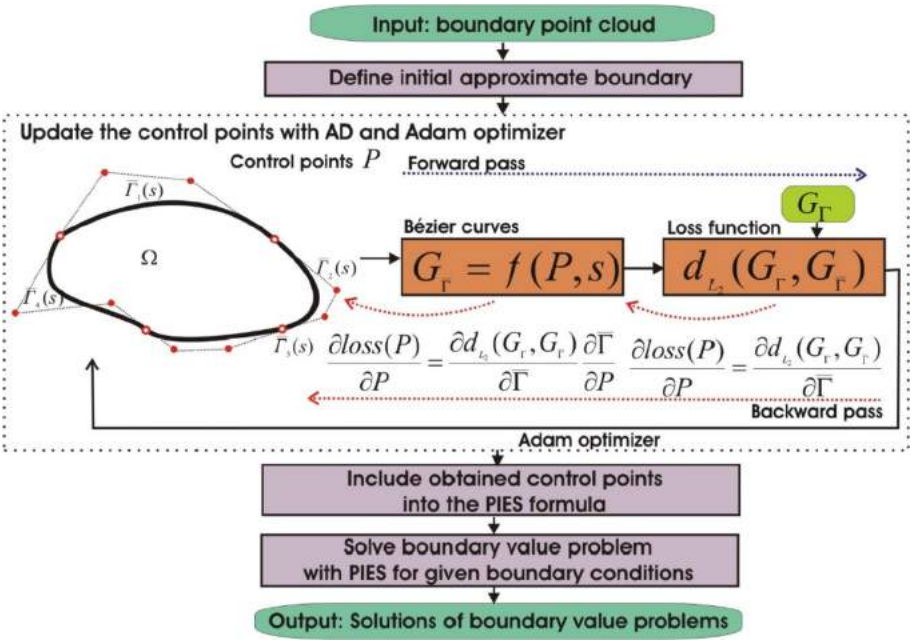



**Fig. 3.** Schematic flowchart of the proposed curve fitting combined with PIES.

To determine the coordinates of control points, the ADAM [14] optimization algorithm is used characterized by fast convergence, and resistance to local minima.

## 4 Results and Evaluation

The scheme given in Fig. 3 is implemented in the PyTorch framework providing access to AD and Adam optimizer modules. Figure 4 shows the results of our experimental evaluation for 4 identified boundaries generated from 16 to 40 cubic Bézier curves.

The curves require 48 and 120 control points to be defined. The data set $G_\Gamma$ consists of evenly sampled points along the boundary. The first three shapes are sampled into 160 points and the last one into 400 points.

As it can be seen from the graphic results, all tested shapes are fitted successfully. The related numerical results are listed in Table 2. The final boundaries are directly used in PIES to simulate BVP governed by the Laplace equation for the following Dirichlet boundary conditions

$$\phi(x_1, x_2) = \cos(x_1)\cosh(x_2) + \sin(x_1)\sinh(x_1). \tag{9}$$

This is possible since the control points are included analytically in the kernels (3, 4). In order to solve the problem on the boundary, the collocation method [3] is adopted with 6 colocation points per each Bézier curve. The last row in Table 2 shows the $L_2$ error norm of the PIES solutions for all shapes from the 400th iteration of the fitting.

**Table 2.** Convergence of the objective functions during iterations and the PIES accuracy.

| $loss_{L2}(G_\Gamma, G_{\overline{\Gamma}})$ | | | | |
|---|---|---|---|---|
| Iteration | A | B | C | D |
| 1 | 75.3377 | 186.8021 | 212.6234 | 195.2782 |
| 50 | 25.1174 | 125.8349 | 147.8667 | 98.5266 |
| 100 | 1.0304 | 81.2112 | 87.9212 | 20.3172 |
| 200 | 0.0935 | 17.3265 | 1.5679 | 0.1621 |
| 400 | 0.0685 | 0.0831 | 0.0657 | 0.0954 |
| PIES ($L_2$ error norm) | | | | |
| 400 | 0.1567 | 0.1235 | 0.3643 | 0.3025 |

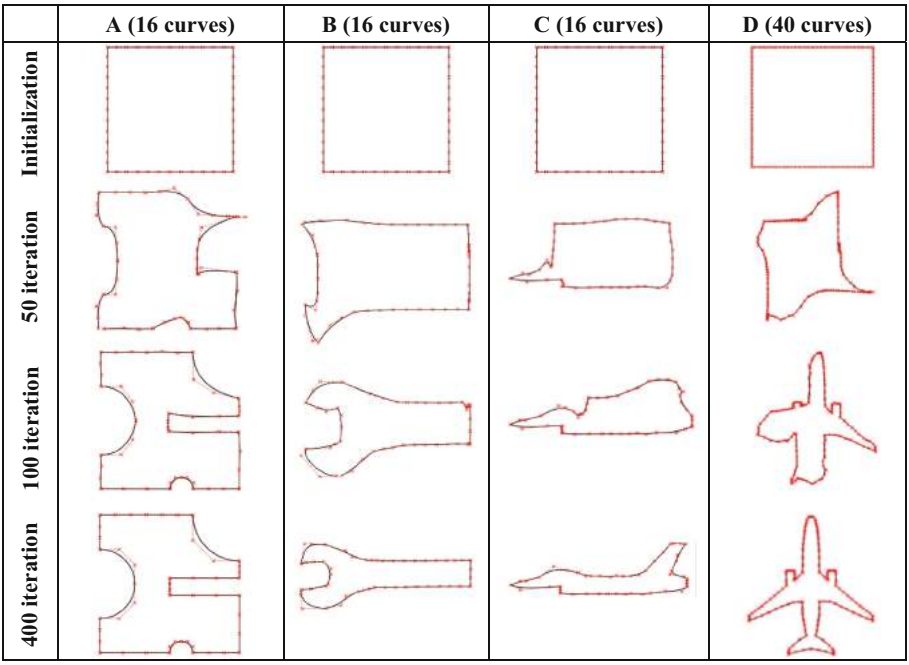



**Fig. 4.** Iterations 1, 50, 100, and 400 in the fitting process.

## 5 Conclusions

The results show that the proposed approach can be applied not only to academic but also to real-life problems with hundreds of design variables considered as the positions of control points. The boundary defined by the Bézier curves is integrated analytically with the PIES computational method so that it can be directly used to solve BVP, as shown in the example. The approach can use different types of objective functions and

optimization algorithms. We also hope to extend the research to study the boundary reconstruction from an unstructured point cloud as well to 3D problems.

## References


1. Zienkiewicz, O.C., Taylor, R.L.: The finite element method for solid and structural mechanics. Elsevier (2005)
2. Brebbia, C.A., Telles, J.C.F., Wrobel, L.C.: Boundary Element Techniques: Theory and Applications in Engineering. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-48860-3
3. Zieniuk, E., Szerszeń, K.: NURBS curves in direct definition of the shapes of the boundary for 2D Stokes flow problems in modified classical BIE. Appl. Numer. Math. **132**, 111–126 (2018)
4. Zieniuk, E., Szerszeń, K.: Nonelement boundary representation with Bézier surface patches for 3D linear elasticity problems in parametric integral equation system (PIES) and its solving using Lagrange polynomials. Numer. Methods Part. Differ. Eqn. **34**(1), 51–79 (2018)
5. Piegl, L., Tiller, W.: The NURBS Book. Springer, Heidelberg (1996). https://doi.org/10.1007/978-3-642-97385-7
6. Borges, C.F., Pastva, T.: Total least squares fitting of Bézier and B-spline curves to ordered data. Comput. Aided Geom. Des. **19**(4), 275–289 (2002)
7. Zhou, M., Wang, G.: Genetic algorithm-based least square fitting of B-Spline and Bézier curves. J. Comput. Res. Dev. **42**(1), 134–143 (2005)
8. Zhang, J., Wang, H.: B-Spline curve fitting based on genetic algorithms and the simulated annealing algorithm. Comput. Eng. Sci. **33**(3), 191–193 (2011)
9. Gálvez, A., Iglesias, A.: Efficient particle swarm optimization approach for data fitting with free knot B-splines. Comput. Aided Des. **43**(12), 1683–1692 (2011)
10. Pandunata, P., Shamsuddin, S.M.H.: Differential evolution optimization for Bézier curve fitting. In: 2010 Seventh International Conference on Computer Graphics, Imaging and Visualization, pp. 68–72 (2010)
11. Iglesias, A., Gálvez, A., Avila, A.: Discrete Bézier curve fitting with artificial immune systems. Stud. Comput. Intell. **441**, 59–75 (2013)
12. Bishop, C.M., Roach, C.M.: Fast curve fitting using neural networks. Rev. Sci. Instrum. **63**(10), 4450–4456 (1992)
13. Samir C., Absil, P.A., Srivastava, A., Klassen, E.: A gradient-descent method for curve fitting on Riemannian manifolds. Found. Comput. Math. **12**(1), 49–73 (2012)
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. Margossian, C.C.: A review of automatic differentiation and its efficient implementation. Wiley Interdisc. Rev. Data Mining Knowl. Disc. **9**, 1305e (2019)
16. Neidinger, R.D.: Introduction to automatic differentiation and MATLAB object-oriented programming. SIAM Rev. **52**(3), 545–563 (2010)
17. Abadi, M., et al.: {TensorFlow}: A System for {Large-Scale} Machine Learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283 (2016)
18. Paszke, A., et al.: Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. **32** (2019)

# Partitioning Dense Graphs with Hardware Accelerators

Xiaoyuan Liu[1,2(✉)], Hayato Ushijima-Mwesigwa[2], Indradeep Ghosh[2], and Ilya Safro[1]

[1] University of Delaware, Newark, DE, USA
{joeyxliu,isafro}@udel.edu
[2] Fujitsu Research of America, Inc., Sunnyvale, CA, USA
{hayato,ighosh}@fujitsu.com

**Abstract.** Graph partitioning is a fundamental combinatorial optimization problem that attracts a lot of attention from theoreticians and practitioners due to its broad applications.In this work, we experiment with solving the graph partitioning on the Fujitsu Digital Annealer (a special-purpose hardware designed for solving combinatorial optimization problems) and compare it with the existing top solvers. We demonstrate limitations of existing solvers on many dense graphs as well as those of the Digital Annealer on sparse graphs which opens an avenue to hybridize these approaches.

**Keywords:** Graph partitioning · Dense graphs · Digital annealer · Quantum-inspired

## 1 Introduction

There are several reasons to be optimistic about the future of quantum-inspired and quantum devices. However, despite their great potential, we also need to acknowledge that state-of-art classical methods are extremely powerful after years of relentless research and development. In classical computing, the development of algorithms, the rich mathematical framework behind them, and sophisticated data structures are relatively mature, whereas the area of quantum computing is still at its nascent stage. Many existing classical algorithms do not have provable or good enough bounds on the performance (e.g., they might not have ideal performance in the worst case), but in many applications, the worst-case scenarios are rather rarely seen. As a result, such algorithms, many of which heuristics, can achieve excellent results in terms of the solution quality or speed. Therefore, when utilizing emerging technologies such as quantum-inspired hardware accelerators and quantum computers to tackle certain problems, it is important to compare them not only with possibly slow but provably strong algorithms but also with the heuristic algorithms that exhibit reasonably good results on the instances of interest.


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 476–483, 2022.
https://doi.org/10.1007/978-3-031-08757-8_40

The graph partitioning [2] is one of the combinatorial optimization problems for which there exists a big gap between rigorous theoretical approaches that ensure best known worst-case scenarios, and heuristics that are designed to cope with application instances exhibiting a reasonable quality-speed trade-off. Instances that arise in practical applications often contain special structures on which heuristics are engineered and tuned. Because of its practical importance, a huge amount of work has been done for a big class of graphs that arise in such areas as combinatorial scientific computing, machine learning, bioinformatics, and social science, namely, *sparse graphs*. Over the years, there were several benchmarks on which the graph partitioning algorithms have been tested and compared with each other to mention just a few [1,3,21]. However, *dense graphs* can be rarely found in them. As a result, most existing excellent graph partitioning heuristics do not perform well in practice on dense graphs, while provable algorithms with complexity that depends on the number of edges (or non-zeros in the corresponding matrix) are extremely slow. As we also show in computational results, a graph sparsification does not necessarily practically help to achieve high-quality solutions.

*Multilevel Algorithms.* This class of heuristics is one of the most successful for a variety of cut-based graph problems such as the minimum linear arrangement [15], and vertex separator [7]. Specifically for a whole variety of (hyper)graph partitioning versions [10,11,16,18] these heuristics exhibit best quality/speed trade-off [2]. In multilevel graph partitioning frameworks, a hierarchy of coarse graph representations is constructed in such a way that each next coarser graph is smaller than the previous finer one, and a solution of the partitioning for the coarse graph can approximate that of the fine graph and be further improved using fast local refinement. Multilevel algorithms are ideally suited for sparse graphs and suffer from the same problems as the algebraic multigrid (which generalizes, to the best of our knowledge, all known multilevel coarsening for partitioning) on dense matrices. In addition, a real scalability of the existing refinement for partitioning is achieved only for sparse local problems. Typically, if the density is increasing throughout the hierarchy construction, various ad-hoc tricks are used to accelerate optimization sacrificing the solution quality. When such things happen at the coarse levels, an error is quickly accumulated. Here we compare our results with KaHIP [17] which produced the best results among several multilevel solvers [2].

*Hardware Accelerators for Combinatorial Problems.* Hardware accelerators such as GPU have been pivotal in the recent advancements of fields such as machine learning. Due to the computing challenges arising as a result of the physical scaling limits of Moore's law, scientists have started to develop special-purpose hardware for solving combinatorial optimization problems. These novel technologies are all unified by an ability to solve the Ising model or, equivalently, the quadratic unconstrained binary optimization (QUBO) problem. The general QUBO is NP-hard and many problems can be formulated as QUBO [14]. It is also often used as a subroutine to model large neighborhood local search [13].

The Fujitsu Digital Annealer (DA) [4], used in this work, utilizes application-specific integrated circuit hardware for solving fully connected QUBO problems. Internally the hardware runs a modified version of the Metropolis-Hastings algorithm for simulated annealing. The hardware utilizes massive parallelization and a novel sampling technique. The novel sampling technique speeds up the traditional Markov Chain Monte Carlo by almost always moving to a new state instead of being stuck in a local minimum. Here, we use the third generation DA, which is a hybrid software-hardware configuration that supports up to 100,000 binary variables. DA also supports users to specify inequality constraints and special equality constraints such as 1-hot and 2-way 1-hot constraints.

*Our Contribution.* The goal of this paper is twofold. First, we demonstrate that existing scalable graph partitioning dedicated solvers are struggling with the dense graphs not only in comparison to the special-purpose hardware accelerators but even sometimes if compared to generic global optimization solvers that are not converged. At the same time, we demonstrate a clear superiority of classical dedicated graph partitioning solvers on sparse instances. Second, this work is a step towards investigating what kind of problems we can solve using combinatorial hardware accelerators. Can we find problems that are hard for existing methods, but can be solved more efficiently with novel hardware and specialized algorithms? As an example, we explore the performance of Fujitsu Digital Annealer (DA) on graph partitioning and compare it with general-purpose solver Gurobi, and also graph partitioning solver KaHIP.

We do not attempt to achieve an advantage for every single instance, especially since at the current stage, the devices we have right now are still facing many issues on scalability, noise, and so on. However, we advocate that hybridization of classical algorithms and specialized hardware (e.g., future quantum and existing quantum-inspired hardware) is a good candidate to break the barriers of the existing quality/speed trade-off.

## 2 Graph Partitioning Formulations

Let $G = (V, E)$ be an undirected, unweighted graph, where $V$ denotes the set of $n$ vertices, and $E$ denotes the set of $m$ edges. The goal of perfect balanced $k$-way graph partitioning (GP), is to partition $V$ into $k$ parts, $V_1, V_2, \cdots, V_k$, such that the $k$ parts are disjoint and have equal size, while minimizing the total number of *cut edges*. A *cut edge* is an edge that has two end vertices assigned to different parts. Sometimes, the quality of the partition can be improved if we allow some imbalance between different parts. In this case, we allow some imbalance factor $\epsilon > 0$, and each part can have at most $(1 + \epsilon)\lfloor n/k \rfloor$ vertices.

*Binary Quadratic Programming Formulation of GP.* We first review the integer quadratic programming formulation for $k$-way GP [8,20]. When $k = 2$, we introduce binary variables $x_i \in \{0, 1\}$ for each vertex $i \in V$, where $x_i = 1$ if vertex

$i$ is assigned to one part, and 0 otherwise. We denote by $\mathbf{x}$ the column vector $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$. The quadratic programming is then given by

$$\min_{\mathbf{x}} \mathbf{x}^T L \mathbf{x} \quad \text{such that } x_i \in \{0,1\},\ \forall i \in V, \tag{1}$$

where $L$ is the Laplacian matrix of graph $G$. For perfect balance GP, we have the following equality constraint:

$$\mathbf{x}^T \mathbb{1} = \left\lceil \frac{n}{2} \right\rceil, \tag{2}$$

where $\mathbb{1}$ is the column vector with ones. For the imbalanced case, we have the following inequality constraint $\mathbf{x}^T \mathbb{1} \leq (1+\epsilon) \left\lceil \frac{n}{2} \right\rceil$.

When $k > 2$, we introduce binary variables $x_{i,j} \in \{0,1\}$ for each vertex $i \in V$ and part $j$, where $x_{i,j} = 1$ if vertex $i$ is assigned to part $j$, and 0 otherwise. Let $\mathbf{x}_j$ denote the column vector $\mathbf{x}_j = (x_{1,j}, x_{2,j}, \cdots, x_{n,j})^T$ for $1 \leq j \leq k$. The quadratic programming formulation is then given by

$$\begin{aligned} \min_{\mathbf{x}} \ & \frac{1}{2} \sum_{j=1}^{k} \mathbf{x}_j^T L \mathbf{x}_j \\ \text{s.t.} \ & \sum_{j=1}^{k} x_{i,j} = 1, \quad \forall i \in V, \\ & x_{i,j} \in \{0,1\}, \quad \forall i \in V, \quad 1 \leq j \leq k. \end{aligned}$$

Again, for perfect balance GP, we have another set of equality constraints:

$$\mathbf{x}_j^T \mathbb{1} = \left\lceil \frac{n}{k} \right\rceil, \quad 1 \leq j \leq k.$$

For the imbalance case, we have the following inequality constraints:

$$(1-\epsilon) \left\lceil \frac{n}{k} \right\rceil \leq \mathbf{x}_j^T \mathbb{1} \leq (1+\epsilon) \left\lceil \frac{n}{k} \right\rceil, \quad 1 \leq j \leq k.$$

*QUBO Formulation.* To convert the problem into QUBO model, we will need to remove the constraints and add them as penalty terms to the objective function [14]. For example, in the quadratic programming (1) with the equality constraint (2), we obtain the QUBO model as follows:

$$\begin{aligned} \min_{\mathbf{x}} \ & \mathbf{x}^T L \mathbf{x} + P \left( \mathbf{x}^T \mathbb{1} - \left\lceil \frac{n}{2} \right\rceil \right)^2 \\ \text{s.t.} \ & x_i \in \{0,1\}, \quad \forall i \in V, \end{aligned}$$

where $P > 0$ is a postive parameter to penalize the violation of constraint (2). For inequality constraints, we will introduce additional slack variables to first convert the inequality to equality constraints, and then add them as penalty terms to the objective function.

## 3 Computational Experiments

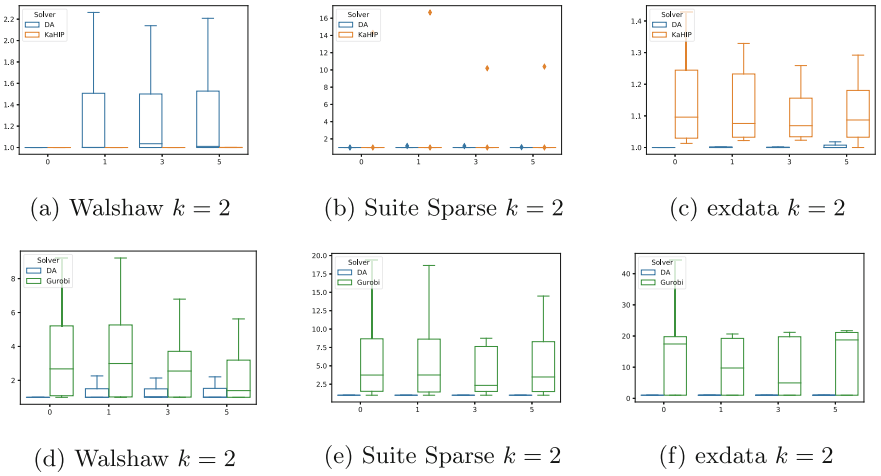
The goal of the experiments was to identify the class of instances that is more suitable to be solved using the QUBO framework and the current hardware. We compare the performance of DA with exact solver Gurobi [5], and the state-of-the-art multilevel graph partitioning solver KaHIP [17]. We set the time limit for DA and Gurobi to be 15 min. For KaHIP, we use KaFFPaE, a combination of distributed evolutionary algorithm and multilevel algorithm for GP. KaFFPaE computes partitions of very high quality when the imbalance factor $\epsilon > 0$, but does not perform very well for the perfectly balanced case when $\epsilon = 0$. Therefore we also enable a recommended by the developers KaBaPE ran with 24 parallel processes, and the time limit of 30 min.

To evaluate the quality of the solution, we compare the approximation ratio, which is computed using the GP cut found by each solver divided by the best-known value. For some graphs, we have the best-known provided from the benchmark [21], otherwise we use the best results found by the three solvers as the best known. Since this is a minimization problem, the minimum possible value of the approximation ratio is 1, the smaller the better. For each graph and each solver used, we also provide the objective function value, i.e., the number of cut edges. Due to space limitation, we present only the summary of the results. Detailed results are available in [12].

*Main Conclusion:* We have focused on demonstrating practical advantage of software and hardware approaches for GP. We found that dense graphs exhibit limitations of the existing algorithms which can be improved by the hardware accelerators.

*Graph Partitioning on Sparse Graphs.* We first test the three solvers on instances from the Walshaw graph partitioning archive [21]. We present the summary of the results with box plots in Fig. 1 (a), (d). We observe that in Fig. 1 (d), where we compare DA and Gurobi, DA can find the best-known partition for most instances, and perform better compared to Gurobi. However, for several sparse graphs, i.e., $d_{avg} < 3$, for example, `uk`, `add32` and `4elt`, DA can not find the best-known solutions. For these sparse graphs, multilevel graph partitioning solvers such as KaHIP can usually perform an effective coarsening and uncoarsening procedure based on local structures of the graph and therefore find good solutions quickly. As shown in Fig. 1 (a), KaHIP performs better than DA. Based on the numerical results, we conclude that for the sparse graphs, generic and hardware QUBO solvers do not lead to many practical advantages. However, graphs with more complex structures, that bring practical challenges to the current solvers might benefit from using the QUBO and hardware accelerators.

*Graph Partitioning on Dense Graphs.* To validate our conjecture, in the next set of experiments, we examine dense graphs from the SuiteSparse Matrix Collection [3] The experimental results are presented in Fig. 1 (b), (e). We observe that for these dense graphs, in general, DA is able to find solutions that are usually at




(a) Walshaw $k = 2$ (b) Suite Sparse $k = 2$ (c) exdata $k = 2$

(d) Walshaw $k = 2$ (e) Suite Sparse $k = 2$ (f) exdata $k = 2$

**Fig. 1.** Comparison of DA with KaHIP (dedicated GP solver), and Gurobi (general-purpose solver) for sparse and dense graphs respectively. The y-axis represents the approximation ratio (solution to best-solution ratio), the minimum possible value of the approximation ratio is 1, the smaller the better. The x-axis represents the imbalance factor as percentage

least as good as those produced by KaHIP and Gurobi. In particular, we find that for one instance, `exdata_1`, KaHIP fails significantly. We therefore use a graph generator MUSKETEER [6] to generate similar instances[1]. The parameters used to generate the instances can be found in the appendix of the full version. In short, MUSKETEER applies perturbation to the original graph with a multilevel approach, the local editing preserves many network properties including different centralities measures, modularity, and clustering. The experiment results are presented in Fig. 1 (c), (f). We find that in most instances, DA outperforms KaHIP and Gurobi, demonstrating that in this class of problems, specialized hardware such as DA is having an advantage.

Currently, to tackle GP on dense graphs, the main practical solution is to first sparsify the graphs (hoping that the sparsified graph still preserves the structure of the original dense graph), solve GP on the sparsified graph, and finally project the obtained solution back to the original graph. We have applied the Forest Fire sparsification [9] available in Networkit [19]. This sparsification is based on random walks. The vertices are burned starting from a random vertex, and fire may spread to the neighbors of a burning vertex. The intuition is that the edges that are visited more often during the random walk are more important in the graph. In our experiments, we eliminate 30% of the edges. Then we solve GP using KaHIP (KaffpaE version) and project the obtained solution back to the original dense graph. Results and details of the experiments can

[1] The `exdata` graph files are available here: https://github.com/JoeyXLiu/dense-graph-exdata.

be found in the full version of the paper. We find that for dense graphs with complex structures, KaHIP does not outperform DA, and graph sparsification does not help to achieve this goal. In this case, we advocate the use of the QUBO framework and specialized hardware.

## References


1. Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D.: 10th DIMACS implementation challenge-graph partitioning and graph clustering (2011). https://www.cc.gatech.edu/dimacs10/
2. Buluç, A., Meyerhenke, H., Safro, I., Sanders, P., Schulz, C.: Recent advances in graph partitioning. In: Kliemann, L., Sanders, P. (eds.) Algorithm Engineering. LNCS, vol. 9220, pp. 117–158. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49487-6_4
3. Davis, T.A., Hu, Y.: The university of Florida sparse matrix collection. ACM Trans. Math. Softw. (TOMS) **38**(1), 1–25 (2011)
4. Fujitsu: Fujitsu Digital Annealer (2022). https://www.fujitsu.com/global/services/business-services/digital-annealer/
5. Gurobi Optimization, I.: Gurobi optimizer reference manual (2018). https://www.gurobi.com/
6. Gutfraind, A., Safro, I., Meyers, L.A.: Multiscale network generation. In: 2015 18th International Conference on Information Fusion, pp. 158–165. IEEE (2015)
7. Hager, W.W., Hungerford, J.T., Safro, I.: A multilevel bilinear programming algorithm for the vertex separator problem. Comput. Optim. Appl. **69**(1), 189–223 (2017). https://doi.org/10.1007/s10589-017-9945-2
8. Hager, W.W., Krylyuk, Y.: Graph partitioning and continuous quadratic programming. SIAM J. Discret. Math. **12**(4), 500–523 (1999)
9. Hamann, M., Lindner, G., Meyerhenke, H., Staudt, C.L., Wagner, D.: Structure-preserving sparsification methods for social networks. Soc. Netw. Anal. Min. **6**(1), 1–22 (2016). https://doi.org/10.1007/s13278-016-0332-2
10. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1), 359–392 (1999)
11. Karypis, G., Kumar, V.: Multilevel algorithms for multi-constraint graph partitioning. In: SC 1998: Proceedings of the 1998 ACM/IEEE Conference on Supercomputing, p. 28. IEEE (1998)
12. Liu, X., Ushijima-Mwesigwa, H., Ghosh, I., Safro, I.: Partitioning dense graphs with hardware accelerators. arXiv preprint arXiv:2202.09420 (2022)
13. Liu, X., Ushijima-Mwesigwa, H., Mandal, A., Upadhyay, S., Safro, I., Roy, A.: Leveraging special-purpose hardware for local search heuristics. Computational Optimization and Applications (2022, to appear)
14. Lucas, A.: Ising formulations of many np problems. Front. Phys. **2**, 5 (2014)
15. Safro, I., Ron, D., Brandt, A.: Graph minimum linear arrangement by multilevel weighted edge contractions. J. Algorithms **60**(1), 24–41 (2006)
16. Safro, I., Sanders, P., Schulz, C.: Advanced coarsening schemes for graph partitioning. ACM J. Exp. Algorithm. (JEA) **19**, 2 (2015)
17. Sanders, P., Schulz, C.: Think locally, act globally: highly balanced graph partitioning. In: Bonifaci, V., Demetrescu, C., Marchetti-Spaccamela, A. (eds.) SEA 2013. LNCS, vol. 7933, pp. 164–175. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38527-8_16

18. Shaydulin, R., Chen, J., Safro, I.: Relaxation-based coarsening for multilevel hypergraph partitioning. SIAM Multiscale Model. Simul. **17**, 482–506 (2019)
19. Staudt, C.L., Sazonovs, A., Meyerhenke, H.: NetworKit: a tool suite for large-scale complex network analysis. Netw. Sci. **4**(4), 508–530 (2016)
20. Ushijima-Mwesigwa, H., Negre, C.F., Mniszewski, S.M.: Graph partitioning using quantum annealing on the D-Wave system. In: Proceedings of the Second International Workshop on Post Moores Era Supercomputing, pp. 22–29 (2017)
21. Walshaw, C.: The graph partitioning archive (2009). https://chriswalshaw.co.uk/partition/

# A Taxonomy Guided Method to Identify Metaheuristic Components

Thimershen Achary(✉) and Anban W. Pillay

University of KwaZulu-Natal, Durban, South Africa
thimershenzn@gmail.com

**Abstract.** A component-based view of metaheuristics has recently been promoted to deal with several problems in the field of metaheuristic research. These problems include inconsistent metaphor usage, non-standard terminology and a proliferation of metaheuristics that are often insignificant variations on a theme. These problems make the identification of novel metaheuristics, performance-based comparisons, and selection of metaheuristics difficult. The central problem for the component-based view is the identification of components of a metaheuristic. This paper proposes the use of taxonomies to guide the identification of metaheuristic components. We developed a general and rigorous method, TAXONOG-IMC, that takes as input an appropriate taxonomy and guides the user to identify components. The method is described in detail, an example application of the method is given, and an analysis of its usefulness is provided. The analysis shows that the method is effective and provides insights that are not possible without the proper identification of the components.

**Keywords:** Metaheuristic · General metaheuristic · Taxonomy

## 1 Introduction

The metaheuristic research field has been criticized for inconsistent metaphor usage, non-standard terminology [1, 2], and use of poor experimental setups, validation, and comparisons [1–3]. These factors have contributed to challenges in the field such as a proliferation of novel metaheuristics and 'novel' approaches being very similar to existing approaches [1, 2, 4]. Several researchers have thus proposed that a component-based view of metaheuristics that explicitly lists metaheuristic components, will assist in identifying novel components [1, 5], promote component-based performance comparison and analyses, and facilitate component-wise selection of metaheuristics for comparative studies [1, 2, 6, 7].

A component-based view is especially important for general metaheuristics, which has enjoyed increasing popularity in recent literature. General metaheuristics, also known as general metaheuristic frameworks [8], unified metaheuristic frameworks [9], and generalized metaheuristic models [10] are used for tasks such as metaheuristic generation [10], performance analysis [11, 12], metaheuristic-similarity analysis [13], and classification of metaheuristics [7]. General metaheuristics are an abstraction of a set of


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 484–496, 2022.
https://doi.org/10.1007/978-3-031-08757-8_41

metaheuristics, i.e., they are generalizations of the components, structure, and information utilized by a set of metaheuristics [6, 12]. They thus also take a component-based view. General metaheuristics make use of a set of component-types, also referred to as general metaheuristics structures [12], component-categories [6], main ingredients [14], or key components [15].

However, general metaheuristics still suffer the challenges outlined above viz. inconsistent metaphor usage and non-standard terminology. They also suffer from similar problems if components are not properly identified. Thus, the identification of components takes on special importance.

This work promotes the systematic use of taxonomies to guide the identification of components. Our proposed method uses formal taxonomy theory, which appears to be absent in several recent metaheuristic studies that involve the creation or incorporation of taxonomies such as [7, 16–19]. Taxonomies, ideally, are built using a rigorous taxonomy building-method e.g. [20, 21]. Taxonomies are intrinsic prerequisites to understanding a given domain, differentiating between objects, and facilitating discussion on the state and direction of research in a domain [22]. Taxonomies may thus help solve the issues affecting metaheuristic research, such as non-standard terminology and nomenclature.

This work proposes the use of taxonomies to guide the identification of metaheuristic components. We developed a general and rigorous method, TAXONOG-IMC, that takes as input an appropriate taxonomy and guides the user to identify components. TAXONOG-IMC promotes the use of taxonomies to guide component identification for any metaheuristic subset, and provides guidance for the proper use of taxonomies to perform component identification.

This paper presents the method, provides an example of its application, and gives an analysis of its usefulness. The rest of the paper is structured as follows: Sect. 2 provides a literature review, Sect. 3 comprehensively describes TAXONOG-IMC, Sect. 4 demonstrates the use of the method by applying it to two taxonomies to showcase its effectiveness, Sect. 5 provides an analysis of the method by showing its effectiveness in analysing nature-inspired, population-based metaheuristics. Section 6 concludes the study.

## 2 Literature Review

The need for a component-based view is best appreciated in general metaheuristics. However, many general metaheuristics lack a rigorous method for identifying components. Many studies proposing a general metaheuristic provide guidance through examples of their usage. Several broad-scoped general metaheuristics follow this trend, such as general metaheuristics for population-based metaheuristics [9] and metaheuristics in general [10, 11, 13]. The general metaheuristics proposed by [6, 9, 10, 13] use mathematical formulations for their component-types. Since these mathematical formulations are sometimes in-part derived from text, the researcher can choose how to formulate a component based on their judgement and interpretation. However, this process can be negatively impacted by inconsistent metaphor usage and non-standard terminology. Components that are essentially the same can be regarded as different. Using examples for guidance may not account for all contingencies.

A general metaheuristic built on the assumption that differentiating the components in detail and using relatable terminology may help resolve challenges in component identification, is presented in [12]. However, most of their component-types of the general metaheuristic were a renaming of the components in [13] and may consequently face the same challenges. Some component-categories in literature were listed, but using them for the general metaheuristic may be difficult; if they consist of combinations of components, then they themselves need to be decomposed, which requires expert knowledge.

Several studies used taxonomies and/or classification-schemes to support the design of general metaheuristics. The advantage of using a taxonomy for this purpose is that it declares a convention by which the components will be identified. It provides a list of possible components that a component-type encompasses. If an issue is taken with the convention, then it can be argued at the taxonomy level. There are studies, such as [23, 24], that propose general metaheuristics whose components make use of a presented taxonomy, and there are studies that make use of existing taxonomies for a proposed general metaheuristic, such as [7, 15]. The studies that proposed both a general metaheuristic and a taxonomy are likely to work well, as the taxonomy is built for the general metaheuristic; however, taxonomies are not necessarily built with general metaheuristics in mind.

Works that use existing taxonomies lack guidance on how to use taxonomies effectively. Existing taxonomies and viewpoints were used in [15] to create a new taxonomy to guide the usage of a proposed general metaheuristic. The taxonomy presented used examples at the lowest level of its hierarchy to illustrate its usage. However, examples do not account for every contingency. The essence of the multi-level classification method proposed in [7] is meritorious; however, a misuse of the behaviour taxonomy presented in [5], led to a classification that is questionable in terms of the taxonomy used, i.e., tabu search is depicted as possessing the differential vector movement behaviour. Some studies consider tabu search as population-based but viewing tabu search as being single-solution based has a stronger consensus [25] and appears to be followed by [5], i.e., the behaviour taxonomy presented by [5] is not applicable to tabu search in its canonical sense.

The study in [14] presents a taxonomy for evolutionary algorithms based on their main components. The same study uses the taxonomy to facilitate the expression of evolutionary algorithms in terms of their main components, and the distinguishing between various evolutionary algorithm classes. This study is notable for its use of a vector representation for its components. Our work uses a similar representation.

## 3 Taxonomy Guided Identification of Metaheuristic Components: TAXONOG-IMC

This section proposes TAXONOG-IMC (see Fig. 1), a general, rigorous method that guides the identification of metaheuristic components using taxonomies.

We use the definition of a taxonomy provided in [20] that lends itself to a flat representation of the metaheuristics or metaheuristic component-types, which facilitates

tabular analysis. A taxonomy T is formally defined in [20] as:

$$T = \sum D_i, (i = 1, \ldots, n) | D_i = \sum C_{ij}, (j = 1, \ldots, k_i); k_i \geq 2 \left(\left( \right. \right. \quad (1)$$

where $T$ is an arbitrary taxonomy, $D_i$ is an arbitrary dimension of $T$, $k_i \geq 2$ is the number of possible characteristics for dimension $D_i$, $C_{ij}$ an arbitrary characteristic for dimension $D_i$. Characteristics for every dimension are mutually exclusive and collectively exhaustive, i.e., each object under consideration must have one and only one $C_{ij}$ for every $D_i$.This organization, using dimensions and characteristics, is likely to be relevant in all cases since they are fundamental to understanding the properties of objects in a domain; hence the definition (1) is used.

Some important terms concerning taxonomies are explained below:

1. Dimensions: A dimension represents some attribute of an object and can be thought of as a variable that has a set of possible values.
2. Characteristics: The characteristics of a given dimension are the possible values that can be assigned to a particular dimension.
3. Taxonomy dimension: A taxonomy dimension refers to a dimension that is part of the taxonomy under consideration. The method has steps where dimensions are proposed – these are not part of the taxonomy but are under consideration to be included. We refer to these as candidate dimensions that may then become part of the taxonomy.
4. Specialized dimension: A specialized dimension is a characteristic of a taxonomy that is promoted to dimension status; specialized dimensions are candidate dimensions.
5. Generalized dimension: A generalized dimension is created by partitioning characteristics of a taxonomy dimension or partitioning the combination of characteristics from multiple taxonomy dimensions. A generalized dimension is a candidate dimension.

To illustrate each term, consider the following dimensions of some metaheuristic: initializer, search operator, and selection. Characteristics of search operator may be, e.g., genetic crossover, swarm dynamic, differential mutation. A taxonomy for evolutionary algorithms in [14] has population, structured population, information sources etc., as its dimensions. Then population would be a taxonomy dimension. Using the behaviour taxonomy presented in [5], solution creation can be thought of as a generalized dimension of the combination and stigmergy dimensions. If we use solution-creation as a taxonomy dimension, then combination would be a specialized dimension.

### 3.1 Comprehensive Description of Method Process

A good start for step 1 (select or create a taxonomy), is to conduct a literature search for relevant taxonomies using keywords, key-phrases, publication titles, etc. However, if no appropriate taxonomy is found, then an appropriate taxonomy building method should be used to create a taxonomy.

Expressing a taxonomy using definition (1), ensures the taxonomy is in a standard format for subsequent steps. The dimensions, and the dimensions' characteristics must be clearly stated to avoid ambiguity.

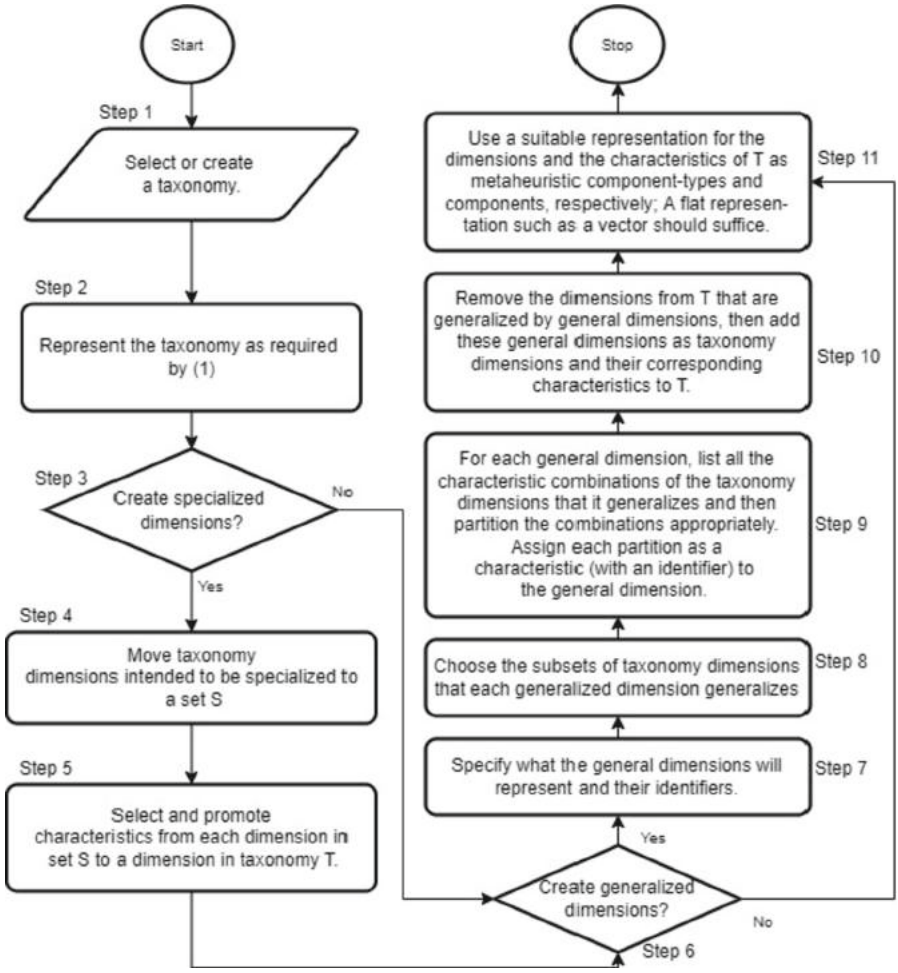



**Fig. 1.** Flowchart depicting the processes of TAXONOG-IMC

Steps 3 to 5 guides the creation of specialized dimensions. Using specialized dimensions will allow for focusing on specific components. The role of set S, introduced in step 4, is to store a collection of dimensions that are to be replaced by one of their characteristics in taxonomy T. In the metaheuristic context, a dimension may be replaced by more than one of its characteristics; this decision accommodates for hybrid-metaheuristics that have more than one characteristic for a dimension. When characteristics become dimensions, they will each need a set of possible characteristics of their own that will be derived from literature or the expertise of the researcher.

The addition of specialized dimensions to the Taxonomy may result in an overwhelmingly large number of taxonomy dimensions. Generalizing an appropriate number of taxonomy dimensions may help with this challenge.

Creating generalized dimensions is guided by steps 7 to 10. It is essential to name the general dimensions clearly and their characteristics to ensure no ambiguities nor confusion arises as to which dimension or characteristic a trait falls under. It is important to note that each subset of taxonomy dimensions, chosen in step 8, must be disjoint. Note that not every taxonomy dimension needs to be integrated into a general dimension.

As an example of when and how general dimensions can be used, consider a chosen set of metaheuristics that have a large diversity on certain taxonomy dimensions. They may be grouped by their characteristic combinations on these dimensions. A generalized dimension could then have two possible values, 1 representing a metaheuristic having a required combination of characteristics for those dimensions, and 0 representing a metaheuristic not having such a combination of characteristics for those dimensions.

## 4 Application of Method

To demonstrate the method, we use it to generate binary component vectors to represent nature-inspired, population-based metaheuristics in terms of their inspiration and behaviour components. We use the behaviour and natural-inspiration taxonomies provided in [5]. In this study, we consider the metaphor/inspiration of a metaheuristic to be a component, but more specifically, a non-functional component. The nature-inspiration taxonomy was created to ascertain the natural-inspiration category of a metaheuristic without ambiguity. The behavioural taxonomy is based on the metaheuristic behaviour, i.e., focusing on the means by which new candidate solutions are obtained, and disregarding its natural inspiration. See Sect. 4.3 for descriptions of all dimensions used by the behaviour and natural-inspiration taxonomies.

### 4.1 Behavior Taxonomy

- Step 1: We use the behavior taxonomy from [5].
- Step 2: We express the taxonomy using the definition given in (1) as follows. A characteristic of 1 means that it is present and 0 means it is not.

  – $b_1$ - Combination (characteristics are {0, 1})
  – $b_2$ - Stigmergy (characteristics are {0; 1})
  – $b_3$ - All population Differential Vector Movement (DVM) (characteristics are {0; 1})
  – $b_4$ - Groups-based (DVM) (characteristics are {0; 1})
  – $b_5$ - Representative based (DVM) (characteristics are {0; 1})

- Step 3: We create specialized dimensions.
- Step 4: S = {Groups-based (DVM)}, The step at this phase dictates that we only select one characteristic to promote to dimension status, but with regards to metaheuristics, which can be hybridized and still be metaheuristics, an exception can be made such that numerous characteristics can be promoted during specialization (this depends on the characteristics, if the characteristics are single-solution and population-based then these can't both be used as component-types for a metaheuristic at the same time, since

there is a possibility that both can be set to 1, which does not make intuitive sense). Therefore, we promote both Sub-population (DVM) and Neighborhood (DVM) to dimensions with their characteristics being binary {0; 1}. $b_4$ is set to Sub-population (DVM) and $b_5$ is set to Neighborhood (DVM), $b_6$ is set to Representative based (DVM).

- Step 5: Groups-based (DVM) is not referenced by any dimension and can thus be discarded. $T = \{b_1; b_2; b_3; b_4; b_5; b_6 \mid b_i = \{0; 1\}; (i = 1, 2, 3, 4, 5, 6)\}$
- Step 6: We do not create generalized dimension.
- Step 11: The vector representation derived from the behavoiur taxonomy is:

$$\left)b_1\ b_2\ b_3\ b_4\ b_5\ b_6\right\} \tag{2}$$

### 4.2 Natural-Inspiration Taxonomy

- Step 1: We use the natural-inspiration taxonomy from [5].
- Step 2: We express the taxonomy using the definition given in (1) as follows:

  - $n_1$ - Breeding-based evolution (characteristics are {0; 1})
  - $n_2$ - Aquatic animals (characteristics are {0; 1})
  - $n_3$ - Terrestrial animals (characteristics are {0; 1})
  - $n_4$ - Flying animals (characteristics are {0; 1})
  - $n_5$ - Microorganisms (characteristics are {0; 1})
  - $n_6$ - Others (characteristics are {0; 1})
  - $n_7$ - Physics-based (characteristics are {0; 1})
  - $n_8$ - Chemistry-based (characteristics are {0; 1})
  - $n_9$ - Social human behaviour algorithms (characteristics are {0; 1})
  - $n_{10}$ - Plants based (characteristics are {0; 1})
  - $n_{11}$ - Miscellaneous (characteristics are {0; 1})

- Step 3: We do not create specialized dimensions.
- Step 6: We create general dimensions.
- Step 7: We create two general dimensions that will be identified as Swarm-intelligence and Physics and Chemistry Based. (This is already done in the taxonomy, but we are redoing it in this process for demonstration).
- Step 8: Aquatic animals, Terrestrial animals, Flying animals, Microorganisms, Others are allocated to the Swarm-intelligence general dimension. Physics-based, Chemistry-based are allocated to the Physics and Chemistry Based general dimension.
- Step 9: The characteristics of Swarm-intelligence are {0; 1}. 1 indicating that either Aquatic animals, Terrestrial animals, Flying animals, Microorganisms, or Others are present, 0 indicating that Aquatic animals, Terrestrial animals, Flying animals, Microorganisms, and Others are absent. The characteristics of Physics and Chemistry Based are {0; 1}. 1 indicating that either Physics-based or Chemistry-based is 1, 0 indicating that Physics-based and Chemistry-based are absent.
- Step 10: Since $n_2$ to $n_8$ are removed, $n_2$ will be the dimension for Swarm-intelligence, $n_3$ will be the dimension for Physics and Chemistry Based, $n_4$ will be the dimension for Social human behavior algorithms, $n_5$ will be the dimension for Plants based, $n_6$

will be the dimension for Miscellaneous; $n_7$ to $n_{11}$ do not refer to any dimensions so they can be discarded. $T = \{n_1; n_2; n_3; n_4; n_5; n_6 \mid n_i = \{0; 1\}, (i = 1, 2, 3, 4, 5, 6)\}$

- Step 11: The vector representation definition derived from the selected taxonomy is:

$$\left\{n_1\ n_2\ n_3\ n_4\ n_5\ n_6\right\} \tag{3}$$

### 4.3 Dimension Descriptions

In this sub-section, the nodes of each hierarchal taxonomy presented in [5] are unambiguously defined as dimensions using the descriptions of each node provided in the same study; from these definitions, we can define the dimensions in the initial steps and proceed to modify them in subsequent steps by adding and/or dropping these dimensions due to using generalized or specialized dimensions.

**Behaviour Dimensions**

- Differential vector movement: New solution is obtained by movement relative to an existing solution
- All population Differential Vector Movement (DVM): All individuals in the population are used to generate the movement of each solution.
- Representative-based (DVM): The movements of each solution are only influenced by a small group of representative solutions, e.g., the best solutions found
- Group-based (DVM): Sub-populations or subsets of the populations are considered, without representative solutions.
- Sub-population (DVM): The movements of each solution are influenced by a subset or group of solutions in the population, and no representative solutions are determined and used in the trajectory calculation at hand.
- Neighborhood (DVM): Each solution is only influenced by solutions in its local neighborhood.
- Combination: New solutions are selected and combined via some method to create new solutions.
- Stigmergy: An indirect communication and coordination strategy is used between different solutions to create new solutions.
- Creation: Exploration of search domain by generating new solution, differential vector movement not present.

**Natural-Inspiration Dimensions**

- Breeding-based evolution: Inspired by the principle of natural evolution and references to producing offspring, successive generations.
- Swarm Intelligence: Inspired by the collective behavior of animal societies.
- Flying animals: Agent movements inspired by flying movements.
- Terrestrial animals: Agent movements inspired by foraging or movements of terrestrial animals.

– Aquatic animals: Agent movements inspired by animals living in aquatic ecosystems.
– Microorganisms: Agent movements inspired by food search by bacteria or how viruses spread infection.
– Others: Very low popularity inspiration sources from the collective behavior of animals.
– Physics and Chemistry Based: Imitate the behavior of physical/chemical phenomena (field of physics and chemistry).
– Social Human Behavior Algorithms: Inspired by human social concepts.
– Plants Based: Inspired by plants, where there is no communication between agents.
– Miscellaneous: Not inspired by any identified category.

## 5 Analysis and Discussion

We now demonstrate the use of the method. Information showing the application frequency of different nature-inspired metaheuristics to feature selection in disease diagnosis is depicted in Table 10 taken from the study in [26]. It is stated that data for the table was obtained by executing various search queries on google scholar. RA is not population-based, and thus is ignored since it is out of scope for the vector derived in the current paper. In this section, the amount of information extracted from Table 10 in [26] is extended using the derived vector. The aim is to reconfigure the table to attribute the frequencies to the component-types of the derived vector. This task is accomplished via the following steps:

1. List all metaheuristic abbreviations and ascertain their full name.
2. Represent each of the nature-inspired, population-based metaheuristics using the vector formats derived, i.e., (2) and (3), as shown in Table 1. If the metaheuristics were not present in the tables, the descriptions of the dimensions of the taxonomies presented in [5] would have to be used to derive their vector representation.
3. Let $B$ be a matrix representing the data of Table 1, i.e., $B\{p\}\{q\}$ will indicate whether the component-type at column index q is present in the metaheuristic at row index p. Let $D$ be a matrix where each intersection of row $i$ and column $j$ is the frequency of application of metaheuristic at row index $i$ to the disease at column index $j$ (D holds the data of Table 10 in [26]). Let $F$ be the matrix that holds the component-type to disease diagnosis application frequencies (Table 2), i.e., where $j$ is index number of the disease in the columns of Table 10 presented in [26] and q is the index number of the component-type in the vector:

$$F\{j\}\{q\} = \sum_{x=0}^{N} B[x]\{q\} \times D[x]\{j\} \tag{4}$$

4. Matrix $F$ contains the data of Table 2 that depicts the table of frequency of appl cation of a component-type to disease diagnosis. From this table, further analysis can be done.

It can be observed from Table 2 that $b_6$ (Representative-based (DVM)) is the dominant behaviour and $n_2$ (Swarm intelligence) is the dominant natural-inspiration. It is

**Table 1.** Representation of nature-inspired, population-based metaheuristics in terms of derived vector formats

KEY: Harmony search (HS), Artificial bee colony (ABC), Glow-worm swarm optimization (GSO), Ant colony optimization (ACO), Firefly algorithm (FA), Monkey algorithm (MA), Cuckoo search (CS), Bat algorithm (BA), Dolphin echolocation (DE), Flower pollination algorithm (FPA), Grey wolf optimizer (GWO), Dragonfly algorithm (DA), Krill herd algorithm (KHA), Elephant search algorithm (ESA), Ant lion optimizer (ALO), Moth-flame optimization (MFO), Multi-verse optimizer (MVO), Runner-root algorithm (RRA), Laying chicken algorithm (LCA), Killer whale algorithm (KWA), Butterfly optimization algorithm (BOA).

| PMBH | b1 | b2 | b3 | b4 | b5 | b6 | n1 | n2 | n3 | n4 | n5 | n6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| HS | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| ABC | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| GSO | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| ACO | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| FA | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| MA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| CS | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| BA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| DE | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| FPA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| GWO | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| DA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| KHA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| ESA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| ALO | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| MFO | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| MVO | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| RRA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| LCA | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| KWA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| BOA | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

interesting to note that in [26], it is stated that ACO is dominant in the use of diagnosis of different human disorders. However, the behaviour associated with ACO is Stigmergy ($b_2$) is not the dominant behaviour; instead, representative-based differential movement ($b_6$) is the dominant behaviour for this application domain.

Literature such as [1] has shown that the names and metaphors of metaheuristics sometimes mask the substantial similarities between the metaheuristics and their differences are so minute that they can be considered marginal variants. ACO is popular, but the problem could lie with many metaheuristics, which have behavioural component-type $b_6$, being diverse in names as this trend is either diluting the core algorithm's popularity or is misguiding users to believe that different metaheuristic names entail that they have nearly orthogonal behaviours.

From Table 2, it can be ascertained that scope for future research lies in applying metaheuristics with behavioural component-types: sub-population (DVM), neighbourhood

**Table 2.** Frequencies of component-type usage, in literature, in various disease diagnosis applications

| Disease diagnosis | $b_1$ | $b_2$ | $b_3$ | $b_4$ | $b_5$ | $b_6$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_6$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Breast cancer | 413 | 619 | 216 | 0 | 0 | 893 | 0 | 1859 | 236 | 0 | 46 | 0 |
| Prostate cancer | 35 | 73 | 9 | 0 | 0 | 68 | 0 | 161 | 21 | 0 | 3 | 0 |
| Lung cancer | 105 | 157 | 41 | 0 | 0 | 154 | 0 | 400 | 51 | 0 | 6 | 0 |
| Oral cancer | 4 | 3 | 2 | 0 | 0 | 6 | 0 | 12 | 3 | 0 | 0 | 0 |
| Neck cancer | 4 | 4 | 0 | 0 | 0 | 9 | 0 | 13 | 3 | 0 | 1 | 0 |
| Skin cancer | 19 | 4 | 15 | 0 | 0 | 53 | 0 | 81 | 8 | 0 | 2 | 0 |
| HIV | 40 | 114 | 24 | 0 | 0 | 80 | 0 | 237 | 18 | 0 | 3 | 0 |
| Stroke | 116 | 120 | 36 | 0 | 0 | 129 | 0 | 330 | 60 | 0 | 11 | 0 |
| Schizophrenia | 8 | 44 | 9 | 0 | 0 | 16 | 0 | 72 | 4 | 0 | 1 | 0 |
| Parkinson | 91 | 144 | 52 | 0 | 0 | 233 | 0 | 434 | 62 | 0 | 24 | 0 |
| Heart disease | 129 | 34 | 58 | 0 | 0 | 234 | 0 | 390 | 55 | 0 | 10 | 0 |
| Anxiety | 17 | 65 | 9 | 0 | 0 | 50 | 0 | 135 | 5 | 0 | 1 | 0 |
| Insomnia | 1 | 6 | 0 | 0 | 0 | 2 | 0 | 9 | 0 | 0 | 0 | 0 |
| Sum | 982 | 1387 | 471 | 0 | 0 | 1927 | 0 | 4133 | 526 | 0 | 108 | 0 |

(DVM), breeding-based evolution, social-human behaviour algorithms, and miscellaneous to disease diagnosis. Even though the three latter component-types are natural-inspirations, and literature has motivated that this category of component-types has little contribution to performance. Applying them increases their presence in a population, from which data can be sampled, i.e., a diverse population is good.

The taxonomies in [5] organized the metaheuristics using their canonical versions. This study relies on the assumption that if two or more metaheuristic-algorithms are associated with the same metaheuristic, then they should possess the behaviour of that metaheuristic. The proposed method can be used to select components for metaheuristic frameworks, classification schemes, representations, and comparative analysis.

## 6 Conclusion

This study proposes TAXONOG-IMC, a structured method that provides guidance for metaheuristic component identification using taxonomies. An example application is provided to showcase how TAXONOG-IMC can aid in metaheuristic analysis.

Identification of metaheuristic components is an important task for the effective use of general metaheuristics, and the metaheuristic component-based view by and large. General metaheuristic publications use strategies such as providing examples, using finer-grain component-types, relying on existing taxonomies or creating new ones to assist in component identification. However, examples don't account for all contingencies that a researcher may encounter, and finer-grain components can also be affected

by non-standard terminology and inconsistent metaphor usage. There are general metaheuristic publications that use taxonomies to assist in component identification; some propose their own taxonomy, and others use an existing taxonomy. The ones that propose their own taxonomy are likely to be compatible with the general metaheuristic since they are created for that purpose; however, some of the publications that use existing taxonomies made questionable decisions during the demonstration of general metaheuristic use – indicating a lack of proper use of taxonomy.

Future research lies in using taxonomies for component-identification for many other metaheuristic subsets, metaheuristics analysis, and use in general metaheuristics.

## References


1. Sörensen, K.: Metaheuristics-the metaphor exposed. Int. Trans. Oper. Res. **22**, 3–18 (2015). https://doi.org/10.1111/itor.12001
2. Aranha, C., et al.: Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intell. (2021). https://doi.org/10.1007/s11721-021-00202-9
3. García-Martínez, C., Gutiérrez, P.D., Molina, D., Lozano, M., Herrera, F.: Since CEC 2005 competition on real-parameter optimisation: a decade of research, progress and comparative analysis's weakness. Soft. Comput. **21**(19), 5573–5583 (2017). https://doi.org/10.1007/s00500-016-2471-9
4. Tzanetos, A., Dounias, G.: Nature inspired optimization algorithms or simply variations of metaheuristics? Artif. Intell. Rev. **54**(3), 1841–1862 (2020). https://doi.org/10.1007/s10462-020-09893-8
5. Molina, D., Poyatos, J., Ser, J.D., García, S., Hussain, A., Herrera, F.: Comprehensive taxonomies of nature- and bio-inspired optimization: inspiration versus algorithmic behavior, critical analysis recommendations. Cogn. Comput. **12**(5), 897–939 (2020). https://doi.org/10.1007/s12559-020-09730-8
6. Peres, F., Castelli, M.: Combinatorial optimization problems and metaheuristics: review, challenges, design, and development. Appl. Sci. **11**, 6449 (2021). https://doi.org/10.3390/app11146449
7. Stegherr, H., Heider, M., Hähner, J.: Classifying Metaheuristics: towards a unified multi-level classification system. Natural Comput. (2020). https://doi.org/10.1007/s11047-020-09824-0.
8. Birattari, M., Paquete, L., Stützle, T.: Classification of metaheuristics and design of experiments for the analysis of components (2003)
9. Liu, B., Wang, L., Liu, Y., Wang, S.: A unified framework for population-based metaheuristics. Ann. Oper. Res. **186**, 231–262 (2011). https://doi.org/10.1007/s10479-011-0894-3
10. Cruz-Duarte, J.M., Ortiz-Bayliss, J.C., Amaya, I., Shi, Y., Terashima-Marín, H., Pillay, N.: Towards a generalised metaheuristic model for continuous optimisation problems. Mathematics **8**, 2046 (2020). https://doi.org/10.3390/math8112046
11. De Araujo Pessoa, L.F., Wagner, C., Hellingrath, B., Buarque De Lima Neto, F.: Component analysis based approach to support the design of meta-heuristics for MLCLSP providing guidelines. In: 2015 IEEE Symposium Series on Computational Intelligence, pp. 1029–1038. IEEE, Cape Town (2015). https://doi.org/10.1109/SSCI.2015.149
12. Stegherr, H., Heider, M., Luley, L., Hähner, J.: Design of large-scale metaheuristic component studies. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 1217–1226. ACM, Lille France (2021). https://doi.org/10.1145/3449726.3463168

13. de Armas, J., Lalla-Ruiz, E., Tilahun, S.L., Voß, S.: Similarity in metaheuristics: a gentle step towards a comparison methodology. Natural Comput. (2021). https://doi.org/10.1007/s11047-020-09837-9
14. Calégari, P., Coray, G., Hertz, A., Kobler, D., Kuonen, P.: A taxonomy of evolutionary algorithms in combinatorial optimization. J. Heuristics **5**, 145–158 (1999). https://doi.org/10.1023/A:1009625526657
15. Raidl, G.R.: A unified view on hybrid metaheuristics. In: Almeida, F., et al. (eds.) HM 2006. LNCS, vol. 4030, pp. 1–12. Springer, Heidelberg (2006). https://doi.org/10.1007/11890584_1
16. Kaviarasan, R., Amuthan, A.: Survey on analysis of meta-heuristic optimization methodologies for node network environment. In: 2019 International Conference on Computer Communication and Informatics (ICCCI), pp. 1–4. IEEE, Coimbatore, Tamil Nadu, India (2019). https://doi.org/10.1109/ICCCI.2019.8821838
17. Fister, I., Perc, M., Kamal, S.M., Fister, I.: A review of chaos-based firefly algorithms: perspectives and research challenges. Appl. Math. Comput. **252**, 155–165 (2015). https://doi.org/10.1016/j.amc.2014.12.006
18. Diao, R., Shen, Q.: Nature inspired feature selection meta-heuristics. Artif. Intell. Rev. **44**(3), 311–340 (2015). https://doi.org/10.1007/s10462-015-9428-8
19. Donyagard Vahed, N., Ghobaei-Arani, M., Souri, A.: Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review. Int J Commun Syst. **32**, e4068 (2019). https://doi.org/10.1002/dac.4068
20. Nickerson, R.C., Varshney, U., Muntermann, J.: A method for taxonomy development and its application in information systems. Eur. J. Inf. Syst. **22**, 336–359 (2013). https://doi.org/10.1057/ejis.2012.26
21. Usman, M., Britto, R., Börstler, J., Mendes, E.: Taxonomies in software engineering: a systematic mapping study and a revised taxonomy development method. Inf. Softw. Technol. **85**, 43–59 (2017). https://doi.org/10.1016/j.infsof.2017.01.006
22. Szopinski, D., Schoormann, T., Kundisch, D.: because your taxonomy is worth it: towards a framework for taxonomy evaluation. Research Papers (2019)
23. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans. Evol. Comput. **9**, 474–488 (2005). https://doi.org/10.1109/TEVC.2005.850260
24. Stork, J., Eiben, A.E., Bartz-Beielstein, T.: A new taxonomy of global optimization algorithms. Natural Comput. (2020). https://doi.org/10.1007/s11047-020-09820-4
25. Glover, F., Laguna, M.: Tabu search background. In: Tabu Search, pp. 1–24. Springer US, Boston, MA (1997). https://doi.org/10.1007/978-1-4615-6089-0_1
26. Sharma, M., Kaur, P.: A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem. Archives Comput. Methods Eng. **28**(3), 1103–1127 (2020). https://doi.org/10.1007/s11831-020-09412-6

# Camp Location Selection in Humanitarian Logistics: A Multiobjective Simulation Optimization Approach

Yani Xue[1], Miqing Li[2], Hamid Arabnejad[1], Diana Suleimenova[1], Alireza Jahani[1], Bernhard C. Geiger[3], Zidong Wang[1], Xiaohui Liu[1], and Derek Groen[1(✉)]

[1] Department of Computer Science, Brunel University London, London, UK
Derek.Groen@brunel.ac.uk
[2] School of Computer Science, University of Birmingham, Birmingham, UK
[3] Know-Center GmbH, Graz, Austria

**Abstract.** In the context of humanitarian support for forcibly displaced persons, camps play an important role in protecting people and ensuring their survival and health. A challenge in this regard is to find optimal locations for establishing a new asylum-seeker/unrecognized refugee or IDPs (internally displaced persons) camp. In this paper we formulate this problem as an instantiation of the well-known facility location problem (FLP) with three objectives to be optimized. In particular, we show that AI techniques and migration simulations can be used to provide decision support on camp placement.

**Keywords:** Facility location problem · Multiobjective optimization · Simulation · Evolutionary algorithms

## 1 Introduction

Forced displacement is a complex global phenomenon, which refers to the movement of people away from their home or origin countries due to many factors, such as conflict, violence, persecution, etc. In 2020, almost 26.4 million people had fled their countries according to the UNHCR (https://www.unhcr.org/uk/figures-at-a-glance.html). In this situation, relocating asylum-seekers/unrecognized refugees to camps becomes an urgent issue to humanitarian organizations or governments. Camps, as important infrastructures, provide protection and allocate available humanitarian resources to thousands of forcibly displaced people. As resources are commonly limited, it is critical to make optimal decisions in seeking the best location for establishing a new camp. Camp placement can be formulated as the well-known facility location problem (FLP) [6]. The FLP can be considered as a multiobjective optimization problem (MOP), which includes two or more objectives to be optimized simultaneously. The objectives of the FLP can include minimizing the total travel distance and maximizing the demand coverage, meanwhile satisfying some constraints [8].


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 497–504, 2022.
https://doi.org/10.1007/978-3-031-08757-8_42

Several MOP-FLP approaches have been proposed, including traditional goal programming, $\epsilon$-constraint approaches and, more recently, metaheuristic optimization algorithms [13] such as particle swarm optimization (PSO) and evolutionary algorithm (EA). As a population-based metaheuristic optimization approach, EA may effectively handle MOPs as it can generate a set of trade-off solutions in a single run. It has specifically been applied to tackle the FLP in disaster emergency management [14], making it natural to employ EA in the context of camp placement. The main challenge here is to have exact number of forcibly displaced persons arriving in destination countries. Due to the ongoing conflicts in origin countries, the number of asylum-seekers/unrecognized refugees or IDPs continuously changes over time.

Here we aim to assist humanitarian organizations and governments in their decision-making on camp placement, and the paper has the following contributions: (1) we present an MOP for camp placement with three objectives regarding travel distance, demand coverage, and idle camp capacity; (2) we use an agent-based simulation to capture the demand uncertainty (i.e., the number of camp arrivals), which is crucial for camp placement but has not been considered in most existing literature; (3) we present a new multiobjective simulation optimization approach for our MOP, which consists of EA and an agent-based forced migration simulation; and (4) we successfully apply the proposed approach to a case study of the South Sudan conflict, and identify a group of optimal solutions for decision-makers.

### 1.1 Related Work

The camp location selection problem is a complex task for the humanitarian organizations to deploy aid. The research areas related to this problem can be generally divided into the modelling the movements of people [11], and the FLP in humanitarian logistics [1,4,7,9]. Here we attempt to address the optimization problem of how to find the optimal locations for establishing a new camp. This problem can be formulated as an MOP. Current approaches for multiobjective FLPs can be classified into two categories. The first is concerned with the traditional single-objective optimization approach, such as the goal programming approach [1], the weighted sum approach [9] and the $\epsilon$-constraint [4]. The second is the multiobjective optimization approach searching for the whole Pareto front, from which the decision makers choose their preferred solution. For example, the classic NSGA-II and a multiobjective variant of the PSO algorithm were applied in the earthquake evacuation planning problem [7]. The reason we consider the second category is that optimization approaches in the first may require prior knowledge, such as the relative importance of the objectives in the weighted sum approach. Such knowledge may not be easy to access, and even if it is available it has been shown that the search aiming for the whole Pareto front may be more promising since it can help the search escape the local optima [3]. Another strand of research is multiobjective optimization under uncertainty. Recently, some studies have proposed a number of robust or stochastic models for FLPs

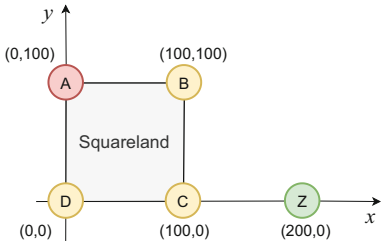



**Fig. 1.** An illustration of the route network for a basic camp placement, where 1) a source country is represented by a square region with one conflict zone (i.e., point A), three towns (i.e., points B, C, and D) and all possible links among these points, and 2) one camp (i.e., point Z) is connected to the nearest location in the source country.

under uncertainty [2]. However, there is a lack of studies on FLPs under uncertainty that take the preferences of people into account. As popular simulation approaches, different agent-based modelling frameworks have been developed to model the movements of displaced persons (or the preferences of those people).

## 2 A Multiobjective Camp Location Selection Model

Our multiobjective model aims to determine the optimal location of a new camp and is constructed according to four main steps. First, we create a source country with conflict zones and towns, and interconnecting links. Second, we add a camp at given coordinates in a destination country. Third, we create a link between the camp and its nearest location in the source country, and lastly we run the Flee simulation [11] and calculate the objectives. Figure 1 illustrates the route network for a basic camp placement problem with one conflict zone, three towns and one camp, and interconnecting roads (lines). The coordinates $(x, y)$ associated with each conflict zone, town or camp are used to indicate their positions.

We have the following model assumptions: the locations of conflict zones and towns, the number of asylum-seekers/unrecognized refugees or IDPs (i.e., agents in Flee simulation), and the conflict period are given, agents are spawned in conflict zones, destination countries are represented by a continuous region, camps have limited capacities, agents move during each time step based on predefined rules in [11], and agents stop moving once they reach the camp. With the notation in Table 1, the MOP can be formulated as follows:

$$\text{minimize}: f_1(j) = \frac{\sum_i^{n_{sim,t,j}} d_{sim,t,i,j}}{n_{sim,t,j}}, \quad t = T \tag{1}$$

$$\text{maximize}: f_2(j) = n_{sim,t,j}, \quad t = T \tag{2}$$

$$\text{minimize}: f_3(j) = \frac{\sum_t |c - n_{sim,t,j}|}{T}, \quad t = 1, 2, \ldots, T \tag{3}$$

**Table 1.** Notations for the MOP.

| Notations | Type | Explanation of notations |
|---|---|---|
| $J$ | Set | The set of candidates sites indexed by $j$ |
| $a$ | Parameter | The total number of agents in all conflict zones |
| $n$ | Parameter | The number of potential camp sites |
| $c$ | Parameter | Camp capacity (unit: agent) |
| $k$ | Parameter | The total number of new camps that will be placed and open |
| $T$ | Parameter | The simulation period or the conflict period (unit: day) |
| $j$ | Decision variable | The index of a candidate site |
| $d_{sim,t,i,j}$ | Dependent variable | The distance travelled by an agent $i \in I_{sim,t,j}$ in the new camp at candidate site $j$ at time $t$ based on the simulation predictions |
| $n_{sim,t,j}$ | Dependent variable | The number of agents served by the new camp at candidate site $j$ at time $t$ based on simulation predictions, indexed by $i$ |

subject to

$$1 \nabla j \nabla n \tag{4}$$

The objective function Eq. (1) minimizes the average distance travelled by each arriving agent in a destination camp at the end of the simulation. This objective focuses on the efficiency (i.e., distance) of allocating people to facilities. The objective function Eq. (2) maximizes the number of people in the camp at the end of the simulation. This objective function can be easily changed to a minimization problem by calculating the negative value of successful arrivals (i.e., $-n_{sim,t,j}, t = T$). The objective function Eq. (3) minimizes the average idle camp capacity over simulation days for the new camp. Note that the new camp can be overpopulated, and if the idle capacity is a negative value, we simply take the absolute value. Constraint (4) restricts the search space of the MOP (i.e., a set of $n$ possible sites), from which we select the optimal camp site. In our MOP, the decision variable $j$ is known as a solution to the problem. Different from the single-objective optimization problem, the MOP has a set of trade-off solutions, called Pareto front, rather than a single optimal solution. In this paper, only one camp will be established (i.e., $k = 1$) and we aim to find the Pareto front of the MOP. This MOP can be further extended to jointly solve the MOP for multiple camps by replacing the current single decision variable with a set of decision variables, expressed as a $k$-dimensional decision vector $\vec{j} = (j_1, j_2, \ldots, j_k)$, and considering all people who arrived at these new camps.

### 2.1 A Simulation-Optimization Approach

We develop a simulation-optimization approach, which combines a (Flee) simulation with a multiobjective optimization algorithm. For the optimization algorithm, we adopt a representative multiobjective evolutionary algorithm, called NSGA-II [5]. Our algorithm works as follows: for each generation of NSGA-II, a group of candidate solutions (each solution is a sequence of $k$ selected sites) are generated, followed by the Flee simulation taking the coordinates corresponding to each solution as input parameters, and assessing and outputting the objective values for the optimization stage. To implement NSGA-II, a candidate solution is represented as a chromosome using a grid-based spatial representation strategy. Each grid cell has longitude and latitude coordinates corresponding to its centroid. The chromosome is then sequentially encoded by the indexes of $k$ selected site(s), where $k$ is the number of camps that will be placed and opened. Note that in this paper we only consider one new camp (i.e., $k = 1$). To automate the simulation process, we utilize FabFlee [12], which is a plugin of FabSim3 (https://github.com/djgroen/FabSim3). Due to data complexity, simulation runs for a group of solutions (i.e., candidate camp locations) are computationally expensive. To reduce the runtime, we employ QCG-PilotJob (http://github.com/vecma-project/QCG-PilotJob) to schedule submitted ensemble runs for different camp locations.

## 3 Test Setup and Results

To demonstrate the application of our MOP, we conducted a case study for the South Sudan conflict in 2013. The geographic coordinates of examined region are $N0^\circ - N16^\circ$ and $E20^\circ - E40^\circ$, and the region was divided into 26842 $0.1^\circ \times 0.1^\circ$ (around 11 km $\times$11 km) grids. Our simulation instances (*ssudan_c1* and *ssudan_c2*) are constructed based on the South Sudan simulation instance presented in [12], which involves almost 2 million fleeing people in a simulation period of 604 days starting from the 15th December 2013, 25 conflict zones and 16 towns in South Sudan, as well as ten camps in neighboring countries Sudan, Uganda and Ethiopia. The *ssudan_c1* has no camp in place yet and aims to establish one new camp with a capacity of $80,000$ (i.e., $c = 80,000$), while the *ssudan_c2* involves all ten existing established camps and aims to add one new camp with a capacity of $12,000$ (i.e., $c = 12,000$). For both simulation instances, the distance between camp and its nearest location in South Sudan was estimated by using the route planning method in [10]. Furthermore, to shorten the execution time, we reduced the number of agents from all conflict zones by a factor 100 (i.e., $a = a/100$), and accordingly, the camp capacity for *ssudan_c2* and *ssudan_c2* are reduced to 800 and 120, respectively. Figure 2 plots the optimal camp locations for the two conflict instances. The objective values of optimal solutions obtained by NSGA-II are summarized in Table 2. For each conflict instance, NSGA-II can find a set of optimal solutions, which are incomparable based on the concept of Pareto optimality. In other words, each solution is a trade-off among average travel distance, the number of camp arrivals, and the average idle camp capacity.

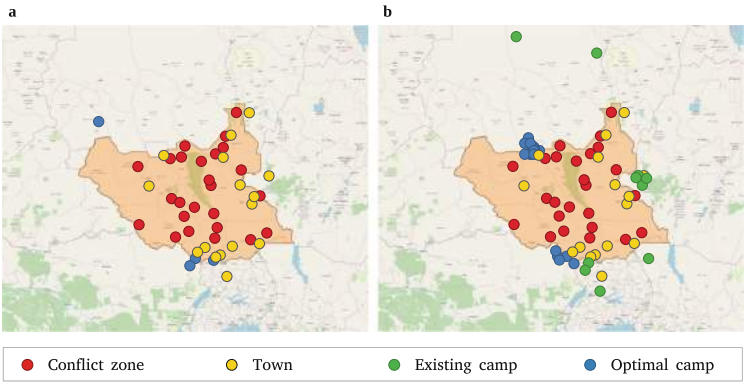



**Fig. 2.** Optimal camp locations (blue circles) obtained by NSGA-II on the (a) ssudan_c1 and (b) ssudan_c2 conflict instances, respectively. (Color figure online)

**Table 2.** The objective values of the optimal solutions obtained by the NSGA-II on the ssudan_c1 and ssudan_c2 conflict instances.

| Conflict instance | Camp location | | Objectives | | |
|---|---|---|---|---|---|
| | Longitude | Latitude | Travel distance | No. camp arrivals | Idle capacity |
| ssudan_c1 | 30.55 | 3.75 | 1380.2211 | 801 | 77.0182 |
| | 25.25 | 11.25 | 6785.469 | 809 | 173.0762 |
| | 31.55 | 3.65 | 1354.2624 | 803 | 82.1556 |
| | 30.25 | 3.35 | 1995.5878 | 804 | 91.2666 |
| ssudan_c2 | 30.35 | 3.85 | 558.905 | 166 | 49.7136 |
| | 29.85 | 3.85 | 651.9379 | 124 | 11.6589 |
| | 29.95 | 3.65 | 598.6553 | 120 | 7.6788 |
| | 28.25 | 10.35 | 440.0152 | 120 | 8.2483 |
| | 28.85 | 9.65 | 226.7078 | 143 | 29.096 |
| | 28.35 | 9.45 | 283.3134 | 150 | 35.351 |
| | 28.55 | 9.55 | 313.1518 | 160 | 44.2268 |
| | 28.45 | 9.55 | 281.1019 | 156 | 40.6904 |
| | 28.65 | 9.55 | 433.7734 | 147 | 32.5613 |
| | 28.05 | 10.05 | 507.1222 | 121 | 8.9636 |
| | 28.45 | 9.65 | 336.701 | 140 | 26.048 |
| | 28.55 | 9.85 | 262.416 | 132 | 19.0679 |
| | 30.75 | 3.45 | 580.9609 | 126 | 13.6225 |
| | 28.55 | 9.75 | 364.0978 | 129 | 16.0894 |
| | 28.45 | 9.45 | 397.1481 | 158 | 42.5331 |
| | 28.35 | 10.05 | 539.0705 | 131 | 18.0646 |
| | 29.75 | 4.15 | 634.4269 | 123 | 10.6474 |
| | 28.55 | 9.65 | 322.0341 | 138 | 24.2169 |
| | 28.05 | 9.45 | 371.0897 | 135 | 21.9901 |
| | 28.55 | 9.45 | 388.0439 | 144 | 29.7268 |

## 4 Conclusion

In this paper, a multiobjective model for the FLP in the context of humanitarian support for forcibly displaced people has been proposed, and the model has been solved by using a simulation-optimization approach. The proposed model has been employed in a case study of South Sudan conflict with a simulation period of 604 days from 15th December 2013. The results obtained by our simulation-optimization approach have demonstrated its ability to provide decision makers with diverse solutions, which strike a balance among the individual travel distance, the number of camp arrivals, and the average idle camp capacity. In the future, other algorithms in multiobjective optimization will be explored. In addition, it would be interesting to consider other factors in the context of forced migration, e.g., construction and transportation costs.

**Acknowledgements.** This work is supported by the ITFLOWS and HiDALGO projects, which have received funding from the European Union Horizon 2020 research and innovation programme under grant agreement nos 882986 and 824115. The authors are grateful to Prof. Simon J E Taylor and Dr. Anastasia Anagnostou for their constructive discussions on this work.

## References

1. Barzinpour, F., Esmaeili, V.: A multi-objective relief chain location distribution model for urban disaster management. Int. J. Adv. Manuf. Technol. **70**(5), 1291–1302 (2014)
2. Boonmee, C., Arimura, M., Asada, T.: Facility location optimization model for emergency humanitarian logistics. Int. J. Disaster Risk Reduct. **24**, 485–498 (2017)
3. Chen, T., Li, M.: The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering. ACM Trans. Softw. Eng. Methodol. **25**(2), 17 (2022)
4. Cilali, B., Barker, K., González, A.D.: A location optimization approach to refugee resettlement decision-making. Sustain. Urban Areas **74**, 103153 (2021)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.A.M.T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
6. Estrada, L.E.P., Groen, D., Ramirez-Marquez, J.E.: A serious video game to support decision making on refugee aid deployment policy. Procedia Comput. Sci. **108**, 205–214 (2017)
7. Ghasemi, P., Khalili-Damghani, K., Hafezalkotob, A., Raissi, S.: Uncertain multiobjective multi-commodity multi-period multi-vehicle location-allocation model for earthquake evacuation planning. Appl. Math. Comput. **350**, 105–132 (2019)
8. Ma, Y., Xu, W., Qin, L., Zhao, X.: Site selection models in natural disaster shelters: a review. Sustainability **11**(2), 399 (2019)
9. Manopiniwes, W., Irohara, T.: Stochastic optimisation model for integrated decisions on relief supply chains: preparedness for disaster response. Int. J. Prod. Res. **55**(4), 979–996 (2017)
10. Schweimer, C., et al.: A route pruning algorithm for an automated geographic location graph construction. Sci. Rep. **11**(1), 1–11 (2021)

11. Suleimenova, D., Bell, D., Groen, D.: A generalized simulation development approach for predicting refugee destinations. Sci. Rep. **7**(1), 1–13 (2017)
12. Suleimenova, D., Groen, D.: How policy decisions affect refugee journeys in south Sudan: a study using automated ensemble simulations. J. Artif. Soc. Soc. Simul. **23**(1) (2020)
13. Xu, W., Zhao, X., Ma, Y., Li, Y., Qin, L., Wang, Y., Du, J.: A multi-objective optimization based method for evaluating earthquake shelter location-allocation. Geomat. Nat. Haz. Risk **9**(1), 662–677 (2018)
14. Zhao, M., Chen, Q.W., Ma, J., Cai, D.: Optimizing temporary rescue facility locations for large-scale urban environmental emergencies to improve public safety. J. Environ. Inform. **29**(1) (2017)

# A Sparse Matrix Approach for Covering Large Complex Networks by Cliques

Wali Mohammad Abdullah(✉) and Shahadat Hossain

University of Lethbridge, Lethbridge, AB, Canada
{w.abdullah,shahadat.hossain}@uleth.ca

**Abstract.** A classical NP-hard problem is the *Minimum Edge Clique Cover (minECC)* problem, which is concerned with covering the edges of a network (graph) with the minimum number of cliques. There are many real-life applications of this problem, such as in food science, computational biology, efficient representation of pairwise information, and so on. Borrowing ideas from [8], we propose using a compact representation, the intersection representation, of network data and design an efficient and scalable algorithm for minECC. Edges are considered for inclusion in cliques in degree-based orders during the clique construction step. The intersection representation of the input graph enabled efficient computer implementation of the algorithm by utilizing an existing sparse matrix package [11]. We present results from numerical experiments on a representative set of real-world and synthetically constructed benchmark graph instances. Our algorithm significantly outperforms the current state-of-the-art heuristic algorithm of [4] in terms of the quality of the edge clique covers returned and running time performance on the benchmark test instances. On some of the largest graph instances whilst existing heuristics failed to terminate, our algorithm could finish the computation within a reasonable amount of time.

**Keywords:** Adjacency matrix · Clique cover · Intersection matrix · Ordering · Sparse graph

## 1 Introduction

The graph kernel operations, such as identification of and computation with dense subgraphs, frequently arise in areas as diverse as sparse matrix determination and complex network analysis [13,14]. In social networks, identification of special interest groups or characterization of information propagation are examples of frequently performed network analytics tasks [23]. The *Edge Clique Cover problem (ECC)* considered in this paper is concerned with finding a collection of complete subgraphs or cliques such that every edge and every vertex of the input graph is included in some clique. The computational challenge is to find an ECC with the smallest number of cliques (*minECC*). The minECC problem is computationally intractable or NP-hard [16].


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 505–517, 2022.
https://doi.org/10.1007/978-3-031-08757-8_43

Effective representation of network data is critical to meeting algorithmic challenges for exactly or approximately solving intractable problems, especially when the instance sizes are large and sparse. In this paper, we use sparse matrix data structures to enable compact representation of sparse network data based on an existing sparse matrix framework [11] to design efficient algorithms for the minECC problem.

Let $G = (V, E)$ be an undirected connected graph, where $V$ is the set of vertices, and $E$ is the set of edges. A clique is a subset of vertices such that every pair of distinct vertices are connected by an edge in the induced subgraph. In graph $G$, an edge clique cover of size $k$ is a decomposition of set $V$ into $k$ subsets $C_1, C_2, \ldots, C_k$ such that $C_i, i = 1, 2, \ldots, k$ induces a clique in $G$ and each edge $\{u, v\} \in E$ is included in some $C_i$. A trivial clique cover with $k = m, |E| = m$ can be specified by the set of edges $E$ with each edge being a clique. Finding a clique cover with the minimum number of cliques (and many variants thereof) is known to be an NP-hard problem [16].

In 1973, Bron and Kerbosch [2] proposed an algorithm to find all maximal cliques of a given graph. That algorithm uses a branch-and-bound technique. The algorithm is made more efficient by cutting off branches of the search tree that will not lead to new cliques at a very early stage. Etsuji Tomita et al. [22] presented a depth-first search algorithm for generating all maximal cliques of an undirected graph, in which pruning methods are employed as in the Bron-Kerbosch algorithm.

Many algorithms have been proposed in the literature to solve the ECC problem approximately. At the same time, there are only a few exact methods that are usually limited to solving small instance sizes. A recent heuristics approach is described by Conte et al. [4] to find an edge clique cover in $O(m\Delta)$ time, where $m$ is the number of edges and $\Delta$ is the highest degree of any vertex in the graph.

In this paper, we use a compact representation of network data based on sparse matrix data structures [11] and provide an improved algorithm motivated by the works of Bron et al. [2], and E. Tomita et al. [22] for finding clique covers. In [1], we used a similar compact representation of network data. In that paper, we employ a "vertex-centric" approach where a vertex, in some judiciously chosen order, together with its edges incident on a partially constructed clique cover, is considered for inclusion in an existing clique. The preliminary implementation produced smaller-sized clique covers when compared with the method of [9] on a set of test instances. While the vertex-centric ECC algorithm frequently produced smaller clique covers compared with other methods, the high memory footprint of the method made it less scalable on very large problem instances. In this paper, we propose an "edge-centric" minECC method. Our method is characterized by a significantly reduced memory footprint and exhibits very good scalability when applied to extremely large synthetic and real-life network instances.

Our approach is based on the simple but critical observation that for a sparse matrix $A \in \mathbb{R}^{m \times n}$, the row intersection graph of $A$ is isomorphic to the adjacency

graph of $AA^\top$, and that the column intersection graph of $A$ is isomorphic to the adjacency graph of $A^\top A$ [11]. Therefore, the subset of rows corresponding to nonzero entries in column $j$ induces a clique in the adjacency graph of $AA^\top$, and the subset of columns corresponding to nonzero entries in row $i$ induces a clique in the adjacency graph of $A^\top A$. Note that matrices $A^\top A$ and $AA^\top$ are most likely dense even if matrix $A$ is sparse. We exploit the close connection between sparse matrices and graphs in the reverse direction. We show that given a graph (or network), we can define a sparse matrix, *intersection matrix*, such that graph algorithms of interest can be expressed in terms of the associated intersection matrix. This structural reduction enables us to use the existing sparse matrix computational framework to solve graph problems [11]. This duality between graphs and sparse matrices has also been exploited where the graph algorithms are expressed in the language of sparse linear algebra [14,15]. However, they use adjacency matrix representation which is different from our intersection matrix representation.

The paper is organized as follows. In Sect. 2, we consider representations of sparse graph data and introduce the notion of intersection representation and cast the minECC problem as a matrix compression problem. Section 3 presents the new edge-centric minECC algorithm. An important ingredient of our algorithm is to select edges incident on the vertex being processed in specific orders. The details of the implementation steps are described, followed by the presentation of the ECC algorithm. The section ends with a discussion on the computational complexity of the algorithm. Section 4 contains results from elaborate numerical experiments. We choose 5 different sets of network data consisting of real-world network and synthetic instances. Finally, the paper is concluded in Sect. 5.

## 2 Compact Representation and Edge Clique Cover

For efficient computer implementation of many important graph operations, representing graphs using adjacency matrix or adjacency lists is inefficient. Adjacency matrix stored as a two-dimensional array is costly for sparse graphs, and typical adjacency list implementations employ pointers where indirect access leads to poor cache utilization [19]. The intersection matrix representation that we propose below enables an efficient representation of pairwise information and allows us to utilize the computational framework DSJM to implement the new ECC algorithm.

### 2.1 Intersection Representation

We require some preliminary definitions. The *adjacency graph* associated with a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is an undirected graph $G = (V, E)$ in which for each column or row $k$ of $A$ there is a vertex $v_k \in V$ and $A(i, j) \neq 0, i \neq j$ if and only if $\{v_i, v_j\} \in E$.

Let $G = (V, E)$ be an undirected and connected graph without self-loops or multiple edges between a pair of vertices. The adjacency matrix $A(G) \equiv A \in \{0,1\}^{|V|\times|V|}$ associated with graph $G$ is defined as,

$$A(i,j) = \begin{cases} 1 & \text{if } \{v_i, v_j\} \in E, i \neq j \\ 0 & \text{otherwise} \end{cases}$$

We now introduce the intersection representation, an enabling and efficient representation of pairwise information. The *intersection representation* of graph $G$ is a matrix $X \in \{0,1\}^{k\times n}$ in which for each vertex $v_j$ of $G$ there is a column $j$ in $X$ and $\{v_i, v_j\} \in E$ if and only if there is a row $l$ for which $X(l,i) = 1$ and $X(l,j) = 1$. A special case is obtained for $k = m$. Then, the rows of $X$ can be uniquely labeled by the edge list sorted by vertex labels. Therefore, matrix $X \in \{0,1\}^{m\times n}$ can be viewed as an assignment to each vertex a subset of $m$ labels such that there is an edge between vertices $i$ and $j$ if and only if the inner product of the columns $i$ and $j$ is 1. Since the input graph is unweighted, the edges are simply ordered pairs and can be sorted in $O(m)$ time. Unlike the adjacency matrix, which is unique (up to fixed labeling of the vertices) for graph $G$, there can be more than one *intersection matrix* representation associated with graph $G$ [1]. We exploit this flexibility to store a graph in a structured and space-efficient form.

Let $X \in \{0,1\}^{m\times n}$ be the intersection matrix as defined above associated with a graph $G = (V, E)$. Consider the product $B = X^\top X$.

**Theorem 1.** *The adjacency graph of matrix $B$ is isomorphic to graph $G$. [1]*

Theorem 1 establishes the desired connection between a graph and its sparse matrix representation. The following result follows directly from Theorem 1.

**Corollary 1.** *The diagonal entry $B(i,i)$ where $B = X^\top X$ and $X$ is the intersection matrix of graph $G$, is the degree $d(v_i)$ of vertex $v_i \in V, i = 1, \ldots, n$ of graph $G = (V, E)$. [1]*

Intersection matrix $X$ defined above represents an edge clique cover of cardinality $m$ for graph $G$. Each edge $\{v_i, v_j\}$ constitutes a clique of size 2. In the intersection matrix $X$, edge $e_l = \{v_i, v_j\}$ is represented by row $l$ with $X(l,i) = X(l,j) = 1$ and other entries in the row being zero. In general, column indices $j'$ in row $l$ where $X(l,j') = 1$ constitutes a clique on vertices $v_{j'}$ of graph $G$. Thus the minECC problem can be cast as a *matrix compression* problem.

**minECC Matrix Problem.** Given $X \in \{0,1\}^{m\times n}$ determine $X' \in \{0,1\}^{k\times n}$ with $k$ minimized such that the intersection graphs of $X$ and $X'$ are isomorphic.

## 3 An Edge-Centric MinECC Algorithm

The algorithm that we propose for the ECC problem is motivated by the maximal clique algorithm due to Bron et al. [2], and E. Tomita et al. [22]. For ease of presentation, we discuss the algorithm in graph-theoretic terms. However, our computer implementation uses a sparse matrix framework of DSJM [11], and all computations are expressed in terms of intersection matrices.

### 3.1 Selection of Uncovered Edges

An edge $\{u, v\} \in E$ is said to be *covered* if both of its incident vertices have been included in some clique; otherwise the edge is *uncovered*. In our algorithm, we select an uncovered edge $\{u, v\}$ and try to construct a maximal clique, $C$, containing the edge. The algorithm selects vertices and edges in a prespecified order during the clique construction process. Note that it may or may not be possible to include additional uncovered edges while building a clique after selecting an uncovered edge. This subsection will give details on how the algorithm selects an uncovered edge.

**Vertex Ordering.** We recall that $d(v)$ denotes the degree of vertex $v$ in graph $G = (V, E)$. Let $Vertex\_Order$ be a list of vertices of graph $G$ using one of the ordering schemes below.

- **Largest-Degree Order (LDO) (see [12]):** Order the vertices such that $\{d(v_i), i = 1, \ldots, n\}$ is nonincreasing.
- **Degeneracy Order (DGO) (see [7,21]):** Let $V' \subseteq V$ be a subset of vertices of $G$. The subgraph induced by $V'$ is denoted by $G[V']$. Assume the vertices $V' = \{v_n, v_{n-1}, \ldots, v_{i+1}\}$ have already been ordered. The $i^{th}$ vertex in DGO is an unordered vertex $u$ such that $d(u)$ is minimum in $G[V \setminus V']$ where, $G[V \setminus V']$ is the graph obtained from $G$ by removing the vertices of set $V'$ from $V$.
- **Incidence-Degree Order (IDO) (see [3]):** Assume that the first $k-1$ vertices $\{v_1 \ldots, v_{k-1}\}$ in incidence-degree order have been determined. Choose $v_k$ from among the unordered vertices that has maximum degree in the subgraph induced by $\{v_1, \ldots, v_k\}$.

**Edge Ordering.** After the vertices have been ordered using one of the above schemes, the algorithm proceeds to choose a vertex in that specific order, which has at least one uncovered incident edge. If there is more than one uncovered edge incident on the vertex being processed, the order in which the edges are processed (i.e., to include in a clique) is as follows. Place all the edges $\{u, v\}$ before $\{p, q\}$ in an ordered edge list, $Edge\_Order$, such that vertex $u$ or vertex $v$ is ordered before vertices $p$ and $q$ in $Vertex\_Order$ list.

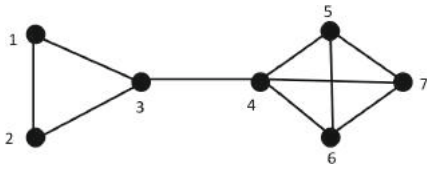



**Fig. 1.** An example of an undirected graph.

Figure 1 shows an undirected graph. $\{4, 3, 5, 6, 7, 1, 2\}$ would be the list with LDO. The edge list induced by the $Vertex\_Order$ will have the following form.

$$Edge\_Order = \{\{4,3\},\{4,5\},\{4,6\},\{4,7\},\{3,1\},\{3,2\},\{5,6\},\{5,7\},\{6,7\},\{1,2\}\}$$

**Edge Selection.** We select an edge to $\{u, v\} \in E$ to include in a new clique if $\{u, v\}$ is uncovered and ordered before all uncovered edges in $Edge\_Order$. The clique that gets constructed with edge $\{u, v\}$ may cover other uncovered edges that are further down the list.

We consider three variants of edge selection for our algorithm, denoted by $L$, $D$, and $I$.

- **L**: In this variant, the set of vertices are ordered using the Largest-Degree Ordering (LDO) scheme. We select a vertex $u$ in that order and then return all the uncovered edges of the form $\{u, v\}$.
- **D**: All the vertices are ordered using Degeneracy Ordering (DGO) scheme. Select a vertex $u$ in that order, and then return all the uncovered edges of the form $\{u, v\}$.
- **I**: Finally, this variant orders the set of vertices using the Incidence-Degree Ordering (IDO) scheme. We select a vertex $u$ in that order and return all the uncovered edges $\{u, v\}$.

### 3.2 The Algorithm

Let $E_{\mathcal{P}} = \{e_1, \ldots, e_{i-1}\}$ be the set of edges that have been assigned to one or more cliques $\{C_1, \ldots, C_{k-1}\}$ and let $e_i = \{v_j, v_{j'}\}$ be the edge currently being processed according to the ordered edge list. Denote by

$$W = \{v_l \mid \{v_j, v_l\}, \{v_{j'}, v_l\} \in E\}$$

the set of common neighbors of $v_j$ and $v_{j'}$.

The complete algorithm is presented below.

**EO-ECC** ($Edge\_Order$)
Input: $Edge\_Order$, set of edges in a predefined order using schemes $L$, $D$, or $I$

```
1:  k ← 0                                          ▷ Number of cliques
2:  for index = 1 to m do                          ▷ m is the number of edges
3:      {u, v} ← Edge_Order[index]
4:      if {u, v} is uncovered then
5:          W ← FindCommonNeighbors(u, v)
6:          if W = ∅ then
7:              k++
8:              C_k ← {u, v}
9:              Mark {u, v} as covered
10:         else
```

11: $k++$
12: $C_k \leftarrow \{u, v\}$
13: Mark $\{u, v\}$ as covered
14: **while** $W \neq \emptyset$ **do**
15: let $t$ be a vertex in $W$
16: $W \leftarrow W \setminus t$
17: **if** $\{t, s\} \in E$ for each $s \in C_k$ **then**
18: Mark $\{t, s\}$ as covered
19: $C_k \leftarrow C_k \cup \{t\}$
20: $FindCommonNeighbors(W, FindNeighbors(t))$
21: **return** $C_1, C_2, ..., C_k$

### 3.3 Discussion

In this subsection, we analyze algorithm EO-ECC to derive it's asymptotic running time. The two kernel operations used in the algorithm are "FindCommonNeighbors" and "FindNeighbors." The *FindCommonNeighbors* operation merges two sorted lists (of integers) and computes the intersection of the lists. The list (of vertices) that this operation returns after each call has at least one fewer vertices. Thus, to construct a clique $C_i$, the total cost would be $(\frac{\gamma_i(\gamma_i-1)}{2})$, where $|C_i| = \rho_i$. Let, $C = \{C_1, C_2, \ldots, C_k\}$ be a clique cover returned by the algorithm EO-ECC. Then the total cost of calling *FindCommonNeighbors* for the algorithm would be $O(\sum_{i=1}^{k} \frac{\gamma_i(\gamma_i-1)}{2})$. The operation *FindNeighbors* in algorithm EO-ECC computes the neighbors set of vertex $v \in V$ [12]. In line 20, *FindNeighbors* operation is used to compute the neighbors of a vertex. Since an uncovered edge gets covered only once, the total cost of *FindNeighbors* operation is at most $O(m)$. Thus, the overall running time of algorithm EO-ECC is $O(m + \sum_{i=1}^{k} \frac{\gamma_i(\gamma_i-1)}{2})$. The following result follows immediately from the above running time expression.

**Theorem 2.** *If the input graph $G$ is triangle-free, then the algorithm EO-ECC runs in $O(m)$ time.*

## 4 Numerical Testing

In this section, we provide results from numerical experiments on selected test instances. 10th Discrete Mathematics and Theoretical Computer Science (DIMACS10) instances and Stanford Network Analysis Platform (SNAP) instances are obtained from the University of Florida Sparse Matrix Collection [5]. (`SNAP`) is a collection of more than 50 large network datasets containing large number of nodes and edges including social networks, web graphs, road networks, internet networks, citation networks, collaboration networks, and communication networks [17]. We also experiment with synthetic graph instances. We generated 182 Erdös-Rényi and Small-World instances using the Stanford

Network Analysis Project (SNAP) [18] instance generator. The number of edges of these generated graphs is varied from 800 to 72 million.

The experiments were performed using a PC with 3.4 GHz Intel Xeon CPU, with 8 GB RAM running Linux. The implementation language was *C++* and the code was compiled using $-O2$ optimization flag with a *g++* version 4.4.7 compiler. We employed the High-Performance Computing system (Graham cluster) at Compute Canada for large instances that could not be handled by the PC.

In what follows, we refer to the vertex-centric ECC algorithm from [1] as Vertex Ordered Edge Clique Cover (`VO-ECC`). We also refer to the ECC algorithm due to Conte et al. as (`Conte-Method`). Finally, the edge-centric minECC algorithm of this paper is identified as Edge Ordered Edge Clique Cover (`EO-ECC`). `EO-ECC` has three variants associated with the three different edge ordering schemes `D`, `L`, and `I`. They are: `EO-ECC-D`, `EO-ECC-L`, and `EO-ECC-I` respectively. In these results, $m$ denotes the number of edges, $n$ denotes the number of vertices of the graph; $|C|$ denotes the number of cliques in the cover, and $t$ is the time in seconds to get the cover. In the presented tables, the smallest cardinality clique cover is marked in **bold**.

**Table 1.** Test Results (Number of cliques) for SNAP instances.

| Graph | | | $\|C\|$ | | |
|---|---|---|---|---|---|
| Name | $m$ | $n$ | VO-ECC using [1] | Conte-Method using [4] | EO-ECC |
| p2p-Gnutella04 | 39994 | 10878 | 38474 | 38491 | **38449** |
| p2p-Gnutella24 | 65369 | 26518 | 63726 | 63725 | **63689** |
| p2p-Gnutella25 | 54705 | 22687 | 53368 | 53367 | **53347** |
| p2p-Gnutella30 | 88328 | 36682 | 85823 | 85822 | **85717** |
| ca-GrQc | 14496 | 5242 | 3777 | 3753 | **3717** |
| as-735 | 13895 | 7716 | 8985 | **8938** | 10130 |
| Wiki-Vote | 103689 | 8297 | 42914 | **39393** | 51145 |
| Oregon-1 | 23409 | 11492 | 15631 | **15491** | 15527 |
| ca-HepTh | 25998 | 9877 | 9663 | 9270 | **9162** |

Table 1 displays the size of clique covers returned by three algorithms: the edge-centric algorithm (`EO-ECC`), the vertex-centric algorithm (`VO-ECC`) discussed in [1] and algorithm (`Conte-Method`) discussed in [4]. `Conte-Method` randomly selects an edge and attempts to build a clique around the selected edge. As the table illustrates, `EO-ECC` produces smaller cardinality edge clique cover than `VO-ECC` except for two instances. On the other hand, it outperforms `Conte-Method` on six out of nine instances.

**Table 2.** Test Results (number of cliques) for DIMACS10 matrices.

| Graph | | | Number of cliques | | | |
|---|---|---|---|---|---|---|
| Name | $m$ | $n$ | Conte-Method | EO-ECC-D | EO-ECC-L | EO-ECC-I |
| chesapeake | 170 | 39 | **75** | 76 | **75** | 76 |
| delaunay_n10 | 3056 | 1024 | 1250 | **1233** | 1275 | 1241 |
| delaunay_n11 | 6127 | 2048 | 2485 | **2449** | 2544 | 2481 |
| delaunay_n12 | 12264 | 4096 | 4993 | **4906** | 5095 | 4939 |
| delaunay_n13 | 24547 | 8192 | 9989 | **9881** | 10211 | 9920 |
| delaunay_n14 | 49122 | 16384 | 19974 | **19672** | 20435 | 19855 |
| delaunay_n15 | 98274 | 32768 | 39923 | **39501** | 40876 | 39782 |
| delaunay_n16 | 196575 | 65536 | 79933 | **78792** | 81528 | 79445 |
| delaunay_n17 | 393176 | 131072 | 159900 | **157792** | 163321 | 158851 |
| delaunay_n18 | 786396 | 262144 | 319776 | **315684** | 326741 | 317987 |
| com-DBLP | 1049866 | 317080 | 238854 | 237713 | **237685** | **237685** |
| belgium_osm | 1549970 | 1441295 | 1545183 | 1545183 | 1545183 | 1545183 |
| delaunay_n19 | 1572823 | 524288 | 639349 | **631354** | 653383 | 635877 |
| delaunay_n20 | 3145686 | 1048576 | 1279101 | **1262843** | 1307080 | 1271229 |
| delaunay_n21 | 6291408 | 2097152 | 2557828 | **2525301** | 2613106 | 2542333 |

Test results for the selected test instances from group `DIMACS10` are reported in Table 2. For comparison, we show the results of `Conte-Method`, `EO-ECC-D`, `EO-ECC-L`, and `EO-ECC-I`. On twelve out of fifteen instances, `EO-ECC-D` gives the least number of cliques to cover all the edges of the given graph. On the graph named `com-DBLP` `EO-ECC-L` and `EO-ECC-I` produce smaller cardinality covers. Overall, `EO-ECC` emerges as the clear winner over `Conte-Method` in terms of the size of the clique covers.

Besides DIMACS10 selected instances, we compare these algorithms on 182 generated instances where the number of edges is varied from 800 to $7.2 \times 10^7$. Using SNAP tool [18], we generated 72 "Small-world" and 110 "Erdös-Rényi" graphs. `EO-ECC` produces smaller (on 47.3%instances) or equal (on 52.7% instances) cardinality clique covers compared with `Conte-Method`.

Rodrigues [20] used different graph instances to evaluate their edge clique cover algorithms. The well-known instances to evaluate edge clique cover problem are from the application "compact letter display" [10]. On thirteen out of fourteen instances, `Conte-Method` [4] gives optimum results. Both Rodrigues's algorithm and our `EO-ECC` give optimum results for all the instances.

The performance comparison between `Conte-Method` and `EO-ECC` is shown in Fig. 2. We compare the time required to find edge clique cover for the given graph.

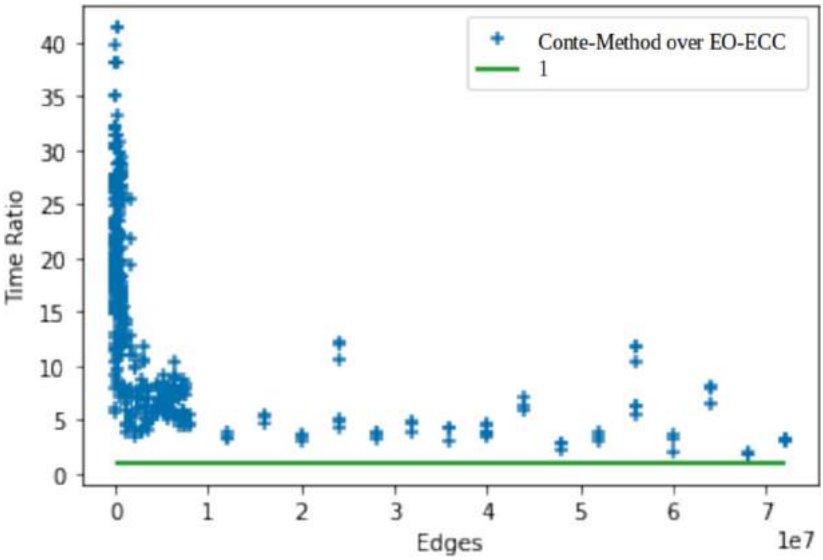



**Fig. 2.** Ratio between the time used by `Conte-Method` and `EO-ECC` for each graph, as a function of the number of the edges (y-axis is in log-scale).

We use fifteen DIMACS10 instances and 182 Erdös-Rényi and Small-World instances. In the figure, a cross mark represents the ratio between the time needed by and `EO-ECC`, as a function of the number of the edges. The green line at height $10^0$ means that `Conte-Method` took the same time as `EO-ECC` to process the corresponding graph, and a cross mark at height $10^1$ means that `Conte-Method` was ten times slower. As the figure clearly demonstrates, `EO-ECC` is always faster than `Conte-Method`, and more than 40 times faster on some of the test instances.

**Table 3.** Graph processing rate (number of edges processed per sec).

| Group | Total instances | Largest rate | Smallest rate | Average rate |
|---|---|---|---|---|
| DIMACS10 | 15 | $2.7E6$ | $3.0E5$ | $1.7E6$ |
| SNAP | 9 | $2.5E6$ | $6.2E4$ | $1.5E6$ |
| Erdös-Rényi | 110 | $2.0E6$ | $1.2E5$ | $8.9E5$ |
| Small World | 72 | $1.7E6$ | $4.3E5$ | $1.1E6$ |

The graph processing rate is one of the quality assessment metrics for an algorithm. We report the processing rate of our algorithm for a selection of real-world (DIMACS10, SNAP) and synthetically generated (Erdös-Rényi, Small World) graphs in Table 3. Table 3 shows the largest rate, the smallest rate, and

the average rate for each set of graph instances. On DIMACS10 instances, the algorithm performs the best, while on Erdös-Rényi instances, the algorithm is not as efficient. This can be explained by the structural properties of graphs. Real-life and Small World synthetic instances display a power-law degree distribution resulting in a large proportion of vertices with very small degrees. Thus, the set intersection operation in our algorithm can be very efficient on those types of graphs.

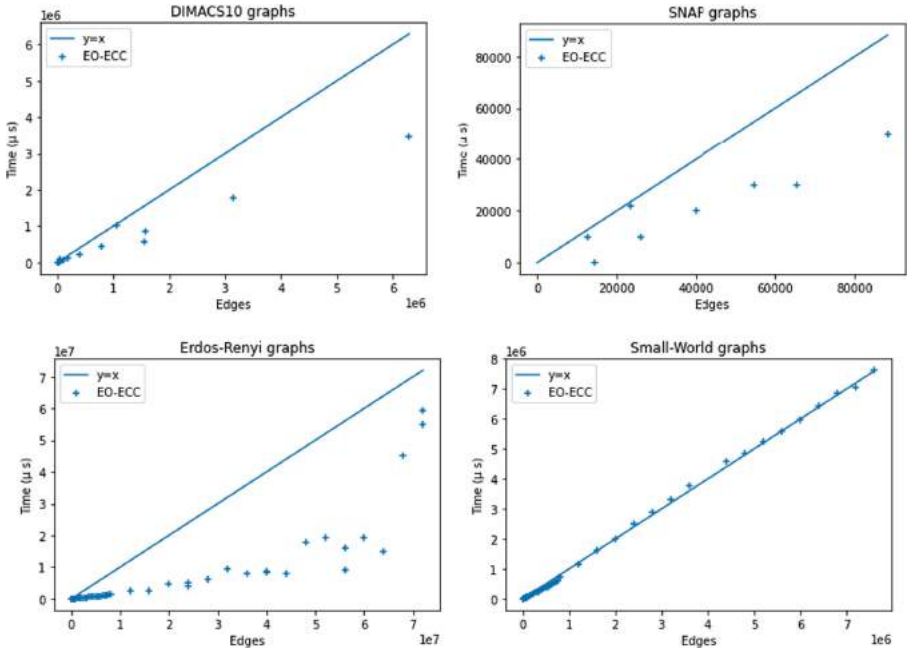



**Fig. 3.** Runtime to find clique cover using `EO-ECC`.

Finally, in Fig. 3, we demonstrate the superior scalability of our algorithm. The figure plots the time used to compute clique covers by `EO-ECC`, where the time is a function of the number of edges in the graph. We report the time in microseconds. A dot $(x, y)$ states that the graph has $x$ edges, and the algorithm spent $y$ microseconds to finish the computation. The figure also displays the line $y = x$ for comparison with the actual running time. On each of the four sets of test instances, the running time shows a linear relationship with the number of edges, demonstrating that the running time of `EO-ECC` is linear in practice.

## 5 Conclusion

In this work, we have proposed a compact representation of network data. The edge clique cover problem is recast as a sparse matrix determination problem.

The notion of *intersection matrix* provides a unified framework that facilitates the compact representation of graph data and efficient implementation of graph algorithms. The adjacency matrix representation of a graph can potentially have many nonzero entries since it is the product of an intersection matrix with its transpose. We have compared our results concerning the clique cover size and runtime with the current state-of-the-art algorithm for minECC [4]. Our algorithm achieves significantly smaller clique covers on the vast majority of the test instances and never returns a clique cover that is larger than the `Conte-Method` [4]. It is also significantly faster than the `Conte-Method`. `EO-ECC` algorithm runs in linear time, which allowed us to process extremely large graphs, both real-life and generated instances. Finally, our algorithm is highly scalable on large problem instances, while the algorithm of `Conte-Method` does not terminate on instances containing $7 \times 10^7$ or more edges within a reasonable amount of time.

A less well-studied but related problem, known as the *Assignment Minimum Edge Clique Cover* arising in computational statistics, is to minimize the number of individual assignments of vertices to cliques It is not always possible to find assignment-minimum clique coverings by searching through those that are edge-clique-minimum. Ennis et al. [6] presented a post-processing method with an existing ECC algorithm to solve this problem. However, their backtracking algorithm becomes costly for large graphs, especially when they have many maximal cliques. Our edge-centric method can be easily adapted, via a post-processing step, to assignment minimum cover calculation. This research is currently being carried out. Results from preliminary computational experiments with a new linear-time post-processing scheme are favourable.

**Acknowledgments.** This research was supported in part by NSERC Discovery Grant (Individual), and the AITF Graduate Student Scholarship. A part of our computations were performed on Compute Canada HPC system (http://www.computecanada.ca), and we gratefully acknowledge their support.

## References

1. Abdullah, W.M., Hossain, S., Khan, M.A.: Covering large complex networks by cliques–a sparse matrix approach. In: Kilgour, D.M., Kunze, H., Makarov, R., Melnik, R., Wang, X. (eds.) Recent Developments in Mathematical, Statistical and Computational Sciences. pp. 117–127. Springer International Publishing, Cham (2021)
2. Bron, C., Kerbosch, J.: Algorithm 457: finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973). https://doi.org/10.1145/362342.362367
3. Coleman, T.F., Moré, J.J.: Estimation of sparse jacobian matrices and graph coloring blems. SIAM J. Numerical Anal. **20**(1), 187–209 (1983)
4. Conte, A., Grossi, R., Marino, A.: Large-scale clique cover of real-world networks. Inf. Comput. **270**, 104464 (2020)
5. Davis, T., Hu, Y.: Suitesparse matrix collection. https://sparse.tamu.edu/. Accessed 10 Feb 2019
6. Ennis, J., Ennis, D.: Efficient Representation of Pairwise Sensory Information. IFPress **15**(3), 3–4 (2012)

7. Eppstein, David, Strash, Darren: Listing all maximal cliques in large sparse real-world graphs. In: Pardalos, Panos M.., Rebennack, Steffen (eds.) SEA 2011. LNCS, vol. 6630, pp. 364–375. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20662-7_31
8. Erdös, P., Goodman, A.W., Pósa, L.: The representation of a graph by set intersections. Canadian J. Math. **18**, 106–112 (1966)
9. Gramm, J., Guo, J., Huffner, F., Niedermeier, R.: Data reduction, exact and heuristic algorithms for clique cover. In: Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX), SIAM, pp. 86–94 (2006)
10. Gramm, J., Guo, J., Huffner, F., Niedermeier, R., Piepho, H., Schmid, R.: Algorithms for compact letter displays: comparison and evaluation. Comput. Stat. Data Anal. **52**, 725–736, 104464 (2007)
11. Hasan, M., Hossain, S., Khan, A.I., Mithila, N.H., Suny, A.H.: DSJM: a software toolkit for direct determination of sparse Jacobian matrices. In: Greuel, G.-M., Koch, T., Paule, P., Sommese, A. (eds.) ICMS 2016. LNCS, vol. 9725, pp. 275–283. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42432-3_34
12. Hossain, S., Khan, A.I.: Exact Coloring of Sparse Matrices. In: Kilgour, D.M., Kunze, H., Makarov, R., Melnik, R., Wang, X. (eds.) AMMCS 2017. SPMS, vol. 259, pp. 23–36. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99719-3_3
13. Hossain, S., Suny, A.H.: Determination of large sparse derivative matrices: structural: orthogonality and structural degeneracy. In: B. Randerath, H., Roglin, B., Peis, O., Schaudt, R., Schrader, F., Vallentin and V. Weil. 15th Cologne-Twente Workshop on Graphs & Combinatorial Optimization, Cologne, Germany, pp. 83–87 (2017)
14. Kepner, J., Gilbert, J.: Graph Algorithms in the Language of Linear Algebra, Society for Industrial and Applied Mathematics. Philadelphia, PA, USA (2011)
15. Kepner, J., Jananthan, H.: Mathematics of Big Data: Spreadsheets, Databases, Matrices, and Graphs. MIT Press (2018)
16. Kou, L., Stockmeyer, L., Wong, C.: Covering edges by cliques with regard to keyword conflicts and intersection graphs. Commun. ACM **21**(2), 135–139 (1978)
17. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection, June 2014. http://snap.stanford.edu/data. Accessed 10 Feb 2019
18. Leskovec, J., Sosič, R.: Snap: A general-purpose network analysis and graph-mining library. ACM Trans. Intell. Syst. Technol. (TIST) **8**(1), 1 (2016)
19. Park, J.S., Penner, M., Prasanna, V.K.: Optimizing graph algorithms for improved cache performance. IEEE Trans. Parallel Distrib. Syst. **15**(9), 769–782 (2004)
20. Rodrigues, M.O.: Fast constructive and improvement heuristics for edge clique covering. Discrete Opt. **39**, 100628 (2021)
21. Rossi, R.A., Gleich, D.F., Gebremedhin, A.H., Patwary, M.M.A.: Fast maximum clique algorithms for large graphs. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 365–366 (2014)
22. Tomita, E., Tanaka, A., Takahashi, H.: The worst-case time complexity for generating all maximal cliques and computational experiments. Theor. Comput. Sci. **363**(1), 28–42 (2006)
23. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press (1994)

# DSCAN for Geo-social Team Formation

Maryam MahdavyRad, Kalyani Selvarajah(✉), and Ziad Kobti

School of Computer Science, University of Windsor, Windsor, ON, Canada
{mahdavy,kalyanis,kobti}@uwindsor.ca

**Abstract.** Nowadays, geo-based social group activities have become popular because of the availability of geo-location information. In this paper, we propose a novel Geo-Social Team Formation framework using DSCAN, named DSCAN-GSTF, for impromptu activities, aim to find a group of individuals closest to a location where service requires quickly. The group should be socially cohesive for better collaboration and spatially close to minimize the preparation time. To imitate the real-world scenario, the DSCAN-GSTF framework considers various criteria which can provide effective Geo-Social groups, including a required list of skills, the minimum number of each skill, contribution capacity, and the weight of the user's skills. The existing geo-social models ignore the expertise level of individuals and fail to process a large geo-social network efficiently, which is highly important for an urgent service request. In addition to considering expertise level in our model, we also utilize the DSCAN method to create clusters in parallel machines, which makes the searching process very fast in large networks. Also, we propose a polynomial parametric network flow algorithm to check the skills criteria, which boosts the searching speed of our model. Finally, extensive experiments were conducted on real datasets to determine a competitive solution compared to other existing state-of-the-art methods.

**Keywords:** Geo-social networks · Geo-social groups · DSCAN

## 1 Introduction

Nowadays, geo-based social group activities have become popular because of the availability of geo-location information. In this paper, we propose a novel Geo-Social Team Formation framework using DSCAN, named DSCAN-GSTF, for impromptu activities, aim to find a group of Geo-Social Networks (GeoSNs) is online social networks that allow geo-located information to be shared in real-time. The availability of location acquisition technologies such as GPS and WiFi enables people to easily share their position and preferences to existing online social networks. Here, the preferences can be common interests, behavior, social relationships, and activities. This information is usually derived from a history of an individual's locations and Geo-tagged data, such as location-tagged photos and the place of the current event [27]. Thus, we have several popular GeoSNs such as Facebook, Twitter, Flickr, Foursquare, Yelp, Meetup,


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 518–533, 2022.
https://doi.org/10.1007/978-3-031-08757-8_44

Gowalla, and Loopt. Consequently, GeoSNs have drawn significant attention in recent years by researchers on many applications, including finding friends in the vicinity [13,23], group-based activity planning [4], and marketing [6].

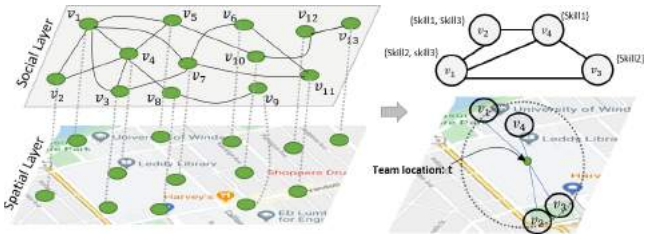



**Fig. 1.** Identifying individuals for impromptu social activity from a GeoSNs.

An impromptu activity planning is one of the popular motivating applications in GeoSNs search. For example, the COVID19 outbreak is affecting every part of human lives. At the initial stage of COVID19, essential services such as health, finance, food, and safety suffered a lot due to unexpected lockdown because they did not have the required human resources as expected. At the same time, fulfilling societies' requirements are also equally important. Therefore, bringing skilled people to the location where the services with diverse demands were crucial and had become a very challenging process. The location of the individual should be close to the place where service is required, and the individual with the required skills needs to be suitable for services. In this example, a service might require several skills. Additionally, each individual may contribute to as many skills as possible in various activities and might have a specific capacity to be involved in multiple activities.

To support this situation, we highly believe that forming groups from GeoSN is an effective solution. In general, this is called the Geo-social group search problem [18,19], which aim to identify groups of individual who are socially cohesive and spatially closest to a location [4]. In other words, the Geo-social group should satisfy that the participants are socially close within the group to confirm good communication between each member and spatially closest to the location of the service to bring them as soon as possible. Figure 1 represents a general Geo-Social Network, where the social layer is to show the social connections between individuals, and the spatial layer is to show the locations of the individuals.

Many existing Geo-social group models considered social cohesiveness and spatial closeness to find successful groups. In addition to these two requirements, recently, Chen et al. [4] incorporated a few essential parameters such as the collective capabilities and capacity of each member. However, to the best of our knowledge, none of the existing Geo-social group models considered the weights of the user's skills which assist in choosing the exact qualified individual. Moreover, efficiently searching required keywords that have high expertise, capacity constraints, social constraint, and spatial closeness altogether have not been

explored in the existing studies. Forming a search framework that can quickly narrow the search space while preserving the correct result is an NP-Hard. It is an open problem and essential to solve in polynomial time [4].

This paper proposes a novel framework to search efficiently on large GeoSNs while preserving the correct result. First, handling large networks is a time-consuming process. So we adopt a recently proposed methodology, Distributed Structural Clustering Algorithm for Networks (DSCAN) [20] algorithm to efficiently manage large networks, which is an extension of SCAN [24]. The basic idea of SCAN is to discover clusters, hubs, and outliers included in a given graph. Initially, in our model, all the nodes of a given graph are randomly divided into equal size sub-graphs and distributed into different machines so that the remaining processing can be conducted simultaneously. Then, by employing the skewness-aware edge-pruning method on the sub-graphs, DSCAN eliminates unwanted edges and moves missing neighbors of nodes from one sub-graph to another. Second, producing socially cohesive groups from these sub-graphs is another essential process. So, DSCAN collects the Core nodes with higher structural similarity and creates a set of clusters from these sub-graphs. Parallelly, the set of skills is collected from each cluster and stored on a map. The third requirement is to choose a spatially close group to the location ($\nabla$) where the service is required. We pick a node randomly from each cluster and evaluate the geographical distance from $\nabla$. The clusters are ranked based on the distance in ascending order. Then a cluster with the lowest rank is selected and tested to see whether it satisfies the requirements of the query or not. If it does not satisfy, move to the following cluster and test the requirement. This process will be repeated till we find the right cluster. Finally, we propose a new polynomial algorithm based on the parametric flow network [8], which checks the skills requirements of the query and contribution capacity of each individual in the selected cluster while considering the user's skills weights.

**Our Contribution:** The followings are the summary of our contributions:

1. We propose a Geo-Social Team Formation (GSTF) model by considering the group's collective capabilities as required for the activities while considering the capacity of contribution from each member and expertise level.
2. We utilize the benefits of Distributed Structural Graph Clustering (DSCAN) to manage the large GeoSNs efficiently.
3. We propose a new polynomial searching algorithm based on the parametric flow network, which satisfies Minimum keywords, expertise level, and capacity constraints.

The rest of the paper is organized as follows. Section 2 discusses related existing work. Section 3 defines the problem definitions of our proposed model. Our framework is presented in Sect. 4. Following that, Sect. 5 illustrates the experimental setup and the corresponding results. Finally, Sect. 6 concludes the research idea of this paper with directions for future work.

## 2 Related Work

Forming a group of individuals for various purposes has been tackled in many different ways. The team formation problem in Social networks was first introduced by Lappas et al. [12]. Later many studies [10,16] have been conducted by incorporating various parameters which influence the successful formation of teams in several applications, including academic collaborations, healthcare [17], and human resource management. However, many of these studies focused highly on minimizing or maximizing some social constraints such as communication cost between members in a team based on their past relationship, profit, and productivity cost.

The concept of GeoSNs services was first introduced by Huang et al. [9]. Many studies have focused on querying geo-social data in order to derive valuable information from both the users' social interactions and current locations [1,21]. Among these, forming Geo-social groups has taken considerable attention of researchers recently since this aims to identify a set of most suitable individuals for various activities which can be planned or unplanned. The unplanned activities such as groups for various purposes during unexpected events, for example, Wildfire and flooding, are relatively complex than the planned activities such as a group for a party or a game. Much existing research proposed various models for both situations while satisfying social constraints and optimizing spatial proximity [4,21]. Many of these focused on forming a group that satisfies a single social constraint while optimizing the spatial proximity. But for impromptu activities, in reality, we require individuals who have diverse demands of skills for multiple tasks or services to serve in a specific location. Recently, Chen et al. [4] introduced a novel framework to discover a set of groups that is socially cohesive while spatially closest to a location for diverse demands. Here, the groups of individuals do not necessarily know each other in the past. However, When there is a tie between two members, their model gives higher priority to the individual who is highly cohesive to the team. The concept of multiple social constraints for various activities has already been studied in [15,17]. However, they considered the frameworks on social networks with known individuals.

Searching cohesive subgroups from a large network is another challenging process in the team formation problem. Structural Clustering Algorithm for Networks (SCAN) algorithm was proposed to detect cohesive subgroups from a network [24]. However, SCAN is a computationally expensive method for a large network because it requires iterative calculation for all nodes and edges. Later, to overcome the limitation with SCAN, many clustering methods have been proposed, such as PSCAN [26], and DSCAN [20]. Since DSCAN is efficient, scalable, and exact, we employ this methodology in our model. To the best of our knowledge, we, for the first time, applied DSCAN in the Geo-Social group search problem.

## 3 Problem Definition

Given an undirected graph $G = (V, E)$, where $V$ is a set of vertices and $E$ is a set of edges. The Graph $G$ incorporates network structures, spatial information, and textual information. In real networks, vertices are users or people, and edges between them may be friendship or previous collaboration. Additionally, each vertex $v \in V$ includes location information, which can be represented as $\nabla = (v.x, v.y)$, where $v.x$ is latitude and $v.y$ is longitude, and a set of keyword attributes which can be represented as $v.A$. The textual information can be a set of skills $S = \{s_1, s_2, \ldots, s_k\}$ of a vertex $v \in V$, where $k$ is the number of skills that a person is expert in. Along with the skills, a vertex has a set weight $W = \{w_1, w_2, \ldots w_k\}$ to represent how much a person expert in each skill.

**Definition 1.** ***Query*** *(Q): The query defines the requirements of skills and number of people in each skills. This includes a Geo-location* $\nabla$ *(latitude (x) and longitude (y)), a set of required skills* $S = \{s_1, s_2, \ldots, s_r\}$*, a set of required expertness in each skills* $P = \{p_1, p_2, \ldots, p_r\}$ *and a contribution capacity of an expert c for every query keyword needs to be assigned.*

**Definition 2.** ***Geo-Social Team*** *(B): For a given location* $\nabla$*, a set of the required number of experts who satisfies social cohesiveness and spatial closeness is selected from a Geo-Social network G while considering contribution capacity c and person's skill weight in each skill.*

In our model, we exploit the DSCAN to handle larger data efficiently. To understand the concept of DSCAN, the following definition are necessary.

**Definition 3.** ***Structural Neighborhood*** *($N_v$): The structural neighborhood $N_v$ of vertex v can be defined as,*

$$N_v = \{w \in V | (v, w) \in E\} \cup \{v\} \tag{1}$$

**Definition 4.** ***Structural similarity:*** *The structural similarity* $\Delta(v, w)$ *between v and w can be defined as,*

$$\Delta(v, w) = |N_v \cap N_w| / \sum \overline{|N_v||N_w|} \tag{2}$$

*If* $\Delta(v, w) \geq \vec{}$*, vertex v shares similarity with w and* $\vec{} \in \mathbb{R}$ *is a density threshold which we assigned.*

When a vertex has enough structurally identical neighbors, it becomes the seed of a cluster, named core node. Core nodes have at least $\mu$ number of neighbors with a structural similarity ($\Delta(v, w)$) that exceeds the threshold $\vec{}$.

**Definition 5.** ***Core:*** *For a given* $\vec{}$ *and* $\mu$*, A vertex* $w \in V$ *is called a core, iff* $N_{w,\epsilon} \geq \mu$*. Where* $N_{w,\epsilon}$ *is the set of neighbor nodes of core node w, and structural similarity of* $N_{w,\epsilon}$ *is greater than* $\vec{}$*.*

**Definition 6.** ***Cluster*** *($\mathcal{C}_w$): Assume node w be a core node. SCAN collects all nodes in* $N_{w,\epsilon}$ *into the same cluster* ($\mathcal{C}_w$) *of node w, initially* $\mathcal{C}_w = \{w\}$*. SCAN outputs a cluster* $\mathcal{C}_w = \{v \in N_{u,\epsilon} | u \in \mathcal{C}_w\}$*.*

**DSCAN Algorithm:** When DSCAN [20] receives a graph, it first deploys disjointed subgraphs of the given graph $G$ to distributed memories on multiple machines $M = \{M_1, M_2, \ldots, M_n\}$ for a given a density threshold $\bar{} \in \mathbb{R}$ and a minimum size of a cluster $\mu \in \mathbb{N}$, where $n$ is number of machines. Initially DSCAN randomly moves a set of vertices $V_i$ in subgraph $G_i = (V_i, E_i)$ for each machine $M_i$. The subgraphs are then processed in a parallel and distributed fashion. Additionally, DSCAN uses edge-pruning based on skewness to improve efficiency further.

**Skewness-Aware Edge-Pruning:** DSCAN applies $\Theta$–skewness edge-pruning to remove unwanted edges and move missing neighbors of nodes from one subgraph to another. Given graph $G = (V, E)$, consider an edge $(u, v)$ be in $E$ and

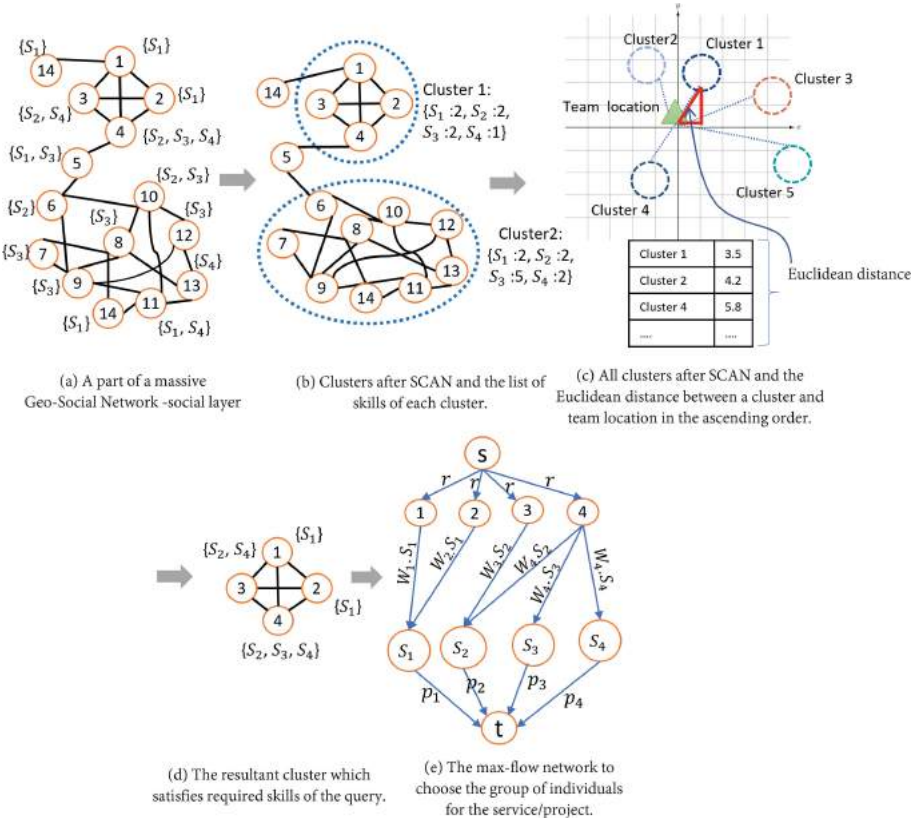



**Fig. 2.** The DSCAN-GSTF framework: (a) a part of a massive geo-social network - social layer, (b) the clusters and the list of skills that each cluster satisfies. (c) The ordered distance between a cluster and the team location (d) selected cluster which satisfies spatial constraint and skill constraint. (e) The max-flow network to choose the successful individuals for the service.

the structural neighborhood of node $u$ is $N_u = \{v \in V | ((u,v) \in E)\} \cup \{u\}$. And $\Theta$–skewness for each edge can be defined as,

$$\Theta(u,v) = min \left( \frac{d_u}{d_v}, \frac{d_v}{d_u} \right) \tag{3}$$

where $d_i = |N_i|$. If $\Theta(u,v) < \epsilon^2$, then the edge $(u,v)$ is considered dissimilar and prune from the graph [20].

## 4 Our Framework

The Geo-Social team formation framework, DSCAN-GSTF consists of three primary processes. 1) Distribute $G$ into multiple machines and perform local clustering. 2) Choose the cluster proximate to the location. 3) Apply parametric flow algorithm to select a competent Geo-Social team of experts. We describe each step one by one in the following sections.

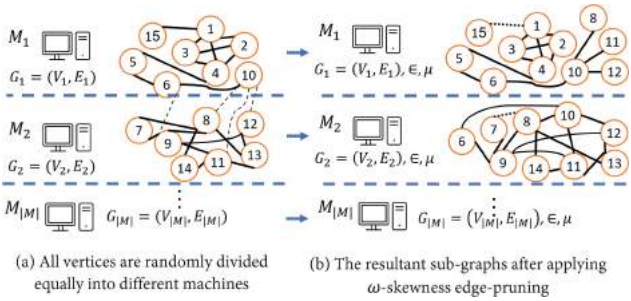



**Fig. 3.** The overview of parallel processing by using DSCAN to replace the process of Fig. 2 (b).

### 4.1 Network Distribution

The GeoSNs are very large networks with millions of edges and vertices. We replicated DSCAN [20] framework in our application. A given large network $G$ is randomly divided into equal size of sub-graphs $\{G_1, G_2, \ldots, G_n\}$. We then deploy each sub-graph into separate machine $M_i$ as shown in Fig. 3 (a). However, sub-graphs $G_i$ and $G_j$ might have neighbor nodes with higher structural similarity ($\geq \mu$). Those nodes should communicate across machines $M_i$ and $M_j$. So, DSCAN employs skewness-aware edge-pruning to keep a low communication cost for billion-edge graphs [20], which is shown in Fig. 3 (b). The skewness-aware edge-pruning drops unnecessary edges to avoid the unwanted communication cost among the machines and moves missing neighbors of nodes which have a high structural similarity, from one sub-graph to another.

The sub-graphs which are placed in each machine, are again clustered based on the structural similarity [26]. Here, DSCAN finds all core nodes in each sub-graph and constructs clusters with the nodes which have high structural similarities. Additionally, we store the list of skills of each cluster. The example of resultant clusters and the list of skills are displayed in Fig. 2 (b).

### 4.2 Suitable Cluster Selection

From the list of clusters, we then select the clusters that satisfy the skill constraints of the given query. To ensure the spatial proximity, we evaluate the Euclidean distance between each cluster and the location $\nabla$ where the service is required (Fig. 2 (c)). We order these distances in ascending order and choose the nearest one that is satisfy the required list of skills as shown in Fig. 2 (d). The selected one is then sent to searching algorithms to find a competent geo-social team. We discuss this process in the following section.

### 4.3 Geo-social Team Formation

We propose a polynomial searching algorithm based on the parametric flow network [8] to find a competent Geo-Social team ($B$). We describe the preliminaries of the parametric flow network one by one.

**Flow Network:** A flow network $N_G = (N_V, N_E)$ is a directed graph that contains a source node $s$, an target node $t$, a set of middle nodes $N_V$, and directed edge set $N_E$. Additionally, each edge has a weight and receive a flow. An edge's weight cannot be exceeded by the amount of flow that passes through it.

**Max Flow and min $s-t$ cut:** Let's say $f$ is a flow of $N_G$, the flow across an $s-t$ cut $(S,T)$ divide its nodes into $S$ and $T$ parts so that the sum of the capacities across $S$ and $T$ is minimized. So, the maximum amount of flow moving from an $s-t$ cut in $N_G$, say $f(S,T)$ is equal to the total weight of the edges in a minimum cut, $\sum_{u\in S, v\in T} f(u,v)$.

**Preflow:** A PreFlow $f$ on $N_G$ is a real-valued function that satisfies the capacity and anti-symmetric constraints on node pairs. The relaxation of the conservation constraint can be defined as, $\sum_{u\in V(D)} f(u,v) \geq 0, \forall u \in V \backslash \{s\}$

**Valid Labelling:** A valid labelling $h$ for a preflow $f$ is a function which is attached to the vertices and has positive integers, such that $h(t) = 0$, $h(s) = |N_V(N_G)|$, where $|N_V(N_G)|$ is the number of vertices in network $N_G$ [3]. For every directed edge from node $v$ to $u$, the relabeling of $h(v) \leq h(u)+1$ should be created to have a valid flow. In other words, for any node $v$ is a valid labelling if $h(v) \leq min\{h_f(v,t), h_f(v,s) + |V(D)|\}$. The purpose of such labelling $h(v)$ is to estimate the shortest distance from the vertex $v$ to $s$ or $t$ [7].

**Calculation of min s-t cut:** After running the max-flow algorithm, a minimum cut can be found as follows. For each node $v \in V$, replace $h(v)$ by $min\{h_f(s,v), h_f(t,v) + |N_V(N_G)|\}$. Now the $s-t$ cut is equal to $S = \{v | h(v) \geq |N_V(N_G)|\}$ where the sink partition $T$ is of the minimum size.

**Parametric Network Flow:** The maximum or minimum value of the flow is determined using a max-flow algorithm based on some criteria. In a parametric-flow network $N_R$, the capacities on arcs out of $s$ and into $t$ are functions of a real-valued parameter $\rho$, and edges possess the following characteristics [8]. For all $v \neq t$ the cost of the edges from source node to $v$ nodes $C_{(s,v)}(\rho)$ is a non-decreasing function of $\rho$. Also, for all $v \neq s$ the cost of the edges from $v$ nodes to target node $t$, $C_{(v,t)}(\rho)$ is a non-increasing function of $\rho$. And finally, for all $v \neq s$ and $v \neq t$ the cost of the edges from node $v$ to node $u$, and $C_{(u,v)}(\rho)$ is a constant. Parametric networks measure maximum flow or minimum cut based on a particular parameter value $\rho$.

**Triangle in Graphs:** A triangle in $G$ is a cycle of length 3. A triangle generated on vertices $u, v, w \in V(G)$ is denoted as $Tri(uvw)$.

**Context Weighted Density** $(CW)$**:** In the selected subgraph $H \subset G$ which satisfies the requirement of query $Q$, vertices that are related to the query $Q$ may be loosely or densely connected. To balance both these situation, we decided to evaluate context weighted density, $(CW)$ so that we can have cohesive group [22]. The context weighted density, $(CW)$ can be calculated with the use of both weighted triangle density and weighted edge density. For a given edge $(u, v) \in E(H)$, $(u, v, w) \in Tri(H)$, the context scores can be defined as below,

$$\text{Edge context score:} w(e(u,v)) = |Q \cap A(u)| + |Q \cap A(v)| \tag{4}$$

$$\text{Triangle context score:} W(T(u,v,w)) = \sum_{e \in \{(u,v),(u,w),(v,w)\}} w(e) \tag{5}$$

where $w(e(u, v))$ is the weight of edge $(u, v)$ and $A(u)$ and $A(u)$ are the set of attributes of vertex $u$ and $v$ respectively.

$$\text{context weighted density: } CW(H) = \frac{\sum_{\Delta \in Tri(\Delta)} w(\Delta) + \sum_{e \in E(H)} w(e)}{|V(H)|} \tag{6}$$

Algorithm 1 shows how to find required skills using a tailored parametric preflow algorithm. It starts by considering the whole input subgraph $H$ as a candidate team. The candidate team is the group of members who satisfies the query criteria. In the line 2, We construct a parametric flow network based on the steps explained in the part below. Then, we use the stop condition in line 3 to check whether the subgraph $H$ itself is a candidate team or not. If not we generate a better solution by solving sub problem $l(ad_0, H_0 \leftarrow H'_0)$, is defined as below [3],

$$l(ad_0, H_0 \leftarrow H'_0) = \sum_{\Delta \in Tri(H'_0)} w(\Delta) + \sum_{e \in E(H'_0)} w(e) - ad_0 \times (|V(H'_0)|) \tag{7}$$

Algorithm 1 considers the progressively modified $ad(H_0)$ as a parameterized capacity in $N_R$. The overall structure of the algorithm is similar to optimization

**Algorithm 1.** Skills Query Search Algorithm

**Input**: cluster $H \in G$, Query $Q$

1: $H_0 \leftarrow H$, $ad_0 \leftarrow AD(H)$
2: Construct an adapted parametric flow network $N_R$ and $\epsilon = AD(H_0)$
3: obtain $H_0'$ from min s-t cut in $N_R$
4: **while** $l(ad_0, H_0 \leftarrow H_0')) \neq 0$ **do**
5: $ad_0 \leftarrow AD(H_0), \epsilon = ad_0$
6: obtain $H_0'$ from min s-t cut in $N_R$
7: **end while**
8: **return** $H_0$

algorithm, i.e., it continuously generates $H_0$ with higher context weighted density until reaching the stop condition. During each iteration, internally the algorithm maintains preflow labels via updating the labels computed from the previous iteration. In order to compute $H_0'$, preflow value and some edge capacities are updated according to $H_0$ generated in the previous iteration. The improved solution gets generated repeatedly until the stop condition is met, i.e., a candidate team is found.

### 4.4 Complexity Analysis

Assume $|V| = n$ and $|E| = m$. In first step of Geo-social team Formation, it takes $O(m^{1.5})$ time to compute structural similarity. As a result, on each machine $M_i$ extracting all the core nodes from $G_i$ takes $O(m^{1.5})$ time. Consequently finding all dissimilar edges of $E_i$ requires $O(\frac{m}{|M|})$ time. The skills checking complexity can be bounded by $O(|V(cluster)^3|)$, making use of the maximum-flow algorithm. However, providing parametric-network flow help us to solve this in a time complexity of solving one min-cut problem.

## 5 Experimental Analysis

We conducted experiments to demonstrate the efficiency and effectiveness of our framework. From the efficiency point of view, we show that the Geo-Social team formation model is faster than the state-of-the-art algorithms on large graphs. The proposed framework finds a resulting team with specified features in a Geo-social network having 1.5 billion edges within $8s$. Furthermore, for demonstrating the effectiveness of DSCAN-GSTF, two illustrative queries are analyzed on various real datasets and various metrics based on spatial and social cohesiveness.

**Dataset:** Table 1 describes the statistical information of five real-world GeoSNs with ground-truth clusters use to evaluate our framework.

**Table 1.** Statistics of real-world datasets.

| Dataset | # of Nodes | # of Edges | Ave-Deg |
|---|---|---|---|
| Gowalla [5] | 196,591 | 950,327 | 9.177 |
| Dianping [2] | 2,673,970 | 1,922,977 | 12.184 |
| Orkut-2007 [25] | 3,072,626 | 34,370,166 | 76.277 |
| Ljournal-2008 [14] | 5,363,260 | 79,023,142 | 14.734 |
| Twitter-2010 [11] | 41,652,098 | 1,468,365,182 | 35.253 |

### 5.1 Experiment Setup

We compare our framework, DSCAN-GSTF, with state-of-the-art models MKC-SSG [4] and geo-social group queries model (GSGQ) [28]. The MKCSSG model satisfies minimum keyword, contribution capacity, as well as social and spatial constraints. The GSGQ did not consider the required number of experts for each skill. Therefore, we change the GSGQ and add the skill constraint to the team's required skills query such that the skills attributes of the members in the resulting team should cover all required skills.

All the above models are implemented in python using NetworkX, Panda, Tensorflow, Numpy, and pyWebGraph libraries. For the distributed and multiple processing in DSCAN-GSTF, we used MPI. All the experiments are performed on a computer cluster of 16 machines with an interconnecting speed of 9.6 GB/s running GNU/Ubuntu Linux 64-bit. Furthermore, each machine's specifications were Intel Xeon E5-2665 64-bit CPU and 256 GB of RAM (8 GB/core). Moreover, MKCSSG and GSGQ are implemented on one machine since they are not distributed algorithms. Each model is executed 20 times, and the average score is recorded.

### 5.2 Effectiveness Evaluation

To show the effectiveness of the Geo-Social team formation framework, we analyzed two representative queries on the Gowalla dataset.

**Query 1:** Parameters are set as follow: location $\nabla = (36.11, -115.13)$, Skills $S = \{salad, chicken, beef, BBQ\}$, Number of each skills $E = \{10, 10, 10, 10\}$, Contribution capacity $c = 4$. This query can be used to find fans of BBQ party around Las Vegas. We assume that the tweets of each user is their favorite dish. We set $\vec{} = 0.5$ and $\mu = 10$ for the first query on Geo-Social team formation framework.

**Query 2:** intends to create a music band around Las Vegas. Query 2 parameters are set as follows: location $\nabla = (36.11, -115.13)$, Skills $S = \{guitar, piano, violin\}$, Number of each skills $E = \{2, 1, 2\}$, Contribution capacity $c = 1$. We set $\vec{} = 0.5$ and $\mu = 10$ for the second query on Geo-Social team formation model.

**Table 2.** Effectiveness evaluation

| Model | SC | GD | ED | Query |
|---|---|---|---|---|
| GSGQ | 0.14 | 0.71 | 0.51 | Query 1: food |
| MKCSSG | 0.18 | 0.41 | 0.67 | |
| DSCAN-GSTF | 0.21 | 0.23 | 0.74 | |
| GSGQ | 0.56 | 0.54 | 0.38 | Query 2: music band |
| MKCSSG | 0.67 | 0.28 | 0.63 | |
| DSCAN-GSTF | 0.56 | 0.23 | 0.81 | |
| GSGQ | 0.27 | 0.62 | 0.43 | Query 3: board game |
| MKCSSG | 0.43 | 0.32 | 0.52 | |
| DSCAN-GSTF | 0.42 | 0.25 | 0.68 | |

**Query 3:** intends to create a board game groups. Query 3 parameters are set as follows: location $\nabla = (36.11, -115.13)$, Skills $S = \{chess, backgammon, monopoly\}$, Number of each skills $E = \{2, 8, 10\}$, Contribution capacity $c = 2$. We set $\bar{} = 0.4$ and $\mu = 9$ for the second query on Geo-Social team formation model.

**Evaluation Metrics:** Here we define the evaluation metrics use to compare the performance of state-of-the-art methods with DSCAN-GSTF.

**Spatial Closeness (SC):** The spatial cohesiveness is to show how closely the team members are located to $\nabla$. Our framework uses the spatial distance of one random member of the result team $B$ to the query location $\nabla$.$SC_{\nabla,B} = \{(Euclidean\_Distance(\nabla, u)), u \in V(B)\}$

**Graph Diameter (GD):** It calculates the topological length or extent of a graph by counting the number of edges in the shortest path between the most distant vertices. In other words, graph diameter indicates the **social closeness** of the team. $GD_B = max\{ShortestPath(v, w))|(v, w) \in V(B)\}$

**Edge Density (ED):** We consider another parameter ED to show the social cohesiveness. $ED_B = |E(B)|/|V(B)|$, where $E(B)$ is the number of edges in team $B$ and $V(B)$ is the number of vertices in team $B$.

The comparison results of analyzed metrics are presented in Table 2. The results are normalized to a value between 0 and 1. Results with a lower score are better except for Edge Density (ED). Overall speaking, we can see DSCAN-GSTF has outperformed in spatial and social cohesiveness in both queries. However, the spatial distance is not significantly better, but the social cohesiveness shows excellent improvement in both queries. Furthermore, applying graph structural communities in DSCAN-GSTF framework improves the social Cohesion

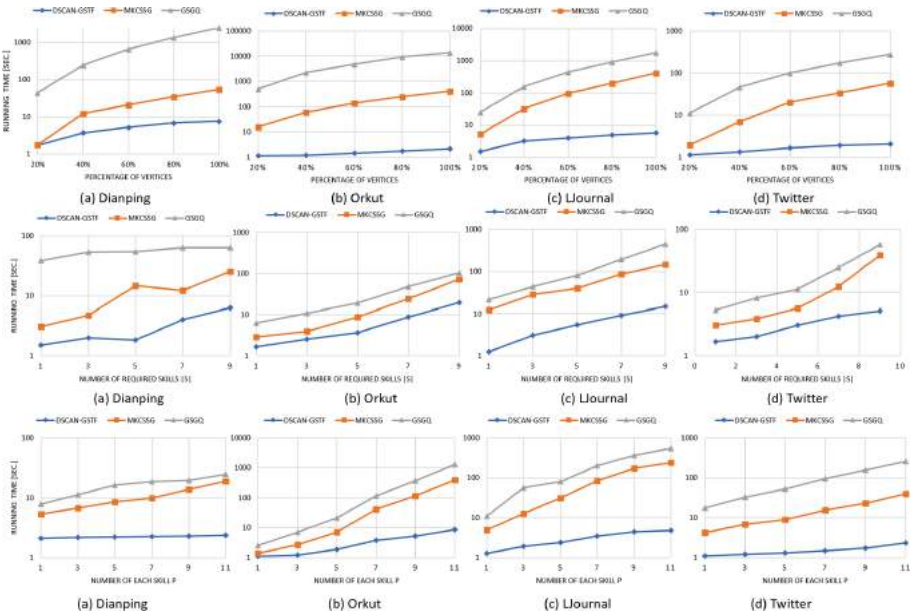



**Fig. 4.** (a) The first raw is to compare the efficiency evaluation based on percentage of vertices. (b) The middle raw is to compare the efficiency evaluation on various number of required skills. (c) The last raw is to compare the efficiency evaluation on various requirement on expert set $R$.

and indicates teams with much less graph diameter than GSGQ and MKCSSG, which utilize the minimum degree and c-truss constraint, respectively.

### 5.3 Efficiency Evaluation

To compare the performance of various models, we use the running time of the queries. We compare the efficiency of GSGQ and MKCSSG with our proposed model, DSCAN-GSTF. Our experiments uses various parameter settings for a query: percentage of vertices, sets of required skills $|S| = \{1, 3, 5, 7, 9\}$, and the minimum number of each skill $E = \{1, 3, 5, 7, 9\}$. We set the default value of both $|S|$ and $E$ to 3. The location for each query is created randomly. We select reasonable values for $\vec{}$and $\mu$ based on each dataset. In Dianping dataset $\vec{}= 0.3$ and $\mu = 2$, in the Orkut dataset $\vec{}= 0.5$ and $\mu = 5$, in the LJournal dataset $\vec{}= 0.6$ and $\mu = 5$, and finally in the Twitter dataset $\vec{}= 0.5$ and $\mu = 6$. When a parameter is changing for evaluation, other parameter values are set to their default value.

We divide each dataset into various percentages to evaluate the scalability of proposed model. The result is presented in Fig. 4 (a) for different datasets while comparing various methods. In general, our DSCAN-GSTF is much more scalable compared to other methods on different datasets. That is because of the

methodology and substantially the properties of DSCAN-GSTF which can limit the search space in quicker time while preserving optimum results.

Figure 4 (b) shows the running time when the number of required skills increases for different datasets; as the required skills increase, the running time for all methods increases. However, this increase is slower in DSCAN-GSTF because checking existing skills on each cluster using the attached summation list of skills has constant time complexity. When $|S|$ is small, the GSGQ requires comparatively high running time to find optimum results. However, when the $|S|$ grows, it provides a result in half a minute. In Fig. 4 (c), changing the number of required skills $E$ on each dataset using various models is presented. Again, for all the datasets, the running time increase as the required number of skills is increased. Nevertheless, because of distributed environment in DSCAN-GSTF, the increasing running time is on a slow increasing slope.

## 6 Conclusions

This paper has explored the Geo-Social team Formation framework and proposed a new model DSCAN-GSTF. In this, we incorporated various criteria to replicate the real-world scenario and exploited DSCAN for the efficient process of large networks. The DSCAN-GSTF introduced a novel polynomial algorithm based on a parametric flow algorithm to identify the successful team members for impromptu activities from GeoSNs. We compared our proposed DSCAN-GSTF model with the state-of-the-art methods, MKCSSG and GSGQ. Extensive experiments were conducted to examine the efficiency and effectiveness of the proposed model on four real-world datasets and recorded the running time under various system settings. Overall, our proposed model generated the output faster than the state-of-the-art methods. As for future work, we plan to extend DSCAN-GSTF to incorporate more sophisticated queries.

## References

1. Armenatzoglou, N., Papadopoulos, S., Papadias, D.: A general framework for geo-social query processing. Proc. VLDB Endow. 913–924 (2013)
2. Bu, J., et al.:: ASAP: a Chinese review dataset towards aspect category sentiment analysis and rating prediction (2021)
3. Chen, L.: Efficient cohesive subgraph search in big attributed graph data (2018)
4. Chen, L., Liu, C., Zhou, R., Xu, J., Yu, J.X., Li, J.: Finding effective geo-social group for impromptu activities with diverse demands. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2020)
5. Cho, E., Myers, S.A., Leskovec, J.: Friendship and mobility: user movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2011)
6. Cliquet, G., Baray, J.: Location-Based Marketing: Geomarketing and Geolocation. Wiley, Hoboken (2020)
7. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2009)

8. Gallo, G., Grigoriadis, M.D., Tarjan, R.E.: A fast parametric maximum flow algorithm and applications. SIAM J. Comput. **18**(1), 30–55 (1989)
9. Huang, Q., Liu, Y.: On geo-social network services. In: 2009 17th International Conference on Geoinformatics, pp. 1–6. IEEE (2009)
10. Kargar, M., An, A., Zihayat, M.: Efficient bi-objective team formation in social networks. In: Flach, P.A., De Bie, T., Cristianini, N. (eds.) ECML PKDD 2012. LNCS (LNAI), vol. 7524, pp. 483–498. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33486-3_31
11. Kwak, H., Lee, C., Park, H., Moon, S: What is Twitter, a social network or a news media? In: Proceedings of the 19th International Conference on World Wide Web (2010)
12. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 467–476 (2009)
13. Liu, W., Sun, W., Chen, C., Huang, Y., Jing, Y., Chen, K.: Circle of friend query in geo-social networks. In: Lee, S., Peng, Z., Zhou, X., Moon, Y.-S., Unland, R., Yoo, J. (eds.) DASFAA 2012. LNCS, vol. 7239, pp. 126–137. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29035-0_9
14. Rossi, R., Ahmed, N.: The network data repository with interactive graph analytics and visualization. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29 (2015)
15. Selvarajah, K.: Investigation of team formation in dynamic social networks (2020)
16. Selvarajah, K., Zadeh, P.M., Kargar, M., Kobti, Z.: Identifying a team of experts in social networks using a cultural algorithm. Procedia Comput. Sci. (2019)
17. Selvarajah, K., Zadeh, P.M., Kobti, Z., Kargar, M., Ishraque, M.T., Pfaff, K.: Team formation in community-based palliative care. In: Innovations in Intelligent Systems and Applications (2018)
18. Shen, C.Y., Yang, D.N., Huang, L.H., Lee, W.C., Chen, M.S.: Socio-spatial group queries for impromptu activity planning. IEEE Trans. Knowl. Data Eng. **28**(1), 196–210 (2015)
19. Shen, C.Y., Yang, D.N., Lee, W.C., Chen, M.S.: Spatial-proximity optimization for rapid task group deployment. ACM Trans. Knowl. Discov. Data (TKDD) **10**(4), 1–36 (2016)
20. Shiokawa, H., Takahashi, T.: DSCAN: distributed structural graph clustering for billion-edge graphs. In: Hartmann, S., Küng, J., Kotsis, G., Tjoa, A.M., Khalil, I. (eds.) DEXA 2020. LNCS, vol. 12391, pp. 38–54. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59003-1_3
21. Sohail, A., Cheema, M.A., Taniar, D.: Geo-social temporal top-k queries in location-based social networks. In: Borovica-Gajic, R., Qi, J., Wang, W. (eds.) ADC 2020. LNCS, vol. 12008, pp. 147–160. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-39469-1_12
22. Tsourakakis, C.: The k-clique densest subgraph problem. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1122–1132 (2015)
23. Valverde-Rebaza, J.C., Roche, M., Poncelet, P., de Andrade Lopes, A.: The role of location and social strength for friendship prediction in location-based social networks. Inf. Process. Manag. **54**(4), 475–489 (2018)
24. Xu, X., Yuruk, N., Feng, Z., Schweiger, T.A.: Scan: a structural clustering algorithm for networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 824–833 (2007)

25. Yang, J., Leskovec, J.: Defining and evaluating network communities based on ground-truth. Knowl. Inf. Syst. **42**(1), 181–213 (2013). https://doi.org/10.1007/s10115-013-0693-z
26. Zhao, W., Martha, V., Xu, X.: PSCAN: a parallel structural clustering algorithm for big networks in MapReduce. In: International Conference on Advanced Information Networking and Applications (AINA) (2013)
27. Zheng, Y.: Location-based social networks: users. In: Zheng, Y., Zhou, X. (eds.) Computing with Spatial Trajectories, pp. 243–276. Springer, New York (2011). https://doi.org/10.1007/978-1-4614-1629-6_8
28. Zhu, Q., Hu, H., Xu, C., Xu, J., Lee, W.-C.: Geo-social group queries with minimum acquaintance constraints. VLDB J. **26**(5), 709–727 (2017). https://doi.org/10.1007/s00778-017-0473-6

# Data Allocation with Neural Similarity Estimation for Data-Intensive Computing

Ralf Vamosi and Erich Schikuta(✉)

Faculty of Computer Science, University of Vienna, Vienna, Austria
ralf.vamosi@cs.univie.ac.at, erich.schikuta@univie.ac.at

**Abstract.** Science collaborations such as ATLAS at the high-energy particle accelerator at CERN use a computer grid to run expensive computational tasks on massive, distributed data sets.

Dealing with big data on a grid demands workload management and data allocation to maintain a continuous workflow. Data allocation in a computer grid necessitates some data placement policy that is conditioned on the resources of the system and the usage of data.

In part, automatic and manual data policies shall achieve a short time-to-result. There are efforts to improve data policies. Data placement/allocation is vital to coping with the increasing amount of data processing in different data centers. A data allocation/placement policy decides which locations sub-sets of data are to be placed.

In this paper, a novel approach copes with the bottleneck related to wide-area file transfers between data centers and large distributed data sets with high dimensionality. The model estimates similar data with a neural network on sparse and uncertain observations and then proceeds with the allocation process. The allocation process comprises evolutionary data allocation for finding near-optimal solutions and improves over 5% on network transfers for the given data centers.

**Keywords:** Data allocation · Data placement · Similarity estimation · Parallel computing · Wide-area file transfers

## 1 Introduction

Physics collaborations such as the ATLAS collaboration [10] at CERN store their data as files in a worldwide computing grid. The LHC Computing Grid [3] provides distributed storage and processing for physics data from extensive experiments. Experts from all over the world submit tasks with the data stored as files across many data centers. The data centers can be considered geographically distant sites that spawn a fully connected network, as shown in Fig. 1 as an example. Many data-intensive, high-performance applications use large numbers of files that are in some way labeled or categorized. In the case of ATLAS, data sets label a various number of files that jobs can collectively handle. A computational job is a process that reads a data set and eventually outputs another


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 534–546, 2022.
https://doi.org/10.1007/978-3-031-08757-8_45

data set. Data sets can be used none or several times. The population of data sets is changing. Some may be removed after their lifespan, and data sets are created from time to time to label files. Because of this changing population of data sets, they cannot be the basic unit of data placement/allocation, but the constituent files are.

Regardless of the particular content, data sets are managed by users. A user defines a class on its own, using particular data sets. Their work interest and funding correlate with data sets. Data sets in a more or less narrow range will be processed to fulfill their research. Those have some commonalities, so some aspects are equal or similar. A metric based on machine learning will be introduced later to find similarities within data sets.

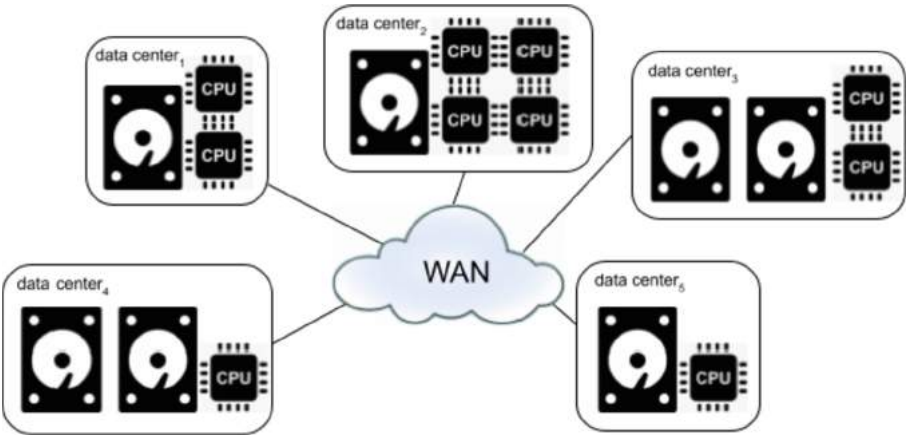



**Fig. 1.** Example of five data centers inter-connected with wide area network (WAN), over which each data center can read files on an external data center, which incurs costs. Each HPC site facilitates different amount of resources.

In our case, there are two types of jobs to be executed in large quantities:

- Producing final data with parameters and calibration data from the experiment. Calibration data is updated from time to time and therefore production must be repeated, for example.
- Analysis to find events in the productive data. Examples around the well-known Higgs boson $H^0$ are $t + \bar{t}$ into $H^0$, or $\Delta + \Delta$ into $H^0$. An $H^0$ boson is formed in both processes, but they are different processes with different probability amplitudes.

Jobs for production and analysis are submitted onto data sets. Computational resources are facilitated in the form of job queues at data centers whose task is to run jobs. A job requires the entire data set to be at the local data center. Missing files of the data set must be transferred to the job, that is, the location of the compute job.

The network latency causes a significant delay in the workflow. The *wide area network (WAN)*, the network between data centers, is a scarce resource and represents a bottleneck in a data-intensive workflow [14]. The way to improve the situation related to the WAN is to leave wide-area transfers out, i.e., local processing for each data set. However, jobs cannot be placed arbitrarily to the appropriate data sets: Jobs are allocated to proper sites. First, they must be compatible; the site must support the requested computation. Say, a high-memory worker is needed. Load balancing takes place according to provided computing volumes in data centers. The one with the majority of files of proper sites is preferred as the target. This process continues, and sites are filled with jobs.

On the other hand, data cannot be freely allocated as there are natural storage limitations. Our focus is to tackle data allocation under the following condition:

- Every time a job kicks in, missing input data must be transferred over the bottleneck, the inter-data center network/WAN, from the source to the target site.
- The network consists of non-uniform sites that provide resources with considerable differences. Some possess more storage capacity, and others provide more processing capabilities due to local funding constraints and even support for various user communities.

In order to deal with a large amount of data, clustering is utilized. Clustering is a broad field applied for different tasks such as classification and segmentation, matching commonalities such as identifying normal samples versus outliers.

The *novel contributions* of this paper are:

A *similarity metric*, a distance, is introduced for data sets to be able to put clustering into action and to decide on proper sub-sets.

Combined with this similarity metric, a *loss function* related to the data allocation task is induced. Based on this loss, allocation can be performed regarding spatial patterns of data set use, that is, locations at data centers. Following patterns of data use is necessary for improving data allocation because data sets will be differently used depending on jobs/tasks on distant sites. The jobs will be distributed in some way across data centers. Some of these data centers are capable of more than others. Say, some store more data sets or provision a processing capability for some job class such as large memory. Large memory jobs require larger memory on the CPU.

## 2 State of the Art

The data allocation problem is synonymously referred to hereto as file allocation. It was investigated when distributed databases were studied, and parallelization had to be utilized.

One of the first data placement/allocation papers was [5]. The work models the task on different abstraction levels. It is further proven that the data placement problem is very difficult to solve and generally NP-Complete.

The file placement problem is investigated under concurrency conditions to build a model with storage cost and communication cost in [17]. Constraints are the multiplicity of databases, variable routing of data, and available capacity of the network nodes.

Some work attempts to cluster data sets according to their interdependency [11]. Subsequently to clustering, clusters are stored on separate machines. A different clustering algorithm is described in [25] that uses k-means algorithm for finding locations for the clustered data, resulting in task allocation to the data centers with most of the input data set. This is comparable to the ATLAS workflow. This paper utilizes data sets as small clusters and imposes several conditions such as uncertainty and sparsity on the data sets.

Evolutionary algorithms have been applied to almost any kind of problem, and it is no surprise that these were applied to this kind of problem as well. In [12], data allocation strategies have been investigated to reduce transaction costs. A genetic algorithm was used here to limit communication effort between data centers by balancing the load.

Placement of files and replication across different nodes may improve on metrics such as job execution, makespan, and bandwidth [8].

Further efforts have been undertaken in previous studies for database optimizations. The authors of [22] discuss database allocation optimization and propose a mathematical model concerning average waiting time. Other database approaches attempt to arrange data effectively over the network nodes, such as in [1,2,6]. However, these studies investigate idealized database cases. For example, they focus on a single query type or do not consider any constraints on communication characteristics. Room for improvement would comprise user behavior and workflow characteristics. Analysis of access patterns can be beneficial for network utilization.

A well-known approach is ranking data according to the number of accesses per time unit. This characteristic is referred to, for example, as data *popularity* [7]. In [7], a successful popularity model is established which uses historical data. A popularity model is implemented as an autonomous service for finding obsolete data and used in the cleaning process [15,20].

Due to complexity and variety, simplified models and local optimizations were studied and applied. A thorough analysis of data usage and data optimization for data grids is necessary. Storage resources and the use of data has to be appropriately treated in the process of file placement [18,19].

Summing up, the actual research on data partitioning and allocation techniques is specific, and often the models aim at a narrow case. Especially in data-intensive high-performance computing (HPC), large volumes of data are processed, and patterns for data usage have to be observed and taken into account to utilize the performance and improve the workflow fully. At this point, we would like to present our research. In general, data management strongly impacts usability, performance, and cost.

# 3 Methods

## 3.1 Background

Data sets collect files in various numbers by giving them a common name (data set name). To certain data sets, jobs will be submitted by a magnitude of 1000 users around the globe. The data set name is a high-dimensional pattern on an abstract level. It consists of a text string in variable lengths, with different sub-strings such as numbers and terms. A data set name (label) looks like

$$mc15.13TeV.362233.Sherpa.CT10.Zee.Pt70$$
$$.140.BFilter.ckkw30.evgen.EVNT.e4558.tid07027172.00$$

There are many identifiers composed into this name to describe the data set. mc15 stands for Monte Carlo; 13 TeV stands for $13 * 10^{12}$ electron volt, which implies the energy of the run; and so on. With this notation, it is challenging to find interrelations between data set names. At least, users manage jobs processing and data sets. The decision is a user-dependent process that was introduced in Sect. 1. User's co-variates are background and funding, and they do usually not select randomly at their whim. Creating data sets and submitting jobs is a confusing process from the outside. However, the common factor is the user between task and data.

Two important points must be addressed in a data allocation policy:

- How likely is it, that data sets will be read by a job of a certain type. It is certain that data sets that are not processed, do not incur any costs from a network's point of view. The opposite case is when a data set is processed at least once or several times.
- What is the impact from the cost perspective, data sets residing on specific nodes.

In the model context, the term node is used for data centers. A node represents a black box, which is not decomposed in its parts, such as network storage, different computing nodes, etc., but rather is described on a top-level with the provision of resources from a job's perspective.

With data sets also comes a lack of information. The use of data sets is uncertain two-folds:

- Data sets change over time. Some of them will overlap.
- Furthermore, there is no information on how likely they will be used for most data sets. Only a tiny minority of data sets have a popularity value that indicates how likely they will be read. There are some investigations on popularity, which are not covered in this work [4].

Our model consists of two sub-models joined together to perform the two steps depicted in Fig. 2:

1. Transform the variable space of data sets into a metric space. This model is introduced to generate variables of data sets that allow better handling. The first model in Sect. 3.2 covers this part.

2. Generate disjoint file sets from data sets and allocate them to nodes. This can be pictured in two dimensions as on the right side in Fig. 2. The allocation is the process that divides files, comprised in data sets, into the number of nodes and maps them to the nodes, which are data centers in grid. The necessary steps are done in the second model in Sect. 3.3.

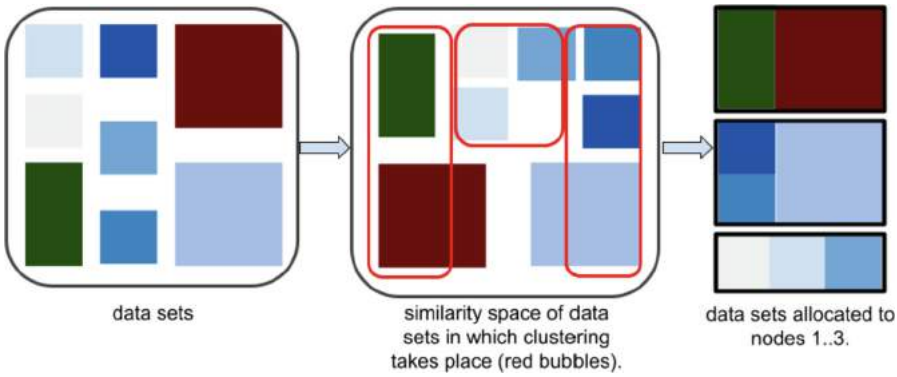



**Fig. 2.** Two models for two transformations of the data sets. First, the data sets are mapped into a similarity space. Second, the allocation model disjoints the data sets into proper sub-sets to allocate to nodes. (Color figure online)

### 3.2 Neural Similarity Estimator

The relationship between users and data sets, and thereby data sets to others, has been discussed. The similarity model proposed in this work relates data sets with others. The neural similarity model is placed first to extend the variables-space for the allocation.

With the similarity model, given a data set as input, another similar data set may be found in the population of data sets. Even new data sets that have not yet been seen can be assigned to other data sets. This is important because new data sets are constantly being created during operation. On the one hand, similarity plays a role in the allocation in order to select a specific kind of data sets. On the other hand, data sets also depend on each other because data sets can overlap several others. They will be more or less mixed, which makes them not separable.

The virtue of such a similarity model is that it can handle complex and inter-related (text) strings. Triplet learning sometimes referred to as triplet loss or triplet comparison, comes in helpful for this task. This technique makes inferences about observations based on commonalities and non-commonalities within the variables [24]. A specific loss function is applied to learn a similarity or distance metric by recognizing similarities (and dissimilarities). It was successfully applied for large learning applications in image similarities [9,23] as well as a variety of digital image processing tasks such as face recognition [16].

In the case of face recognition and person re-identification, similar samples are images of the same person. Figure 3 illustrates the core idea. The so-called anchor a and the positive sample + are from the same user, and the other example - is from another user. The triplet comparison operates on three different example inputs. On average, the $a$ and + are more similar to each other. During network training, the cost function applied to a triplet of samples is gradually decreased. The distance between the anchor and the positive sample becomes smaller, while the distance between the anchor and the negative sample increases.

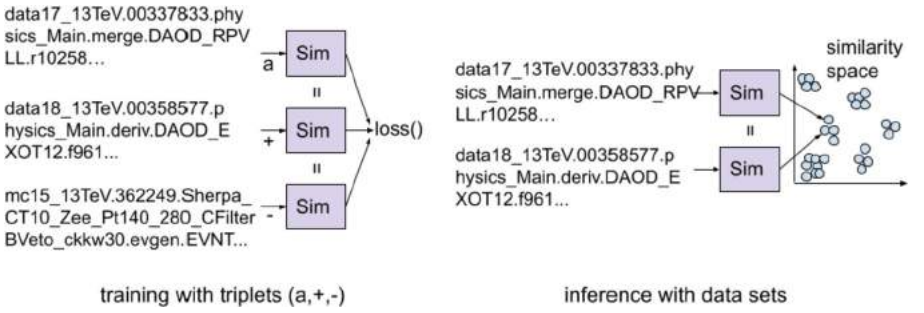



**Fig. 3.** Training of the model is shown on the left, in which triplets of data sets (a, +, −) are fed from the training set while the model tries to lower the loss(). The inference is shown on the right. Data sets are mapped into the latent similarity space, where next data sets have higher similarity.

In our case, data sets with their names are the input from users. As data set names are depicted in Sect. 3.1, the first step is text pre-processing to obtain a tuple of word vectors that are the representations for the data set names. Terms and numbers occurring only once in the training set of data set names are replaced by a default value since they are not informative. For example, 933 and aabb are replaced as default value.

The model consists of several steps:

1. The data set names are tokenized. This process makes a discrete set with unique numbers from each alpha-numeric subtext. In a standard text case, this would be the word level, and each word would convert into a number in the discrete co-domain.
2. Each token is converted into a word vector.
3. In training, a latent metric space is spawned under the loss function from Eq. 1. Data sets are just passed through the trained model in the inference phase. Data sets become labeled with the vector of the low-dimensional neural co-domain of the similarity model Fig. 3.

The model's network architecture is strongly inspired by image recognition projects with a triplet loss, like [13], we do not need a particular network type such as CNN for the short input vectors at hand. Even though a CNN would

be possible, we go with a fully connected multi-layer feed-forward network: a multi-layer perceptron. As for other recognition tasks, as mentioned, a triplet loss is used in the network:

$$loss = max[d(\ a,\ +) - d(\ a,\ -) + margin, 0] \tag{1}$$

where a triplet is $(a, +, -)$ with some data set, $a$, a positive example data set from the same user $+$, and a negative example data set from some other user, $-$.

The training phase takes a magnitude of one day on a single GPU. In the prediction phase, the model maps each piece of input data set into its output space, a metric space in which similar inputs are placed closer to each other. This is depicted on the right in Fig. 3. Inference on one data set is just a mapping with the fixed weights in the model and takes a magnitude of 1 ms.

### 3.3 Data Allocation Model

For the allocation process, an evolutionary clustering method has been developed. The goal is to find improved data allocations over the course of the run time in terms of a loss according to the following loss function decreases:

$$cost_{tx}(\boldsymbol{S}) = \sum_{j \in \{jobs\}} cost_{tx}(S, j) \tag{2}$$

where $cost_{tx}(S, j)$ denotes an affine function of the number of non-local files for job j given $\boldsymbol{S}$, that is, the number of triggered file transfers over the network (WAN) by j.

The data allocation model generates an allocation matrix for all files, $\boldsymbol{S}$. As explained, network transfers between nodes are expensive, which is why the total number of these transfers is regarded in the total loss denoted as $cost_{tx}(S)$. Data center-internal transfers are much faster and do not need to be considered (zero cost). Given the amount of work, i.e., a set of computational jobs, the problem can be formalized as minimization problem with parameter $S$ specifying the allocation, and a loss function $cost_{tx}(S)$ depending on system variables and the set of jobs:

$$\begin{aligned} &\underset{\boldsymbol{S}}{argmin}\ cost_{tx}(\boldsymbol{S}) \\ &so\ that \\ &\boldsymbol{S}^T \times \mathbf{w}_{file} \leq \mathbf{w}_{store} \end{aligned} \tag{3}$$

where $cost_{tx}(\boldsymbol{S})$ is the loss function introduced before, $\mathbf{w}_{file}$ is a column vector which represents the file sizes for $file_1, ..., file_M$ and $\mathbf{w}_{store}$ is a column vector which represents the storage capacities of data centers 1, 2, ..., N. $cost_{tx}(\boldsymbol{S})$ penalizes external heavy transfers. $\mathbf{w}_{store}$ is needed by the model to control the cluster sizes. Each cluster becomes parameterized for the sake of collecting files for sub-sets in the allocation process. No other variables are needed except relative costs, i.e., $cost(\boldsymbol{S}_{new}) - cost(\boldsymbol{S})$ while the model probes the nodes iteratively with variations of data allocations and descent on the plane of the loss function.

The allocation process can be outlined in the following steps:

- Parameterize and prioritize clusters. Set a seed according to popularity and type variable for each cluster.
- Disjoint the data sets into the number of clusters into subsets to be allocated. This produces the allocation matrix $S$ (Eq. 3). Boundary conditions $\mathbf{w}_{store}$ are needed to control the cluster sizes.
- Calculate costs by Eq. 2 and repeat the process.

The presented model here relies on work management and solves the data allocation problem. The model does not see the actual workload, but the cost that comes with the workload related to the data allocation. The consideration by only costs allows for a more flexible way of not having to expose the underlying grid and jobs.

The evolution of the model comes into the described allocation process. After finding a $S$ which betters the situation, it is placed into an internal pool with the label of the cost that indicates the quality or fitness:

- Populate the pool and delete the bottom solutions
- Vary single solutions
- Combine parameters of two solutions into one novel solution

The model runs on a single PC with parallelization. In-memory data on files and data sets are fixed and can be shared. Each data allocation iteration is done on a worker process, which just communicates the parameters of a solution. Other workers take the work on combining solutions.

## 4 Justification and Evaluation

Data centers/HPC sites (in some jargon, just sites) are referred to here as nodes. The number of nodes in the optimization has been reduced from over 100 to a low 2-digit number, see Fig. 4. Details are listed below. This is a feasible number, and the set can cover the most important ones, say, nodes with the highest capacities. There is an exponential drop-off from the largest to the smallest data centers. At the peak, there are large ones working on the majority of data sets, and at the tail, there are many small ones. Large ones covered in the optimization process imply a high potential for improvement.

For each iteration of the experiment, the following parameters are defined for nodes in such a way that the values are sampled in a uniformly random way:

- the provisioned work performance in terms of average jobs
- the provisioned queues for jobs for two types (A and B)
- the provisioned storage capacity for files

Data is defined in the following way:

- 10 thousand data sets from 100 users are taken. This is used for the validation of the model.

- Each data set possesses 1 to 100 files.
- For training of the similarity model, a subset of the data set user records are taken, a sample size of 2 GB plain text. This record indicates the data set, their files, and the associated user to each data set. This sub-set covers 100 users from the full user records comprising about 25 GB per month. The selected time window of one month is short for picking up short correlations for the demonstration.
- For these data sets, a popularity value is chosen between 0 and 1. Highly active users possess data sets with higher popularity on average, and more frequent data sets are represented with higher popularity. The model can only see 10% of the popularity values.

With the data and the provision on the computer side, the work of the model can be outlined:

- Similarity estimation is done once, and the final similarity values are kept in memory beside the original variables.
- The data allocation part produces a data distribution while observing the total costs according to Eq. 2. Over the course of evolution, the allocation model permutates the output to find a fitter solution. In other words, a solution with a smaller cost.

The improvement by the data allocation can be depicted in the Fig. 4. The model runs three configurations, and in the smallest one, it performs better. This is probably due to the fact that the solution space is smaller. A smaller cardinality makes the model performs iteration quicker, and more vectors in the solution space can be probed. The improvement is shown to be at least 5.5% despite uncertainties in the system that the allocation model cannot control, on the input side, where sparse information comes from the data sets, and second, the work on each node that depends on the jobs and the nodes.

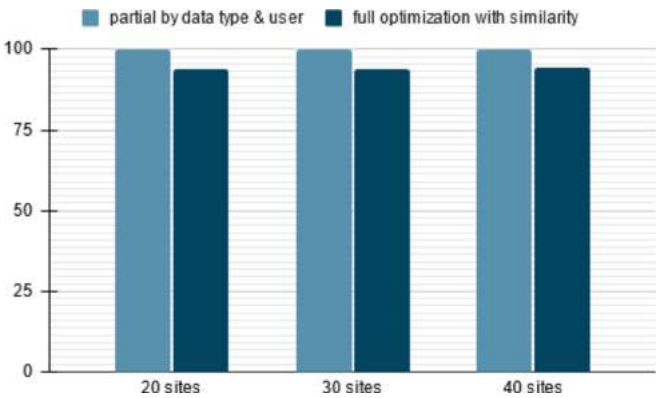



**Fig. 4.** On the left, data sets are allocated by clustering by type and user. The full optimization, on the right, is done with similarity model and the exploitation of popularity. It saves 6%, ranging from almost 6.3% for with 20 sites, and 5.5% points for 40 sites.

## 5 Conclusion and Outlook

Six percent improvement in the most important data centers is a perceivable saving on time and cost in terms of money. This amount corresponds to the provisioned traffic of one to two large data centers to get an idea. Provisioned bandwidth of data centers is often 10 Gb/s. Our experiment shows that savings are possible without making a structural change or manual investigation. In order to contribute to a better solution, more detailed exploitation of data centers, e.g., including different types of storage, compute nodes, etc., for example, is possible in the following way: A second optimization level can be added, and both algorithms will go hand in hand. Our model works on the top level regarding inter-data center traffic. Another model, with the proper cost metric also a variation of this model, will act on the second level, an entity for the internal costs per data center.

The first part of the composed model, the similarity model from Sect. 3.2, can be updated from time to time. This update process would run in the background to update to the next version of this sub-model.

The core model is the allocation model from Sect. 3.3. It can also be set up in such a way that it operates on a subset of the data sets. Say, new data sets shall be placed according to its policy. Free resources must be observed by the model. A flag would be added to distinguish between data sets to be moved and fixed data sets, i.e., data sets allocated/placed recently [21].

The volume of data sets is small compared to the population in the real grid. Anyhow, the outcome of the data allocation can be used in a top-down approach, in which missing data sets of the entire population would be clustered to the initial data sets. So, for example, 1 data set becomes 10, and 10 becomes 100, until all data sets are included. The solution contains clusters with contiguous elements, as with the solution of the '1' data sets with the heuristic variables' similarity and popularity. In other words, the sparse '1' data sets have a high dependency on the nearby data sets of the entire population.

The presented model shows the potential to perform data placement/allocation with high-dimensional data in the background. Further, the loss function allows the model to be more abstract. The model operates solely regarding the loss and only sees the data and the relevant resources; these are the capacities. The complexity of the data processing with all relevant resources hides in the loss.

## References

1. Abdel-Ghaffar, K.A.S., Abbadi, A.E.: Optimal allocation of two-dimensional data (extended abstract). In: Afrati, F., Kolaitis, P. (eds.) ICDT 1997. LNCS, vol. 1186, pp. 409–418. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-62222-5_60
2. Atallah, M.J., Prabhakar, S.: (almost) Optimal parallel block access to range queries. In: Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, pp. 205–215. ACM (2000)

3. Atlas, C., et al.: Atlas computing: technical design report (2005)
4. Beermann, T., et al.: Methods of data popularity evaluation in the atlas experiment at the LHC. In: EPJ Web of Conferences (2021)
5. Bell, D.A.: Difficult data placement problems. Comput. J. **27**(4), 315–320 (1984)
6. Berchtold, S., Böhm, C., Braunmüller, B., Keim, D.A., Kriegel, H.P.: Fast parallel similarity search in multimedia databases. In: Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data. SIGMOD 1997, pp. 1–12. ACM, New York (1997). https://doi.org/10.1145/253260.253263
7. Bonacorsi, D., et al.: Exploiting CMS data popularity to model the evolution of data management for run-2 and beyond. J. Phys. Conf. Ser. **664**, 032003 (2015). IOP Publishing
8. Chang, R.S., Chang, H.P.: A dynamic data replication strategy using access-weights in data grids. J. Supercomput. **45**(3), 277–295 (2008)
9. Chechik, G., Sharma, V., Shalit, U., Bengio, S.: Large scale online learning of image similarity through ranking. J. Mach. Learn. Res. **11**(3) (2010)
10. Collaboration, A., et al.: The atlas experiment at the cern large hadron collider (2008)
11. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. **15**(3), 200–222 (2001)
12. Guo, W., Wang, X.: A data placement strategy based on genetic algorithm in cloud computing platform. In: 2013 10th Web Information System and Application Conference (WISA), pp. 369–372. IEEE (2013)
13. Hoffer, E., Ailon, N.: Deep metric learning using triplet network. In: Feragen, A., Pelillo, M., Loog, M. (eds.) SIMBAD 2015. LNCS, vol. 9370, pp. 84–92. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24261-3_7
14. Liu, Y., Liu, Z., Kettimuthu, R., Rao, N., Chen, Z., Foster, I.: Data transfer between scientific facilities-bottleneck analysis, insights and optimizations. In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), pp. 122–131. IEEE (2019)
15. Megino, F.B., et al.: Implementing data placement strategies for the CMS experiment based on a popularity model. J. Phys. Conf. Ser. **396**, 032047 (2012). IOP Publishing
16. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition (2015)
17. Ram, S., Marsten, R.E.: A model for database allocation incorporating a concurrency control mechanism. IEEE Trans. Knowl. Data Eng. **3**(3), 389–395 (1991)
18. Sato, H., Matsuoka, S., Endo, T.: File clustering based replication algorithm in a grid environment. In: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pp. 204–211. IEEE Computer Society (2009)
19. Sato, H., Matsuoka, S., Endo, T., Maruyama, N.: Access-pattern and bandwidth aware file replication algorithm in a grid environment. In: Proceedings of the 2008 9th IEEE/ACM International Conference on Grid Computing, pp. 250–257. IEEE Computer Society (2008)
20. Spiga, D., Giordano, D., Barreiro Megino, F.H.: Optimizing the usage of multi-petabyte storage resources for LHC experiments. In: Proceedings of the EGI Community Forum 2012/EMI Second Technical Conference (EGICF12-EMITC2), 26–30 March 2012. Munich, Germany (2012). https://pos.sissa.it/162/107/
21. Vamosi, R., Lassnig, M., Schikuta, E.: Data allocation service ADAS for the data rebalancing of atlas. In: EPJ Web of Conferences, vol. 214, p. 06012. EDP Sciences (2019)

22. Wang, J.Y., Jea, K.F.: A near-optimal database allocation for reducing the average waiting time in the grid computing environment. Inf. Sci. **179**(21), 3772–3790 (2009)
23. Wang, J., et al.: Learning fine-grained image similarity with deep ranking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1386–1393 (2014)
24. Weinberger, K.Q., Sha, F., Saul, L.K.: Convex optimizations for distance metric learning and pattern classification [applications corner]. IEEE Sig. Process. Mag. **27**(3), 146–158 (2010)
25. Yuan, D., Yang, Y., Liu, X., Chen, J.: A data placement strategy in scientific cloud workflows. Futur. Gener. Comput. Syst. **26**(8), 1200–1214 (2010)

# Computer Graphics, Image Processing and Artificial Intelligence

# Your Social Circle Affects Your Interests: Social Influence Enhanced Session-Based Recommendation

Yan Chen[1,2], Wanhui Qian[1,2], Dongqin Liu[1,2], Mengdi Zhou[1,2], Yipeng Su[2(✉)], Jizhong Han[2], and Ruixuan Li[2]

[1] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
{chenyan1,qianwanhui,liudongqin,zhoumengdi,suyipeng}@iie.ac.cn
[2] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{hanjizhong,liruixuan}@iie.ac.cn

**Abstract.** Session-based recommendation aims at predicting the next item given a series of historical items a user interacts with in a session. Many works try to make use of social network to achieve a better recommendation performance. However, existing works treat the weights of user edges as the same and thus neglect the differences of social influences among users in a social network, for each user's social circle differs widely. In this work, we try to utilize an explicit way to describe the impact of social influence in recommender system. Specially, we build a heterogeneous graph, which is composed of users and items nodes. We argue that the fewer neighbors users have, the more likely users may be influenced by neighbors, and different neighbors may have various influences on users. Hence weights of user edges are computed to characterize different influences of social circles on users in a recommendation simulation. Moreover, based on the number of followers and PageRank score of each user, we introduce various computing methods for weights of user edges from a comprehensive perspective. Extensive experiments performed on three public datasets demonstrate the effectiveness of our proposed approach.

**Keywords:** Session-based recommendation · Social recommendation · Social influence

## 1 Introduction

In an era of information explosion, the rapid growth of online shopping and services makes it difficult for users to choose what they prefer from innumerable goods and services. As is known to all, different people's interests and preferences are completely different, and in some scenarios, individuals are not very certain about their needs. Driven by the above backgrounds, recommendation system emerges as the times require [4,25,38].


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 549–562, 2022.
https://doi.org/10.1007/978-3-031-08757-8_46

Recommendation system is applied to capture users' interests based on their personal information and historical interactions with the items [12,13]. Further, it predicts the next item to users that they may interact with according to user preferences [3,14,30].

Session-based recommendation (SR) is first introduced to tackle the case that users' personal information (even the user IDs) can not be acquired [8]. A session is defined as a continuous interaction sequence of items in close proximity [23]. User preferences are usually captured by mining sequential transition patterns in a session [18,21]. Recurrent Neural Network (RNN) is proposed to model the sequence dependencies to learn users' preferences in [27]. Domain-Aware Graph Convolutional Network (DA-GCN) [5] builds a graph and applies a graph neural network to gain users' interests.

Fortunately, user IDs can be obtained in many cases, but we can still make use of SR to conduct a recommendation [1]. Even given the same session historical items, various people may interact with different items out of their personalized interests. So customized session-based recommendation can be made for users [6,22,34].

When user IDs are available, users' social network can be acquired as well. It is obvious that user's interest are easily influenced by their friends [17,24,35]. As a result, a better recommendation can be made when considering the preferences of users' friends. Lots of works have made efforts to take advantage of users' social network to get a more accurate recommendation [11,26,37]. However, when it comes to SR, brand new methods should be considered due to the above-mentioned special features (modeling sequential dependencies for recommendation). In social SR, recent work [2] builds a heterogeneous graph, which consists of social network and all historical user behaviors. It learns social-aware user and item representations, and gets a state-of-the-art (SOTA) performance. Although exiting works take the social influence into account, they set the weights of user-user edges as the same value, and do not detail the differences of social influences among users in social network, instead using a model to capture the impact of social influence on user preferences.

In this situation, we argue that the probabilities of users being affected are different among various users in SR and diverse users may have various influences on their followers. Therefore, we analyze social influence for SR in an explicit manner. Specially, we argue that the fewer people one follows, the more likely he/she may be affected by his/her social circle. Moreover, the more influential users are, the larger impact they will impose on their followers. To verify our thought, we propose a social influence enhanced model, which uses Social-aware Efficient Recommender (SERec) [2] as a backbone. The in-degree and PageRank [19] score for each user node in the social network are primarily computed, which reflect the social influences. Then we take the results as weights among connected user nodes in the graph. Finally, the social network obtains the knowledge of social influences, and we can utilize it to get a more accurate preference for each user.

Our contributions are summarized as follows:

1. We incorporate influence degree of each user affecting and being affected into the social network to capture a more accurate user preference in an explicit way.
2. We come up with some simple but effective methods to obtain the influence weights of social network to make our approach practical.
3. Extensive experiments are performed to verify the effectiveness of our proposed method.

In the rest of this paper, related work is introduced in Sect. 2. We detail our method in the following Sect. 3. Last but not least, we conduct extensive experiments in Sect. 4 and sum up our work in Sect. 5.

## 2 Related Works

Since we focus on social session-based recommendation (SR), we first discuss the relative works of SR and then talk about social recommendation.

**Session-Based Recommendation:** Session-based recommendation can be regarded as a sequential modeling task for the reason that a session is composed of a series of user historical interactions with the items. Naturally, RNNs are preferred to model item transition patterns [7,9]. In [8], Hidasi et al. first give a formal definition of session-based recommendation and come up with a multi-layered Gated Recurrent Unit (GRU) model, which is a variant of RNN to capture sequential dependencies in a session. This work is generally treated as a pioneer attempt for SR. Following Hidasi's work, an improved RNN [27] creatively points out a data augmentation technique to improve the performance of RNN. Subsequent works take Convolutional Neural Networks (CNNs) into consideration to model sequential dependencies [28,29]. A Dilated CNN [36] proposes a stack of holed convolutional layers to learn high-level representation from both short- and long-range item dependencies. Furthermore, attentive mechanism is introduced into recommendation to reduce noise item impact and focus on users' main purposes, i.e. Neural Attentive Recommendation Machine (NARM) [15], and Short-Term Attention/Memory Priority model (STAMP) [16]. Recently, Graph Neural Networks (GNNs) have been widely used in a large number of tasks on account of their superior performances besides session-based recommendation [10,20,39]. SR-GNN [33] is a typical GNN work for SR. It builds all session data into graphs, in which items are regarded as nodes and an edge is added if there is a transit between two items. SR-GNN employs a gated neural network to capture complex transitions of items. These methods try to model item dependencies in sessions, but do not take social influence into account.

**Social Recommendation:** As a widely studied research field, social network has been applied for recommendation in depth [31,32]. However, when social session-based recommendation is mentioned, there is not yet much work for the

reason that SR is a relatively new topic, and previous social recommendation methods are not suitable for SR due to their lack of modeling sequence dependencies. Recently Dynamic Graph Recommendation (DGRec) [26] models dynamic user behaviors with a recurrent neural network, and captures context-dependent social influence with a graph-attention neural network. DGRec gets a better recommendation performance, but its inefficiency can not be ignored, because it has to deal with lots of extra sessions to make a recommendation. To solve the efficiency issue that DGRec meets, SERec [2] implements an efficient framework for session-based social recommendation. In detail, SERec precomputes user and item representations from a heterogeneous graph neural network that integrates the knowledge from social network. As a result, it reduces computations during predicting stage. Efficient as it is, SERec just adds users' social network into the interaction graph and sets weights among user nodes as the same value without considering influence differences among various users.

## 3 Modeling Methods

### 3.1 Problem Definition

In session-based recommendation, we define $U = \{u_1, u_2..., u_N\}$ as the set of users, $I = \{i_1, i_2, ...i_M\}$ as the set of items. Each user $u \in U$ generates a set of sessions, $S^u = \{s_1^u, s_2^u, ...s_T^u\}$. For each session, there are a series of items that a user interacts with sorted by timestamp. For example, $s_1^u = \{i_1^u, i_2^u, ...i_L^u\}$, and $L$ is the length of session $s_1^u$. All sessions of users constitute users' historical behaviors dataset $B$. The goal of session-based recommendation is as follows: given a new session of user $u$, $S = \{i_1^u, i_2^u, ...i_n^u\}$, predict $i_{n+1}^u$ for the user $u$ by recommending top- $K$ items ($1 \leq K \leq M$) from all items $I$ that might be interesting to the user $u$.

In addition, in social session-based recommendation, besides users' historical behaviors $B$, a social network can be utilized to improve recommendation. Let $SN = (U, E)$ denote the social network, where $U$ is the nodes of users, $E$ is the set of edges. There is an edge if a social link exits between two users. For example, an edge $(u, v)$ from $u$ to $v$ means that $u$ is followed by $v$, in other words, $v$ follows $u$.

### 3.2 Social Influence Modeling

**Social Session-Based Behaviors Graph Building.** We first construct a heterogeneous graph from all users' behaviors $B$ and social network $SN$. Then we apply a GNN (Graph Neural Network) [37] to learn representations of users and items. The user representations can capture user preferences and more accurate social influences. Item representations can learn useful information from user-item interactions and cross-session item transition dependencies.

In the heterogeneous graph, all the users and items in $B$ and $SN$ make up the graph nodes, and the set of edges consists of four kinds of edges. A user-user edge

$(u, v)$ exists if $v$ follows $u$ (in other words, $u$ is followed by $v$). It is worth noting that we make such a design because in our model a user node representation is learned by the incoming edges and users are more influenced by those they follow than those following them. If a user $u$ has ever clicked an item $i$ in any session, there will be two edges, namely $(u, i)$ and $(i, u)$. Lastly, there is an edge $(i, j)$ if item $i$ transmits directly to item $j$.

Now, we take edge weights into consideration. The weight of user-item $(u, i)$ and item-user $(i, u)$ is the times of user $u$ clicked item $i$. And the weight of item-item $(i, j)$ is the times of item $i$ transmitted to $j$.

When considering weights of user-user $(u, v)$, SERec [2] defines all the weights of user-user as an identical number 1. However, we argue that different weights should be designed among users in the social network, for various influences may have on different users in the social network. As a result, we compute every weight between user-user edge to explicitly represent the social influence.

**Social Influence Computing.** After the heterogeneous graph is constructed, we compute the weights among user nodes in the following method inspired by [40].

Given a user node $v$, the node in-degree $d$ denotes the number of users that $v$ follows. We view all the incoming edges' weights $W$ as the user $v$'s degree of being influenced. In other words, the weight of edge $(u, v)$ is calculated in the following equation:

$$W(u, v) = C/d, \tag{1}$$

where $C$ is a positive constant to control the range of weight.

For example, if a node $v$ follows three users $u_1, u_2, u_3$, then there are three edges $(u_1, v), (u_2, v), (u_3, v)$. So the in-degree of $v$ is 3, and the weights of the three edges are all set to $W = C/3$.

We further make a research on the ability of users to influence their followers. We apply PageRank [19] to calculate the importance of a user in the social network, which is a way of measuring the importance of website pages. Intuitively, the larger of the importance value users get, the more influence users may have on their followers. In detail, the influence of node $u$ is computed as follows:

$$F_u = (1 - A)/N + A * (F_{v_1}/out(v_1) + F_{v_2}/out(v_2) + ... + F_{v_n}/out(v_n)), \tag{2}$$

where $out(v_i)$ denotes the out-degree of node $v_i$. $v_1...v_n$ is the followers of user $u$, $A$ is a coefficient, $N$ is the number of user nodes.

For example, if a node $u$ is followed by three users $v_1, v_2, v_3$, then there are three edges $(u, v_1), (u, v_2), (u, v_3)$. Then weights of the three edges are all set to be $F_u$.

### 3.3 Model Architecture and Training

In this section, we briefly illustrate how to capture user preferences by modeling user sequential patterns. Our model selects SERec as a backbone, and utilizes

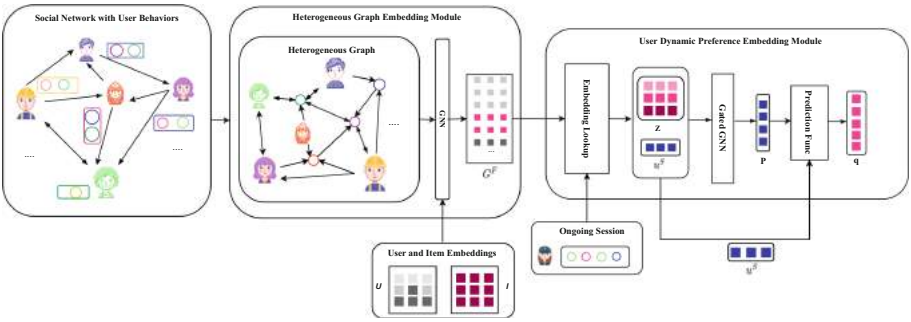



**Fig. 1.** An overview of model architecture. Heterogeneous graph embedding module is applied to learn representations of users and items. User dynamic preference embedding module applies a gated GNN to obtain a user preference embedding in the ongoing session.

a Graph Neural Network (GNN) [37] and gated GNN [33] to capture user preferences. As is demonstrated in Fig. 1, our model is composed of two modules: heterogeneous graph embedding module and user dynamic preference embedding module.

**Heterogeneous Graph Embedding Module.** We apply GNN (graph neural network) to model user and item embedding, which has fused social influence. Supposing GNN is comprised of $F$ layers, let $G^f[x]$ denote the representation of node $x$ at layer $f$, where $x$ may be a user or an item. The new node representation $G^f[x]$ is computed as follows:

$$G^f[x] = ReLU(W_1^f(G^{f-1}[x]||\hat{G}^f[x]) + b_1^f), \tag{3}$$

where $G^{f-1}[x]$ is the old node representation. $\hat{G}^f[x]$ is the aggregated information from node $x$'s neighbors $N(x)$, and $y$ belongs to $N(x)$ if there is an edge $(y, x)$ pointing to $x$. $W_1^f$ and $b_1^f$ are learnable parameters.

The aggregation information from node $x$'s neighbors is calculated by the equation below:

$$\hat{G}^f[x] = \sum_{y \in N(x)} Attention(y, x) * (W_2^f G^{f-1}[y] + b_2^f), \tag{4}$$

where $W_2^f$ and $b_2^f$ are learnable parameters. *Attention* function is detailed in [2].

**User Dynamic Preference Embedding Module.** Given an ongoing session $S$ of user $x$, a graph $G = (V, E)$ is constructed in the same way mentioned in Sect. 3.2, where $V$ denotes items in $S$, $E$ denotes item transitions, and the weight of an edge is the times of item transitions. Since we build a heterogeneous graph on all historical sessions of users and social network, global user and item

representations are obtained. Then for the ongoing session $S$, we first retrieve the relative user and item representations to initialize node representations $Z$. We utilize a gated GNN [33] to model the session-specialized item representation:

$$g_i = v_i \odot tanh(W_c(\hat{N}[z_i]||z_i) + b_c) + (1 - v_i) \odot W_z z_i, \tag{5}$$

$$v_i = sigmod(W_i(\hat{N}[z_i]||z_i) + b_i), \tag{6}$$

where $z_i$ is the node vector of an item in session $S$, $\hat{N}[z_i]$ is the aggregated information from $z_i$'s neighboring nodes, $g_i$ represents the vector of node $i$ for specialized session $S$. $W_c$, $b_c$, $W_z$, $W_i$ and $b_i$ are learnable parameters. Based on $g_i$, we get a user preference embedding in the ongoing session $S$:

$$P = \sum_{1 \le i \le |S|} b_i * g_i, \tag{7}$$

$$b_i = softmax(r^T sigmod(W_v g_i + W_{last} g_{last} + W_u u^S)), \tag{8}$$

where $g_{last}$ is the embedding of the last item in session $S$ to capture user's recent interest, $u^S$ is the user embedding to capture user $u$'s general preference. $r$, $W_v$, $W_{last}$ and $W_u$ are learnable parameters.

Finally, we generate the score $q$ for every item in $I$ via multiplying its embedding $e_i$ by user preference embedding in the ongoing session $S$:

$$q = softmax(P^T e_i). \tag{9}$$

We apply a cross-entropy of the prediction and the ground truth as the loss function in the following form:

$$L = -\sum_{m=1}^{M} q_m log(\hat{q}) + (1 - q_m) log(1 - \hat{q}), \tag{10}$$

where $\hat{q}$ denotes the one-hot encoding vector of the ground truth item.

## 4 Experiments

### 4.1 Experimental Setup

**Datasets.** Following the existing classic social recommendation works [2,26], we evaluate our proposed method on three public real-world benchmark datasets:

(1) **Foursquare:** The Foursquare[1] dataset is a publicly online available dataset which consists of users' check-in records on different venues in a period of time. Records are regarded as the same session if the check-in time interval is shorter than a day, and records are viewed as different sessions if interval is longer than a day.

[1] https://sites.google.com/site/yangdingqi/home/foursquare-dataset.

**Table 1.** Datasets statistics

| Datasets | All users | All items | All clicks | Social links | All sessions |
|---|---|---|---|---|---|
| Foursquare | 39,302 | 45,595 | 3,627,093 | 304,030 | 888,798 |
| Gowalla | 33,661 | 41,229 | 1,218,599 | 283,778 | 258,732 |
| Delicious | 1313 | 5793 | 266,190 | 9130 | 60,397 |

(2) **Gowalla:** Gowalla[2] is another check-in data, and the social network is based on location social network website. Sessions are extracted by the same way as Foursquare.
(3) **Delicious:** Delicious[3] is an online bookmarking system. We consider a sequence of tags a user has marked to a book as a session (marking time is recorded).

We split each dataset into training/validation/test sets, following the settings in [2]. And we conduct our experiments on augmented datasets. To be specific, for a session $S = (v_1, v_2, v_3, ..., v_n)$, we generate a series of interaction sequences and labels $([v_1], v_2), ([v_1, v_2], v_3), ..., ([v_1, v_2, ..., v_{n-1}], v_n)$, where $[v_1, v_2, ..., v_{n-1}]$ is user's historical sequence, $v_n$ is the next-clicked item, namely the label.

The statistics of datasets are summarized in Table 1.

**Evaluation Metrics.** We evaluate all models with three widely used ranking-based metrics:

(1) **REC@K:** It measures the recall of the top-$K$ ranked items in the recommendation list over all the testing instances. In our experiments, only one item is set as the label, so REC@$K$ is used to measure whether the label item is contained in the top-$K$ ranked items according to the scores.
(2) **MRR@K:** It measures the mean reciprocal rank of the predictive position of the true target item on the top-$K$ ranked items in the recommendation list. The target item is expected to rank ahead in terms of ranking scores.
(3) **NDCG@K:** NDCG is a standard ranking metric. In the context of session-based recommendation, it also measures the position of target item in the recommendation list.

$K$ is set to 10 and 20 in our experiments.

**Comparison Methods.** SERec [2] and DGRec [26] are regarded as two typical works related to social session-based recommendation, and SERec as the SOTA social SR model has proved it enjoys a more effective performance than DGRec [2]. Consequently, we only compare our method with SERec. Moreover, our work is realized on the basis of graph structure by taking social influence

[2] https://snap.stanford.edu/data/loc-gowalla.html.
[3] https://grouplens.org/datasets/hetrec-2011/.

explicitly into consideration, we also want to verify if it is effective compared with the existing none-social SR methods like SR-GNN [33], which applies a gated graph convolutional layer to learn item transitions. Last but not least, as mentioned above in Sect. 3.2, we put forward two ways (in-degree and PageRank) on how to compute the social influences, and we test the two different types and combination of both on SERec and SR-GNN to prove our proposed method.

**Implementation Details.** Following the backbone method, we set the model hyper-parameters as mentioned in [2]. Generally, user and item IDs are made an embedding into low dimensional latent spaces with the same dimensionality 128. Adam optimizer was used to train the models and the batch size for mini-batch is 128. The above models' performance are reported under their optimal hyper-parameter settings [2]. Specially, we find that our model gets an optimal performance when C is set to 100 and A to 0.85, so we report our model performance under this optimal settings. To compute PageRank scores of user nodes, we first initialize PageRank score of each user node to $1/N$ ($N$ is the number of user nodes in the social network), then update all user nodes' PageRank scores using Eq. (2) for five times iteratively. Since we filter out user nodes without followers or followees, each user node can obtain a PageRank score in the social network.

### 4.2 Experimental Evaluations

**Weights Calculating for Social Network.** We first compute $W(u, v)$ and $F_v$ for each user in the social network using Eqs. (1) and (2), and set the weights of user-user edges in the following three ways to verify our proposed method:

(1) Using $W(u, v)$ only as the weight for edge $(u, v)$ to demonstrate our idea that the fewer people that users follow, the more influence their social circles may have on their interests;
(2) Using $F_u$ only as the weight of edge $(u, v)$ to prove that the more influential that users are (the PageRank scores are high), the more impact they may have on their followers' behaviors;
(3) Using the sum of $W(u, v)$ and $F_u$ as the weight of edge $(u, v)$ to test the relation between the above two weight computing ways.

**Results Analysis.** As is mentioned above, the explicit social influence is evaluated in three manners, and experiments are implemented based on SERec and SR-GNN to check if our method can achieve a high performance. The experimental results of overall performance are reported in Tables 2, 3 and 4. The optimal and suboptimal results of each column are highlighted in boldface and underline for SERec and SSR-GNN respectively. We denote SSR-GNN for social-aware SR-GNN and denote model with the three weight computing ways by adding a postfix 'W, F, C' respectively. And 'R, M, N' is short for 'REC, MRR, NDCG'. The following observations can be drawn from the results.

**Table 2.** Performance on foursquare

| Model | Foursquare | | | | | |
|---|---|---|---|---|---|---|
| | R@10 | M@10 | N@10 | R@20 | M@20 | N@20 |
| SERec | 61.67 | 34.11 | 40.69 | 70.07 | 34.71 | 42.84 |
| SERecF | 63.15 | 36.83 | 43.11 | 70.86 | 37.37 | 45.11 |
| SERecC | 63.21 | 36.82 | 43.12 | 70.89 | 37.37 | 45.11 |
| SERecW | **63.34** | **36.91** | **43.27** | **70.94** | **37.44** | **45.20** |
| SSRGNN | 60.94 | 33.62 | 40.15 | 69.28 | 34.21 | 42.29 |
| SSRGNNF | 61.02 | 33.56 | 40.13 | 69.53 | 34.16 | 42.29 |
| SSRGNNC | 61.04 | 33.55 | 40.12 | **69.65** | 34.15 | 42.31 |
| SSRGNNW | **61.21** | **33.83** | **40.39** | 69.60 | **34.42** | **42.52** |

**Table 3.** Performance on Gowalla

| Model | Gowalla | | | | | |
|---|---|---|---|---|---|---|
| | R@10 | M@10 | N@10 | R@20 | M@20 | N@20 |
| SERec | 45.88 | 25.26 | 30.06 | 53.48 | 25.78 | 31.93 |
| SERecF | 46.76 | **26.54** | 31.34 | 53.85 | **27.05** | 33.17 |
| SERecC | 46.75 | 26.14 | 31.00 | **54.07** | 26.65 | 32.85 |
| SERecW | **46.87** | 26.53 | **31.49** | 53.92 | 27.03 | **33.27** |
| SSRGNN | 45.32 | 24.81 | 29.68 | 52.94 | 25.33 | 31.59 |
| SSRGNNF | **45.50** | 24.85 | 29.74 | **53.01** | 25.37 | 31.62 |
| SSRGNNC | 45.45 | **25.01** | **29.88** | 52.87 | **25.53** | **31.76** |
| SSRGNNW | 45.29 | 24.76 | 29.65 | 52.79 | 25.28 | 31.53 |

First of all, let us focus on the performance of our method on the SOTA model SERec. In general, our proposed method outperforms SERec. It is proved that the model can learn more accurate user preferences by explicitly adding social influence as the weights of user-user edges. Furthermore, considering users' influence to their followers has a similar improvement on the model with thinking about the degree of users being influenced. To be more specific, SERecW may win a little bit than SERecF. However, to our surprise, simply summing up the first two social influence computing results as the weights does not gain a best performance. One possible reason may be that the two weights computing methods represent different views of social influence, and can not be added on the same dimension.

Secondly, when it comes to SSR-GNN, after applying our proposed method to original SSR-GNN, it can make a progress on recommendation results. Other conclusions are nearly the same as SERec, except that SSR-GNNC achieves the best results on some metrics. This further illustrates the uncertainty of summing up the two different weights for their various practical significance.

**Table 4.** Performance on delicious

| Model | Delicious | | | | | |
|---|---|---|---|---|---|---|
| | R@10 | M@10 | N@10 | R@20 | M@20 | N@20 |
| SERec | **40.22** | 21.22 | 25.70 | **49.50** | 21.87 | 28.05 |
| SERecF | 39.79 | **21.39** | **25.80** | 49.33 | **22.05** | **28.19** |
| SERecC | 40.12 | 21.11 | **25.80** | 48.96 | 21.75 | 28.15 |
| SERecW | 40.15 | 21.31 | 25.64 | 49.18 | 21.98 | 28.09 |
| SSRGNN | 39.73 | **21.55** | **25.85** | 48.78 | **22.18** | **28.13** |
| SSRGNNF | 39.92 | 21.53 | 25.83 | 49.23 | 22.10 | 28.06 |
| SSRGNNC | **40.09** | 21.48 | 25.83 | **49.62** | 22.11 | **28.13** |
| SSRGNNW | 39.77 | 21.35 | 25.68 | 49.33 | 22.01 | 28.10 |

We have to mention that among the three datasets, our method performs better on Foursquare and gowalla, but is not stable on Delicious. Through deep research, we find that the data characters may lead to such a result. Let us review the statistics of the three datasets in Table 1, Foursquare and Gowalla have rich social links, which can help models learn user preferences by considering the social influence in a more explicit way. On the contrary, Delicious has far less social links than the other two, so models learn little extra information from social network by adding social influence among users, even performs worse possibly due to large data variance. All in all, our proposed method can gain a significant performance promotion on large amounts of datasets, but may meet unstableness on less amounts of datasets.

## 5 Conclusion

In this paper, we propose an explicit view to discuss how users' social network influences their behaviors. Based on the sense that the smaller one person's social circle is, the more influence his/her social network may have on his/her interests, and the more influential users are, the more likely that they may affect their followers. To verify our idea, we build a heterogeneous social graph, and explicitly compute the influences as weights in social network graph according to in-degrees and PageRank scores of user nodes. Finally, extensive experiments are conducted on three public datasets. It is demonstrated that modeling social influences in an explicit way can outperform the SOTA model on large datasets while get a degradation on a small dataset. In the future, we will explore the influence of social network formation to session-based recommendation. We consider how a user's social network is formed. For example, they share the same topic. This kind of social network may lead to a more stable influence on users' interest, we would study such influence to session-based recommendation.

**Acknowledgements.** We gratefully thank the reviewers for their insightful comments. This research is supported in part by the National Key Research and Development Program of China under Grant 2018YFC0806900.

## References

1. Chen, T., Wong, R.C.W.: Handling information loss of graph neural networks for session-based recommendation. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1172–1180 (2020)
2. Chen, T., Wong, R.C.W.: An efficient and effective framework for session-based social recommendation. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 400–408 (2021)
3. Chipchagov, M., Kublik, E.: Model of the cold-start recommender system based on the Petri-Markov nets. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12743, pp. 87–91. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77964-1_7
4. Choi, M., Kim, J., Lee, J., Shim, H., Lee, J.: Session-aware linear item-item models for session-based recommendation. In: Proceedings of the Web Conference 2021, pp. 2186–2197 (2021)
5. Guo, L., Tang, L., Chen, T., Zhu, L., Nguyen, Q.V.H., Yin, H.: DA-GCN: a domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation. In: Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence. IJCAI, pp. 2483–2489 (2021)
6. Guo, L., Yin, H., Wang, Q., Chen, T., Zhou, A., Quoc Viet Hung, N.: Streaming session-based recommendation. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1569–1577 (2019)
7. Hidasi, B., Karatzoglou, A.: Recurrent neural networks with top-k gains for session-based recommendations. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp. 843–852 (2018)
8. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. In: 4th International Conference on Learning Representations, ICLR (2016)
9. Hidasi, B., Quadrana, M., Karatzoglou, A., Tikk, D.: Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 241–248 (2016)
10. Hsu, C., Li, C.T.: RetaGNN: relational temporal attentive graph neural networks for holistic sequential recommendation. In: Proceedings of the Web Conference, pp. 2968–2979 (2021)
11. Jiang, B., Lu, Z., Liu, Y., Li, N., Cui, Z.: Social recommendation in heterogeneous evolving relation network. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12137, pp. 554–567. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50371-0_41
12. Kużelewska, U.: Effect of dataset size on efficiency of collaborative filtering recommender systems with multi-clustering as a neighbourhood identification strategy. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12139, pp. 342–354. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50420-5_25

13. Kuzelewska, U.: Quality of recommendations and cold-start problem in recommender systems based on multi-clusters. In: Computational Science - ICCS 2021–21st International Conference, pp. 72–86 (2021)
14. Landin, A., Parapar, J., Barreiro, Á.: Novel and diverse recommendations by leveraging linear models with user and item embeddings. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12036, pp. 215–222. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45442-5_27
15. Li, J., Ren, P., Chen, Z., Ren, Z., Lian, T., Ma, J.: Neural attentive session-based recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 1419–1428 (2017)
16. Liu, Q., Zeng, Y., Mokhosi, R., Zhang, H.: Stamp: short-term attention/memory priority model for session-based recommendation. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1831–1839 (2018)
17. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: Proceedings of the fourth ACM International Conference on Web Search and Data Mining, pp. 287–296 (2011)
18. Meng, W., Yang, D., Xiao, Y.: Incorporating user micro-behaviors and item knowledge into multi-task learning for session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1091–1100 (2020)
19. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank citation ranking: bringing order to the web. Technical report, Stanford InfoLab (1999)
20. Pan, Z., Cai, F., Chen, W., Chen, H., de Rijke, M.: Star graph neural networks for session-based recommendation. In: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, pp. 1195–1204 (2020)
21. Pan, Z., Cai, F., Ling, Y., de Rijke, M.: Rethinking item importance in session-based recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1837–1840 (2020)
22. Quadrana, M., Karatzoglou, A., Hidasi, B., Cremonesi, P.: Personalizing session-based recommendations with hierarchical recurrent neural networks. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 130–137 (2017)
23. Ren, P., Chen, Z., Li, J., Ren, Z., Ma, J., De Rijke, M.: RepeatNet: a repeat aware neural recommendation machine for session-based recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 4806–4813 (2019)
24. Sanz-Cruzado, J., Macdonald, C., Ounis, I., Castells, P.: Axiomatic analysis of contact recommendation methods in social networks: an IR perspective. In: Jose, J.M., et al. (eds.) ECIR 2020. LNCS, vol. 12035, pp. 175–190. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45439-5_12
25. Sato, M., Singh, J., Takemori, S., Zhang, Q.: Causality-aware neighborhood methods for recommender systems. In: Hiemstra, D., Moens, M.-F., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (eds.) ECIR 2021. LNCS, vol. 12656, pp. 603–618. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72113-8_40
26. Song, W., Xiao, Z., Wang, Y., Charlin, L., Zhang, M., Tang, J.: Session-based social recommendation via dynamic graph attention networks. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 555–563 (2019)

27. Tan, Y.K., Xu, X., Liu, Y.: Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, pp. 17–22 (2016)
28. Tuan, T.X., Phuong, T.M.: 3D convolutional networks for session-based recommendation with content features. In: Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 138–146 (2017)
29. Twardowski, B., Zawistowski, P., Zaborowski, S.: Metric learning for session-based recommendations. In: Hiemstra, D., Moens, M.-F., Mothe, J., Perego, R., Potthast, M., Sebastiani, F. (eds.) ECIR 2021. LNCS, vol. 12656, pp. 650–665. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-72113-8_43
30. Wang, S., Cao, L., Wang, Y., Sheng, Q.Z., Orgun, M.A., Lian, D.: A survey on session-based recommender systems. ACM Comput. Surv. (CSUR) **54**(7), 1–38 (2021)
31. Wang, W., Yin, H., Du, X., Hua, W., Li, Y., Nguyen, Q.V.H.: Online user representation learning across heterogeneous social networks. In: Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 545–554 (2019)
32. Wang, X., Hoi, S.C., Ester, M., Bu, J., Chen, C.: Learning personalized preference of strong and weak ties for social recommendation. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1601–1610 (2017)
33. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., Tan, T.: Session-based recommendation with graph neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 346–353 (2019)
34. Wu, S., Zhang, M., Jiang, X., Xu, K., Wang, L.: Personalizing graph neural networks with attention mechanism for session-based recommendation. CoRR abs/1910.08887 (2019)
35. Xiao, L., Min, Z., Yongfeng, Z., Yiqun, L., Shaoping, M.: Learning and transferring social and item visibilities for personalized recommendation. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 337–346 (2017)
36. Yuan, F., Karatzoglou, A., Arapakis, I., Jose, J.M., He, X.: A simple convolutional generative network for next item recommendation. In: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, pp. 582–590 (2019)
37. Zhang, C., Song, D., Huang, C., Swami, A., Chawla, N.V.: Heterogeneous graph neural network. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 793–803 (2019)
38. Zhong, J., Ma, C., Zhou, J., Wang, W.: PDPNN: modeling user personal dynamic preference for next point-of-interest recommendation. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12142, pp. 45–57. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50433-5_4
39. Zhou, H., Tan, Q., Huang, X., Zhou, K., Wang, X.: Temporal augmented graph neural networks for session-based recommendations. In: SIGIR 2021: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1798–1802 (2021)
40. Zhu, Y., Xu, Y., Yu, F., Liu, Q., Wu, S., Wang, L.: Graph contrastive learning with adaptive augmentation. In: Proceedings of the Web Conference 2021, pp. 2069–2080 (2021)

# Action Recognition in Australian Rules Football Through Deep Learning

Stephen Kong Luan(✉), Hongwei Yin, and Richard Sinnott

School of Computing and Information Systems, University of Melbourne, Melbourne, Australia
rsinnott@unimelb.edu.au

**Abstract.** Understanding player's actions and activities in sports is crucial to analyze player and team performance. Within Australian Rules football, such data is typically captured manually by multiple (paid) spectators working for sports data analytics companies. This data is augmented with data from GPS tracking devices in player clothing. This paper focuses on exploring the feasibility of action recognition in Australian rules football through deep learning and use of 3-dimensional Convolutional Neural Networks (3D CNNs). We identify several key actions that players perform: kick, pass, mark and contested mark, as well as non-action events such as images of the crowd or players running with the ball. We explore various state-of-the-art deep learning architectures and developed a custom data set containing over 500 video clips targeted specifically to Australian rules football. We fine-tune a variety of models and achieve a top-1 accuracy of 77.45% using R2+1D ResNet-152. We also consider team and player identification and tracking using You Only Look Once (YOLO) and Simple Online and Realtime Tracking with a deep association metric (DeepSORT) algorithms. To the best of our knowledge, this is the first paper to address the topic of action recognition in Australian rules football.

**Keywords:** Action recognition · 3D CNN · Australian rules football

## 1 Introduction

Action recognition has been explored by many researchers over the past decade. The typical objective is to detect and recognize human actions in a range of environments and scenarios. Action recognition, unlike object detection, needs to consider both spatial and temporal information in order to make classifications. In this paper we focus on using 3-dimensional Convolutional Neural Networks (3D CNNs) to achieve action recognition for players in Australian rules football.

Australian rules football, commonly referred to as "footy" in Australia, is a popular contact sport played between two 18-player teams on a large oval. The premier league is the Australian Football League (AFL). The ultimate aim is to kick the ball between 4 goal posts for a score (6 points if the ball goes through


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 563–576, 2022.
https://doi.org/10.1007/978-3-031-08757-8_47

the middle two posts) or a minor score (1 point if the ball goes through the one of the inner/outer posts). This is achieved by players doing a range of actions to move the ball across the pitch. These include kicking, passing (punching the ball), catching, running (up to 15 m whilst carrying the ball) and tackling.

The understanding of player actions and player movements in sports are crucial to analyse player and team performances. Counting the number of effective actions that take place during a match is key to this. This paper focuses on development of a machine learning application that is able to detect and recognize player actions through the use of deep artificial neural networks.

## 2 Literature Review

Prior to deep learning, approaches based on hand-engineered features for computer vision tasks were the primary method used for action recognition. Improved Dense Trajectories (IDT) [26] is representative of such approaches. This achieved good accuracy and robustness, however hand engineering features is limited. Deep learning architectures based on CNNs have achieved unparalleled performance in the field of computer vision. Deep Video developed by Karpathy et al. [17] was one of the first approaches to apply 2D CNNs for action recognition tasks. This used pre-trained 2D CNNs applied to every frame of the video and fusion techniques to learn spatio-temporal relationships. However, its performance on the UCF-101 data set [20] was worse than IDT, indicating that 2D CNNs alone are sub-optimal for action recognition tasks since they do not adequately capture spatio-temporal information.

Two-stream networks such as [19] add a stream of optical flow information [11] as a representation of motion besides the conventional RGB stream. The approach used two parallel streams that were combined with fusion based techniques. This approach was based on 2D CNNs and achieved similar results to IDT. This approach sparked a series of research efforts focused on improving two-stream networks. This included works focused on improvement in fusion [6], and use of recurrent neural networks including Long Short-Term Memory (LSTM) [4,15]. Other methods include Temporal Segment Networks (TSN) [27] capable of understanding long range video content by splitting a video into consecutive temporal segments, and multi-stream networks that consider other contextual information such as human poses, objects and audio in video. The framework of two-stream networks was widely adopted by many researchers, however, a major limitation of two-stream networks was that optical flows require pre-processing and hence require considerable hand-engineering of features. Generating optical flows for videos can be both computationally and storage demanding. This also affected the scale of training data sets required.

3D CNNs can be thought of as a natural way to understand video content. Since video is a series of consecutive frames of images, a 3-dimensional convolutional filter can be applied to both the spatial and temporal domain. Initial research was explored by [13] in 2012, then in 2015 by Tran et al. [22] who proposed a 3D neural network architecture called C3D using $3 \times 3 \times 3$ convolutional kernels. They demonstrated that 3D CNNs were better at learning

spatio-temporal features than 2D CNNs. The introduction of C3D marked the start of a new chapter in action recognition. 3D CNNs were shown to be suited to extracting and learning spatio-temporal features from video - a core demand for real-time action recognition. However C3D were difficult to train with the training process usually taking weeks on large data sets, due to the cost incurred in training with an overwhelming number of parameters in the full 3D architecture.

In 2017, Carreira et al. [2] proposed Inflated 3D ConvNets (I3D), which utilized transfer learning and outperformed all other models using the UCF-101 data set. I3D avoided the necessity for training from scratch by using some well-developed 2D CNN architectures that were pre-trained on large scale data sets such as the ImageNet [3]. I3D added an additional temporal dimension, where the model weights were used. The proposed I3D model was implemented for both the two-stream and single stream approach. Weights from an Inception-V1 model [12] pre-trained on ImageNet were used and trained on the Kinetics-400 data set [18]. This was subsequently fine-tuned on the UCF-101 data set to achieve a top-1 accuracy of 95.1% with RGB stream only. I3D demonstrated that 3D CNNs could benefit from the weights of 2D CNNs pre-trained on large scale data. This has since become a popular strategy adopted by many that has sparked a model benchmark standard based on the Kinetics-400 data.

Tran et al. [24] proposed the R2+1D architecture in 2018. This focused on factorizing spatio-temporal 3D convolutions into 2D spatial convolutional blocks and 1D temporal convolutional blocks. This decomposition provided simplicity for model optimization and improved the efficiency of training, while also enhancing the model's ability to represent complex functions by increasing the number of non-linearities through adding Rectified Linear Unit activation functions (ReLU) between the 2D and 1D blocks. The R2+1D model used the Deep Residual Network (ResNet) [10] architecture as the backbone and achieved similar performances to I3D on data sets such as Kinetics-400 and UCF-101.

Non-local blocks proposed by Wang et al. in 2018 [28] introduced a new form of operational building block that was able to capture long range temporal features similar to the self attention mechanism [25]. This was compatible to most architectures with minimal effort. The authors implemented their model by adding non-local blocks into the I3D architecture and achieved consistent improvement of performance over the original model using several data sets. In 2019 Tran et al. [23] proposed Channel Separated Networks (CSN) which focused on factorizing 3D CNNs by separating channel-wide interactions and spatio-temporal interactions by introducing regularization measures into the architecture to improve the overall accuracy. CSN are regarded as an efficient and lightweight architecture, where the model interaction-reduced channel-separated network (ir-CSN) using a ResNet-152 backbone reported a top-1 accuracy of 79.2% on the Kinetics-400 data set.

Feichtenhofer et al. [5] proposed the SlowFast networks framework. This consisted of a fast and a slow stream. The fast stream was used for extracting temporal motion features at a high frame rate, whilst the slow stream was used for extracting spatial features at a low frame rate. These two streams were later fused

by lateral connections, commonly seen in two-stream network models. However, the architecture of SlowFast networks was fundamentally different to two-stream networks since it was based on streams of different temporal frame rates and not two separate streams of spatial and temporal features. The SlowFast network provided a generic and efficient framework that could be applied to various spatio-temporal architectures. Furthermore, the fast stream was lightweight as the channel capacity was greatly reduced by only focusing on temporal features. The proposed network used ResNet architecture as the backbone and achieved a better performance than I3D and R2+1D on the Kinetics-400 data set.

Another similar framework was the Temporal Pyramid Network (TPN) proposed by Yang et al. [30]. This used a pyramid structure for processing frames at multiple feature levels to capture the variation in speed for different actions - so called visual tempos. TPN had the ability to use various 3D or 2D architectures as the backbone, where the set of hierarchical features extracted by the backbone undergoes down-sampling with a spatial module and a temporal rate module for processing features rich in both visual tempos and spatial information. These could then be aggregated by an information flow process. TPN used a ResNet-101 backbone and achieved better performance in Kinetics-400 over the SlowFast network.

## 3 Australian Rules Football Data Set

A well-defined and high-quality data set is crucial for action recognition tasks. This should contain enough samples for deep neural networks to extract motion patterns, and offer enough variance for different scenarios and camera positions for performance analysis. No such data set exists for AFL, hence we construct our own action recognition data set for AFL games. In this process, we referred to some well-known data sets for video content understanding including Youtube-8M [1], UCF 101 [20], Kinetics-400 [18], SoccerNet [8] and others. All the training and testing videos used here were retrieved from YouTube.

As AFL games are popular in Australia, there are more than enough videos on YouTube, including real match recordings, training session recordings, tutorial guides etc. However, manually creating and labelling data from video content (individual frames) is a challenging and time-consuming task. In order to feed enough frames and information for temporal feature extraction into deep learning models, we set the standard that each video clip should be at least 16 frames in length and it should be not a long-distance shot with low resolution of action tasks.

Players in an AFL match are highly mobile hence actions only exist for a very limited amount of time and are often interfered with by other players through tackles. As a result, actions sometime may end up in failure. This brings significant challenges to the construction process of the data set, e.g. judging the actual completeness of actions. This work focuses on recognizing the patterns and features of attempted actions, and pays less attention to whether the action has been completed or not. All action clips within the data set have a high level

of observable features, where the actual completeness of those actions was less of a concern.

In AFL games, some actions like marks (catching the ball kicked by a player on the same team) have a specific condition that needs to be met. According to AFL rules, a mark is only valid when a player takes control of the ball for a sufficient amount of time, in which the ball has been kicked from at least 15 m away and does not touch the ground and has not been touched by another player. We aim to identify specific action patterns based only on the camera images and as such we do not consider the precision of whether the kicker was 15 m away. Marks can be separated into marks and contested marks, where the latter is when multiple players attempt to catch (or knock the ball away) at the same time.

The videos from YouTube comprise many meaningless frames. We clip videos from longer videos and label them into five different classes:

**(1) Kick:** This class refers to the action whereby a player kicks the ball. The ball could come from various sources: the player himself holding the ball in front and dropping/kicking it, or kicking it directly off the ground.

**(2) Mark:** A player catches a kicked ball for sufficient time to be judged to be in control of the ball and without the ball being touched/interfered with by another player.

**(3) Contested mark:** Contested mark, is a special form of mark. This refers to the action that one player is trying to catch the ball and one or more opponents are either also trying to catch the ball at the same time or they are trying to punch the ball away.

**(4) Pass:** A player passes (punches) the ball to another player in the same team.

**(5) Non-Action:** This class includes players running, crowds cheering etc. This class is used to control the model performance as during the match there are many non-action frames. Without this class, the model would always try to classify video content into the previous four classes.

The details of each class in the data set are shown in Table 1, and example of each action class is shown in Fig. 1. Compared to other classes, the non-action class has a relatively low number of instances in the data set. The reason is that this class spans many different scenes, and too many instances in this class would drive the attention of the model away from key features of the four key action classes.

There are several challenges when using a data set for action recognition. Some actions share the same proportion of representations. One example is marking and passing the ball. In a video clip of relatively long distance passing, if the camera does not capture the whole passing process, e.g. it starts from somewhere in the middle, the representing features of this action might be similar to a mark action, i.e. someone catches the ball. The data set could also be modified by combing two classes of mark and contested mark, as sometimes it is hard to identify a mark compared to a contested mark. If a player is trying to catch the ball, and in the background an opponent is also trying to catch the ball, but

**Table 1.** Number of instances of each class

| Class | # of instances | | |
|---|---|---|---|
| | Training | Testing | Total |
| Kick | 158 | 20 | 178 |
| Contested mark | 94 | 20 | 114 |
| Mark | 61 | 20 | 81 |
| Pass | 83 | 21 | 104 |
| Non-action | 66 | 21 | 87 |
| **Total** | **462** | **102** | **564** |

**Fig. 1.** Kick, contested mark, mark, completed pass

they do have not any physical contact at any time from one angle it may be considered as a mark. From a different camera angle, where there appears to be some degree of physical contact, it might seem more like a contested mark.

## 4 Implementation and Discussion of Results

Given the complexity and diversity of the architectures mentioned above, we use the Gluon CV toolkit [9]. This provides a Pytorch model implementation, and importantly, the ability to train custom data sets. In order to fully utilize the benefit of transfer learning and to compensate for the limited amount of data, we used models pre-trained on largely scaled action recognition data sets such as the Kinetics-400, and then fine-tune those models using the custom AFL data set. The final implementation involves a slightly modified version of Gluon CV which includes a few algorithmic alterations and some minor bug fixes. The architectures and pre-trained models we used along with their specifications and top-1 accuracy on Kinetics-400 are listed below in Table 2 [31]. Here R2+1D ResNet-50 model was calculated using a $112 \times 112 \times 3 \times 16$ input data size, R2+1D ResNet-152 model was calculated using a $112 \times 112 \times 3 \times 32$ input data size, and all other models were calculated based on a $224 \times 224 \times 3 \times 32$ input data size.

**Table 2.** Model specifications

| Model | Pre-trained | #Mil parameters | GFLOPS | Accuracy (%) |
|---|---|---|---|---|
| I3D ResNet-50 | ImageNet | 28.863 | 33.275 | 74.87 |
| I3D ResNet-101 Non-Local | ImageNet | 61.780 | 66.326 | 75.81 |
| I3D SlowFast ResNet-101 | ImageNet | 60.359 | 342.696 | 78.57 |
| R2+1D ResNet-50 | – | 53.950 | 65.543 | 74.92 |
| SlowFast-8x8 ResNet-101 | – | 62.827 | 96.794 | 76.95 |
| TPN ResNet-101 | – | 99.705 | 374.048 | 79.70 |
| R2+1D ResNet-152* [7] | IG65M | 118.227 | 252.900 | 81.34 |
| irCSN ResNet-152* [7] | IG65M | 29.704 | 74.758 | **83.18** |

All model architectures are in 3D. I3D and I3D SlowFast models were based on inflated 2D ResNet pre-trained on ImageNet. irCSN and R2+1D ResNet-152 were pre-trained on IG-65M, and all other models were trained from scratch. All models used the Kinetics-400 data set for training [9].

The final training dataset was randomly split into training and validation data sets in the ratio of 70% and 30% respectively. A sub-clip of 16 frames was evenly sampled from each video clip at a regular interval depending on the clip's length. The number of input frames was selected as most actions happen in a short time period. If the sampled frames were less than 16, replacements would be randomly selected from the rest of the frames. The sampled frames would then be processed by standard data augmentation techniques, where it would

be first resized to a resolution of $340 \times 256$, while R2+1D resized the frames to $171 \times 128$. The frames were then subject to a random resize with bi-linear interpolation and a random crop size $224 \times 224$. The crop size for R2+1D was $112 \times 112$. Following this, the frames were randomly flipped along the horizontal axis with a probability of 0.5, and normalized with means of $(0.485, 0.456, 0406)$ and standard deviations of $(0.229, 0.224, 0.225)$ with respect to each channel.

The training process used stochastic gradient descent (SGD) as the optimizer, with custom values of learning rate, momentum and weight decay, which were specific to each model. The value of learning rate plays a very important role in the model training process, where the correct learning rates will allow the algorithms to converge, whereas the wrong learning rates will result in the model not generalizing at all. Since we fine-tune pre-trained models, the initial learning rate was set much lower than the original model. The common values of the learning rate were 0.01 and 0.001, with a momentum of 0.9, a weight decay of $1e^{-5}$, and learning rate policy set to either step or cosine, depending on each model's architecture and level of complexity. Cross entropy loss was used for the model criterion with class weights taken into consideration since the training data set was imbalanced between the different classes. The number of epochs was set at 30 with an early stopping technique used to prevent over-fitting. The epoch with the lowest validation loss was saved as the best weight.

The top-1 accuracy on the testing data set for the fine-tuned models is shown in Table 3.

**Table 3.** Top-1 accuracy on the AFL test data set

| Model | Accuracy (%) |
|---|---|
| I3D ResNet-50 | 56.86 |
| I3D ResNet-101 Non-local | 61.77 |
| SlowFast-8x8 ResNet-101 | 69.61 |
| TPN ResNet-101 | 70.59 |
| I3D SlowFast ResNet-101 | 71.57 |
| R2+1D ResNet-50 | 72.55 |
| irCSN ResNet-152 | 74.51 |
| R2+1D ResNet-152 | **77.45** |

As seen, the best performing model was the R2+1D ResNet-152 model pre-trained on the (very large) IG65M dataset. This achieved a top-1 accuracy of 77.45%. The final classification of action recognition results are shown in Table 4. As seen, the classification for marks had the lowest recall of 0.55, while contested marks had a recall of 0.85. This is possibly due to marks and contested marks being difficult to distinguish in some circumstances due to the presence of other players in the background. The classification for non-action has the lowest precision of 0.57 and the lowest f1 score of 0.65. The reason for this is that the

non-action class is very broad and contains many sub-classes, such as scenes of audiences and players running and cheering. Splitting the class into multiple distinct classes in the future may improve the non-action accuracy. Among all classes, the classification of kicks has the highest f1-score at 0.89, since a kick has arguably the most distinct and recognizable features.

**Table 4.** Final classification results

| Action | Precision | Recall | F1-score |
|---|---|---|---|
| Kick | 1.00 | 0.80 | 0.89 |
| Contested mark | 0.74 | 0.85 | 0.79 |
| Mark | 0.85 | 0.55 | 0.67 |
| Pass | 0.86 | 0.90 | 0.88 |
| Non-action | 0.57 | 0.76 | 0.65 |

The results for the top-1 accuracy of the AFL testing data set are generally consistent with the model performance using the Kinetics-400 dataset, however the R2+1D ResNet-50 model achieved some noteworthy improvements. The model I3D ResNet-50 performed poorly with a top-1 accuracy of 56.86%, whilst the model I3D ResNet-101 Non-Local only achieved an accuracy of 61.77%. It might be inferred that the inflated 2D ResNets (I3D) are limited in their ability to capture spatio-temporal features, while R2+1D is more capable in this regard as it utilizes the factorization of the 3D ResNet architecture. It was also found that non-local blocks may not be suitable for Australian rules football, as they are designed to capture long range temporal features. Actions in AFL are relatively fast and diverse which results in the model under-performing.

It was found that the performance of models generally depends on their backbone architecture. The complexity of the ResNet architecture is closely related to the prediction accuracy, hence it could be argued that the more complex the architecture is, the more likely the model will generalize and make the right predictions. Comparing ResNet-50 with ResNet-152, there is a significant difference in complexity and number of parameters, which could be one reason for the relatively large performance difference. Another major factor to consider is that both R2+1D ResNet-152 and irCSN used IG65M for model pre-training and hence benefit from the very largely scaled data set. It is also interesting to note that R2+1D uses a $112 \times 112$ resolution input after data augmentation, whilst the rest of the models use a $224 \times 224$ input. Despite this R2+1D is still able to produce some of the best results overall.

SlowFast and TPN networks both model visual tempos in video clips. When incorporating I3D into SlowFast network, the model I3D SlowFast ResNet-101 performed evidently better than the other I3D models, indicating that the SlowFast networks are capable at better extracting spatio-temporal features and that modelling visual tempos improves the overall model performance. However, SlowFast is a more strict framework that limits the number of frames of different

streams, whilst TPN is more flexible due to its pyramid structure. As a result, TPN ResNet-101 performed slightly better than SlowFast ResNet-101.

There are several important limitations to the presented models. Firstly, incomplete actions will likely be classified as actions. As shown in Fig. 2(a), an incomplete contested mark has been classified as a contested mark. This is due to the incomplete action sharing a lot of similar features to a completed action. The model does not always possess the ability to recognise whether the ball has been cleanly caught (or not). Secondly, the model tends to perform poorly in complex scenes and environments. From Fig. 2(b), it can be seen that there are many players present in the background and a player is tackling another player who has the ball. In this case, the model mis-classifies the scenario into a pass as it is similar to the scenarios of pass in the training data set.


**(a)** Incomplete contested mark **(b)** Realistic complex environment

**Fig. 2.** Mis-classified actions

## 5 Team Identification and Associated Limitations

Many action events depend on distinguishing teams, e.g. a completed pass requires the ball to be passed by a player within the same team. Team identification is thus important to any Australian football model. In this work, we utilize the You-Only-Look-Once (YOLO) v5 [14] framework and the DeepSORT algorithm [29] to identify and track multiple objects at the same time. The implementation of this module inputs raw frames to be classified, filters and then keeps player location information in each frame. The DeepSORT algorithm is capable of tracking object movement across different frames, and assign unique IDs to team players.

As with many team sports, AFL players wear team jerseys with colors representing their team. In this way, audiences are able to identify (distinguish) players from the two teams. We apply color distribution extractors to images to extract the differences in player jersey colors. The distribution can then be used as an input to construct a high-dimensional features such as KMeans clustering [16] to cluster players into groups. A screenshot of the results of the application is shown in Fig. 3.


**Fig. 3.** Team identification classification example

The performance of this module is limited by the resolution of the video. With a higher resolution, the jersey color of players in the foreground is clear but those in the far background is less clear. Another challenge faced is rapid camera movement and viewpoint changes. In real matches, sudden viewpoint changes from long-distance views to close-up views (and vice versa) happens continually. Ideally (from the model perspective) there would be a single camera angle - akin to what a spectator sees in a game, but this never happens in reality when games are shown on television. These continuous viewpoint changes make it challenging to track a specific player's movement. Nevertheless, the team identification is able to distinguish the teams within a few milliseconds. The performance of the system also greatly depends on teams wearing clearly identifiable jerseys. This is always the case however so does not limit the model. If players get especially muddy for example this might be an issue, but this is a rarity in Australia.

## 6 Conclusions and Future Work

This paper explored the feasibility of action recognition for Australian rules football using 3D CNN architectures. Various action recognition models including state-of-the-art models pre-trained on large-scale data sets were utilised. We fine-tune those models on a newly developed AFL data set, and reported a 77.45% top-1 accuracy for the best performing model R2+1D ResNet-152. A smoothing strategy allowed the algorithm to localize the frame range for actions in long video segments. We also developed a team identification solution and an action recognition application that showed both the potential and viability of applying real time end-to-end action recognition to AFL matches.

There are many future extensions to the work. The team identification framework opens up further improvements on action recognition in AFL matches for specific teams. Actions such as pass and contested mark require additional

team information in order to be classified correctly. Moreover, the use of attention mechanisms in machine learning and use of transformers such as Bidirectional Encoder Representations from Transformers (BERT) [21] has the ability to model contextual information with mechanisms for self attention. This could be useful in scenes that contain multiple players and allow to achieve a higher prediction accuracy.

Examples of the application of the models and the source code are available at: https://youtu.be/I7490fyuiK8 and https://github.com/stephenkl/Research-project respectively. This research was undertaken using the LIEF HPC-GPGPU Facility hosted at the University of Melbourne. This Facility was established with the assistance of LIEF Grant LE170100200.

## References

1. Abu-El-Haija, S., et al.: YouTube-8M: a large-scale video classification benchmark. CoRR abs/1609.08675. arXiv: 1609.08675 (2016)
2. Carreira, J., Zisserman, A.: Quo Vadis, action recognition? A new model and the kinetics dataset. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4724–4733. IEEE, Honolulu (July 2017). https://doi.org/10.1109/CVPR.2017.502, http://ieeexplore.ieee.org/document/8099985/
3. Deng, J., Dong, W., Socher, R., Li, L.J., Kai Li, Li Fei-Fei: ImageNet: a large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. IEEE, Miami (June 2009). https://doi.org/10.1109/CVPR.2009.5206848, https://ieeexplore.ieee.org/document/5206848/
4. Donahue, J., et al.: Long-term recurrent convolutional networks for visual recognition and description. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2625–2634. IEEE, Boston (June 2015). https://doi.org/10.1109/CVPR.2015.7298878, http://ieeexplore.ieee.org/document/7298878/
5. Feichtenhofer, C., Fan, H., Malik, J., He, K.: SlowFast networks for video recognition. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 6201–6210. IEEE, Seoul (October 2019). https://doi.org/10.1109/ICCV.2019.00630, https://ieeexplore.ieee.org/document/9008780/
6. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1933–1941. IEEE, Las Vegas (June 2016). https://doi.org/10.1109/CVPR.2016.213, http://ieeexplore.ieee.org/document/7780582/
7. Ghadiyaram, D., Tran, D., Mahajan, D.: Large-scale weakly-supervised pre-training for video action recognition. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12038–12047. IEEE, Long Beach (June 2019). https://doi.org/10.1109/CVPR.2019.01232, https://ieeexplore.ieee.org/document/8953267/
8. Giancola, S., Amine, M., Dghaily, T., Ghanem, B.: SoccerNet: a scalable dataset for action spotting in soccer videos. CoRR abs/1804.04527. arXiv: 1804.04527 (2018)
9. Guo, J., et al.: GluonCV and GluonNLP: deep learning in computer vision and natural language processing. arXiv:1907.04433 (February 2020)

10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778. IEEE, Las Vegas (June 2016). https://doi.org/10.1109/CVPR.2016.90, http://ieeexplore.ieee.org/document/7780459/
11. Horn, B.K., Schunck, B.G.: Determining optical flow. Artif. Intell. **17**(1–3), 185–203 (1981). https://doi.org/10.1016/0004-3702(81)90024-2, https://linkinghub.elsevier.com/retrieve/pii/0004370281900242
12. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning, ICML 2015, JMLR.org, Lille, France, vol. 37, pp. 448–456 (July 2015)
13. Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for human action recognition. IEEE Trans. Pattern Anal. Mach. Intell. **35**(1), 221–231 (2013). https://doi.org/10.1109/TPAMI.2012.59
14. Jocher, G., Stoken, A., Chaurasia, A., Borovec, J.: NanoCode012, Taoxie, Kwon, Y., Michael, K., Changyu, L., Fang, J., V, A., Laughing, tkianai, yxNONG, Skalski, P., Hogan, A., Nadar, J., imyhxy, Mammana, L., AlexWang1900, Fati, C., Montes, D., Hajek, J., Diaconu, L., Minh, M.T., Marc, albinxavi, fatih, oleg, wanghaoyang0106: ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support (October 2021). https://doi.org/10.5281/zenodo.5563715
15. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4694–4702. IEEE, Boston (June 2015). https://doi.org/10.1109/CVPR.2015.7299101, http://ieeexplore.ieee.org/document/7299101/
16. Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., Wu, A.: An efficient k-means clustering algorithm: analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002). https://doi.org/10.1109/TPAMI.2002.1017616
17. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732. IEEE, Columbus (June 2014). https://doi.org/10.1109/CVPR.2014.223, https://ieeexplore.ieee.org/document/6909619
18. Kay, W., et al.: The Kinetics Human Action Video Dataset. arXiv:1705.06950 (May 2017)
19. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Proceedings of the 27th International Conference on Neural Information Processing Systems, NIPS 2014, vol. 1, pp. 568–576. MIT Press, Montreal (December 2014)
20. Soomro, K., Zamir, A.R., Shah, M.: UCF101: a dataset of 101 human actions classes from videos in the wild. arXiv:1212.0402 (December 2012)
21. Su, W., et al.: VL-BERT: pre-training of generic visual-linguistic representations. CoRR abs/1908.08530. arXiv: 1908.08530 (2019)
22. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 4489–4497. IEEE, Santiago (December 2015). https://doi.org/10.1109/ICCV.2015.510, http://ieeexplore.ieee.org/document/7410867/

23. Tran, D., Wang, H., Feiszli, M., Torresani, L.: video classification with channel-separated convolutional networks. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 5551–5560. IEEE, Seoul (October 2019). https://doi.org/10.1109/ICCV.2019.00565, https://ieeexplore.ieee.org/document/9008828/
24. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6450–6459 (June 2018). https://doi.org/10.1109/CVPR.2018.00675, iSSN: 2575-7075
25. Vaswani, A., et al.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, pp. 6000–6010. Curran Associates Inc., Long Beach (December 2017)
26. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: 2013 IEEE International Conference on Computer Vision, pp. 3551–3558 (December 2013). https://doi.org/10.1109/ICCV.2013.441, iSSN: 2380-7504
27. Wang, L., et al.: Temporal segment networks: towards good practices for deep action recognition. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016, Part VIII. LNCS, vol. 9912, pp. 20–36. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_2
28. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7794–7803. IEEE, Salt Lake City (June 2018). https://doi.org/10.1109/CVPR.2018.00813, https://ieeexplore.ieee.org/document/8578911/
29. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International Conference on Image Processing (ICIP), pp. 3645–3649. IEEE, Beijing (September 2017). https://doi.org/10.1109/ICIP.2017.8296962, http://ieeexplore.ieee.org/document/8296962/
30. Yang, C., Xu, Y., Shi, J., Dai, B., Zhou, B.: Temporal pyramid network for action recognition. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 588–597. IEEE, Seattle (June 2020). https://doi.org/10.1109/CVPR42600.2020.00067, https://ieeexplore.ieee.org/document/9157586/
31. Zhu, Y., et al.: A comprehensive study of deep video action recognition. arXiv:2012.06567 (December 2020)

# SEGP: Stance-Emotion Joint Data Augmentation with Gradual Prompt-Tuning for Stance Detection

Junlin Wang[1,2], Yan Zhou[1(✉)], Yaxin Liu[1,2], Weibo Zhang[1,2], and Songlin Hu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{wangjunlin,zhouyan,liuyaxin,zhangweibo,husonglin}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

**Abstract.** Stance detection is an important task in opinion mining, which aims to determine whether the author of a text is in favor of, against, or neutral towards a specific target. By now, the scarcity of annotations is one of the remaining problems in stance detection. In this paper, we propose a Stance-Emotion joint Data Augmentation with Gradual Prompt-tuning (SEGP) model to address this problem. In order to generate more training samples, we propose an auxiliary sentence based Stance-Emotion joint Data Augmentation (SEDA) method, formulate data augmentation as a conditional masked language modeling task. We leverage different relations between stance and emotion to construct auxiliary sentences. SEDA generates augmented samples by predicting the masked words conditioned on both their context and auxiliary sentences. Furthermore, we propose a Gradual Prompt-tuning method to make better use of the augmented samples, which is a combination of prompt-tuning and curriculum learning. Specifically, the model starts by training on only original samples, then adds augmented samples as training progresses. Experimental results show that SEGP significantly outperforms the state-of-the-art approaches.

**Keywords:** Stance detection · Data augmentation · Curriculum learning

## 1 Introduction

The goal of stance detection is to classify a piece of text as either being in support, opposition, or neutrality towards a given target, the target may not be directly contained in the text. With the rapid development of social media, more and more people post online to express their support or opposition towards various targets. Stance detection is known to have several practical application areas such as polling, public health surveillance, fake news detection, and so on. These conditions motivate a large number of studies to focus on inferring


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 577–590, 2022.
https://doi.org/10.1007/978-3-031-08757-8_48

**Table 1.** Examples of stance detection task.

| Text | Stance |
|---|---|
| Wearing a mask is common sense and kind to your fellow human. We all have to do our part to slow the spread of COVID-19 | Favor |
| Spend the day outside, get some sun and fresh air. Without a face mask. Best way to keep up your immune system | Against |
| Any skincare suggestions for breakouts because of face masks? | Neutral |

the stances of users from their posts. Table 1 shows some examples on target "Wearing a Face Mask", annotated with the stance labels.

One of the biggest challenges in stance detection task is the scarcity of annotated samples. Data augmentation is commonly used to address data scarcity, which aims to generate augmented samples based on limited annotations. Zhang et al. [37] replace words with WordNet [19] synonyms to get augmented sentences. Wei et al. [33] propose EDA, which is a combination of token-level augmentation approaches. These methods are effective, but the replacement strategies are simple, thus can only generate limited diversified patterns. To enhance the consistency between augmented samples and labels, Wu et al. [35] propose CBERT, the segmentation embeddings of BERT [11] are replaced with the annotated labels during augmentation. However, these methods fail to take targets into consideration. To solve this problem, Li et al. [16] propose ASDA, which uses the conditional masked language modeling (C-MLM) task to generate augmented samples under target and stance conditions.

Although ASDA [16] achieves highly competitive performance, there still exist two limitations. First, they neglect the emotional information during augmentation. It should be noted that emotion can affect the judgment of stance. There exists a number of studies that use emotional information to assist stance detection and achieve good results [6,14,20]. Thus, we posit that in addition to stance and target information, the introduction of emotional information through auxiliary sentences can further improve the label consistency of augmented samples. Second, they neglect the linguistic adversity problem [17,31] during training. This problem is introduced by data augmentation method and therefore can be seen as a form of noising, where noised data is harder to learn from than unmodified original data.

In this paper, we propose a Stance-Emotion joint Data Augmentation with Gradual Prompt-tuning (SEGP) model to address the above limitations. Specifically, we present an auxiliary sentence based Stance-Emotion joint Data Augmentation (SEDA) method that generates target-relevant and stance-emotion-consistent samples based on C-MLM task. We suppose that there are "Consistency", "Discrepancy" and "None" relations between stance and emotion. The auxiliary sentences are constructed on the premise of these relations as well as the target. With the help of C-MLM task, SEDA augment the dataset by predicting

the masked words conditioned on both their context and the auxiliary sentences. Furthermore, to address the linguistic adversity problem in augmented samples, we propose a Gradual Prompt-tuning method, which combines prompt-tuning with curriculum learning to train our model. We design a template that contains target and stance information. After that, we create an artificial curriculum in the training samples according to the disturbance degree in data augmentation. Starting by training on original samples, we feed augmented samples with a higher level of noising into the model as training progresses. The model learns to explicitly capture stance relations between sentence and target by predicting masked words. Our main contributions can be summarized as follows:

- We propose a Stance-Emotion joint Data Augmentation (SEDA) method, which introduces emotional information in the conditional data augmentation of stance detection.
- We further propose a Gradual Prompt-tuning method to overcome the linguistic adversity problem in augmented samples, which combines prompt-tuning with curriculum learning.
- Experimental results show that our methods significantly outperform the state-of-the-art methods.

## 2 Related Work

### 2.1 Stance Detection

Stance detection aims to automatically infer the stance of a text towards specific targets [1,13], which is related to argument mining, fact-checking, and aspect-level sentiment analysis. Early stance detection tasks concentrate on online forums and debates [27,29]. Later, a series of studies on different types of targets emerge. The targets become political figures [15,26], controversial topics [7], and so on. At present, the research tasks are mainly divided into three types, in-target stance detection [36], cross-target stance detection [3], and zero-shot stance detection [2]. In this paper, we focus on in-target stance detection, which means the test target can always be seen in the training stage.

### 2.2 Data Augmentation

Lexical substitution is a commonly used augmentation strategy, which attempts to substitute words without changing the meaning of the entire text.

The first commonly used approach is the thesaurus-based substitution, which means taking a random word from the sentence and replacing it with its synonym using a thesaurus. Zhang et al. [37] apply this and search synonyms in WordNet [19] database. Mueller et al. [21] use this idea to generate additional training samples for their sentence similarity model. This approach is also used by Wei et al. [33] as one of the four random augmentations in EDA.

The second approach is the word-embedding substitution, which replaces some words in a sentence with their nearest neighbor words in the embedding

space. Jiao et al. [10] apply this with GloVe embeddings [23] to improve the generalization of their model on downstream tasks, while Wang et al. [30] use it to augment tweets needed to learn a topic model.

The third approach is based on the masked language model, which has to predict the masked words based on their context. Therefore, the model can generate variations of a text using the mask predictions. Compared to previous approaches, the generated text is more grammatically coherent as the model takes context into account when making predictions. Grag et al. [8] use this idea to generate adversarial samples for text classification. Wu et al. [35] formulate the data augmentation as a C-MLM task. Li et al. [16] propose an Auxiliary Sentence based Data Augmentation (ASDA) method that generates samples based on C-MLM task. Inspired by ASDA, we investigate how to introduce more information via auxiliary sentences.

### 2.3 Curriculum Learning

Curriculum learning is proposed by Bengio et al. [4], which is a training strategy that imitates the meaningful learning order in human curricula. It posits that models train better when training samples are organized in a meaningful order. In the beginning, researchers assume that there exists a range of difficulties in the training samples [28,34]. They leverage various heuristics to sort samples by difficulty and train models on progressively harder samples. Korbar et al. [12] propose instead of discovering a curriculum in existing samples, samples can be intentionally modified to dictate an artificial range of difficulty. Wei et al. [32] combine this idea with data augmentation and propose a curriculum learning strategy, but the performance is still constricted by the gap of objective forms between pre-training and fine-tuning.

### 2.4 Prompt-Tuning

Pre-trained language models like GPT [5] and BERT [11] capture rich knowledge from massive corpora. To make better use of the knowledge, prompt-tuning is proposed. In prompt-tuning, downstream tasks are also formalized as some objectives of language modeling by leveraging language prompts. The results of language modeling can correspond to the solutions of downstream tasks. With specially constructed prompts and tuning objectives [18,24], we can further inject and stimulate the task-related knowledge in pre-trained models, thus boosting the performance. To our knowledge, there is currently a lack of research on applying prompt-tuning to the stance detection task.

## 3 Method

In this section, we first introduce the variables and definitions that appear in this paper. Then provide the overall architecture of SEGP and explain it in detail.

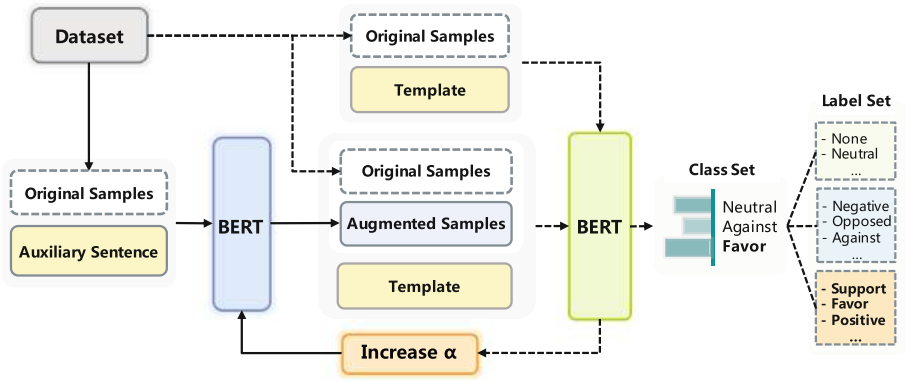



**Fig. 1.** The overall architecture of SEGP, where $\alpha$ represents the degree of disturbance in the augmentation stage. Solid arrows indicate Stance-Emotion joint Data Augmentation stage and dashed arrows indicate Gradual Prompt-tuning stage.

### 3.1 Preliminaries

We first give some essential preliminaries. Suppose a given training dataset of size n is $D_{\text{train}} = \{X, S, T, E\}$, where $X = \{x_1, x_2, \ldots, x_n\}$ is the set of input samples. For each $x_i \in X$, it consists of a sequence of $l$ words $x_i = [w_i^1, w_i^2, \ldots, w_i^l]$. We define a stance label set $S = \left(s_1, s_2, \ldots, s_{|M|}\right)$, a target set $T = \left(t_1, t_2, \ldots, t_{|C|}\right)$ and an emotional label set $E = \left(e_1, e_2, \ldots, e_{|N|}\right)$, where the values of $|M|$, $|C|$ and $|N|$ depend on the dataset settings.

### 3.2 Overall Architecture

In this paper, we propose a Stance-Emotion joint Data Augmentation with Gradual Prompt-tuning (SEGP) model, and the overall architecture is shown in Fig. 1. SEGP consists of two stages, as we can see from Fig. 1, they are indicated by solid arrows and dashed arrows respectively. The first stage is to get more training samples using the SEDA method. The second is the training stage, which uses the Gradual Prompt-tuning method to overcome the linguistic adversity problem in augmented samples.

### 3.3 Stance-Emotion Joint Data Augmentation

The objective of a data augmentation method is to generate training samples based on the existing limited annotations. In this paper, we propose a novel conditional data augmentation method called SEDA, which is based on C-MLM task. We leverage stance, emotion, and target information to construct auxiliary sentences. SEDA generates target-relevant and stance-emotion-consistent augmented samples by predicting masked words conditioned on context and auxiliary sentences.

**Construction of Auxiliary Sentences.** Many approaches achieve better results by taking emotional information as auxiliary information. It should be noted that stance could be inferred independently from the emotional state, the emotions contained in a text may be positive but expresses an opposition stance to a given target. This is due to the complexity of interpreting a stance because it is not always directly consistent with the emotional polarity. We analyze the distribution of stance and emotional labels in COVID-19-Stance dataset. As shown in Fig. 2, there is a large gap in the distribution of these two types of labels.

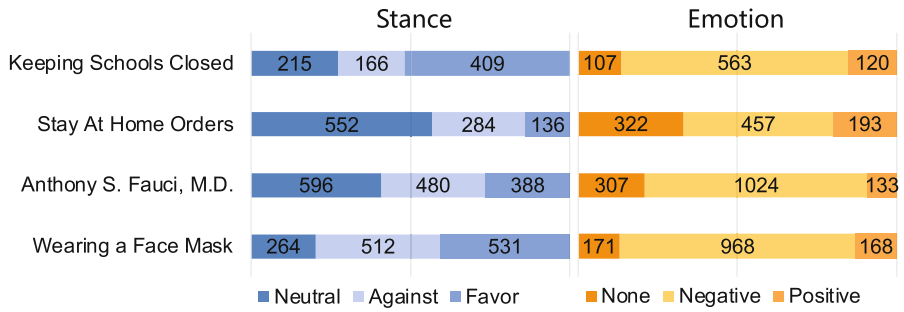



**Fig. 2.** Stance and emotion distribution in COVID-19-Stance dataset.

Our research is based on stance label set S = {"Neutral", "Against", "Favor"} and emotional label set E = {"None", "Negative", "Positive"}. In order to integrate these two types of information, we define a cross label set $C = \{S - E\}$, which is generated by stance label $s$ and emotional label $e$. For example, given $s$="Favor" and $e$="Negative", we can obtain the cross label $c$="Favor-Negative". Before constructing auxiliary sentences, we put forward the following relations between stance and emotion:

- Consistency: When cross label $c$ is in {"Favor-Positive", "Against-Negative", "Neutral-Positive", "Neutral-Negative"}, we suppose that the stance is consistent with emotion, so emotional information can be directly introduced into the auxiliary sentence.
- Discrepancy: When cross label $c$ is in {"Favor-Negative", "Against-Positive"}, we suppose that there is a difference between stance and emotion, so we need to consider this contradiction when constructing auxiliary sentences.
- None: When the emotional label $e$ = "None", we suppose that the emotional information is not helpful. In this case, the auxiliary sentence only needs to introduce stance information.

Therefore, we leverage the above mentioned relations to construct three kinds of auxiliary sentences regarding target, stance, and emotion. We also place slots in the auxiliary sentences, $\{a_i\}$ is used to fill target words, $\{s_i\}$ is used to fill stance label, and $\{e_i\}$ is used to fill emotional label. Experiments show that grammar correctness is not important. Table 2 shows how to select the corresponding

auxiliary sentence according to a cross label. After obtaining the auxiliary sentence, we prepend both another training sample $x_j$ that has the same target and cross label with $x_i$. The complete input form for each training sample $x_i$ is: Auxiliary sentence+$x_j$+"the text is":+$x_i$.

**Table 2.** Correspondence between relations, cross labels, and auxiliary sentences.

| Relations | Cross labels | Auxiliary sentences |
|---|---|---|
| Consistency | Favor-Positive<br>Against-Negative<br>Neutral-Positive<br>Neutral-Negative | The following texts have $\{s_i\}$ stance and $\{e_i\}$ emotion to $\{a_i\}$. |
| Discrepancy | Favor-Negative<br>Against-Positive | Although the emotion is $\{e_i\}$, the following texts are both $\{s_i\}$ to $\{a_i\}$. |
| None | Favor-None<br>Neutral-None<br>Against-None | The following texts have $\{s_i\}$ stance to $\{a_i\}$. |

For example, given the input $x_i$: *I don't need to wear a mask to live a healthy life.* with the stance label $s$ = "Against" and emotional label $e$ = "Positive". The corresponding target is "Wearing a face mask". First, we get its cross label $c$ = "Against-Positive" and choose the discrepancy auxiliary sentence. Second, we find another training sample $x_j$: *The death rate is falling so fast, we don't need to wear masks at all.* So the complete input is: *Although the emotion is* {*positive*}*, the following texts are both* {*against*} *to* {*wearing a face mask*}*. The death rate is falling so fast, we don't need to wear masks at all. The text is: I don't need to wear a mask to live a healthy life.* The introduction of the auxiliary sentence and $x_j$ not only helps to generate more diversified samples, but also provides a strong guideline to help generate target-relevant and label-compatible samples.

**Data Generation.** We fine-tune the pre-trained model via C-MLM task. For a training sample $x_i$ from $X$, we specify that the model can only randomly mask words in the input sample $x_i$ and the mask radio is $\alpha$. Because we want to preserve all of the target, stance, and emotional information. After prepending the corresponding auxiliary sentence and $x_j$ to obtain the masked sentence, a pre-trained language model like BERT is used to predict the masked words. The prediction of masked words depends not only on the context of $x_i$, but also on their target, stance, and emotion.

After fine-tuning the model on the training dataset for a few epochs, we use the well-trained model for augmentation. Similar to the fine-tuning procedure, the model randomly masks words of the training sample, then prepend the auxiliary sentence and another training sample. The model is used to predict the masked words, we repeat these steps over training samples to get augmented samples.

### 3.4 Gradual Prompt-Tuning

In this paper, we apply the training strategy of curriculum learning to prompt-tuning. We aim to solve the linguistic adversity problem [17,31] in augmented samples as well as make better use of the knowledge contained in pre-trained language models.

**Prompt-Tuning.** In order to bridge the gap of objective forms between pre-training and fine-tuning, prompt-tuning is proposed. By tuning a pre-trained language model with the cloze-style task, prompt-tuning can manipulate the model behavior to fit various downstream tasks that more fully utilize task-related knowledge in pre-trained language models. Formally, prompt is consists of a template $P(\cdot)$ and a set of stance labels $S$. For stance detection task, a pre-trained language model uses input sentences and prompt to predict the stance label for a given target. In order to provide more information, we place two slots into the template, $\{t_i\}$ is used to fill target words, and $[MASK]$ is for the model to fill a label word. We set the template $P(\cdot)$ = "The stance to $\{t_i\}$ is $[MASK]$", and map $x$ to the prompt input $x_{prompt} = x+$"The stance to $\{t_i\}$ is $[MASK]$". After that, $x_{prompt}$ is fed into a pre-trained model.

The model first converts the input $x_{prompt} = \left(w_i^1, w_i^2, \ldots, [MASK], \ldots, w_i^l\right)$ to sequence $\left([CLS], w_i^1, w_i^2, \ldots, [MASK], \ldots, w_i^l, [SEP]\right)$, then compute the hidden vector $h_{[MASK]}$ of $[MASK]$. Given $s \in S$, the model calculates the probability for $s$ can fill the masked position, where $\mathbf{s}$ is the embedding of $s$ in a pre-trained language model. The probability is calculated as follows:

$$p\left([MASK] = s \mid x_{\text{prompt}}\right) = \frac{\exp\left(s \cdot h_{[MASK]}\right)}{\sum_{s \in S} \exp\left(\mathbf{s} \cdot h_{[MASK]}\right)} \tag{1}$$

There also exists an injective mapping function $\sigma$ that bridges the set of classes $Y$ and the set of label words $S$, we define $\sigma = Y \rightarrow S$. With the verbalizer $\sigma$, we can formalize the probability distribution over $Y$ with the probability distribution over $S$ at the masked position. i.e., $p(y \mid x) = p([MASK] = \gamma(y) \mid x_{prompt})$. We map the supporting stance to "Favor", the opposing stance to "Against" and other stances to "Neutral". According to model fills the masked position of $x_{prompt}$ with "Favor", "Against" or "Neutral", we can get the stance of $x$. For prompt-tuning, with a template P$(\cdot)$, a label set $S$ and verbalizer $\sigma$, the learning objective is to maximize $\frac{1}{|X|} \sum_{x \in X} \log p\left([MASK] = \gamma\left(y_x\right) \mid P(x)\right)$.

**Curriculum Learning.** The data augmentation method might introduce linguistic adversity and can be seen as a form of noising, where noised data is harder to learn from than unmodified original data. Curriculum learning posits that the model train better when training samples are organized in a meaningful order that gradually shows more concepts and complexity. Therefore, we apply the training strategy of curriculum learning to prompt-tuning. We define the mask radio $0.0 \leq \alpha \leq 0.15$ as disturbance degree for SEDA stage, create an artificial curriculum in training samples according to the disturbance degree of

the augmented samples. A larger mask ratio $\alpha$ represents a larger variation in the training samples, thus harder to learn from than unmodified original samples. During training, we begin with a disturbance degree of $\alpha = 0.0$ (equivalent to no augmentation), then linearly increase $\alpha$ by 0.05 every time validation loss plateaus, up to a final of $\alpha = 0.15$.

## 4 Experiment

In this section, we first present the dataset used for evaluation and several baseline methods. Then introduce experimental details and analyze the results.

### 4.1 Dataset and Baseline Methods

We carry out experiments on the stance detection dataset COVID-19-Stance [9], which is collected by crawling Twitter, using Twitter Streaming API. It contains the tweets of four targets (i.e., "Stay At Home Orders", "Wearing a Face Mask", "Keeping Schools Closed" and "Anthony S. Fauci, M.D"), and the stance label of each tweet is either "Favor" or "Against" or "Neutral".

We compare SEGP with the following baseline methods:

- BiLSTM [25]: Bi-Directional Long Short Term Memory Network takes tweets as input and is trained to predict the stance towards a target, without explicitly using the target information.
- CT-BERT [22]: A pre-trained language model that predicts the stance by appending a linear classification layer to the hidden representation of $[CLS]$ token, pre-trained on a corpus of messages from Twitter about COVID-19.
- CT-BERT-v2 [22]: It is identical to CT-BERT, but trained on more data, resulting in higher downstream performance.
- EDA [33]: A simple data augmentation method that consists of four operations: synonym replacement, random deletion, random swap, and random insertion.
- ASDA [16]: A data augmentation method that generates target-relevant and label-consistent data samples based on C-MLM task.

### 4.2 Experimental Results

SEGP is implemented based on CT-BERT-v2 [22], using a batch size of 8. The learning rate of Adam optimizer is 1e−5 and the maximum sequence length is 256. Experimental results are shown in Table 3, the best model configuration is selected according to the highest performance on the development set.

We first compare SEGP with BiLSTM [25], CT-BERT [22] and CT-BERT-v2 [22]. It can be seen that SEGP is superior to all baselines in accuracy and F1 score, which demonstrates the validity of our model in stance detection tasks. Besides, we compare SEGP with different data augmentation methods, i.e., EDA and ASDA. We can observe that SEGP performs the best, while EDA and ASDA

methods have limited improvement in performance. Furthermore, when target = "Anthony S. Fauci, M.D.", the result is even worse than CT-BERT that only trained on original samples.

SEGP has better performance on all targets, which proves it can not only generate more diversified samples but also have the ability to overcome the linguistic adversity problem and better utilize task-related knowledge in pre-trained language models.

**Table 3.** Performance of SEGP and different baseline methods for stance detection on four targets in the COVID-19-Stance dataset. The performance is reported in terms of accuracy(Acc), precision(P), recall(R), and F1 score(F1). We highlight the best results in bold.

| Model | Wearing a face mask | | | | Stay at home orders | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | P | R | F1 | Acc | P | R | F1 |
| BiLSTM | 57.80 | 56.90 | 58.00 | 56.70 | 73.50 | 67.90 | 64.00 | 64.50 |
| CT-BERT | 81.00 | 81.80 | 80.30 | 80.30 | 84.30 | 81.60 | 78.80 | 80.00 |
| CT-BERT-v2 | 81.25 | 80.49 | 81.99 | 80.13 | 86.00 | 82.56 | 88.00 | 84.78 |
| EDA | 81.50 | 79.77 | 78.61 | 79.07 | 85.50 | 81.96 | 84.50 | 83.09 |
| ASDA | 82.50 | 80.96 | 80.24 | 80.53 | 87.00 | 83.04 | 85.09 | 83.99 |
| SEGP | **84.50** | **83.20** | **83.49** | **83.34** | **89.00** | **86.33** | **89.37** | **87.71** |
| Model | Anthony S. Fauci, M.D. | | | | Keeping schools closed | | | |
| | Acc | P | R | F1 | Acc | P | R | F1 |
| BiLSTM | 63.80 | 63.90 | 63.10 | 63.00 | 62.70 | 57.00 | 54.50 | 54.80 |
| CT-BERT | 81.70 | 81.60 | 83.00 | 81.80 | 77.20 | 76.50 | 76.10 | 75.50 |
| CT-BERT-v2 | 80.25 | 80.16 | 81.36 | 80.42 | 81.00 | 78.81 | 79.14 | 78.85 |
| EDA | 80.50 | 80.82 | 81.01 | 80.55 | 83.00 | 80.92 | 81.66 | 80.98 |
| ASDA | 81.00 | 81.49 | 81.04 | 81.06 | 83.50 | 81.29 | 80.95 | 81.01 |
| SEGP | **82.50** | **82.60** | **82.57** | **82.57** | **86.00** | **84.04** | **84.45** | **84.23** |

### 4.3 Analysis of Stance-Emotion Joint Data Augmentation

We conduct experiments to prove the following two points: (1) the effectiveness of introducing emotional information into data augmentation; (2) the effectiveness of introducing emotional information through different types of auxiliary sentences.

In order to prove the first point, we compare the results of Stance-Emotion joint Data Augmentation (SEDA) with ASDA, which does not take emotional information into account. We present several augmented samples generated by these two methods in Table 4. It can be observed that the generated words of SEDA are more consistent with the label information. Furthermore, according

to the experimental results in Table 5, SEDA outperforms ASDA on all targets, which further demonstrates the validity of emotional information.

**Table 4.** Examples generated by ASDA and SEDA. Italicized texts represent generated words.

| Target | Wearing a face mask |
|---|---|
| Source | In the USA, Walmart will now serve mask-less customers. Hopefully the same will happen in the UK |
| ASDA | In the USA, Walmart will *today* serve mask-less customers. Hopefully the *fight* will *spread* in the UK |
| SEDA | In the USA, Walmart will now serve mask-less customers. Hopefully the same will happen *sooner to* the *globe* |

In order to prove the second point, we compare the results of using different auxiliary sentences. The auxiliary sentences are constructed based on the relations between stance and emotion. "Consistency only" means we only use the "Consistency" relation between stance and emotion to introduce emotional information, thus SEDA(Consistency only) only contains the auxiliary sentence: The following texts have $\{s_i\}$ stance and $\{e_i\}$ emotion to $\{a_i\}$. "Discrepancy only" means we only use the "Discrepancy" relation, thus SEDA(Discrepancy only) only contains: Although the emotion is $\{e_i\}$, the following texts are both $\{s_i\}$ to $\{a_i\}$. SEDA is what we propose in this paper, which introduces emotional information based on "Consistency", "Discrepancy" and "None" relations. Therefore, as shown in Table 2, SEDA contains three types of auxiliary sentences. The experimental results in Table 5 show the performance impact of different auxiliary sentences, we can see that SEDA performs the best, indicating the effectiveness of the way we introduce emotional information.

### 4.4 Analysis of Gradual Prompt-tuning

We further explore the effectiveness of curriculum learning by comparing SEGP with SEP, which does not use the training strategy of curriculum learning. Curriculum learning requires a series of training samples with different disturbance degrees. In our method, the disturbance degree is determined by the mask ratio $\alpha$ in augmentation stage. Therefore, the artificial curriculums in the training samples are created according to $\alpha$. Experimental results are shown in Table 6, which indicates that we can further improve performance by combining prompt-tuning with curriculum learning.

**Table 5.** Performance comparison of introducing emotional information in different ways. We highlight the best results in bold.

| Model | Wearing a face mask | | | | Stay at home orders | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | P | R | F1 | Acc | P | R | F1 |
| ASDA | 82.50 | 80.96 | 80.24 | 80.53 | 87.00 | 83.04 | 85.09 | 83.99 |
| SEDA(Consistency only) | 81.50 | 79.57 | 80.18 | 79.83 | 86.50 | 83.05 | 86.40 | 84.51 |
| SEDA(Discrepancy only) | 80.50 | 78.98 | 77.74 | 78.25 | 86.00 | 82.33 | 86.68 | 84.17 |
| SEDA | **83.50** | **82.50** | **82.26** | **82.36** | **87.50** | **84.51** | **86.32** | **85.36** |
| Model | Anthony S. Fauci, M.D. | | | | Keeping schools closed | | | |
| | Acc | P | R | F1 | Acc | P | R | F1 |
| ASDA | 81.00 | 81.49 | 81.04 | 81.06 | 83.50 | 81.29 | 80.95 | 81.01 |
| SEDA(Consistency only) | 80.00 | 79.93 | 81.17 | 80.32 | 83.50 | 80.89 | 81.51 | 81.14 |
| SEDA(Discrepancy only) | 80.50 | 80.31 | 81.66 | 80.79 | 82.00 | 80.38 | 81.77 | 80.85 |
| SEDA | **82.00** | **82.11** | **82.20** | **82.09** | **85.50** | **83.79** | **83.09** | **83.40** |

**Table 6.** Performance comparison of applying different training strategies. We highlight the best results in bold.

| Model | Wearing a face mask | | | | Stay at home orders | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | P | R | F1 | Acc | P | R | F1 |
| SEP | 83.50 | 82.50 | 82.26 | 82.36 | 87.50 | 84.51 | 86.32 | 85.36 |
| SEGP | **84.50** | **83.20** | **83.49** | **83.34** | **89.00** | **86.33** | **89.37** | **87.71** |
| Model | Anthony S. Fauci, M.D. | | | | Keeping schools closed | | | |
| | Acc | P | R | F1 | Acc | P | R | F1 |
| SEP | 82.00 | 82.11 | 82.20 | 82.09 | 85.50 | 83.79 | 83.09 | 83.40 |
| SEGP | **82.50** | **82.60** | **82.57** | **82.57** | **86.00** | **84.04** | **84.45** | **84.23** |

## 5 Conclusion

In this paper, we propose SEGP to address the scarcity of annotations problem in stance detection. SEGP is mainly composed of two stages, i.e., Stance-Emotion joint Data Augmentation (SEDA) and Gradual Prompt-tuning. With the help of C-MLM task, SEDA generates target-relevant and label-compatible samples by predicting the masked word conditioned on both their context and the auxiliary sentences. Gradual Prompt-tuning can make better use of the augmented samples as well as the knowledge contained in pre-trained models. The experimental results show that SEGP obtains superior performance over all baseline methods. Since our methods are not designed for a certain model, we will investigate how to extend them to other tasks in the future.

## References

1. AlDayel, A., Magdy, W.: Stance detection on social media: state of the art and trends. Inf. Process. Manage. **58**(4), 102597 (2021)
2. Allaway, E., McKeown, K.: Zero-shot stance detection: a dataset and model using generalized topic representations. arXiv preprint arXiv:2010.03640 (2020)

3. Augenstein, I., Rocktäschel, T., Vlachos, A., Bontcheva, K.: Stance detection with bidirectional conditional encoding. arXiv preprint arXiv:1606.05464 (2016)
4. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning, pp. 41–48 (2009)
5. Brown, T.B., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 (2020)
6. Chauhan, D.S., Kumar, R., Ekbal, A.: Attention based shared representation for multi-task stance detection and sentiment analysis. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) ICONIP 2019. CCIS, vol. 1143, pp. 661–669. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36802-9_70
7. Du, J., Xu, R., He, Y., Gui, L.: Stance classification with target-specific neural attention networks. In: International Joint Conferences on Artificial Intelligence (2017)
8. Garg, S., Ramakrishnan, G.: Bae: Bert-based adversarial examples for text classification. arXiv preprint arXiv:2004.01970 (2020)
9. Glandt, K., Khanal, S., Li, Y., Caragea, D., Caragea, C.: Stance detection in covid-19 tweets. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, vol. 1 (2021)
10. Jiao, X., et al.: Tinybert: distilling bert for natural language understanding. arXiv preprint arXiv:1909.10351 (2019)
11. Kenton, J.D.M.W.C., Toutanova, L.K.: Bert: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, pp. 4171–4186 (2019)
12. Korbar, B., Tran, D., Torresani, L.: Cooperative learning of audio and video models from self-supervised synchronization. In: Advances in Neural Information Processing Systems 31 (2018)
13. Küçük, D., Can, F.: Stance detection: a survey. ACM Comput. Surv. (CSUR) **53**(1), 1–37 (2020)
14. Li, Y., Caragea, C.: Multi-task stance detection with sentiment and stance lexicons. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6299–6305 (2019)
15. Li, Y., Caragea, C.: A multi-task learning framework for multi-target stance detection. In: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pp. 2320–2326 (2021)
16. Li, Y., Caragea, C.: Target-aware data augmentation for stance detection. In: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1850–1860 (2021)
17. Li, Y., Cohn, T., Baldwin, T.: Robust training under linguistic adversity. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 21–27 (2017)
18. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: a systematic survey of prompting methods in natural language processing. arXiv preprint arXiv:2107.13586 (2021)
19. Miller, G.A.: Wordnet: a lexical database for English. Commun. ACM **38**(11), 39–41 (1995)
20. Mohammad, S.M., Sobhani, P., Kiritchenko, S.: Stance and sentiment in tweets. ACM Trans. Internet Technol. (TOIT) **17**(3), 1–23 (2017)

21. Mueller, J., Thyagarajan, A.: Siamese recurrent architectures for learning sentence similarity. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)
22. Müller, M., Salathé, M., Kummervold, P.E.: Covid-twitter-bert: a natural language processing model to analyse covid-19 content on twitter. arXiv preprint arXiv:2005.07503 (2020)
23. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
24. Reynolds, L., McDonell, K.: Prompt programming for large language models: Beyond the few-shot paradigm. In: Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, pp. 1–7 (2021)
25. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**(11), 2673–2681 (1997)
26. Sobhani, P., Inkpen, D., Zhu, X.: A dataset for multi-target stance detection. In: Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pp. 551–557 (2017)
27. Somasundaran, S., Wiebe, J.: Recognizing stances in ideological on-line debates. In: Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text, pp. 116–124 (2010)
28. Tsvetkov, Y., Faruqui, M., Ling, W., MacWhinney, B., Dyer, C.: Learning the curriculum with bayesian optimization for task-specific word representation learning. arXiv preprint arXiv:1605.03852 (2016)
29. Walker, M., Anand, P., Abbott, R., Grant, R.: Stance classification using dialogic properties of persuasion. In: Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 592–596 (2012)
30. Wang, W.Y., Yang, D.: That's so annoying!!!: a lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using# petpeeve tweets. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 2557–2563 (2015)
31. Wang, X., Pham, H., Dai, Z., Neubig, G.: Switchout: an efficient data augmentation algorithm for neural machine translation. arXiv preprint arXiv:1808.07512 (2018)
32. Wei, J., Huang, C., Vosoughi, S., Cheng, Y., Xu, S.: Few-shot text classification with triplet networks, data augmentation, and curriculum learning. arXiv preprint arXiv:2103.07552 (2021)
33. Wei, J., Zou, K.: Eda: easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196 (2019)
34. Weinshall, D., Cohen, G., Amir, D.: Curriculum learning by transfer learning: Theory and experiments with deep networks. In: International Conference on Machine Learning, pp. 5238–5246. PMLR (2018)
35. Wu, X., Lv, S., Zang, L., Han, J., Hu, S.: Conditional bert contextual augmentation. In: International Conference on Computational Science, pp. 84–95. Springer (2019)
36. Zhang, B., Yang, M., Li, X., Ye, Y., Xu, X., Dai, K.: Enhancing cross-target stance detection with transferable semantic-emotion knowledge. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3188–3197 (2020)
37. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Adv. Neural. Inf. Process. Syst. **28**, 649–657 (2015)

# Image Features Correlation with the Impression Curve for Automatic Evaluation of the Computer Game Level Design

Jarosław Andrzejczak(✉), Olgierd Jaros, Rafał Szrajber, and Adam Wojciechowski

Institute of Information Technology, Lodz University of Technology, 215 Wólczańska Street, 90-924 Lodz, Poland
{jaroslaw.andrzejczak,rafal.szrajber}@p.lodz.pl
http://it.p.lodz.pl

**Abstract.** In this study, we present the confirmation of existence of the correlation of the image features with the computer game level Impression Curve. Even a single image feature can describe the impression value with good precision (significant strong relationship, Pearson $r > 0{,}5$). Best results were obtained using by combining several image features using multiple regression (significant very strong positive relationship, Pearson $r = 0{,}82$ at best). We also analyze the different set of image features at different level design stages (from blockout to final design) where significant correlation (strong to very strong) was observed regardless of the level design variant. Thanks to the study results, the user impression of virtual 3D space, can be estimated with a high degree of certainty by automatic evaluation using image analysis.

**Keywords:** Image analysis · Virtual reality · Impression curve · Level design · Automatic evaluation

## 1 Introduction

In [1] study, we have shown that Virtual Reality space affects different users in a similar way. That sense can be stored and described as Impression Curve[1] for this space. Therefore, Impression Curve can be used in 3D VR space evaluation such as 3D level design. Still, it requires tests with many users to gather proper data. It would be a great improvement if designers could estimate the sense of 3D space during the development process and then verify it at the end with the users. Especially with the growing popularity of level designs generated by

[1] Impression Curve is a measure of the visual diversity and attractiveness of a game level. It assesses subjective attraction of a given space. For the detailed information about the Impression Curve, its acquisition method, its strengths and weaknesses in the domain of the 3D space evaluation, please refer to [1].


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 591–604, 2022.
https://doi.org/10.1007/978-3-031-08757-8_49

algorithms [16]. We focused our efforts to provide such computationally low cost tool for 3D space evaluation in the context of estimating user experience.

The purpose of the research was to verify the existence of the correlation of the image features (gathered using automatic image analysis) with the Impression Curve (obtained during previous studies conducted on 112 people). The study described in this article involves examining the impact of various image features such as mean brightness and contrast, features based on saliency and movement maps (such as complexity or density), as well as descriptive statistics like entropy, skewness and kurtosis.

The contributions to research concerning automatic evaluation of the immersive Virtual Reality space, especially in case of the Impression Curve estimation presented in this article, are:

- Confirmation of the existence of a correlation between data gathered using image analysis and user-generated Impression Curve.
- Tests verifying the correlation between individual image feature and the Impression Curve for the VR space.
- Tests verifying the correlation between combined image features and the Impression Curve for the VR space.
- Analysis of usability of each image features depending on the level design stages and changing factors of the 3D space.
- Proposition of the best image features (with the highest correlation values with Impression Curve) for evaluation of individual level design stages.

We start with a related work overview in the domain of image analysis for feature extraction in the next section. Then we describe hypotheses and an evaluation method. Next, both test results and their discussion will be presented, as well as observations about data gathered. Finally, ideas for further development and final conclusions will be given.

## 2 Image Features

There are many image features available to consider in terms of image analysis for automatic feature extraction and image description. Our goal was to test as diverse set of features as possible. The three groups of features were used: color and luminance-based (such as mean brightness, mean color contrast) [6], features based on saliency and motion maps (such as balance and density) [5], as well as descriptive statistics (entropy, skewness and kurtosis) [7]. Therefore, a total of thirteen features were selected for this study:

- **Color and luminance group:** Average Contrast, Average Luminance and Average Saturation.
- **Saliency and motion maps group:** Alignment Complexity, Balance Complexity, Density Complexity, Grouping Complexity, Size Complexity and Total Complexity.
- **Descriptive statistics group:** Entropy, Kurtosis, Skewness and Fractal Complexity.

For the calculations of the image **Average Contrast**, **Average Luminance** and **Average Saturation**, the definitions for the HSL color palette were used. Average Contrast was calculated using the mean square of the Luminance of individual pixels [6].

Image features from the second group are based on classification and analysis of areas indicated in saliency maps [2] and motion maps [8]. Those maps are combined (with a weight of 50% of each, as we considered them equally important) and classified to be used with the metrics described in [5]. This stage requires the greatest number of computations. We start with the creation of the saliency map using the fast background detection algorithm [2], which is then denoised using the method described in [3]. The result is a black and white image, with white pixels representing the relevant ones. A motion map is created as a difference of the pixels of two subsequent video frames converted to grayscale with a Gaussian blur applied to them (which allows limiting the influence of details and noise on motion detection) [4]. The resulting image is denoised and thresholded to obtain a black and white image and combined with the saliency map to obtain the final visual attention saliency map [14]. Then the classification of regions, objects and their contours as well as shape recognition is made.

Regions of attention (representing grouped objects) and their centroids are calculated using K-Means with 30 starting points (pixels) picked randomly on visual attention saliency map white pixels. For each iteration, the closest region centroid for each point is calculated and the region centroid weights are updated. The algorithm runs for 1000 epochs or until each region centroid remains unchanged in two subsequent epochs. During this process, centroids, which for two ages were not the closest one for any point, are permanently removed from the set to optimize the calculations. Centroids calculated for one frame become the starting points for the next frame, with one new random starting point added (to allow the new area recognition).

Objects of attention are found by applying erosion filter and OpenCV shape detection [10] on the final visual attention saliency map. Next, the object's contour is calculated using the contour approximation method [10]. For each of the identified object, a centroid is calculated. Please note that object's centroid is usually different from region centroid, as one region can contain many objects.

Localized object's contours are therefore used for shape recognition [10]. Only simple geometric shapes are taken into account, and every object with a number of vertices greater than or equal to five is classified as a circle (for the purpose of further analysis).

All of the above final visual attention saliency map characteristic is then used with the metrics for UI complexity analysis described in [5]. Each metric gives a final score in the range [0,1] where a score closer to zero means less complexity. The **Alignment Complexity** determines the complexity of the interface in terms of the position of the found shapes relative to each other. The evaluation consists of the calculation of the local and global alignment coefficients for grouped and ungrouped objects. The **Density Complexity** determines the comparison of the visual attention object size to the entire image frame size.

The **Balance Complexity** describes the distribution of visual attention objects on the quarters of the screen. It is calculated as the arithmetic mean of two mean values: the proportion of the number of objects between pairs of quarters and the proportion of the size of objects between pairs of quarters. The **Size Complexity** is calculated due to the grouping of objects on the screen in terms of shape. For each shape type, the number of occurrences of the size of objects is checked. Then The sum of the occurrences of unique object regions is divided by the number of objects in the particular group of shapes. The **Grouping Complexity** determines how many of the objects are grouped into shape type groups. It is the sum of the ratio of ungrouped objects to all occurring and the number of groups of shapes occurring in the region of objects from all possible shapes types. The **Total Complexity** is a combined metric of all previous with weights as proposed in [5]:

$$TotalComplexity = \begin{array}{l} 0,84 \times Alignment + 0,76 \times Balance + \\ 0,8 \times Density + 0,72 \times Size + 0,88 \times Grouping \end{array} \quad (1)$$

The third group of image features is based on statistical descriptors of a data set's distribution. The **Skewness** is a measure of the asymmetry of a distribution of the mean. The higher the Skewness, the more asymmetric data distribution. The **Kurtosis** is a measure of how results are concentrated around the mean. The high Kurtosis value would suggest outliers in the data set and low Kurtosis value the lack of outliers [11]. The **Entropy** of an image is used as a measure of the amount of information it contains [7]. The more detailed the image, the higher the value of the Entropy will be. Entropy, Kurtosis and Skewness were counted separately for Hue, Saturation and Luminosity as they operate on the single variable (grayscale image as input). The **Fractal Complexity** is a measure of self-similarity. It determines how much it is possible to break an image or fractal into parts that are (approximately) a reduced copy of the whole. This parameter was used to assess the complexity of the image [9][2].

## 3 Evaluation

The goal of the evaluation was to verify the existence of the correlation of the image features (gathered using automatic image analysis) with the Impression Curve. For this purpose, the Pearson and Spearman correlation were used [13]. All the level design stages as well as the influential factors on the 3D space impression (such as lightening condition changes, geometrical and material changes) described in [1] were used (Fig. 1).

[2] At this stage of the Impression Curve automatic evaluation study, we have used the controlled 3D space designs to minimize the influence of the such factors as action, gameplay rules and restrictions, story and lore present in commercial game designs. After confirmation of existence of the correlation of the image features with the computer game level Impression Curve described in this article, we moved to testing level design from popular games. The results of this study will be published in the future, as it is in development at the time of writing this article.

The study was divided into two parts. First, the correlation of the individual image features with the Impression Curve was analyzed. After that, image features with the highest correlation value were combined into sets and once again tested for correlation with Impression Curve to see if there is any gain in the strength of the correlation.

The hypotheses in individual parts were as follows:

1. First part: there is a significant correlation (positive or negative) between an individual image feature and the Impression Curve for the same VR space.
2. Second part: the correlation (positive or negative) with the Impression Curve is higher for the combined image features than for the individual image features.
3. Additional observation: different set of image features presents the highest correlation values for different level design stages.

What is more, different level design stages and changing factors of the 3D space (for example: lightening condition, geometrical detail or material changes) of the same game level allow us to observe if there is any difference in correlation between data gathered using image analysis and user-generated Impression Curve. Thanks to this, we were able to point out the best automatic evaluation measures in the form of selected image features, to use at each design stage (blockout, models without materials, textured models as well as lightning and atmospheric effects such as rain).

The twelve level variants showing successive design stages were used according to our previous research, described in details in [1]. There were as follows: simple blockout (A), advanced blockout (B), main models without materials (C), main models with monochromatic materials (D) and final materials (E) as well as with extra fine detailed models (called final level version) (F), main models with geometrical changes (G) and final level with changes of visual factors as lightening condition (L), weather condition (W), different materials (M), added expression (X) as well as with extra models and objects in the environment (O). Existence of correlation between image features and Impression Curve values would allow creation of a tool to automatically estimate Impression Curve for a VR space with a high degree of probability. And as a result, to automatically evaluate expected user impression even on an early Virtual Reality space design stage.

## 4 Results and Analysis

During the study, hundreds of correlation plots were gathered and analyzed. We assumed that per frame comparison will be sensitive to rapid image changes, effecting low or no correlation at all. That is why, the mean and median of an image feature for a few consecutive frames were calculated. A small range of 4–5 frames allow us to eliminate minor fluctuations, where a larger range of 20–30 frames softened the charts quite significantly. However, a larger range considerably reduces the number of data samples, which had an impact on the


**Fig. 1.** The twelve level variants showing successive design stages used in this study for image analysis and correlation with Impression Curve. A - simple blockout; B - advanced blockout; C - models without materials; D - models with monochromatic materials; E - models with final materials; F - final level version; G - geometrical changes; L - lightening condition changes; W - weather changes; M - material changes; X - expression added; O - extra models added.

significance value $p$. Thus, we started from a range of four frames and increased this interval by four from that point. As a result, four to twenty frames, we observed increased correlation value for most of the image features while preserving low value of $p < 0{,}05$. For frame range greater than twenty, results were not significant anymore ($p > 0{,}05$). Also, above this point, the correlation value for many image features dropped below the value of 0,3. Thus, we choose a range of twenty frames for our study, as it shows the highest correlation values with significance $p < 0{,}05$ (in many cases $p < 0{,}01$). In the other hand, we gathered image data more often (thirty times per second - video recorded with a 30 FPS frame rate) than during study with users. Thus, the Impression Curve data had to be interpolated between measure points (as we assumed linear change). This way we were able to compare this data even per frame.

The experiment stages were as follows: first, for each of the video game level variants the Impression Curve data (gathered with users) was interpolated between the measure points to match the frequency of data calculated using image analysis for this level variant walkthrough video; next, the image features were calculated and refined using respectively mean and median for 20 subsequent frame intervals; finally, the Pearson and Spearman correlation between those data were calculated.

The recordings of twelve variants of the video game level variants (used in [1]), including twenty-nine thousand three hundred and thirty-nine frames in total, were analyzed. As a result, thirty-six data sets were obtained and used to generate two hundred and ninety-nine correlation plots.

### 4.1 Individual Image Features Correlation

The first part of the study involved testing each of the thirteen image features individually for correlation with an interpolated Impression Curve for each of twelve variants of the video game level described earlier. The result of a single feature-variant pair was stored in numerical way and also as a correlation plot for easier analysis (Fig. 2). Each data point in the graph shows respectively the mean or median (depending on which one was used) over an interval of 20 frames of the video data. The feature values are marked in red, while the values of the Impression Curve are marked in green. The charts contain the calculated Pearson correlation for a whole Impression Curve. When this value is below 0,5 the Spearman correlation is calculated as well to compensate possible outliers and check for nonlinear relation. Two numbers are presented for each correlation. The first is the mean correlation value, the second is the calculated $p$ value of this correlation.

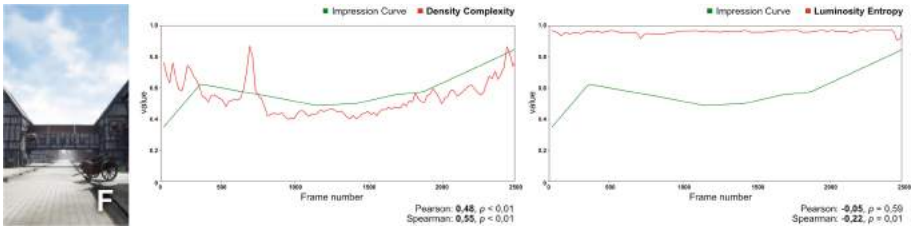



**Fig. 2.** Correlation plot examples for final level design variant (F variant, on the left). Two image feature correlation plots are presented: one with significant strong positive relationship - Density Complexity (Pearson r = 0,48 with $p < 0,01$, center) other with no significant linear relationship and weak non-linear relationship - Luminosity Entropy (Pearson r = -0,05 with $p = 0,59$, right). A linear relationship can be observed for Density Complexity. (Color figure online)

Then the correlation values of every image feature tested for a single level design variant were juxtaposed with each other (Table 1 shows the results for only one variant as an example - the same was done for each of twelve level design variants).

We observed many significant correlation values (positive and negative) between image features and Impression Curve value. Observation varied from a few weak relationships ($r$ value between 0,20 and 0,29) to moderate relationship in most cases ($r$ value between 0,30 and 0,39) and even over a dozen strong relationship ($r$ value between 0,40 and 0,69). There was not a single variant without at least one significantly related image feature, and in most cases there were several moderate relationships. What is more, some image features tend to correlate more often than others, where others given at least weak relationship only once or twice (Table 2). We did not observe a significant difference between

**Table 1.** Pearson's correlation values for individual image features of the final version of the level (F). Feature values were calculated respectively as the mean and median for the intervals of twenty frames. The highest correlation results are marked with a gray background color and bold text. The significant $p$ values are marked with a gray background color. We can observe that the same image features show the highest correlation and similar values for both the mean and the median, with only one feature (Grouping Complexity) presenting lower correlation using the median. $r$ - Pearson correlation coefficient value; $p$ - significance value.

| | Mean | | Median | |
|---|---|---|---|---|
| Image feature | $r$ | $p$ | $r$ | $p$ |
| Alignment Complexity | 0,07 | 0,463 | 0,06 | 0,516 |
| Balance Complexity | **0,33** | <0,001 | **0,32** | p <0,001 |
| Density Complexity | **0,48** | <0,001 | **0,52** | p <0,001 |
| Grouping Complexity | **0,28** | 0,002 | 0,16 | 0,087 |
| Size Complexity | **0,43** | <0,001 | **0,46** | p <0,001 |
| Total Complexity | **0,53** | <0,001 | **0,48** | p <0,001 |
| Average Contrast | **-0,28** | 0,002 | **-0,28** | 0,002 |
| Average Luminance | 0,01 | 0,951 | -0,01 | 0,908 |
| Average Saturation | **-0,36** | <0,001 | **-0,35** | 0,000 |
| Fractal Complexity | -0,05 | 0,551 | -0,06 | 0,544 |
| Hue Entropy | **0,35** | <0,001 | **0,35** | p <0,001 |
| Hue Kurtosis | -0,04 | 0,651 | -0,03 | 0,737 |
| Hue Skewness | 0,05 | 0,582 | 0,07 | 0,457 |
| Saturation Entropy | -0,16 | 0,083 | -0,15 | 0,103 |
| Saturation Kurtosis | -0,03 | 0,719 | -0,07 | 0,477 |
| Saturation Skewness | -0,04 | 0,701 | -0,04 | 0,652 |
| Luminosity Entropy | 0,03 | 0,719 | 0,05 | 0,587 |
| Luminosity Kurtosis | **0,33** | <0,001 | **0,33** | p <0,001 |
| Luminosity Skewness | 0,06 | 0,508 | 0,07 | 0,456 |

mean and median values ($t - test\ p = 0{,}52$) thus only median will be used in further analysis as less valuable for outliers.

For all but one image features, we observe no significant difference between Pearson and Spearman correlation coefficient values, which suggest a linear nature of the relationship. Thus, in further combined image features we focused on Pearson correlation coefficient as linear relationship is more desired for the future video game level design automatic evaluation system. Only for Density Complexity feature, we observed significant difference ($t-test\ p = 0{,}05$) between Pearson and Spearman results with Spearman correlation coefficient values being higher most of the time giving moderate to high positive relationship (also with much lower $p$ value). This indicates the existence of a non-linear relationship between Density Complexity feature and the Impression Curve.

The results are dominated by a positive correlation, with six image features tending to present a negative relationship more often than positive. Those are: Grouping Complexity, Fractal Complexity, Average Contrast, Average Saturation and Entropy (for Saturation and Luminosity). Most of them present many moderate to strong relationships (also variants with low correlation value results were not significant with $p > 0{,}05$). The highest single image features correlation value observed was 0,57 (strong positive relationship, $p < 0{,}01$) for a Size Complexity feature in variant of models with the final materials (E).

**Table 2.** Pearson's correlation values for individual image features for all twelve level design variants. Feature values were calculated as median for the intervals of twenty frames. The significant correlation results (with $p =< 0{,}01$) are marked with a grayscale background color (the darker the color, the higher the correlation value) and bold text. Strong relationship ($r$ value between 0,40 and 0,69) was outlined with a white text color. We can observe that some image features as Size Complexity or Grouping Complexity tend to present high correlation value in many variants. A - simple blockout; B - advanced blockout; C - models without materials; D - models with monochromatic materials; E - models with final materials; F - final level version; G - geometrical changes; L - lightening condition changes; W - weather changes; M - material changes; X - expression added; O - extra models added.

| | Level Design Variant | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Image Feature | A | B | C | D | E | F | G | L | W | M | X | O |
| Alignment Complexity | -0,15 | -0,01 | -0,04 | 0,13 | 0,14 | 0,06 | 0,10 | 0,09 | **-0,25** | 0,02 | -0,07 | -0,10 |
| Balance Complexity | 0,04 | 0,06 | **0,33** | **0,25** | 0,18 | **0,32** | 0,08 | 0,14 | 0,19 | **0,47** | **0,33** | **0,31** |
| Density Complexity | 0,23 | 0,17 | 0,20 | **0,33** | **0,33** | **0,52** | 0,18 | 0,20 | **0,39** | 0,20 | -0,09 | 0,15 |
| Grouping Complexity | -0,14 | **-0,25** | -0,40 | **-0,33** | 0,06 | 0,16 | **-0,27** | **-0,36** | **-0,24** | -0,02 | 0,18 | 0,06 |
| Size Complexity | -0,21 | -0,19 | **-0,36** | **-0,32** | **0,57** | **0,46** | -0,16 | -0,42 | **0,38** | **0,24** | **0,34** | **0,39** |
| Total Complexity | -0,21 | 0,05 | 0,00 | 0,23 | **0,38** | **0,48** | -0,06 | -0,20 | 0,19 | **0,28** | 0,04 | 0,14 |
| Average Contrast | **-0,55** | -0,28 | -0,14 | 0,10 | -0,21 | -0,28 | -0,30 | -0,37 | -0,27 | -0,06 | 0,12 | 0,02 |
| Average Luminance | -0,10 | -0,15 | **0,42** | -0,24 | 0,25 | -0,01 | 0,03 | -0,03 | -0,03 | 0,19 | 0,03 | 0,08 |
| Average Saturation | **-0,41** | -0,10 | -0,20 | **0,47** | -0,34 | -0,35 | **-0,40** | 0,26 | 0,29 | -0,31 | -0,07 | -0,16 |
| Fractal Complexity | 0,22 | 0,01 | -0,26 | -0,25 | -0,02 | -0,06 | 0,00 | -0,06 | -0,28 | -0,05 | -0,21 | -0,19 |
| Hue Entropy | -0,08 | -0,18 | 0,11 | 0,32 | 0,27 | 0,35 | -0,12 | 0,21 | 0,06 | 0,30 | 0,32 | 0,30 |
| Hue Kurtosis | 0,28 | 0,29 | 0,01 | 0,37 | 0,37 | -0,03 | 0,17 | 0,06 | **0,43** | 0,05 | -0,11 | -0,04 |
| Hue Skewness | -0,25 | -0,32 | 0,03 | **0,47** | 0,38 | 0,07 | -0,04 | -0,08 | **-0,56** | 0,12 | -0,05 | -0,09 |
| Saturation Entropy | -0,31 | -0,07 | -0,14 | 0,05 | -0,38 | -0,15 | -0,25 | 0,19 | 0,13 | -0,22 | 0,21 | 0,00 |
| Saturation Kurtosis | 0,04 | 0,06 | 0,23 | **-0,55** | 0,21 | -0,07 | 0,17 | -0,05 | 0,28 | 0,34 | -0,09 | 0,07 |
| Saturation Skewness | 0,06 | -0,01 | 0,29 | **-0,49** | 0,20 | -0,04 | 0,22 | 0,29 | 0,35 | 0,37 | 0,11 | 0,21 |
| Luminosity Entropy | -0,36 | -0,32 | 0,07 | 0,16 | 0,05 | 0,05 | -0,25 | -0,22 | **-0,43** | 0,16 | 0,07 | 0,16 |
| Luminosity Kurtosis | 0,27 | 0,18 | 0,35 | 0,12 | **0,55** | 0,33 | 0,15 | **0,53** | 0,22 | 0,15 | 0,12 | 0,09 |
| Luminosity Skewness | 0,17 | 0,13 | **-0,41** | 0,13 | -0,33 | 0,07 | 0,00 | **0,45** | 0,10 | -0,12 | -0,09 | -0,11 |

We also observed that the earlier the level creation stage, the lower the correlation values of most image features (Table 2). The materials used in the virtual space design has a great influence on the correlation value. In the case of variant C (3D models without materials), a significant strong relationship weak relationship with the Average Luminance can be noticed. This correlation decreases after adding materials to the models (variants D with monochromatic materials and E with final materials) effecting with no significant relation in final level variant (F with lightning). Similar observation can be made with Saturation Kurtosis and Saturation Skewness giving the highest correlation values for variant with monochromatic materials (D) and also no significant relation in the final level variant. Another interesting observation can be made in first design stage (simple blockout - variant A). In such a simple block design, the color-based image features gave the highest correlation values with significant strong negative relationship for Average Contrast (Pearson $r$ = -0,55, $p < 0{,}01$). This relation weakens with the addition of final models and textures. It is also worth

paying attention to the fact that with the appearance of the final materials, the sign of the correlation for the Size Complexity image feature changes from negative to positive relationship.

It must be remembered that the value at a given point for the correlating images feature shows the general tendency of the Impression Curve (increase or decrease of it) - not the exact values of it. To reproduce the value of the curve, it is necessary to know its value at one point at least. At the same time, the change in perception of virtual space (increase or decrease) is a feature shared by users (as shown in the research presented in [1]), while the assignment of a numerical value to the Impression Curve may depend on the user and the definition of the rating scale. Therefore, the use of the Immersion Curve value change in the automatic evaluation system of the game level is not only a more reliable, but also more universal (less dependent on the user).

### 4.2 Combined Image Features Correlation

Among the image features tested, the most common correlation between them and Impression Curve can be observed in seven cases (Table 2). They were divided into two groups:

- The most promising that gives the highest correlation values, especially in final level design variant (F). Those are: **Density Complexity, Size Complexity, Total Complexity and Balance Complexity**. This group formed a base set for all the combined set (and will be referred to as DTSBC hereinafter).
- The second most promising with a little lower correlation value than the first group or high relationship with variants other than final level design (F). Those are: **Grouping Complexity, Average Contrast, Average Saturation**. They were added, in every possible combination, to the first group and checked for improvement in relationship strength.

In addition to the above, color-based image features of Entropy, Kurtosis and Skewness for Hue, Saturation and Luminosity were also included in described sets as they presented significant correlation values in different stages of design (especially in early stages A to E). Image features in those sets were combined using multiple regression. From all the combined sets, those with the best Pearson's correlation values were selected (Table 3).

There was significant strong or very strong positive relationship in all cases. The best results overall were achieved for the sets *DSTBC + Average Contrast + Average Saturation + Hue Entropy* and *DTSBC + Average Contrast + Average Saturation + Hue Entropy + Luminosity Kurtosis* where the latter works for a larger number of variants (thus it is more universal). In almost all cases, the combined feature sets correlated significantly better than the single ones included in them (Fig. 3). These isolated opposite cases arise when one feature in a combination did not correlate individually. It can be observed in variant G (geometrical changes) where combined result of DSTBC + Average Contrast is

**Table 3.** The best Pearson's correlation values for combined image features for all twelve level design variants. Features were combined using multiple regression. Three best correlated image features: Density Complexity + Size Complexity + Total Complexity + Balance Complexity (called DTSBC for short) were bases for all four combined sets. The correlation results are color-coded as a heatmap with a grayscale background color (the darker the color, the higher the correlation value). Very strong relationship ($r$ value higher than 0,70) was outlined with a white text color and bold text. All the results were significant ($p =< 0{,}01$). There was significant correlation in all cases, where the best results were achieved for sets DTSBC + Average Contrast + Average Saturation + Hue Entropy + Luminosity Kurtosis. We can observe that the more advanced level design stage (B to F) the stronger the correlation. Also, the combination of HSL Entropy, Kurtosis and Skewness can be useful for variants with lightning and weather changes. DTSBC - image features: Density Complexity + Size Complexity + Total Complexity + Balance Complexity; A - simple blockout; B - advanced blockout; C - models without materials; D - models with monochromatic materials; E - models with final materials; F - final level version; G - geometrical changes; L - lightening condition changes; W - weather changes; M - material changes; X - expression added; O - extra models added.

| | Level Design Variant | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | B | C | D | E | F | G | L | W | M | X | O |
| DSTBC + Average Contrast | 0,66 | 0,42 | 0,61 | 0,55 | 0,63 | **0,73** | 0,40 | 0,64 | 0,60 | 0,65 | 0,52 | 0,55 |
| DSTBC + Average Contrast + Average Saturation + Hue Entropy | 0,67 | 0,43 | 0,63 | 0,60 | **0,72** | **0,81** | 0,53 | 0,68 | 0,62 | **0,75** | 0,66 | 0,63 |
| DSTBC + Average Contrast + Average Saturation + Hue Entropy + Luminosity Kurtosis | 0,68 | 0,43 | 0,65 | 0,61 | **0,82** | **0,82** | 0,54 | **0,70** | 0,62 | **0,75** | **0,71** | 0,63 |
| DSTBC + Hue Kurtosis + Hue Skewness + Saturation Entropy | 0,63 | 0,47 | 0,62 | 0,59 | **0,71** | **0,74** | 0,46 | 0,55 | **0,77** | 0,68 | 0,52 | 0,57 |
| DSTBC + Hue Kurtosis + Hue Skewness + Saturation Entropy + Luminosity Entropy | 0,63 | 0,48 | 0,63 | 0,62 | **0,71** | **0,77** | 0,50 | **0,77** | **0,80** | 0,69 | 0,52 | 0,66 |

equal to single Average Saturation correlation value (but with negative sign). On the other hand, the combined sets presented strong and very strong relationship for those level variants that for a single feature had only a few weak or moderate relationships: M (material changes), X (added expression) and O (extra models added). We can also observe that the more advanced level design stage (B to F) the stronger the correlation (Table 2). Even the worst level design variant for single feature - geometrical changes (G) - now shows significant strong positive relationship (Pearson $r = 0{,}54$ with $p < 0{,}01$ at best).

There is significant difference in correlation values for color-based features (color, luminance as well as descriptive statistics for HSL) between single feature correlation (Table 2) and combined value using those image features (Table 3). The single feature correlation values are rather small or even not significant on later design variants (G to O). However, when they are combined with other image features, they have shown the highest or the second-highest correlation value. This happens even if, for a given variant of the level design, a single color-based feature did not show a correlation with the Impression Curve (mostly due to the high values of $p$).

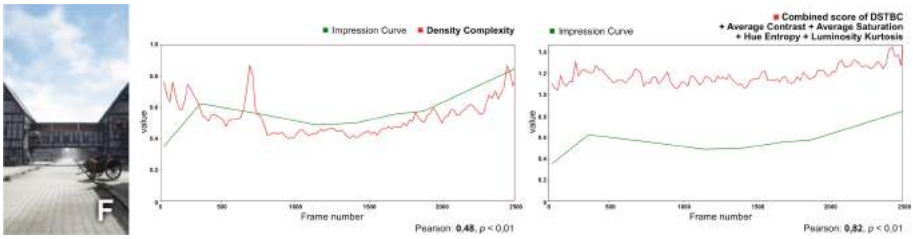



**Fig. 3.** Correlation plot examples for final level design variant (F variant, on the left). Two image feature correlation plots are presented: single image feature - Density Complexity (Pearson r = 0,48 with $p < 0{,}01$, center) and combined score of Density Complexity, Size Complexity, Total Complexity, Balance Complexity (DSTBC for short), Average Contrast, Average Saturation, Hue Entropy and Luminosity Kurtosis (Pearson r = 0,82 with $p < 0{,}01$, right). The combined score presents much higher correlation value than the component features separately with significant very strong positive relationship.

### 4.3 Best Features for Different Level Design Stages

Another aspect of the evaluation of the results was the changes of individual feature correlation at the subsequent stages of the game level design. Thanks to such approach, it was possible to assess the usefulness of the automatic evaluation method at different stages of the level design (from simple blockout with gray objects, trough materials and textures, to final design with lightning and atmospheric effects). The best image features or their combination to be used in such evaluation system will be the ones correlating regardless of the variant we are dealing with. The results for similar versions (such as first stage simple design been analyzed together) of the level were also compared. The image features with similar correlation values for each variant were considered the most promising. Such approach allowed us to eliminate those image features that correlated only in a single case.

For early stages of design that use blockout (A and B) we observed a significant correlation with the color-based image features, where at later stages (C to F) features from saliency and motion maps group showed better results (Table 2). What is more, most of the color-based image features tends to not show significant correlation at later stages, especially at the final level design. The exception here are the values of Hue Kurtosis, Hue Skewness and Luminosity Entropy for the weather change variant (W) with strong relationship. This showed that those image features could be added to the combined set to help verify how atmospheric effects affects users' impression of virtual space.

It is worth noticing that combination of HSL Entropy, Kurtosis and Skewness showed high or very high significant correlation results for variant the most visually different from the rest - L - where lightning conditions are changed (day to night). For example, set combined of DSTBC + Hue Kurtosis + Hue Skewness + Saturation Entropy + Luminosity Entropy resulted for weather changes level design variant in very strong relationship (Pearson's $r = 0{,}80$, $p < 0{,}01$).

Even this set is not universal for the whole process (other combined sets have given higher correlation results), it could improve Impression Curve estimation at design with rapid lightning or weather changes. At the same time, changes in lighting, weather or materials did not affect the shape of the Impression Curve, but did have a significant effect on the image features correlation.

## 5 Conclusion

The aim of this study was to investigate the existence of the correlation of the image features with the Impression Curve for game level design. The study shows that even a single image feature can describe the impression value with good precision (strong relationship, Pearson $r > 0{,}5$) for final level design. Best results were obtained by combining several image features using multiple regression (for image features: Density Complexity, Size Complexity, Total Complexity, Balance Complexity, Average Contrast, Average Saturation, Hue Entropy and Luminosity Kurtosis combined using multiple regression). Such set produced very strong positive relationship with Impression Curve values (Pearson $r = 0{,}82$ with $p < 0{,}01$ at best). What is more, significant correlation (strong to very strong) was observed regardless of level design variant, which makes it possible to apply image analysis at every stage of the level design process, making such solution more universal. The study also analyzed the possibility to use a different set of image features at different level design stages to get the highest results. The color-based image features were the best in this regard to be used at blockout stage of design (A and B, moderate to strong relationship) and HSL Entropy, Kurtosis and Skewness at stages with lightning and weather changes (L and W, moderate to strong relationship).

We saw many development opportunities for the idea of the automatic evaluation of game level design. The tests can be performed on production versions of game levels (taken from popular games), data about Impression Curve as well as image analysis could be obtained in real time or the study could be to extend with an Eye-tracker (to verify if there is a relation between the eye movement and Impression Curve). Also, the joined signals of EEG and Eye-tracker data can be analyzed as in [15]. Those four research ideas are carried out by us at the time of writing this article and the results will be published in the future. Improvements can be made in terms of calculation time as well, with a goal of real time analysis. For example, by applying faster classified like the one used in [12] for HUD detection, to obtain saliency maps in short time.

To sum up, the study has shown that Impression Curve value, and hence, the user impression of virtual 3D space, can be estimated with a high degree of certainty by automatic evaluation using image analysis of such level walkthrough. We propose usage of the combined image feature set for better estimation of Impression Curve. For early stages of design (blockout and models without textures) different set can be used to increase the relationship strength.

## References


1. Andrzejczak, J., Osowicz, M., Szrajber, R.: Impression curve as a new tool in the study of visual diversity of computer game levels for individual phases of the design process. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12141, pp. 524–537. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50426-7_39
2. Dudek, P., Wang, B.: A fast self-tuning background subtraction algorithm. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 401–404 (2014)
3. Jamil, N., Sembok, T.M.T., Bakar, Z.A.: Noise removal and enhancement of binary images using morphological operations. In: Proceedings of the International Symposium on Information Technology, pp. 1–6 (2008)
4. Milanfar, P.: A tour of modern image filtering: new insights and methods, both practical and theoretical. IEEE Signal Process. Mag. **30**(1), 106–128 (2013)
5. Magel, K., Alemerien, K.: GUIEvaluator: A metric-tool for evaluating the complexity of graphical user interfaces. In: Proceedings of the International Conference on Software Engineering and Knowledge Engineering, SEKE (2014)
6. Peli, E.: Contrast in complex images. J. Opt. Soc. Am. A Opt. Image Sci. Vis. **7**, 2032–2040 (1990)
7. Kumari, S., Vijay, R.: Image quality estimation by entropy and redundancy calculation for various wavelet families. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. **4**, 027–034 (2012)
8. Simons, D.J., Franconeri, S.L.: Moving and looming stimuli capture attention. Percept. Psychophys. **65**, 999–1010 (2003)
9. Della-Bosca, D., Patterson, D., Roberts, S.: An analysis of game environments as measured by fractal complexity. In: Proceedings of the Australasian Computer Science Week Multiconference (ACSW 2017), Association for Computing Machinery, USA, Article 63, pp. 1–6 (2017)
10. Kaehler, A., Bradski, G.: Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library. O'Reilly Media, Sebastopol (2016). ISBN 9781491937969
11. Engineering Statistics Handbook: NIST/SEMATECH e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/. Accessed 02 Feb 2022
12. Kozłowski, K., Korytkowski, M., Szajerman, D.: Visual analysis of computer game output video stream for gameplay metrics. In: Krzhizhanovskaya, V.V., et al. (eds.) ICCS 2020. LNCS, vol. 12141, pp. 538–552. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-50426-7_40
13. Lazar, J., et al.: Research Methods in Human-Computer Interaction, 2nd edn. Wiley, Hoboken (2017). ISBN 9780128053904
14. Rogalska, A., Napieralski, P.: The visual attention saliency map for movie retrospection. Open Phys. **16**(1), 188–192 (2018)
15. Szajerman, D., Napieralski, P., Lecointe, J.-P.: Joint analysis of simultaneous EEG and eye tracking data for video images. COMPEL Int. J. Comput. Math. Electr. Electron. Eng. **37**(5), 1870–1884 (2018)
16. Ølsted, P.T., Ma, B., Risi, S.: Interactive evolution of levels for a competitive multiplayer FPS, evolutionary computation (CEC). In: 2015 IEEE Congress on IEEE, pp. 1527–1534 (2015)

# ACCirO: A System for Analyzing and Digitizing Images of Charts with Circular Objects

Siri Chandana Daggubati and Jaya Sreevalsan-Nair(✉)

Graphics-Visualization-Computing Lab (GVCL), International Institute of Information Technology, Bangalore (IIITB), 26/C, Electronics City, Bengaluru 560100, Karnataka, India
{daggubati.sirichandana,jnair}@iiitb.ac.in
http://www.iiitb.ac.in/gvcl

**Abstract.** Automated interpretation of digital images of charts in documents and the internet helps to improve the accessibility of visual representation of data. One of the approaches for automation involves extraction of graphical objects in the charts, *e.g.,* pie segments, scatter points, etc., along with its semantics encoded in the textual content of the chart. The scatter plots and pie charts are amongst the widely used infographics for data analysis, and commonly have circle objects. Here, we propose a chart interpretation system, ACCirO (Analyzer of Charts with Circular Objects), that exploits the color and geometry of circular objects in scatter plots, its variants, and pie charts to extract the data from its images. ACCirO uses deep learning-based chart-type classification and OCR for text recognition to add semantics, and templatized sentence generation from the extracted data table for chart summarization. We show that image processing and deep learning approaches in ACCirO have improved the accuracy compared to the state-of-the-art.

**Keywords:** Image processing · Scatter plots · Pie charts · Dot plots · Bubble plots · Circle geometry · Circle Hough Transform (CHT) · Spectral clustering · Chart data extraction · Text recognition

## 1 Introduction

Given the ubiquity of charts for visualizations, there is recent interest in automating its interpretation. The motivating applications include filtering significant charts from image databases, generating visual question-answering (QA) systems, etc., given the raster format of the charts. Though many image processing techniques have been used, there are still gaps in existing technology for automated chart interpretation owing to the diversity and complexity of chart content

This work has been generously supported by the Machine Intelligence and Robotics (MINRO) Grant, Government of Karnataka. This paper has benefitted from discussions with GVCL members and peers at IIITB.


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 605–612, 2022.
https://doi.org/10.1007/978-3-031-08757-8_50

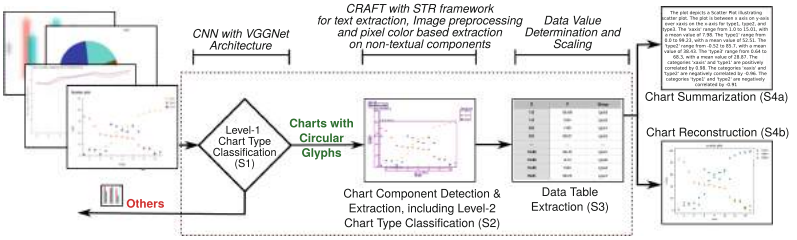



**Fig. 1.** Our proposed workflow for ACCirO is an automated system for data extraction from images of charts with circular objects and color, and for which geometric and color-based information extraction techniques are used.

and the requirement of human-in-the-loop in most cases. Here, we consider pie charts, scatter plots, and its variants, *dot plots*, and *bubble plots*, with circular objects. The source images are from documents in portable document format (.pdf), websites, outputs from plotting tools, and curated image databases [7]. The data extracted from images comes from its non-textual and textual content in charts. The non-textual content implies the geometric objects as per chart type, *e.g.,* pie segments and scatter points. The text content is from chart, legend, and axes titles, which are localized using annotation and extracted using image processing and text recognition.

When using the second-order gradient tensor field-based approach for object extraction from charts [4,5], we observe that only regions with high color gradients (edges and corners) are extracted, *e.g.,* boundaries of bar objects. But in pie charts, the gradients are concentrated in the corners of the largest rectangle enclosed within the pie owing to the high curvature gradient. At the same time, the pie chart has a circle geometry that can be exploited for sector extraction. Thus, we propose using color and geometry information in pie charts for automated annotation and data extraction. Given that circle geometry is predominantly used for scatter points in scatter, dot, and bubble plots, our proposed method is generalized for our selected four chart types.

Thus, our contribution is integrating an end-to-end system, ACCirO (Analyzer for Chart images with C*ir*cular Objects), generalized for four chart types. ACCirO has a four-step workflow (Fig. 1): $\mathbf{S_1}$ the chart classification, $\mathbf{S_2}$ a novel color-based annotation along with text extraction, $\mathbf{S_3}$ a novel color-based data extraction, $\mathbf{S_{4a}}$ text summarization and $\mathbf{S_{4b}}$ chart reconstruction. ACCirO specifically works for charts where color encodes class information and improves on $\mathbf{S_2}$ and $\mathbf{S_3}$ in BarChartAnalyzer [4], and ScatterPlotAnalyzer [5]. *Alpha blending* leads to the blended colors in bubble plots, which are different from the colors given in the chart legend. Our algorithm in $\mathbf{S_3}$ addresses the challenge of computing the color, radius, and center of constituent circles in overlap regions.

***Related Work:*** We generalize data extraction for different chart types, as is the current focus [2,4,5,8], but by using color information exclusively. CHT [6] has been used for object extraction from charts [8]. For the chart-types with circular objects in the foreground with non-textured background, CHT suffices.

## 2 The Workflow of ACCirO

We propose a fully automated workflow for ACCiro for pie charts and dot, bubble, and scatter plots. We consider bubble and dot plots as variants of scatter plots, owing to the similarity in using circular objects as graphical objects with positional information. The four key components of our workflow (Fig. 1) are: ($\mathbf{S_1}$) chart type classification, $\mathbf{S_2}$ chart component detection and extraction, and $\mathbf{S_3}$ data table extraction, optionally followed by $\mathbf{S_{4a}}$ chart summarization or $\mathbf{S_{4b}}$ chart reconstruction. We use the implementation from our previous work [4,5] for $\mathbf{S_1}$ to pick scatter plots and pie charts, $\mathbf{S_{4a}}$, and $\mathbf{S_{4b}}$. Scatter plots are further subclassified to its variants based on position and size variations of scatter points.

***$S_2$: Chart Component Detection and Extraction***

A chart is structurally composed of specific elements, referred to as *chart components*, whose characteristic properties in its raster format are exploited for their extraction. The seven components are: *canvas*, *legend*, *chart title*, *XY-axis titles*, and *XY-axis labels*. The region of the chart that *contains* the graphical objects, *e.g.,* pie sectors, scatter points, and bounded by axes, is the *canvas*. The process of localizing and retrieving them from the chart images is called *chart component extraction*. A separate component-wise analysis is more effective for stepwise chart interpretation than joint extraction using the entire image.

We first extract textual content by using a DL-based Optical Character Recognition (OCR), namely, Character Region Awareness for Text Detection (CRAFT), followed by a scene text recognition framework (STR), as used in ScatterPlotAnalyzer. This text is now removed to extract the canvas and graphical markers/objects in the legend. But, the filtered image still contains "noise" such as axis lines, gridlines, ticks, and small text fragments missed by the OCR.

In pie charts, we use CHT to extract the entire pie from the image, and the unique pixel colors in the pie pixels are used to locate objects in the legend. In scatter plots and their variants, object extraction using geometry is not reliable owing to the variety in marker styles and the presence of overlapping scatter points. So we initially remove axes and gridlines based on the property of the periodic arrangement of their straight pixels lines. We then locate the color-based clusters of pixels and tag them as "objects." The color histogram of the image gives colors of high frequency needed for the localization of pixel clusters. The bounding box of pie and axes in scatter plots gives the canvas.

Legend extraction follows after the canvas extraction in our workflow to accommodate cases of legend being placed in the canvas. The color-based clusters with relatively smaller pixel coverage and placed adjacent to text are identified as legend markers with corresponding labels. The final step is to semantically classify the textboxes outside the canvas region based on their role, *i.e.,* chart title, axes titles, and labels. The axes are usually found at the bottom, and the left of the canvas region, and the chart title at its top. In legend-free pie charts, text boxes in the proximity of arc centers of the pie sectors give the class information.

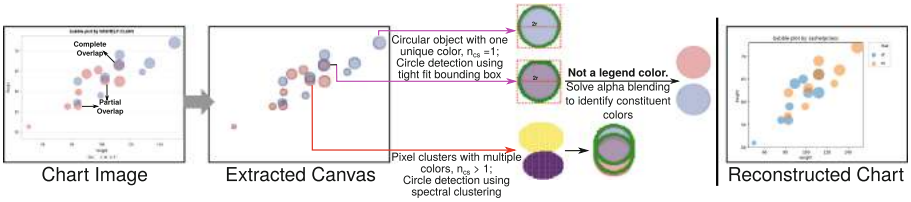



**Fig. 2.** Our proposed method for circular object extraction from bubble chart images.

### *$S_3$: Data Table Extraction from Graphical Objects*

We extract information from the pixel clusters in the image space and convert them to data space using the cues from the extracted text.

***Pixel-Based Data Extraction from Pie Charts:*** Percentage data is obtained from the sector area, which is determined using the fraction of pixel counts in the sector [2]. We implement this on the "donut" with one-third of the radius of the pie removed from the center. The donut is used instead of the entire pie to reduce the discrepancy from the missing pixels owing to text removal in the pie region. The sectors are then mapped to their labels based on colors. In legend-free charts, sectors are mapped to the closest text labels.

***Pixel-Based Data Extraction from Scatter, Dot, and Bubble Plots:*** Here, the 2D data is encoded in the positional vectors of the scatter object. The variants use additional visual encodings for more attributes. Such as, the height of stacks of dots in dot plots represents bar height in an equivalent bar chart, and bubble plots use the size and color of objects to encode additional attributes. Contours are extracted for color pixel clusters from ]step2. The positional information of scatter points is determined using the contour centroids of the clusters. In the case of overlapping points, the number of points involved is computed based on the ratio of contour area with the smallest contour observed in the chart. We then use k-means clustering to get centroids in the overlapping region. In the case of a dot plot, we determine the count of objects stacked with the same x-coordinate value of contour centroids. This count is given a class label using the legend color or the x-tick mark label.

Bubble plots have the unique challenge of overlapping circular objects with varying sizes and transparencies. The resultant alpha blending of overlapped scatter object regions poses a challenge in identifying the number and parameters of the constituent scatter objects. The resultant color from alpha blending is given by: $C = \alpha.F + (1-\alpha).B$, where $F$ and $B$ are the foreground and background colors respectively; and $0 \leq \alpha \leq 1$. We resolve the challenge using contour colors (Fig. 2). To estimate the number of overlapping points in a contour, we get $n_{cs}$ segments based on the unique colors in contours. Using the value of $n_{cs}$, we extract the center, radius, and class label of the scatter point. If $n_{cs} = 1$, the center and radius of the scatter point are given by a tight-fitting bounding box of the contour. When $n_{cs} > 1$, we use spectral clustering in the pixel cluster of contour and determine circle parameters with best-fit circle regression of $n_{cs}$

clusters. In all cases, we use the class label corresponding to its contour color, as given in the legend.

For those circles which do not correspond to legend colors in the above two cases, we use the blending equation to solve for the legend colors and corresponding transparency, $\alpha$, values that give the resultant blended colors $C$. We use an *exhaustive* search of the solution space to find the closest solution. We assume that the color $C$ of the spectral cluster is a blend of two or more legend colors. Hence, we solve for $2 \leq p \leq m = |C_L|$, for $m$ legend colors and the set of legend colors $C_L$. We use $mC_p$ combinations of colors, where each experimental run uses a subset of legend colors $\{C_1, C_2, \ldots, C_p\}$, where $C_i \in C_L$, $\forall i \in [1, p]$. The blending equation is now modified as a linear combination of $C_i$:

$$C^* = \sum_{i=1}^{p} \alpha_i . C_i, \text{ where } \sum_{i=1}^{p} \alpha_i = 1 \text{ and } 0 < \alpha_i < 1, \forall i \in [1, p].$$

We can reduce the search space by limiting the value of $\alpha_1$ to be a value in the interval $[0.5, 0.95]$, with a step size of 0.05. We also implement a greedy algorithm terminating the search when the closest color is obtained.

***Data Transformation for Scatter Plots:*** Finally, to transform data in pixel space to the numerical space, we use the scaling factor computed from the semantics of the text, as done in ScatterPlotAnalyzer. However, this does not work for circle size encoding in bubble charts, as the factor for circle sizes can be obtained from either the size legend (as present in some charts) or the parameter setting used in different plotting tools *e.g.,* DPI, for the chart generation. Thus, the exact data mapping for bubble charts will be explored in future.

## 3 Experiments and Results

***Qualitative Assessment:*** Through the visualizations, we observe that our method performs superior to the tensor field-based method in ScatterPlotAnalyzer [5] (Fig. 3). The color-based data extraction technique is an improvement over tensor fields [5] in the case of cluttered scatter points. Figure 4 shows outputs at different stages of ACCirO for a sample of each chart type, demonstrating similarities between the reconstructed and sources chart images. Visual analysis of the results of ACCiro on pie charts and scatter plots images in the FigureQA [7] gives reconstruction accuracy of ∼90.11% and ∼90.5%, respectively. To improve the circle detection using CHT [6] in a pie chart, which has an average of 96% accuracy, advanced circle detection methods such as RANSAC may be used.

***Quantitative Assessment:*** For quantitative assessment, we use synthetically generated chart images from publicly available data sources, *e.g., Kaggle*. We have generated a set of 15 images each for pie charts, dot and bubble charts, and 24 for scatter plots. In total, we use a test set of 69 images here.

We use the F1 Score and MAPE (Mean Absolute Percentage Error) metrics to measure the success and failure rates of the performance of ACCirO (Table 1). We

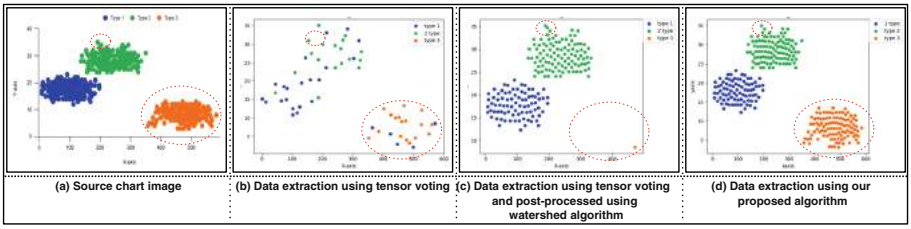



**Fig. 3.** Comparison of (a) source image and charts reconstructed using similar methods: (b) ScatterPlotAnalyzer [5], (c) modified ScatterPlotAnalyzer, and (d) our color-based method, for a multiclass scatter plot with a high degree of overlap of scatter points. The cluster of scatter points (red dotted ellipses) extracted is highlighted. (Color figure online)

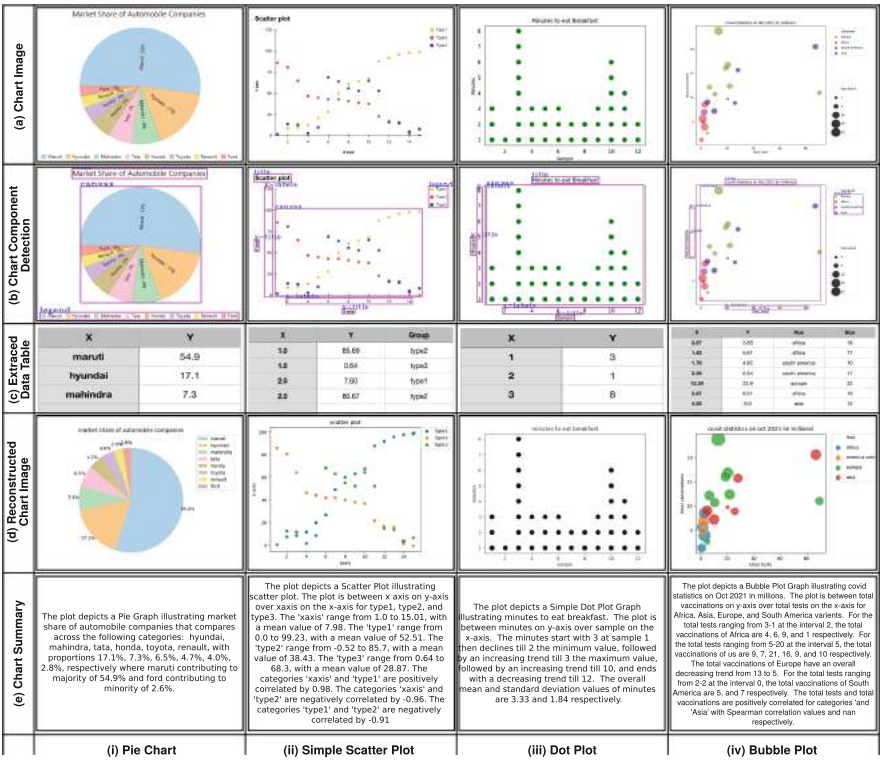



**Fig. 4.** Different stages of chart data extraction followed by chart reconstruction and summarization from the sample source images of (i) pie chart, (ii) scatter plot, (iii) dot plot, and (iv) bubble plot. The text summary is best visible at 220+% zoom level.

**Table 1.** Accuracy of data table extraction using ACCirO

| Chart type → Accuracy measure ↓ | Pie chart | Dot plot | Bubble plot | Scatter plot | Overall measures |
|---|---|---|---|---|---|
| Average precision | 0.94 | 1.00 | 1.00 | 0.97 | **0.96** |
| Average recall | 1.00 | 1.00 | 0.99 | 0.95 | **0.95** |
| Success rate: F1 score $> 0.8$ | 93% | 100% | 100% | 96% | **96.4**% |
| Success rate: MAPE $< 0.2$ | 100% | 100% | 93% | 84% | **88.7**% |

consider the data extraction as a *success* with F1 Score $> 0.8$ as in [3]. Despite the smaller test dataset, the data extraction accuracy of ACCirO from scatter plots surpasses the state-of-the-art methods with F1-Score 97%, compared to 90.5% and 88% for MECDG [1] and Scatteract [3], respectively. Our data extraction for dot plots is 100% accurate, owing to the structured point layout. Even for bubble plots, despite the complex challenges due to transparency and overlapping points, we get an F1-Score of 100%. It must be noted that, since the bubble/object size is in pixel measure, we exclude it from the F1 score computation. We observe that the normalized radius values are closer to raw data.

To determine the numerical precision errors in $\mathbf{S_3}$, we compare the difference between the source and extracted data values using the Mean Absolute Percentage Error (MAPE) as in [4]. Here, our alternative definition for the *success* of data extraction is when the error rate MAPE $< 0.2$. MAPE is augmented in the case of omission and precision errors owing to cluster centroids overlapping with scatter points and pixel space to data space transformations, respectively. Pie charts have been an exception for error-free data extraction of percentage values.

## 4 Conclusions

ACCirO has two known limitations which are to be resolved in future work. Firstly, owing to ACCiro being a color-based method, it fails in cases where the shape and texture of scatter points encode class or type information. Secondly, the STR text recognition model fails to interpret superscript symbols, and recognition of 'o', '0', and '−.' In summary, our proposed color-based end-to-end chart image interpretation system, ACCirO, has been generalized for the chart with circular objects, such as pie charts, and scatter, dot and bubble plots.

## References

1. Chen, L., Zhao, K.: An approach for chart description generation in cyber-physical-social system. Symmetry **13**(9), 1552 (2021)
2. Choi, J., Jung, S., Park, D.G., Choo, J., Elmqvist, N.: Visualizing for the non-visual: enabling the visually impaired to use visualization. In: Computer Graphics Forum, vol. 38, pp. 249–260. Wiley Online Library (2019)

3. Cliche, M., Rosenberg, D., Madeka, D., Yee, C.: Scatteract: automated extraction of data from scatter plots. In: Ceci, M., Hollmén, J., Todorovski, L., Vens, C., Džeroski, S. (eds.) ECML PKDD 2017. LNCS (LNAI), vol. 10534, pp. 135–150. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71249-9_9
4. Dadhich, K., Daggubati, S.C., Sreevalsan-Nair, J.: BarChartAnalyzer: digitizing images of bar charts. In: International Conference on Image Processing and Vision Engineering (IMPROVE), pp. 17–28. INSTICC, SciTePress (2021)
5. Dadhich, K., Daggubati, S.C., Sreevalsan-Nair, J.: ScatterPlotAnalyzer: digitizing images of charts using tensor-based computational model. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12746, pp. 70–83. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77977-1_6
6. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM **15**(1), 11–15 (1972)
7. Kahou, S.E., Atkinson, A., Michalski, V., Kádár, Á., Trischler, A., Bengio, Y.: FigureQA: an annotated figure dataset for visual reasoning. CoRR abs/1710.07300 (2017)
8. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: Revision: automated classification, analysis and redesign of chart images. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software & Technology, pp. 393–402 (2011)

# Learning Scale-Invariant Object Representations with a Single-Shot Convolutional Generative Model

Piotr Zieliński(✉) and Tomasz Kajdanowicz

Wrocław University of Science and Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
p.zielinski@pwr.edu.pl

**Abstract.** Contemporary machine learning literature highlights learning object-centric image representations' benefits, i.e. interpretability, and the improved generalization performance. In the current work, we develop a neural network architecture that effectively addresses the task of multi-object representation learning in scenes containing multiple objects of varying types and sizes. In particular, we combine SPAIR and SPACE ideas, which do not scale well to such complex images, and blend them with recent developments in single-shot object detection. The method overcomes the limitations of fixed-scale glimpses' processing by learning representations using a feature pyramid-based approach, allowing more feasible parallelization than all other state-of-the-art methods. Moreover, the method can focus on learning representations of only a selected subset of types of objects coexisting in scenes. Through a series of experiments, we demonstrate the superior performance of our architecture over SPAIR and SPACE, especially in terms of latent representation and inferring on images with objects of varying sizes.

**Keywords:** Deep autoencoders · Representation learning · Generative models · Scene analysis

## 1 Introduction

The ability to discriminate and reason about individual objects in an image is one of the important tasks of computer vision, which is why object detection and instance segmentation tasks have drawn vast attention from researchers throughout the years. The latest advances in artificial intelligence require a more insightful analysis of the image to provide more profound reasoning about its contents. It can be achieved through representation learning, which facilitates extracting useful information about objects, allowing transferring more general knowledge to other tasks [2]. One can see multi-object representation learning as a natural extension to the aforementioned computer vision tasks. Here, the objective is to produce a valuable abstract feature vector of each of the inferred


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 613–626, 2022.
https://doi.org/10.1007/978-3-031-08757-8_51

objects and hence produce a structured representation of the image, allowing for its more insightful understanding.

Recently, the most successful methods are based on the variational autoencoder (VAE) framework [16,21], with structured latent space, which includes individual objects' representations. The original approach consists in extracting object latent vectors with a recurrent network [1,3,7–9]. Alternatively, each object's representation can be produced with a single forward pass through the network by employing a convolution-based single-shot approach [4,18]. However, these methods are limited by a single feature map utilized to create objects' latent vectors and hence cannot be used when object sizes vary.

In this paper, we propose a single-shot method for learning multiple objects' representations, called *Single-Shot Detect, Infer, Repeat* (SSDIR[1]). It is a convolutional generative model applying the single-shot approach with a feature pyramid for learning valuable, scale-invariant object representations. By processing multi-scale feature maps, SSDIR can attend to objects of highly varying sizes and produce high-quality latent representations directly, without the need of extracting objects' glimpses and processing them with an additional encoder network. The ability to focus on individual objects in the image is improved by leveraging knowledge learned in an SSD [19] object detection model. In experiments, we compare the SSDIR model on multi-scale scattered MNIST digits, CLEVR [15] and WIDER FACE [23] datasets with other single-shot approaches, proving the ability to focus on individual objects of varying sizes in complicated scenes, as well as the improved quality of objects' latent representations, which can be successfully used in other downstream problems, despite the use of an uncomplicated convolutional backbone.

We summarize our contributions as follows. We present a model that enhances multi-object representation learning with a single-shot, feature pyramid-based approach, retaining probabilistic modeling of objects. We provide a framework for generating object representations directly from feature maps without extracting and processing glimpses, allowing easier scaling to larger images. We compare the method with other single-shot multi-object representation learning models and show its ability to attend to objects, the improved latent space quality, and applicability in various benchmark problems.

## 2 Related Works

Multi-object representation learning has recently been tackled using unsupervised, VAE-based models. Two main approaches include sequential models, attending to a single object or part of the image at a time, and single-shot methods, which generate all representations in a single forward pass through the network.

The original approach to this problem was presented by Ali Eslami *et al.* in [1]. The *Attend, Infer, Repeat* (AIR) model assumes a scene to consist of objects,

[1] Code available at: https://github.com/piotlinski/ssdir.

represented with *what* vector, describing the object's appearance, *where* vector indicating its position on the image and *present* vector, describing if it is present in the image, controlling termination of the recurrent image processing. The model attends to a single object at a time, generating representations sequentially with a recurrent network until a non-present object is processed. Other studies, including [10] and [22] proposed a different approach, where objects representations are learned using Neural Expectation-Maximization, without structuring the latent representations explicitly. These methods suffer from scaling issues, not being able to deal with complex scenes with multiple objects.

Alternatively, an image might be described with a scene-mixture approach, as in MONet [3], IODINE [9] and GENESIS [7,8]. Here, the model does not explicitly divide the image into objects but instead generates masks, splitting the scene into components, which the model encodes. In the case of MONet and GENESIS, each component is attended and encoded sequentially, while IODINE uses amortized iterative refinement of the output image. However, these methods are not a good fit for learning object representations in an image, as scene components usually consist of multiple objects. Furthermore, masks that indicate particular objects limit the model's scalability due to this representation requiring more memory than bounding box coordinates.

GENESIS belongs to a group of methods, which focus on the ability to generate novel, coherent and realistic scenes. Among them, one should notice recent advances with methods leveraging generative adversarial networks (GANs), such as RELATE [6] or GIRAFFE [20]. Compared to VAE-based methods, they can produce sharp and natural images, which are more similar to original datasets. However, these models do not include an explicit image encoder, and therefore cannot be applied for multi-object representation learning directly. What is more, the process of training GANs tends to be longer and more complicated than in the case of VAEs.

Recently, methods such as GMAIR [24] postulate that acquiring valuable *what* object representations is crucial for the ability to use objects encodings in other tasks, such as clustering. Here, researchers enhanced the original *what* encoder with Gaussian Mixture Model-based prior, inspired by the GMVAE framework [11]. In our work, we also emphasize the importance of the *what* object representation and evaluate its applicability in downstream tasks.

One of the promising methods of improving model scalability of VAE-based multi-object representation learning models was presented in SPAIR [4], where the recurrent attention of the original AIR was replaced with a local feature maps-based approach. In analogy to single-shot object detection models like SSD [19], the SPAIR first processes image with a convolutional backbone, which returns a feature map with dimensions corresponding to a fixed-sized grid. Each cell in the grid is then used to generate the locations of objects. Objects representations' are inferred by processing these cells sequentially, generating *what*, *depth* and *present* latent variables, describing its appearance, depth in the scene, and the fact of presence. This approach has recently been extended in SPACE [18], which fixes still existing scalability issues in SPAIR by employing parallel

latent components inference. Additionally, the authors used the scene-mixture approach to model the image background, proving to be applicable for learning objects' representations in more complex scenes. However, both methods rely on a single grid of fixed size, which makes it difficult for this class of models to attend to objects of highly varying sizes. What is more, both of them employ glimpse extraction: each attended object is cut out of the input image and processed by an additional encoder network to generate objects' latent representations; this increases the computational expense of these methods.

Latest advances in the field of multi-object representation learning try to apply the aforementioned approaches for inferring representations of objects in videos. SQAIR [17] extends the recurrent approach proposed in AIR for sequences of images by proposing a propagation mechanism, which allows reusing representations in subsequent steps. A similar approach was applied to single-shot methods by extending them with a recurrent network in SILOT [5] and SCALOR [14]; here, the representations were used in the object tracking task. An interesting approach was proposed by Henderson and Lambert [12]. Authors choose to treat each instance within the scene as a 3D object; the image is then generated by rendering each object and merging their 2D views into an image. This allows for a better understanding of objects' representations, at the cost of significantly higher computational complexity.

## 3 Method

SSDIR (**S**ingle-**S**hot **D**etect, **I**nfer, **R**epeat) is a neural network model based on a variational autoencoder architecture [16,21] as shown in Fig. 1; its latent space consists of structured objects' representations $\boldsymbol{z}$, enhanced by leveraging knowledge learned in a single-shot object detection model SSD [19], both sharing the same convolutional backbone.

### 3.1 The Proposed Model: SSDIR

Our model extends the idea of single-shot object detection. Let $\boldsymbol{x}$ be the image representing all relevant (i.e. detected by the SSD) objects present in the image. SSDIR is a probabilistic generative model, which assumes that this image is generated from a latent representation $\boldsymbol{z}$ according to a likelihood distribution. This representation consists of a set of latent vectors assigned to each grid cell in the feature pyramid of SSD's convolutional backbone and is sampled from a prior distribution $p(\boldsymbol{z})$. Since the likelihood distribution is unknown, we approximate it using the decoder network $\theta$, which parametrizes the likelihood $p_\epsilon(\boldsymbol{x}|\boldsymbol{z})$. Then, the generative model can be described as a standard VAE decoder (1).

$$p(\boldsymbol{x}) = \int p_\epsilon(\boldsymbol{x}|\boldsymbol{z})\, p(\boldsymbol{z})\, d\boldsymbol{z} \tag{1}$$

To do inference in this model, SSDIR applies variational method and approximates the intractable true posterior with a function $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p(\boldsymbol{z}|\boldsymbol{x})$,

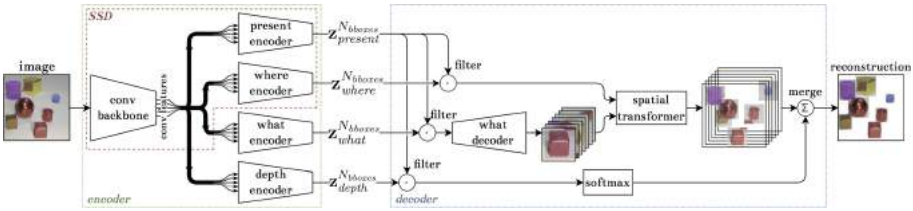



**Fig. 1.** Illustration of the SSDIR model. It consists of two fully-convolutional neural networks: an *encoder* and a *decoder*. The *encoder* uses a convolutional backbone as a feature extractor, which builds a pyramid of multi-scale features processed by each latent component encoder. Each object's position $\boldsymbol{z}_{where}$ and presence $\boldsymbol{z}_{present}$ latent vectors are computed using a trained object detection model *SSD*, indicating grid cells, which refer to detected objects; $\boldsymbol{z}_{what}$ and $\boldsymbol{z}_{depth}$ are computed with additional convolutional encoders, which process the feature maps from the pyramid in a similar manner to *SSD*. In the decoder, all latents are filtered to include only present objects for reconstructions. *What* decoder reconstructs appearances of each present object, which are then put in their original place with an affine transformation in the *spatial transformer* module. Finally, object reconstructions are merged using weighted sum, created by applying *softmax* on objects' *depth* latents.

parametrized by $\phi$ (encoder parameters). This allows us to use ELBO (Evidence Lower Bound) as the loss function (2):

$$\mathcal{L}(\theta, \phi) := \mathbb{E}_{z \sim q_\phi(\boldsymbol{z}|\boldsymbol{x})} \left[\log p_\cdot \left(\boldsymbol{x}|\boldsymbol{z}\right)\right] - D_{KL}\left(q_\phi\left(\boldsymbol{z}|\boldsymbol{x}\right) \| p\left(\boldsymbol{z}\right)\right) \tag{2}$$

where $D_{KL}$ is the KL divergence.

**Object Representation.** SSDIR extends the grid-based approach with a feature pyramid for object detection proposed in SSD to produce objects' latent representations. We assume each object can be described by four latent variables:

- $\boldsymbol{z}_{where} \in \mathbb{R}^4$ – the object's bounding box position and size,
- $\boldsymbol{z}_{present} \in \{0, 1\}$ – a binary value indicating if given cell contains any object,
- $\boldsymbol{z}_{what} \in \mathbb{R}^D$ – $D$-sized vector describing the object appearance,
- $\boldsymbol{z}_{depth} \in \mathbb{R}$ – a real number indicating how deep in the scene the given object was observed (we assume, that objects with a bigger value of $\boldsymbol{z}_{depth}$ appear in front of those with a lower value).

To simplify the process of objects discovery, we reuse a trained SSD model to get bounding box position and size, as well as the detected object class. SSDIR utilizes detections to produce $z_{where}$ and $z_{present}$ as shown in (3) and (4).

$$\boldsymbol{z}^i_{where} = \left[cx_i \; cy_i \; w_i \; h_i\right] \tag{3}$$

$$z^i_{present} \sim Bernoulli\left(\beta^i\right) \tag{4}$$

where:

i refers to the cell in the feature pyramid,
$cx$, $cy$ are the bounding box' center coordinates,
w, h are the bounding box' width and height dimensions,
$\beta^i = \begin{cases} \arg\max_k \boldsymbol{c}_i & \text{if an object detected in the cell,} \\ 0 & \text{otherwise,} \end{cases}$
$\boldsymbol{c}$ are the object's predicted class confidences.

The two remaining latent components: $\boldsymbol{z}_{what}$ and $\boldsymbol{z}_{depth}$ are modeled with Gaussian distributions, as shown in (5) and (6).

$$\boldsymbol{z}^i_{what} \sim \mathcal{N}\left(\boldsymbol{\mu}^i_{what}, \boldsymbol{\sigma}^i_{what}\right) \tag{5}$$

$$\boldsymbol{z}^i_{depth} \sim \mathcal{N}\left(\boldsymbol{\mu}^i_{depth}, \boldsymbol{\sigma}^i_{depth}\right) \tag{6}$$

where:

$\boldsymbol{\mu}_{what}$, $\boldsymbol{\mu}_{depth}$ are means, encoded with *what* and *depth* encoders,
$\boldsymbol{\sigma}_{what}$, $\boldsymbol{\sigma}_{depth}$ are standard deviations, which are treated as model's hyperparameters.

**SSDIR Encoder Network.** To generate the latent representation of objects contained in an image, we apply the feature pyramid-based object detection approach. The function of the encoder $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is implemented with a convolutional backbone (VGG11) accepting images of size $300 \times 300 \times 3$, extended with a feature pyramid, and processed by additional convolutional encoders, as shown in Fig. 1. Specifically, *where*, *present* and *depth* encoders contain single convolution layer with $3 \times 3$ kernels (1 in case of *present* and *depth* and 4 for *where* encoder) per each feature map in the pyramid, whereas *what* encoder may include sequences of convolution layers with ReLU activations, finally returning $D$-sized vector for each cell in each feature pyramid grid. The outputs of these encoders are used to generate latent vectors $\boldsymbol{z}_{where}$, $\boldsymbol{z}_{present}$, $\boldsymbol{z}_{what}$ and $\boldsymbol{z}_{depth}$.

The backbone's, as well as *where* and *present* encoders' weights are transferred from an SSD model trained with supervision for detection of objects of interest in a given task and frozen for training; *what* and *depth* encoders, which share the same pretrained backbone, are trained with the decoder network. Such architecture allows parallel inference, since neither latent component depends on any other, without the need of extracting glimpses of objects and processing them with a separate encoder network – in SSDIR latent representations are contained within feature maps directly, improving its scalability.

**SSDIR Decoder Network.** Latent representations of objects in the picture are forwarded to the decoder network to generate reconstructions of areas in the input image that contain objects of interest, i.e. those detected by the SSD network. First, the latent variables are filtered according to $\boldsymbol{z}_{present}$, leaving only those objects, which were found present in the image by the SSD network.

Next, per-object reconstructions are generated by passing filtered $\boldsymbol{z}_{what}$ vectors through a convolutional *what* decoder, producing $M$ images of size $64 \times 64 \times 3$, representing each detected object's appearance. These images are then translated and scaled according to the tight bounding box location $\boldsymbol{z}_{where}$ in the *spatial transformer* module [13]. The resulting $M$ $300 \times 300 \times 3$ images are merged using a weighted sum, with softmaxed, filtered $\boldsymbol{z}_{depth}$ as the weights. The output of the model might then be normalized with respect to the maximum intensity of pixels in the reconstruction to improve the fidelity of the reconstruction.

SSDIR does not require special preprocessing of the image, apart from the standard normalization used widely in convolutional neural networks. Originally, the background is not included in the reconstruction phase, since its representation is not crucial in the task of multi-object representation learning; we assume that this way SSDIR learns to extract the key information about all objects from the image. The background might however be reconstructed as well by including an additional $z_{what}$ encoder and treating the background as an extra object, which is transformed to fill the entire image and put behind all other objects.

The parallel nature of the model is preserved in the decoder. The operations of filtering, transforming, and merging are implemented as matrix operations, allowing good performance and scalability.

**Training.** The SSDIR model is trained with a modified ELBO loss function. We extend the original form (2), which intuitively includes reconstruction error of an entire image and KL divergence for latent and prior distributions with a normalized sum of each detected object's reconstruction error. This allows the model to reach high quality of reconstructions (and as a result – high quality of $\boldsymbol{z}_{what}$ latent representations) and correct order of objects' $\boldsymbol{z}_{depth}$, preserving transformation function continuity thanks to KL divergence-based regularization. The final form of the loss function is shown in (7).

$$\begin{aligned}\mathcal{L}(\boldsymbol{x}, \theta, \phi) = \alpha_{obj}\mathbb{E}_{\boldsymbol{z}}[\log p_{\theta}(\boldsymbol{x}|\boldsymbol{z})] + \alpha_{rec}\frac{1}{M}\sum_{i}^{M}\mathbb{E}_{z_i}[\log p_{\theta}(x_i|z_i)] \\ - \alpha_{what}D_{KL}(q_{\phi}(\boldsymbol{z}_{what}|\boldsymbol{x}) \| p(\boldsymbol{z}_{what})) \\ - \alpha_{depth}D_{KL}(q_{\phi}(\boldsymbol{z}_{depth}|\boldsymbol{x}) \| p(\boldsymbol{z}_{depth}))\end{aligned} \tag{7}$$

where:

$\mathbb{E}_{\boldsymbol{z}}[\log p_{\theta}(\boldsymbol{x}|\boldsymbol{z})]$ is the likelihood of the reconstruction generated by the decoder,
$\mathbb{E}_{z_i}[\log p_{\theta}(x_i|z_i)]$ is the likelihood of an $i$-th detected object reconstruction,
$\alpha_{obj}$, $\alpha_{rec}$, $\alpha_{what}$, $\alpha_{depth}$ are loss components coefficients, modifying the impact of each one on the learning of the model,
M is the number of objects detected by the SSD model in a given image.

In case of both $\boldsymbol{z}_{what}$ and $\boldsymbol{z}_{depth}$ we assume the prior to be a standard normal distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. The training objective is described by (8) for each image $\boldsymbol{x}_i$ in the training dataset. The model is trained jointly with gradient ascent

using Adam as the optimizer, utilizing the reparametrization trick for back-propagating gradients through the sampling process. The process of learning representations is unsupervised, although the backbone's and *where* and *present* encoders' weights are transferred from a pretrained SSD model.

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_i \mathcal{L}\left(\boldsymbol{x}_i, \theta, \phi\right) \tag{8}$$

**Table 1.** Differences between **SSDIR** and baseline methods. "*semi-*" indicates that the object detection model is trained with supervision, while the representation learning procedure is unsupervised. "*glimpses*" refers to the process of learning object's $\boldsymbol{z}_{what}$ by extracting a sub-image containing the object (based on its $\boldsymbol{z}_{where}$ latent vector) and encoding it with a separate VAE; "*single-shot*" is the approach adopted in SSDIR.

| Criterion | Basic VAE | SPAIR [4] | SPACE [18] | **SSDIR** |
|---|---|---|---|---|
| Unsupervised | Semi- | ✓ | ✓ | Semi- |
| Inferring representations | Glimpses | Glimpses | Glimpses | Single-shot |
| Varying sizes | ✓ | ✗ | ✗ | ✓ |
| Particular objects type | ✓ | ✗ | ✗ | ✓ |
| Parallel encoding | ✗ | ✗ | ✗ | ✓ |

## 4 Experiments

In this section, we evaluate the performance of SSDIR and compare it with two baseline methods: SPAIR [4] and SPACE [18]. We focus on verifying the ability to learn valuable representations of objects, which sizes vary; this is conducted by analyzing the quality of reconstructions produced by the decoder of each method and applying the produced representations in a downstream task. Besides, we conduct an ablation study to analyze the influence of the dataset characteristics on SSDIR performance.

Our implementation of SPAIR is enhanced with a convolutional encoder instead of the original, fully-connected network, which should improve its performance on more complicated datasets. Since in this work we focus on learning objects' representation, we consider models without background: SPAIR does not explicitly model it, whereas in SPACE we analyze the foreground module outputs, which tries to reconstruct individual objects in the image. In Table 1 we included a comparison between the analyzed methods, together with an approach employing an object detector, a spatial transformer for extracting glimpses, and a VAE for learning their representations (denominated as *SSD+STN+VAE*).

The datasets used in the research were chosen to resemble common choices among recent multi-object representation learning methods. Among them, we decided to include datasets of various complexity, providing the ability to validate the model on simple images and prove its performance on complex, realistic images. Therefore, we conducted our experiments using three datasets: 1) multi-scale, scattered MNIST digits (with configured minimum and maximum digit

size, as well as grids for scattering digits), 2) CLEVR dataset [15] (containing artificially generated scenes with multiple objects of different shape, material, and size, used widely in the field of scene generation and multi-object representation learning), 3) WIDER FACE [23] (face detection benchmark dataset, with images containing multiple people; the dataset was used to demonstrate the ability of SSDIR to focus on objects of a particular type).

### 4.1 Per-object Reconstructions

In this section, we present a comparison of images' and objects' reconstructions for the proposed model and the baseline methods. In Fig. 2 we show inputs and reconstructions of representative images from each dataset (test subset, i.e. images not used for training), as well as some individual object reconstructions. Note, that due to the number of objects presented in the image and the nature of the models, it would not be possible to show all reconstructed objects.

Both SPAIR and SPACE can reconstruct the scattered MNIST dataset's image correctly. However, looking at the *where* boxes inferred by these models it is visible, that due to their limited object scale variability they are unable to attend to individual objects with a single latent representation, often reconstructing one digit with multiple objects. This is confirmed by the analysis of object reconstructions: SPAIR builds object reconstructions by combining reconstructed parts of digits, whereas SPACE can reconstruct digits of sizes similar to its preset, but divides bigger ones into parts. SSDIR is able to detect and reconstruct the MNIST image accurately: the use of a multi-scale feature pyramid allows for attending to entire objects, creating scale-invariant reconstructions, which are then mapped to the reconstruction according to tight *where* box coordinates.

SPAIR did not manage to learn object representations in the other two datasets. Instead, it models the image with rectangular boxes, containing a bigger part of an image. The aberrations visible in CLEVR dataset with SPAIR are caused by a transparency mask applied in this model and the fact, that these objects are heavily transformed when merging into the reconstruction. The tendency to model the image with rectangles is even more visible in the WIDER FACE dataset, where SPAIR divides the image in almost equal rectangles, aligned with the reconstruction grid. This effect allows for a fair quality of overall image reconstructions but does not yield valuable object representations.

In the case of SPACE, the model was not able to learn objects' representation in the CLEVR dataset, despite an extensive grid search of the hyperparameters relevant to the foreground module (especially the object's size). Instead, it models them using the background module, which cannot be treated as object representations since they gather multiple objects in one segment (this lies in line with problems reported in the GitHub repository[2]). Hence, objects reconstructions visible in Fig. 2 for this dataset contain noise. When applied to the WIDER FACE dataset, SPACE tends to approach image reconstruction in the

[2] https://github.com/zhixuan-lin/SPACE/issues/1.

same way as SPAIR, dividing the image into rectangular parts, reconstructed as foreground objects. Similarly, this leads to an acceptable reconstruction quality but does not provide a good latent representation of the image's objects.

SSDIR shows good performance on the CLEVR dataset: it can detect individual objects and produce their latent representations, which results in good quality reconstructions. Similarly, in the case of the WIDER FACE dataset, the model is able to reconstruct individual faces. However, due to the simple backbone design and low resolution of object images, the quality of reconstructed faces is low. Additionally, as a result of using a multi-scale feature pyramid, SSDIR returns multiple image reconstructions for individual objects.

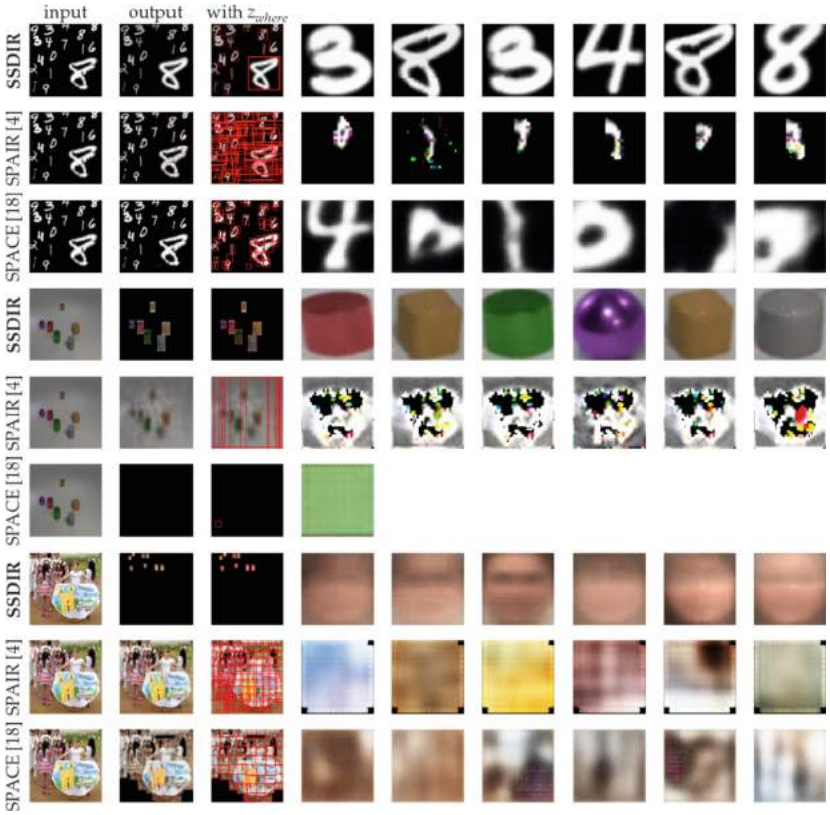



**Fig. 2.** Model inference comparison between **SSDIR**, SPAIR [4], and SPACE [18] for three typical images from each dataset. The first column presents the input image, the second and third contain image reconstruction without and with inferred bounding boxes; the remaining columns include some of the reconstructed individual objects. The number of images is limited due to the number of objects reconstructed by each model; for SSDIR, objects are meaningful and visually sound, while SPAIR and SPACE tend to divide bigger objects into smaller ones, or, in case of more complicated scenes, reconstruct them by dividing into rectangles, returning a redundant number of latents.

### 4.2 Latent Space

In this section, we present the analysis of the SSDIR model's latent space and compare it with the latent space of SPAIR and SPACE. Figure 3 visualizes latent spaces for the scattered multi-scale MNIST dataset. For each model, we process the test subset to generate latent vectors of each image. Then, individual objects' $\boldsymbol{z}_{where}$ vectors were compared with ground truth bounding boxes, and labels were assigned to latent representations by choosing the maximum intersection over union between predicted and true boxes. Each $\boldsymbol{z}_{what}$ vector was then embedded into two-dimensional space using t-SNE.

**Table 2.** Comparison of metrics for digit classification task using latent objects' representations and logistic regression. Results are averaged over 3 random seeds.

| Method | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **SSDIR** | **0.9789** $\pm$ 0.0016 | **0.9787** $\pm$ 0.0017 | **0.9786** $\pm$ 0.0016 | **0.9786** $\pm$ 0.0016 |
| SPAIR [4] | 0.1919 $\pm$ 0.0073 | 0.1825 $\pm$ 0.0087 | 0.2019 $\pm$ 0.0092 | 0.1803 $\pm$ 0.0102 |
| SPACE [18] | 0.2121 $\pm$ 0.0432 | 0.2020 $\pm$ 0.0431 | 0.2158 $\pm$ 0.0435 | 0.1992 $\pm$ 0.0462 |

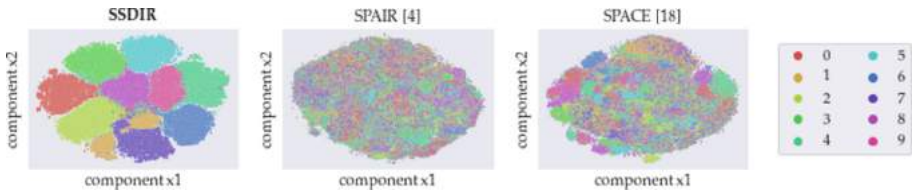



**Fig. 3.** Visualization of $\boldsymbol{z}_{what}$ latent space for scattered MNIST test dataset. Each object representation was converted using t-SNE to a two-dimensional space and plotted; the labels were inferred by choosing maximum intersection over union of predicted $\boldsymbol{z}_{where}$ and the ground truth bounding box and label. SSDIR shows a structured latent space, allowing easier distinguishing between digits.

Comparing the latent spaces, it is visible that SSDIR embeds the objects in a latent space, where digits can be easily distinguished. What is more, the manifold is continuous, without visible aberrations. The baseline methods' latent spaces are continuous as well, but they do not allow easy discrimination between each object class. The main reason is probably the fact, that both SPAIR and SPACE tend to divide large objects into smaller parts, according to the preset object size, as shown in Sect. 4.1.

Next, we tried to use the latent representations of objects in images for a downstream task of digit classification. For each of the methods, we trained models on the scattered MNIST dataset using three random seeds and produced latent representations for both train and test subset, assigning labels to each object's $\boldsymbol{z}_{what}$ based on intersection over union between $\boldsymbol{z}_{where}$ and ground truth boxes. Then, for each model and seed, we trained a logistic regression model to

classify the digits based on their latent representations. Test subset classification metrics are gathered in Table 2. SSDIR latent space proves to be more valuable than the baseline methods', reaching high values of each metric.

### 4.3 Ablation Study

To test the influence of the dataset's characteristics on the model performance, we performed an ablation study. The scattered MNIST dataset is generated by drawing random cells in a preset grid and inserting a random-sized MNIST digit inside it with a random offset. The number and size of grids, as well as the minimum and maximum size of a digit, are the hyperparameters of the dataset generation researched in the ablation study.

An SSDIR model was trained on each of the generated datasets and evaluated on a test subset with regard to the mean square error of reconstructions. The results of the study are shown in Fig. 4.

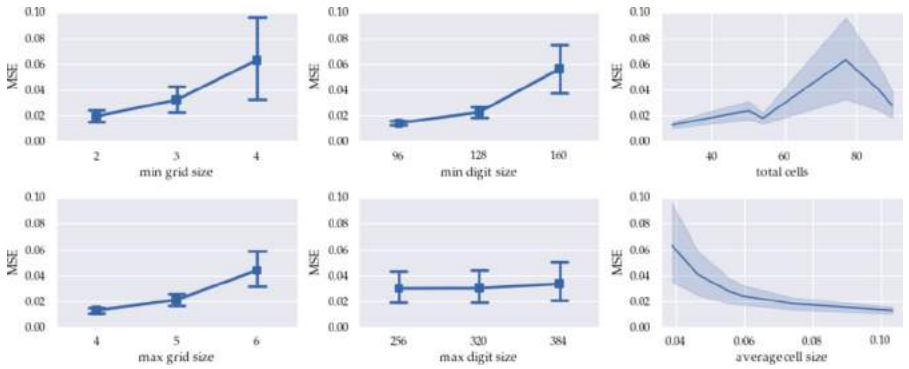



**Fig. 4.** Influence of the dataset generation parameters on the model performance. Parameters generating a dataset with larger or more occluded digits causes the model's performance to mitigate. SSDIR works best for non-occluded, small digits.

It is visible, that the model is sensitive to the size of objects in images. Bigger objects cause the mean square error to rise, mainly due to the transformation of small-sized reconstructions to the output image. Another factor that causes the error to increase is the number of digits in the image, which usually leads more occlusions to appear in the final image. The upturn is visible with increasing the minimum and maximum grid size, as well as the total number of cells.

## 5 Conclusions

In this paper, we proposed SSDIR, a single-shot convolutional generative model for learning scale-invariant object representations, which enhances existing solutions with a multi-scale feature pyramid-based approach and knowledge learned

in an object detection model. We showed the improved quality of latent space inferred by SSDIR by applying it in a downstream task and proved its ability to learn scale-invariant representations of objects in simple and complex images.

Among the method's drawbacks, one should mention limited input image size, which makes it struggle with very complicated scenes, especially in case of occlusions. What is more, learning representations of objects in complex scenes could be improved by more advanced modeling of objects' interactions. These issues will be addressed in future works, which include applying a more advanced convolutional backbone and larger input images for improving the ability to detect objects and the quality of their representations. The latent vectors inferred by SSDIR could potentially be used in other advanced tasks, such as object tracking or re-identification. In such a case, the model could benefit from the increased sophistication of the model architecture. Additionally, SSDIR could be extended for processing videos by utilizing a recurrent network to consider temporal dependencies between subsequent frames.

## References


1. Ali Eslami, S.M., et al.: Attend, infer, repeat: fast scene understanding with generative models. In: Advances in Neural Information Processing Systems (Nips), pp. 3233–3241 (2016)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013). https://doi.org/10.1109/TPAMI.2013.50
3. Burgess, C.P., et al.: MONet: Unsupervised Scene Decomposition and Representation, pp. 1–22 (2019). http://arxiv.org/abs/1901.11390
4. Crawford, E., Pineau, J.: Spatially invariant unsupervised object detection with convolutional neural networks. In: 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, pp. 3412–3420 (2019)
5. Crawford, E., Pineau, J.: Exploiting spatial invariance for scalable unsupervised object tracking. In: Proceedings of the AAAI Conference on Artificial Intelligence 34(04), pp. 3684–3692, April 2020. https://doi.org/10.1609/aaai.v34i04.5777, https://ojs.aaai.org/index.php/AAAI/article/view/5777
6. Ehrhardt, S., et al.: RELATE: physically plausible multi-object scene synthesis using structured latent spaces. NeurIPS (2020)
7. Engelcke, M., Kosiorek, A.R., Jones, O.P., Posner, I.: Genesis: generative scene inference and sampling with object-centric latent representations. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=BkxfaTVFwH
8. Engelcke, M., Parker Jones, O., Posner, I.: GENESIS-V2: inferring unordered object representations without iterative refinement. arXiv preprint arXiv:2104.09958 (2021)
9. Greff, K., et al.: Multi-object representation learning with iterative variational inference. In: 36th International Conference on Machine Learning, ICML 2019 2019-June, pp. 4317–4343 (2019)

10. Greff, K., Van Steenkiste, S., Schmidhuber, J.: Neural expectation maximization. Adv. Neural Inf. Process. Syst. **2017**-Decem(Nips), 6692–6702 (2017)
11. Gu, C., Xie, H., Lu, X., Zhang, C.: CGMVAE: Coupling GMM prior and GMM estimator for unsupervised clustering and disentanglement. IEEE Access **9**, 65140–65149 (2021). https://doi.org/10.1109/ACCESS.2021.3076073
12. Henderson, P., Lampert, C.H.: Unsupervised object-centric video generation and decomposition in 3D. In: Advances in Neural Information Processing Systems (NeurIPS) 33 (2020)
13. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS 2015, pp. 2017–2025. MIT Press, Cambridge (2015)
14. Jiang*, J., Janghorbani*, S., Melo, G.D., Ahn, S.: Scalor: generative world models with scalable object representations. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=SJxrKgStDH
15. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C.L., Girshick, R.: Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1988–1997 (2017). DOI: https://doi.org/10.1109/CVPR.2017.215
16. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, Conference Track Proceedings (2014)
17. Kosiorek, A.R., Kim, H., Posner, I., Teh, Y.W.: Sequential attend, infer, repeat: Generative modelling of moving objects. Advances in Neural Information Processing Systems 2018-Decem (NeurIPS), pp. 8606–8616 (2018)
18. Lin, Z., et al.: Space: unsupervised object-oriented scene representation via spatial attention and decomposition. In: International Conference on Learning Representations (2020). https://openreview.net/forum?id=rkl03ySYDH
19. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: SSD: single shot MultiBox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
20. Niemeyer, M., Geiger, A.: Giraffe: representing scenes as compositional generative neural feature fields. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2021)
21. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, ICML 2014, pp. II-1278-II-1286. JMLR.org (2014)
22. Van Steenkiste, S., Greff, K., Chang, M., Schmidhuber, J.: Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In: 6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings, pp. 1–15 (2018)
23. Yang, S., Luo, P., Loy, C.C., Tang, X.: Wider face: a face detection benchmark. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5525–5533 (2016). https://doi.org/10.1109/CVPR.2016.596
24. Zhu, W., Shen, Y., Yu, L., Sanchez, L.P.A.: Gmair: unsupervised object detection based on spatial attention and gaussian mixture (2021)

# How to Sort Them? A Network for LEGO Bricks Classification

Tomasz Boiński(✉), Konrad Zawora, and Julian Szymański

Faculty of Electronics, Telecommunication and Informatics, Gdańsk University of Technology, 11/12 Narutowicza Street, 80-233 Gdańsk, Poland
{tomboins,julszyma}@pg.edu.pl

**Abstract.** LEGO bricks are highly popular due to the ability to build almost any type of creation. This is possible thanks to availability of multiple shapes and colors of the bricks. For the smooth build process the bricks need to properly sorted and arranged. In our work we aim at creating an automated LEGO bricks sorter. With over 3700 different LEGO parts bricks classification has to be done with deep neural networks. The question arises which model of the available should we use? In this paper we try to answer this question. The paper presents a comparison of 28 models used for image classification trained to classify objects to high number of classes with potentially high level of similarity. For that purpose a dataset consisting of 447 classes was prepared. The paper presents brief description of analyzed models, the training and comparison process and discusses the results obtained. Finally the paper proposes an answer what network architecture should be used for the problem of LEGO bricks classification and other similar problems.

**Keywords:** Image classification · LEGO · Neural networks

## 1 Introduction

LEGO bricks are highly popular among kids and adults. They can be used to build vast array of, both very simple and very complex, constructions. This is achieved by availability of multiple, sometimes very different, yet compatible brick shapes. For the smooth build process the bricks need to properly sorted and arranged - constant searching for proper bricks in a big pile of LEGO is discouraging and limits creativity. Usually the sorting is done by shape. The colors and decals can be easily distinguished even in a big pail of bricks [2]. Still, with over 3700 different LEGO parts [24] (and the number is constantly growing) even disregarding the color makes the problem complex.

No solution for this problem was proposed so far. LEGO Group provides only a simple sorting mechanism, based on the brick size, in form of the 2011 released, now discontinued, LEGO Sort and Store item. Fan offered solutions usually rely on optimization of the manual sorting process (e.g. [1]). Some fans tried to build AI powered sorting machines [10,38] with some success. Independently from the way of building the sorting machine, it requires a well-trained neural network


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 627–640, 2022.
https://doi.org/10.1007/978-3-031-08757-8_52

able to distinguish between different, often very similar bricks. The solution should at least divide them into smaller number of categories aggregating bricks similar in shape and usage, allowing further manual selection of proper bricks. Thus LEGO oriented object classification solution is needed.

Problems like object detection, image segmentation, content-based image retrieval, or most commonly, object classification lie in domain of computer vision. In the last case the given, previously detected object, is assigned a one or more labels. The objects can have either one label assigned (multi-class classification) or many labels assigned (multi-label classification).

Computer vision is an actively research sub-domain of machine learning. It originated as far as in late 60ties of the 20-th century [27]. What was at the beginning portrayed as a simple task, assigned to students in summer school, currently remains a complex and not yet fully solved problem.

Across the recent years multiple deep neural network architectures emerged. For their comparison a standardised approach was established - *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) competition [29]. During the competition the models should classify objects to one of the 1000 classes based on 1.2 million of training images. The model accuracy is tested on 150000 images. Two metrics are calculated – Top1 (the percentage of directly correctly classified images) and Top5 (the percentage of images that were classified among the 5 with the highest probability). There are other commonly used datasets like CIFAR-10 and CIFAR-100 [20], SIFT10M [8], Open Images Dataset [19,21], Microsoft Common Objects in Context (COCO) [23]. As each dataset contains photos from different categories, with different size etc., good standing with one of the datasets does not guarantee the same results with the other. Furthermore the datasets try to be very general whereas in some cases the images contain similar objects. That is why further evaluation is still required.

In our research we undertook construction of AI-powered sorting machine [6] treating LEGO recognition as multi-class classification. To search for the best architecture that matches our scenario we decided to base our dataset in that prepared for ILSVRC. This way we could speed up training process thanks to *transfer learning* approach. As candidate architectures we selected the ones that achieved the best results in the aforementioned competition.

The structure of this paper is as follows. In Sect. 2 a description of compared network topologies is given. Later on, in Sect. 3, the used dataset is presented. Further in Sect. 4 details how the training was done and the testing methodology are presented. Section 5 discusses results obtained during the tests. Finally, some conclusions are given.

## 2 Network Topologies

In this paper we tested 28 network topologies from 7 families:

– EfficientNet – EfficientNetB0, EfficientNetB1, EfficientNetB2, EfficientNetB3, EfficientNetB4, EfficientNetB5, EfficientNetB6 and EfficientNetB7 variants,

- NASNet – NASNetMobile and NASNetLarge variants,
- ResNet – ResNet50, ResNet50V2, ResNet101, ResNet101V2, ResNet152 and ResNet152V2 variants,
- MobileNet – MobileNet, MobileNetV2, MobileNetV3Large and MobileNetV3-Small variants,
- Inception – InceptionV3, InceptionResNetV2 and Xception variants,
- DenseNet – DenseNet121, DenseNet169 and DenseNet201 variants,
- VGG – VGG16 and VGG19 variants.

In this section a brief introduction to each family and variant is given, portraying its strengths and rationale behind the used architecture.

EfficientNet architecture was defined in 2019 [37]. The model aims at efficient scaling of convolutional deep neural networks. The authors distinguished three dimensions of scaling: depth scaling, width scaling and resolution scaling. Depth scaling is the most commonly used approach, as it allows increase in number and complexity of detected features by increasing the number of convolutions. However, with increasing network depth, the training process gets longer and a problem of vanishing gradient can be observed [13]. Width scaling relies on increase of number of channels in each convolution. It is commonly used in shallow networks, where width scaling increased both training speed and classification quality [39]. Resolution scaling allows potential extraction of additional features. With all three scaling approaches there is a point of diminishing returns, beyond which additional computational overhead is not being compensated by better accuracy. EfficientNet uses so-called compound scaling, where all three parameters are equally scaled using $\phi$ parameter.

The base model here is similar to MnasNet [36] and MobileNetV2 [30]. Each model in this family differs by the $\phi$ parameter value (starting with $\phi = 0$).

Care needs to be taken when using the model in TensorFlow framework [31], as zero-padding is used for convolutions with resolutions that cannot be divided by 8. The number of channels also needs to be divisible by 8. The real compound scaling parameters applied when using TensorFlow are thus different.

ResNet50 was proposed in 2015 [11], as a solution to vanishing and exploding gradient problems. Thanks to so-called residual connections, it allows training of very deep networks (over 1000 convolutional layers). Residual connections perform elementwise addition of identity function between convolution blocks. This improves gradient flow, by skipping non-linear activation functions usually placed in convolutional blocks.

In 2016 a revision of the original model was proposed (called ResNet V2) [12]. The whole family of this model (in both ResNet and ResNet V2 revisions) achieves very high results in ILSVR competition reaching 74.9%–78% accuracy in Top1 and 92.1%–94.2% accuracy in Top5 categories.

DenseNet was defined in 2016 [16]. Similarly as in ResNet, the aim is to solve the vanishing gradient by shortening its flow path. DenseNet uses so-called *dense blocks* to achieve it. The dense block consists of $1 \times 1$ and $3 \times 3$ blocks and output of every block within it is connected with input of every next block. Each layer within a dense block has thus direct access to its output which limits the

flow path. DenseNet has also low width of the convolutional layers. Each variant of DenseNet architecture differs in terms of size of the last two dense blocks.

In 2018 DenseNet achieved the highest score in ILSVR competition Top1 category reaching accuracy of 75% for DenseNet-121, 76.2% for DenseNet-169, 77.42% for DenseNet-201 and 77.85% for DenseNet-264 77.85%.

Inception architecture was defined in 2014 [34] with Inception v1/GoogLeNet. The aim was to reduce the risk of overfitting and eliminate the problems with gradient flow. A special Inception block was proposed - it is composed of three layers with different filters ($1 \times 1$, $3 \times 3$ and $5 \times 5$). This led to high calculation complexity so a reduction was introduced that limited the number of entry channels. 9 Inception v1 blocks were combined as GoogLeNet architecture.

Inception v2 and v3 were defined in 2015 [35]. They increased performance, limiting information loss and computational complexity. Inception v3 achieves 77.9% accuracy in ILSVR competition Top1 category and 93.7% in Top5.

In 2016 Inception v4, InceptionResNetV1 and InceptionResNetV2 architectures were proposed [33]. The main goal was simplification and unification of the Inception models. ResNet residual connections were also included in the model. The best results were obtained by InceptionResNetV2 model. In Top1 category of the ILSVRC competition it achieved accuracy of 80.3% and in Top5 95.3%.

In 2017 an extension to Inception V3, by replacing the inception block with so-called extreme inception, was defined [7]. The original block was modified so that for each $1 \times 1$ convolution output corresponds one $3 \times 3$ convolution. This architecture, called Xception, proved to be easier to define and modify in software frameworks than the original Inception model.

Xception achieved better results in ILSVR competition than the original Inception v3 model. For Top1 category the accuracy was 79% and for Top5 94.5%. It also had less parameters (22.86 million vs 23.63 million).

NASNet model was defined in 2017 [41]. It was created thanks to Google AI's *AutoML* [28] and Neural Architecture Search [40]. The creation of optimal network architecture is treated here as reinforcement learning problem, with the final network accuracy as a reward. This induced a very high computational cost, so the search space had to be narrowed considerably. Based on the analysis of other models the authors first defined a general architecture, which composed of only 2 blocks - *normal cell* and *reduction cell*.

This significantly reduced the time needed to find the optimal model. Still, the training time remained very long. However, the model achieved good results. For ILSVRC Top 1 category it reached accuracy of 74.4% and 82.5% for smaller NASNetMobile variant, and larger NASNetLarge variant respectively.

MobileNet model was defined in 2017 [15]. It was designed to allow fast inference on mobile and embedded devices. The authors of this solution point out that after a certain level of network complexity, the increase in inference time is much bigger than the increase in accuracy, making the potential gain computationally unprofitable. To further increase the performance of inference, authors defined a special convolution, called *depthwise separable convolution*. It separates the operation into two phases - filtering and combination. This

approach allowed up to 9 times lower computational complexity with only a 1% lower accuracy [15] (for ILSVRC Top 1 category).

Few versions of MobileNet architecture were proposed, each introducing usage of different approaches (like residual connections) or different numbers of channels. The original MobileNet model achieved for Top1 category of ILSVRC competition the accuracy equal to 70.6%. MobileNetV2 [30] achieved 72.0% with around 20% lower number of parameters and 47% lower computational cost. MobileNetV3 [14] introduced 2 versions - Small with 2.5 million parameters and Large with 5.4 million parameters. The accuracy for Top 1 category of ILSVRC competition was 75.2% for MobileNetV3Large and 67.4% for MobileNetV3Small.

VGG is one of the oldest architectures, was defined in 2014 [32]. Different variants of this model vary by the number of trainable layers. For Top 1 category of ILSVRC competition, VGG16 and VGG19 reach accuracy of 71.3%. For Top5 category, VGG16 reaches accuracy of 90.1%, whereas VGG19 of 90%.

As we can see, all of the aforementioned models achieved very good result in the ILSVRC competition. At the time of their publication they gained the highest score and usually became the state of the art. As mentioned in Sect. 1 it doesn't always translate to the same results for other datasets.

## 3 The Dataset

During the training we used custom dataset containing both real photos and renders of LEGO bricks, belonging to 447 classes. The bricks were taken from authors personal collection of over 150 LEGO sets and represents the most commonly available brick shapes. The whole dataset consists of 620082 images, where 52601 were real photos and 567481 were life-like renders. The renders were created using Blender tool [9] based on 3D models from LDraw library [17].

The renders were used to speed up data gathering. We created a script that randomly selected a brick type, color and alignment simulating its move on a conveyor belt below a fixed positioned camera. Thanks to Blender and its extension called ImportLDraw [26] we managed to generate realistic images of LEGO bricks. Sample renders, after being cropped, can be seen in Fig. 1.

Real photos were created to increase the representativeness of the training set. For that we created a dedicated Android app allowing quick tagging and automatic cropping of LEGO bricks on pictures taken with phone camera. Sample real photos can be seen in Fig. 2.

The full set of rendered images (before cropping) and real photos are publicly available – [5] and [3] respectively. The complete dataset is also available [4].

Before the training the dataset was prepared so that all networks would be trained on the same images. The images need to be standardised in terms of size and proportions. As some of the bricks are long and narrow (e.g. brick 3002), we decided to scale the longer edge to the desired size, and the shorter edge proportionally (otherwise we could loose some information). Then, the image canvas was extended to form a square and was filled with white background. Next, all images were augmented using imgaug library [18]. The transformation included the following operations applied with 50% probability:

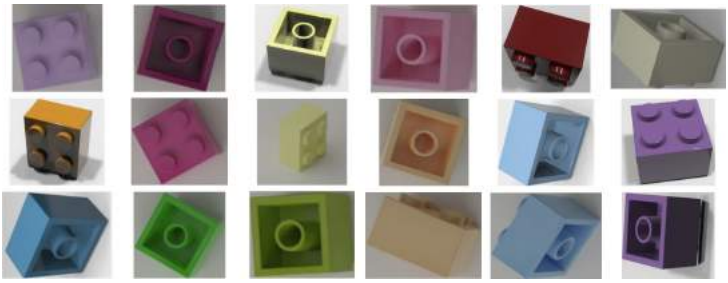


**Fig. 1.** Sample renders for brick number 3003

**Fig. 2.** Sample real photos of brick number 3003

– scaling to randomly selected size (80%–120% of the original size),
– random rotation between −45° and 45°,
– random shift by up to 20%,
– random transformation into a trapezoid with an angle of up to 16°.

Next, 5 randomly selected operations were applied, from the following list:

– Gaussian, median or averaged blur with a random intensity,
– sharpening filter with random blending factor and brightness,
– emboss filter with random blend factor and brightness,
– superimpose the contours detected by the edge detection filter, with a probability of 50%,
– Gaussian noise of random intensity,
– dropout of random pixels or a group of pixels,
– inversion of every image channel, with probability of 5%
– addition of a random value to each pixel,
– random brightness change of the image,
– random contrast change of the image,
– generation of a grayscale image and overlaying it with random transparency over the original photo.

The augmentations were done once, so that the results will be comparable. During the training process, to reduce the risk of overfitting, we performed

additional augmentation before each epoch - the images were rotated by random angle up to 15° and the contrast was changed by random value (up to 10%).

The data gathered were divided into training and validation sets. The training set contains 447000 images (1000 images each class, 650 renders and 350 real photos). The validation set contained 44700 images (100 images each class, 50 renders, 50 real photos). The numbers were obtained experimentally.

The test set consisted of real photos created independently. It contains 4000 images of bricks belonging to 20 classes (200 images each). The set was created in separate session using bricks from other set (Lego Creative Box Classic – 10698). We used 2 variants of the set - easy and hard. Both have the same number of photos, however the hard set contained images that are hard or even impossible to distinguish but belongs to different classes (e.g. bricks 3001 and 3010).

## 4 Training Process

All models presented in Sect. 2 were trained using transfer learning approach. It consisted of 2 phases:

- pre-training – done with the base model locked, only the newly added top layers are trained,
- fine tuning – the base model was partially or completely unlocked, all unlocked layers could be trained.

Pre-training is characterised by a high learning rate (we've used 0.01) with relatively low computation cost, as the backward pass needs to be calculated only for the newly added layers. After this stage, we could observe Top1 accuracy for the 447 LEGO classes at around 50–70%. During the fine-tuning stage, some of the layers are unlocked and the training is repeated for those layers. The problem here is how many layers should be unlocked. If the number will be low, then the training process will be faster, but we might not get to the desired accuracy. The number of unlocked layers also depends on the initial size of the base model.

We aimed at comparing different architectures so we designed adaptive fine tuning algorithm. It goes as follows:

1. `N := 0`
2. `N := N + unfreeze_interval`
3. `top1_history := []`
4. Unlock `N` top layers and recompile the model
5. Perform 1 training epoch
6. Perform 1 validation epoch
7. Add the Top1 accuracy on the validation set to `top1_history` list
8. If `top1_history` contains no less than `patience` elements and the Top1 accuracy on validation set did not increase by at least `min_delta` during last `patience` epochs, go to step 2
9. If `top1_history` contains `max_epochs_per_fit` elements, go to step 2.
10. Go to step 5.

where:

- `unfreeze_interval` (15 by default) – the number of layers to be unlocked within on fine-tuning iteration,
- `max_epochs_per_fit` (50 by default) – max number of training epochs in one fine-tuning iteration,
- `patience` (5 by default) – the number of epochs in one iteration, after which the model quality is evaluated,
- `min_delta` (0.01 by default) – minimal requested Top1 accuracy improvement reached in `patience` epochs .

The aforementioned algorithm was run to train each model for limited time. To increase the training speed and limit memory footprint we used so called *mixed precision training* [25] and XLA [22] compiler. Both approaches allowed us to train the networks with larger batch sizes.

For fine tuning we've used `learning_rate` = 0.0001. Both phases were trained using categorical cross-entropy loss function and Adam optimizer. All networks were trained with `batch_size` = 128, except for EfficientNetB5, EfficientNetB6 and EfficientNetB7, which used 64, 32 and 32 respectively.

## 5 The Results

In total 28 network topologies were tested. The comparison process was divided into two stages. First, all models were trained for four hours using adaptive fine tuning approach described in Sect. 4. The second stage lasted twelve hours. It was done with the same approach as stage 1, but only 5 best models and the best out of each family was trained. All tests were done on dual Intel Xeon Gold 6130 server with 256 GiB RAM and dual NVIDIA GeForce RTX 2080 (8 GiB GDDR6 RAM each) GPU cards. Each training was done on single GPU (two models were trained at once). The default batch size was 128. Due to the memory constraints some models used smaller batch size, namely: EfficientNetB5 (64), EfficientNetB6 (32) and EfficientNetB7 (32). In both stages we used transfer learning, where for the first stage we used a model trained on ImageNet data.

### 5.1 Stage I - The Four-Hour Training

Summary of obtained results (ranked from best to worst) are presented in Table 1. The best model in each family is marked with bold font.

EfficientNet models achieved varied results. The best variant was EfficientNetB1. It reached 84.4% Top1 accuracy and 95.85% Top5 accuracy, giving it the 8th place. EfficientNetB3 and EfficientNetB0 got slightly worse results, whereas EfficientNetB7 and EfficientNetB6 were one of the worst models. The reason for such outcome was the compound scaling which caused small number of frames (images) processed in the give time frame. This led to relatively small number of epochs and thus lower accuracy. The differences between EfficientNet variant are sustainable. For the next stage only EfficientNetB1 variant was selected.

**Table 1.** Results after the first stage (measured on the validation set)

| Model | Top1 accuracy | Top5 accuracy | Epochs run | Training time |
|---|---|---|---|---|
| **VGG16** | **92.99%** | **99.00%** | **11** | **04:01:07** |
| VGG19 | 91.70% | 98.64% | 11 | 04:05:07 |
| **ResNet50** | **87.40%** | **96.96%** | **14** | **04:04:07** |
| ResNet101V2 | 87.20% | 96.94% | 14 | 04:10:42 |
| ResNet152V2 | 86.92% | 96.74% | 15 | 04:11:10 |
| ResNet50V2 | 86.57% | 96.72% | 14 | 04:05:15 |
| ResNet152 | 86.00% | 96.28% | 13 | 04:09:27 |
| **EfficientNetB1** | **84.40%** | **95.85%** | **15** | **04:12:06** |
| ResNet101 | 84.09% | 95.44% | 14 | 04:16:57 |
| EfficientNetB3 | 83.58% | 95.63% | 10 | 04:14:06 |
| EfficientNetB0 | 82.87% | 95.03% | 15 | 04:15:20 |
| **MobileNetV3Large** | **82.32%** | **94.80%** | **17** | **04:09:28** |
| **Xception** | **82.31%** | **94.95%** | **9** | **04:27:31** |
| EfficientNetB2 | 81.33% | 94.50% | 11 | 04:04:16 |
| InceptionResNetV2 | 79.43% | 93.90% | 7 | 04:01:10 |
| MobileNet | 78.56% | 94.04% | 14 | 04:09:26 |
| **DenseNet201** | **77.41%** | **92.18%** | **14** | **04:01:57** |
| MobileNetV2 | 75.75% | 92.29% | 16 | 04:06:31 |
| DenseNet169 | 75.44% | 91.38% | 13 | 04:14:47 |
| DenseNet121 | 74.01% | 90.49% | 14 | 04:06:56 |
| MobileNetV3Small | 73.07% | 89.97% | 17 | 04:13:45 |
| InceptionV3 | 71.19% | 89.22% | 10 | 04:16:48 |
| EfficientNetB5 | 66.72% | 87.58% | 3 | 04:14:28 |
| EfficientNetB4 | 65.60% | 86.67% | 6 | 04:25:51 |
| **NASNetMobile** | **59.60%** | **82.08%** | **16** | **04:11:54** |
| EfficientNetB6 | 58.34% | 82.15% | 2 | 04:35:13 |
| EfficientNetB7 | 54.38% | 78.60% | 1 | 04:03:32 |
| NASNetLarge | 53.39% | 77.89% | 4 | 04:26:13 |

ResNet models achieved very good results. The best variant was ResNet50 reaching 87.40% Top1 and 96.96% Top5 accuracy. The other variants achieved similar results. 3 models were selected: ResNet50, ResNet101V2 and ResNet152V2.

Inception models reached mediocre results. The best one was Xception reaching 82.31% Top1 and 94.95% Top5 accuracy. All models finished pre-training stage and reached the fine-tuning phase. However, we observed very low performance in

terms of processed images per second, which might have been the cause of mediocre accuracy. Thus only the Xception model was selected.

DenseNet models did not perform too well. The best results were obtained by DenseNet201 variant (77.41% Top1 and 92.18% Top5 accuracy). Contrary to other models, the poor quality did not come from performance problems. DenseNet training showed one of the highest images per second rate. The problem lies in low increase of accuracy between epochs. We suspect it is caused by the design of DenseNet, specifically the concatenation operation. Unlike other tested architectures, in DenseNet, last convolutional layers are just a small part of the final feature map. By tuning a small amount of top convolutional layers, we're potentially leaving a big part of the feature map intact. This could be fixed by changing the training methodology and training all convolutional layers, but it has not been attempted in this phase.

NASNet models also got poor results. The best one, NASNetMobile, reached 59.60% Top1 and 82.08% Top5 accuracy placing 25 out of 28 tested models. Once again performance was the reason for the results. For the second stage only NASNetMobile was selected.

MobileNet scored averagely, the best variant being MobileNetV3Large reaching 82.32% Top1 and 94.8% Top5 accuracy. This variant, despite being targeted for mobile devices, outperforms deeper models like Xception or DenseNet201 thanks to the highest images per second rate and thus the highest training performance. For the second stage MobileNetV3Large was selected.

The best results were obtained by the VGG network variants - VGG16 placed first (with 92.99% Top1 and 99% Top5 accuracy) and VGG19 placed second (with 91.70% Top1 and 98.64% Top5 accuracy). The results came unexpected, as this is the oldest tested architecture. During the ILSVRC competition it was outperformed over the years by all other tested models, with exception of some MobileNet variants. The VGG are relatively shallow, but very wide. This allows fast unlocking of many layers in the fine-tuning approach and thus leads to very fast learning times. For the second stage both VGG16 and VGG19 were selected.

### 5.2 Stage II - The Twelve-Hour Training

During this stage 10 models were further trained. The aggregated results (ranked from best to worst) can be seen in Table 2.

All models managed to get better results. In most cases (except NASNetMobile and DenseNet201) twelve-hour limit was sufficient to achieve convergence.

During this stage, we observed the similar results as in the previous one. Once again, VGG16 and ResNet50 proved to be the best. However, the quality difference, both in Top1 and Top5 accuracy, between models that reached convergence is not big - the biggest difference is only 2.18% points. This is true even for mobile models, like MobileNetV3Large. This network required however more epochs to reach convergence.

What came as a surprise is that, once again, VGG16 model achieved the best results. In ILSVRC competition this model is outperformed by every other non-mobile approach presented in this paper. In the problem presented here

**Table 2.** Result after the second stage (measured on the validation set)

| Model | Top1 accuracy | Top5 accuracy | Epochs run | Training time | Parameters |
|---|---|---|---|---|---|
| VGG16 | 94.56% | 99.21% | 31 | 12:13:49 | 138.3M |
| ResNet50 | 93.81% | 99.10% | 41 | 12:13:23 | 25.6M |
| ResNet101V2 | 93.19% | 98.77% | 45 | 12:08:03 | 44.6M |
| MobileNetV3Large | 92.65% | 98.68% | 48 | 12:02:37 | 5.4M |
| VGG19 | 92.62% | 98.79% | 29 | 12:26:21 | 143.6M |
| Xception | 92.49% | 98.69% | 27 | 12:06:12 | 22.9M |
| ResNet152V2 | 92.45% | 98.54% | 40 | 12:02:22 | 60.3M |
| EfficientNetB1 | 92.38% | 98.51% | 37 | 12:19:43 | 7.8M |
| DenseNet201 | 85.26% | 95.85% | 41 | 12:07:23 | 20.2M |
| NASNetMobile | 78.59% | 93.46% | 41 | 12:09:57 | 5.3M |

(distinguishing LEGO bricks), VGG16 model trains very fast and reaches superb accuracy. This model is, however, characterised by high number of parameters and thus costly in terms of calculation time both at the time of training and inference. For practical application, the second model, ResNet50, might be thus a better choice, as it has Top1 accuracy lower only by 0.75% point, while 5.4 times lower the number of parameters. This model might also be a better choice after extending the training set with images representing other LEGO bricks, that currently are not taken into consideration (and thus extending the number of classes almost tenfold).

Very good results were also obtained by a mobile-oriented models, especially MobileNetV3Large, which had only 1.91% point lower Top1 accuracy than VGG16 model. Furthermore, it contains only 5.4 million parameters (in contrast to 138.3 million for VGG16). Thus in applications where computing performance is scarce, MobileNetV3Large should be used over any more complicated model. Despite its size, it outperforms in terms of accuracy other, more complicated models (VGG19, Xception, ResNet152V2 and EfficientNetB1).

DenseNet201 and NASNetMobile did not reach convergence in the twelve-hour time limit and thus did not achieve good results. DenseNet201 suffered from overfitting and NASNetMobile had very slow accuracy increase and would require much longer training time.

### 5.3 Final Tests

We performed some final tests on the two best models. The results for the easy and hard sets are presented in Table 3. As can be seen, both models reach similar accuracy.

To test the models in real life application we implemented a mobile app which took photos of LEGO bricks laying on a white background and combined it with the pre-trained models. VGG16 correctly recognized 39 out of 40 bricks. Wrongly labeled 822931 brick was classified as 3003 due to their similarity from the camera

**Table 3.** VGG16 and ResNet50 accuracy for easy and hard tests

| | Easy set | | Hard set | |
|---|---|---|---|---|
| | Top1 accuracy | Top5 accuracy | Top1 accuracy | Top5 accuracy |
| VGG16 | 92.37% | 99.02% | 86.08% | 98.22% |
| ResNet50 | 90.40% | 99.05% | 86.30% | 98.60% |

perspective. ResNet50 correctly classified all bricks. The networks were tested in different conditions. We used an intensive pink light to illuminate the test environment. This made the background pink and most of the bricks appeared as having different, not seen before, colors. VGG16 correctly recognized 37 bricks out of 40. ResNet50 model once again correctly classified all of the bricks.

## 6 Conclusions

The paper presents extensive analysis of deep neural network architectures in order to verify their suitability for classification of LEGO bricks. The problem is characterized with the need to distinguish objects between multiple, often similar classes, as there are over 3700 different LEGO brick shapes. For this purpose, a new dataset was created containing 447 classes and a set of tools automating the analysis process were implemented. In total, 28 network architectures, belonging to 7 families, were analyzed and compared. For the comparison, we used our proposed training algorithm with adaptive fine-tuning approach.

Results showed that VGG16 model proved to be the best with its Top1 accuracy of 94.56% and Top5 accuracy of 99.21%). Surprisingly, in ILSVRC competition this model was outperformed by other solutions. The model is characterized, however, with very big number of parameters (138.3 million) and high number of floating point operations during training and inference process (15.3 GFLOPs). Not falling far behind was ResNet50 model (Top1 93.81%, Top5 99.10%) which had lower parameter count (25.6 million) and required far lower system performance (3.87 GFLOPs). In many cases, this might be the best choice for similar problems, where there are a lot of similar objects to classify. Surprisingly, also the smaller, mobile models proved to be worthwhile. MobileNetV3Large achieved very good accuracy (Top1 92.65%, Top5 98.68%), with very low parameter count (5.4 million) and low performance requirements (0.21 GFLOPs).

The two best models also were tested in real life application. They proved to be very accurate in both synthetic test on predefined test sets and during live classification of LEGO bricks.

In the near future we plan on extending the dataset with additional classes to cover as much LEGO brick shapes as possible to provide a deep neural network able to classify any type of LEGO bricks. Such network could be used in LEGO sorting machines, software recommending constructions based on the bricks available, automatic brick database creation and many more.

**Acknowledgment.** The authors would like to thank Bartosz Śledź and Sławomir Zaraziński for help with part of the implementation and dataset creation.

## References

1. Adam: LEGO sorting chart (2019). https://go.gliffy.com/go/publish/12232322. Accessed 24 Mar 2022
2. Aplhin, T.: The LEGO storage guide (2020). https://brickarchitect.com/guide/. Accessed 34 Mar 2022
3. Boiński, T.: Images of LEGO bricks (2021). https://doi.org/10.34808/xz76-ez11. Accessed 24 Mar 2022
4. Boiński, T., Zaraziński, S., Śledź, B.: LEGO bricks for training classification network (2021). https://doi.org/10.34808/3qfs-rt94. Accessed 24 Mar 2022
5. Boiński, T., Zawora, K., Zaraziński, S., Śledź, B., Łobacz, B.: LDRAW based renders of LEGO bricks moving on a conveyor belt (2020). https://doi.org/10.34808/jykr-8d71. Accessed 24 Mar 2022
6. Boiński, T.M.: Hierarchical 2-step neural-based LEGO bricks detection and labeling. In: Proceedings of 37th Business Information Management Association Conference, pp. 1344–1350 (2021)
7. Chollet, F.: Xception: Deep learning with depthwise separable convolutions (2017)
8. Dua, D., Graff, C.: UCI machine learning repository (2017). http://archive.ics.uci.edu/ml. Accessed 22 Nov 2021
9. Foundation, T.B.: Blender (2002). https://www.blender.org/. Accessed 22 Nov 2021
10. Garcia, P.: LEGO Sorter using TensorFlow on Raspberry Pi (2018). https://medium.com/@pacogarcia3/tensorflow-on-raspbery-pi-lego-sorter-ab60019dcf32. Accessed 22 Nov 2021
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition (2015)
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks (2016)
13. Hochreiter, S.: The vanishing gradient problem during learning recurrent neural nets and problem solutions. Internat. J. Uncertain. Fuzziness Knowl. Based Syst. **6**(02), 107–116 (1998)
14. Howard, A., et al.: Searching for mobilenetv3 (2019)
15. Howard, A.G., et al.: MobileNets: efficient convolutional neural networks for mobile vision applications (2017)
16. Huang, G., Liu, Z., van der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks (2018)
17. Jessiman, J.: LDraw. http://www.ldraw.org (1995). Accessed 22 Nov 2021
18. Jung, A.: imgaug source code. https://github.com/aleju/imgaug. Accessed 22 Nov 2021
19. Krasin, I., et al.: Openimages: a public dataset for large-scale multi-label and multi-class image classification. Dataset available from https://storage.googleapis.com/openimages/web/index.html (2017)
20. Krizhevsky, A., Nair, V., Hinton, G.: Cifar-10 and cifar-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html. Accessed 24 Mar 2022
21. Kuznetsova, A., et al.: The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. IJCV (2020)

22. Li, M., et al.: The deep learning compiler: a comprehensive survey. IEEE Trans. Parallel Distrib. Syst. **32**(3), 708–727 (2021). https://doi.org/10.1109/tpds.2020.3030548
23. Lin, T., et al.: Microsoft COCO: common objects in context. CoRR abs/1405.0312 (2014). http://arxiv.org/abs/1405.0312
24. Maren, T.: 60 fun LEGO facts every LEGO fan needs to know (2018). https://mamainthenow.com/fun-lego-facts/
25. Micikevicius, P., et al.: Mixed precision training (2018)
26. Nelson, T.: ImportLDRaw. https://github.com/TobyLobster/ImportLDraw. Accessed 16 June 2021
27. Papert, S.A.: The summer vision project (1966). https://dspace.mit.edu/handle/1721.1/6125. Accessed 22 Nov 2021
28. Le, Q., Zoph, B., Research Scientists, G.B.t.: Using machine learning to explore neural network architecture (2017). https://ai.googleblog.com/2017/05/using-machine-learning-to-explore.html. Accessed 22 Nov 2021
29. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
30. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv 2: Inverted residuals and linear bottlenecks (2019)
31. Shukla, N., Fricklas, K.: Machine learning with TensorFlow. Manning Shelter Island, Ny (2018)
32. Simonyan, K., Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition (2015)
33. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.: Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning (2016)
34. Szegedy, C., et al.: Going Deeper with Convolutions (2014)
35. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the Inception Architecture for Computer Vision (2015)
36. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: Mnasnet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828 (2019)
37. Tan, M., Le, Q.V.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks (2020)
38. West, D.: LEGO sorting machine (2019). https://twitter.com/JustASquid/status/1201959889943154688. Accessed 08 Feb 2021
39. Zagoruyko, S., Komodakis, N.: Wide residual networks (2017)
40. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning (2017)
41. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning Transferable Architectures for Scalable Image Recognition (2018)

# Novel Photoplethysmographic Signal Analysis via Wavelet Scattering Transform

Agnieszka Szczęsna[1(✉)], Dariusz Augustyn[2], Henryk Josiński[1], Adam Świtoński[1], Paweł Kasprowski[2], and Katarzyna Harężlak[2]

[1] Department of Computer Graphics, Vision and Digital Systems, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
agnieszka.szczesna@polsl.pl

[2] Department of Applied Informatics, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland

**Abstract.** Photoplethysmography (PPG) is a non-invasive optical technique, applied in clinical settings to measure arterial oxygen saturation. Using modern technology, PPG signals can be measured by wearable devices. This paper presents a novel procedure to study the dynamics of biomedical signals. The procedure uses features of a wavelet scattering transform to classify signal segments as either chaotic or non-chaotic. To this end, the paper also defines a chaos measure. Classification is made using a model trained on a dataset consisting of signals generated by systems with known characteristics. Using an example PPG signal, this paper demonstrates the usefulness of the wavelet scattering transform for the analysis of biomedical signals, and shows the importance of correctly preparing the training set.

**Keywords:** Wavelet scattering transform · Chaos · PPG · Classification · Biomedical signals

## 1 Introduction

Photoplethysmography (PPG) is an non-invasive optical measurement technique. By using a light source to illuminate skin tissue, either the transmitted or reflected light intensity is collected by a photodetector to record the photoplethysmogram. Traditionally, PPG signals have been recorded using red or near infra-red light. In recent years, green light has been used for wearable devices, such as wristbands and smartwatches, to provide highly usable and accessible

This publication was supported by the Department of Graphics, Computer Vision and Digital Systems, under statue research project (Rau6, 2022), Silesian University of Technology (Gliwice, Poland).


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 641–653, 2022.
https://doi.org/10.1007/978-3-031-08757-8_53

daily health monitoring [8,10,12,16,31]. As such, a proper understanding of green light PPG is of critical importance. Recorded light intensity variations have traditionally been associated with blood volume pulsations in the microvascular bed of the tissue. The tissue penetration depth of green light is approximately 530 nm. The source of the chaotic properties of PPG is unclear. Such properties could originate in the upper layers of the skin, due to changes in capillary density caused by arterial transmural pressure, or in deeper layers, due to changes in vessel blood volume [27].

Despite uncertainty concerning the mechanisms of PPG, the technique is generally accepted to provide valuable clinical information about the cardiovascular system. PPG signals are used to monitor pulse rate, heart rate, oxygen saturation, blood pressure, and blood vessel stiffness [2,7,13,15,17,21–23,28]. Unfortunately, such signals are often corrupted by noise, motion artifacts, and missing data.

Biological signals contain deterministic and stochastic components, both of which contribute to the underlying dynamics of the physiological system. All biological signals contribute information on the underlying physiological processes. Therefore, by studying such signals, the physiological systems that generate them can be better understood.

In early studies, PPG as well as ECG (electrocardiogram) and HRV (heart rate variability) were claimed to be chaotic mostly based on the results of time-delay reconstructed trajectory, correlation dimension and largest Lyapunov exponent [29]. Subsequently, with the development of nonlinear time series analysis methods for real-world data, further evidence of the chaotic nature of such biological signals has emerged. However, many tools that were previously thought to provide clear evidence of chaotic motion have been found to be sensitive to noise and prone to producing misleading results. Thus, controversy remains concerning the topic of chaos in biological signals [11,26,27].

Sviridova and Sakai [26] applied nonlinear time series analysis methods to PPG signals to identify the unique characteristics of the underlying dynamical system. Such methods included time delay embedding, largest Lyapunov exponent, deterministic nonlinear prediction, Poincaré section, the Wayland test, and the method of surrogate data. Results demonstrated that PPG dynamics are consistent with the definition of chaotic motion, and the chaotic properties were somewhat similar to Rössler's single band chaos with induced dynamical noise.

A more recent approach to signal analysis is the use of machine learning or deep learning methods. Such methods can generalize knowledge acquired from a training dataset, and apply it to the analysis of a testing dataset. Boullé et al. [3] used a deep neural network to classify univariate time series' generated by discrete and continuous dynamical systems based on the presence of chaotic behavior. The study suggests that deep learning techniques can be used to classify time series' obtained by real-life applications into chaotic or non-chaotic.

De Pedro-Carracedo et al. [9] found that the dynamics of a PPG signal were predominantly quasi-periodic over a small timescale (5000 data points 250 Hz).

Over a longer timescale (600000 data points 250 Hz), more diverse and complex dynamics were observed, but the signal did not display chaotic behavior. This analysis used a deep neural network to classify the PPG signals. The following dynamics classes were defined: periodic, quasi-periodic, non-periodic, chaotic, and random. Unfortunately, the dataset used to train the network contained only one system for each class. Given that chaotic systems are difficult to generalize [6], this is not sufficient to accurately classify the dynamics of the real-life signal.

De Pedro-Carracedo et al. [20] applied a modified 0–1 test to the same PPG time series' as the above study. They also found that the majority of PPG signals displayed quasi-periodic behavior across a small timescale, and that as the timescale increased the dynamics became more complex, due to the introduction of additional cardiac rhythm modulation factors. Under specific physiological conditions, such as stress, illness, or physical activity, a transition from quasi-periodicity to chaos can be possible. This phenomenon provides the motivation for measuring the presence of chaos within PPG signals under various conditions.

The objective of this study is to analyze the dynamics of PPG signals during different everyday activities. We propose a novel approach to classify signals using features of a wavelet scattering transform (WST) and a support vector machine (SVM) classifier. This approach was simplified by defining only two classes of signals: chaotic and non-chaotic. Compared to previous research, the training data was prepared in greater detail, and included noise, which was omitted in previous works.

Wavelet analysis provides a unifying framework for the description of many time series phenomena [25]. Introduced by Mallat [18], WST has a similar architecture to convolutional neural network. Despite requiring no parameter learning, WST performs strongly, particularly in constrained classification tasks. WST is a cascade of complex wavelet transforms and modulus non-linearities. At a chosen scale, averaging filters provide invariance to shifts and deformations within signals [1]. Hence, WST can be applied accurately and efficiently to small datasets, whereas convolutional neural network require a large amount of training data. Consequently, WST features possess translation invariance, deformation, stability, and high-frequency information [4]. As such, WST is highly suitable feature extractors for non-linear and non-stationary signals, and has been widely used in audio, music, and image classification.

Moreover, WST is often used to analyze time series', including biomedical signals. By inputting WST features to an SVM classifier, electroencephalography signals were correctly classified as belonging to alcoholic or non-alcoholic patients [5]. In addition, a WST was used to classify heart beats based on ECG signals, with an accuracy of 98.8–99.6%. Jean Effil and Rajeswari [14] used a WST and a deep learning long short-term memory algorithm to accurately estimate blood pressure from PPG signals.

## 2 Materials and Methods

### 2.1 The Wavelet Scattering Transform

The wavelet transform is convolutions with dilated wavelets. For 2D transformations, the wavelets are also rotated. Being localized waveforms, wavelets are stable to deformations, unlike Fourier sinusoidal waves. A scattering transform creates nonlinear invariants using wavelet coefficients with modulus and averaging pooling functions. Such transforms yield representations that are time-shift invariant, robust to noise, and stable to time-warping deformations. These attributes are highly useful for many classification tasks, and wavelet transforms are the most common method applied to limited datasets. Andén and Mallat [1] provide a brief overview of the key properties of scattering transforms, including stability to time-warping deformation and energy conservation, and describe a fast computational algorithm.

The WST consists of three cascading stages. In the first stage, the signal $x$ undergoes decomposition and convolution with a dilated mother wavelet $\Delta$ of center frequency $\triangleright$, giving $x * \Delta_{\phi}$. Following this, the convolved signal is subjected to a nonlinear modulus operator, which typically increases the signal frequency and can compensate for the loss of information due to down sampling. Finally, a time-average/low-pass filter in the form of a scaling function $\rho$ is applied to the absolute convolved signal, giving $|x * \Delta_{\phi}| * \rho$.

The zero-order scattering coefficients $S_0$ describe the local translation invariance of the signal:

$$S_0 = x * \rho.$$

At each level, the averaging operation causes the high-frequency parts of the convolved signal to be lost. These parts can be recovered via the convolution of the signal with the wavelet in the following level.

The first-order scattering coefficients $S_1$ are therefore defined as the average absolute amplitudes of wavelet coefficients for any scale $1 \leq j \leq J$, over a half-overlapping time window of size $2^j$:

$$S_1 = |x * \Delta_{\phi_1}| * \rho.$$

The second-order scattering coefficients $S_2$ are calculated by repeating the above steps:

$$S_2 = ||x * \Delta_{\phi_1}| * \Delta_{\phi_2}| * \rho.$$

The higher order wavelet scattering coefficients can be calculated by iterating the above process.

The scattering coefficients for each level of the wavelet scattering transform are obtained by processing the defined constant-$Q$ filter bank, where $Q$ is the number of wavelets per octave. Each level can have a filter bank with different $Q$ parameters.

During implementation we used the MATLAB (version R2021b) *waveletScattering* function. The two-layer WST was obtained using Gabor wavelet. For the first and second levels $Q_1 = 8$ and $Q_2 = 1$ respectively. The transform is invariant

to translations up to the invariance scale, which is set to half of the signal length in the default implementation. The scaling function determines the duration of the invariant in time. Moreover, the invariance scale affects the spacing of the wavelet center frequencies in the filter banks. The output $R^{paths \times windows \times signals}$ is a feature tensor. This tensor was reshaped into a matrix which is compatible with the SVM classifier. The columns and rows of the matrix correspond to scattering paths and scattering time windows respectively. This results in a feature matrix of $signals \cdot windows$ rows and $paths - 1$ columns.

The zero-order scattering coefficients are not used. Given that multiple scattering windows are obtained for each signal, repeated labels were created that corresponded to the labels (0, 1). Following this, normalization was applied. Scattering coefficients of order greater than 0 were normalized by their parents along the scattering path. Using the defined parameters for the $N$ input signals (runs), each composed of 1000 samples, this procedure produced a $102 \times 8 \times N$ WST feature tensor, which was then transformed into a $N \cdot 8 \times 101$ matrix.

### 2.2 Classification Model

The testing and training datasets were created using 13 dynamical systems (five chaotic and eight non-chaotic) of first, second, or third order. Table 1 shows the training set characteristics. Each system was provided with 1000 created test files, each of which contained 1000 samples. Augmentation was applied by randomizing the initial conditions, defined by the $\mathbf{x}_0$ vector, according to the formula $[2 \cdot rand() - 1] \cdot \mathbf{x}_0$, where $rand()$ generates pseudorandom numbers that are uniformly distributed in the interval $(0, 1)$. The chaotic systems are represented by driven or autonomous dissipative flows. Previously described as A.4.5, A.5.1, A.5.2, A.5.13, and A.5.15 [24], we describe these flows as CHA_1,CHA_2, CHA_3, CHA_4, and CHA_5, respectively. The non-chaotic systems were divided into the following classes: i) periodic, including the OSC_1, OSC_2, DOSC_1, and IOSC systems; ii) quasi-periodic, including QPS_1 and QPS_2; and iii) non-periodic, including DS_1 and DS_2. The quasi-periodic systems are described by the general function

$$x = f(t) = A_1 \cdot sin(\omega_1 \cdot t + \varphi_1) + A_2 \cdot sin(\omega_2 \cdot t + \varphi_2),$$

where the ratio $\omega_1/\omega_2$ is irrational.

Based on previous PPG signal analysis, we made the following experimental design choices:

- A signal with a length of 1000 samples was obtained from each system using 1000 runs with different initial parameters. Each dimension of the multidimensional systems was treated separately. This corresponded to analysis using windows with a short time horizon of 31.2 s for 32 Hz PPG signal.
- We used SVM classification with a radial basis kernel similar to that proposed by Buriro et al. [5]. The classification is made based on WST features.
- Two classes were defined for the classification task: chaotic (class 1) and non-chaotic (class 0). The decision to use just two classes, and therefore fold

periodic, quasi-periodic, and non-periodic behavior into the same class, was made to test the thesis that PPG signals are never chaotic [9]. In further work, a larger number of more distinct classes will be used.
– Sviridova and Sakai [26] show that PPG signals display some similarity to Rössler's chaos with induced dynamical noise. As such, we used Rössler's system as one of the signals with chaotic behavior, as shown in Fig. 1. Furthermore, all signals with additive white Gaussian noise were added to the whole set.

The accuracy of the trained models was checked by 10-fold cross-validation.

To investigate the properties of the training set, the following models were trained:

– *Model01* was trained without output signals from Rössler's system (CHA_3). Using 10-fold cross-validation, accuracy was validated as 100%. Testing using Rössler's system signal showed 32% accuracy. Based on the model, it is impossible to effectively classify the signals produced by chaotic systems. It is therefore important to include the signals from CHA_3 within the training set.
– *Model02* was trained without the quasi-periodic systems QPS_1 and QPS_2. Using 10-fold cross-validation, accuracy was validated as 100%. Testing using QPS_1 and QPS_2 showed 78.5% accuracy. On this basis, we determine that quasi-periodic systems are easier to correctly classify.
– *Model03* was trained on signals without additional noise. Using 10-fold cross-validation, accuracy was validated as 100%. Testing using signals with additive Gaussian noise with a signal to noise ratio of 7dB showed 94.03% accuracy. Although the analytical methods for the assessment of chaotic behavior are highly sensitive to noise, the prepared model is not, and even noisy signals can be classified with high accuracy.
– *Model01N* and *Model02N* are variants of models *Model01* and *Model02* respectively, trained additionally with noisy signals.
– *ModelAll* was trained using all signals, both with and without additive Gaussian noise, with a signal to noise ratio of 7 dB. Using 10-fold cross-validation, accuracy was validated as 99.88%.

### 2.3 PPG Dataset

The dataset used in this work is the public available PPG dataset for motion compensation and heart rate estimation in daily life activities (PPG-DaLiA[1]) [21]. Given that the database contains a reference ECG measurement, it is often used to test heart rate estimation algorithms [31]. The dataset contains a total of 36 h of recording for 15 study participants undertaking eight different types of physical everyday life activities: working, sitting, walking, eating lunch, driving, cycling, playing football, and climbing stairs. The sensor data was obtained from commercially available devices. In our case, 64 Hz PPG signals which we used for testing the trained models were recorded by the wrist-worn Empatica E4 device.

[1] https://ubicomp.eti.uni-siegen.de/home/datasets/sensors19/, accessed July 2021.

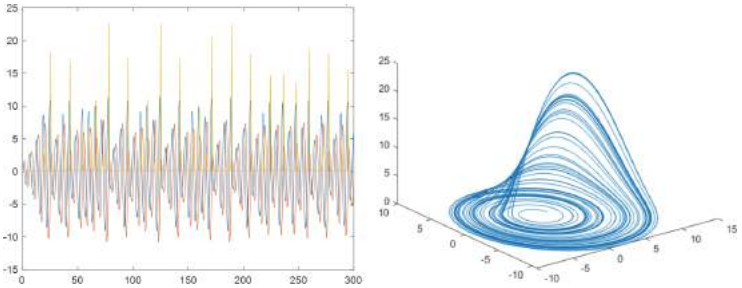


**Fig. 1.** 3D signals and a phase portrait of the Rössler system (CHA_3).

**Table 1.** The training set characteristics.

| Name and symbol | Class | Dimension | Short description |
|---|---|---|---|
| Ueda oscillator CHA_1 | Chaotic (class 1) | 2 | Driven dissipative flow |
| Lorenz attractor CHA_2 | Chaotic (class 1) | 3 | Autonomous dissipative flow |
| Rössler attractor CHA_3 | Chaotic (class 1) | 3 | Autonomous dissipative flow |
| Halvorsen's cyclically symmetric attractor CHA_4 | Chaotic (class 1) | 3 | Autonomous dissipative flow |
| Rucklidge attractor CHA_5 | Chaotic (class 1) | 3 | Autonomous dissipative flow |
| Undamped oscillator 1 OSC_1 | Periodic (class 0) | 2 | Slow oscillations with constant amplitude |
| Undamped oscillator 2 OSC_2 | Periodic (class 0) | 2 | Fast oscillations with constant amplitude |
| Damped oscillator 1 DOSC_1 | Periodic (class 0) | 2 | Fast oscillations with decreasing amplitude |
| Oscillator with increasing amplitude of oscillations IOSC | Periodic (class 0) | 2 | Oscillations with growing amplitude |
| Damped system 1 DS_1 | Non-periodic (class 0) | 3 | Slow fading signals |
| Damped system 2 DS_2 | Non-periodic (class 0) | 3 | Fast fading signals |
| Quasi-periodic system 1 QPS_1 | Quasi-periodic (class 0) | 1 | Irrational ratio: $\omega_1/\omega_2 = \pi$ |
| Quasi-periodic system 2 QPS_2 | Quasi-periodic (class 0) | 1 | Irrational ratio: $\omega_1/\omega_2 = (1 + \sqrt{5})/2$ |
| 16 000 000 samples | 0 | | All samples of signals with non-chaotic behavior |
| 16 000 000 samples | 1 | | All samples of signals with chaotic behavior |

The base frequency of the PPG signals was adjusted to match the training set. We counted the number of signal zero crossings within a given time interval when using the $CHA\_5$ system. We were required to increase the frequency of zero crossings by a factor of two, giving a final PPG signal frequency 32 Hz.

## 3 Results

The PPG signal was split into 1000 samples (31.2 s) segments, following fitting and resampling. For each segment, the WST features were obtained. As part

**Table 2.** Results for all tested models—the ratio of windows classified as chaotic to the total number of windows within the PPG signal. The highest values for each model have been marked.

| Participant | *Model01* | *Model01N* | *Model02* | *Model02N* | *Model03* | *ModelAll* |
|---|---|---|---|---|---|---|
| *S01* | 0.27 | 0.07 | 0.25 | 0.14 | 0.26 | 0.11 |
| *S02* | 0.29 | 0.12 | 0.28 | 0.15 | 0.29 | 0.15 |
| *S03* | 0.22 | 0.07 | 0.17 | 0.12 | 0.19 | 0.09 |
| *S04* | 0.30 | 0.16 | 0.28 | 0.20 | 0.29 | 0.19 |
| *S05* | **0.56** | **0.33** | **0.53** | **0.34** | **0.54** | **0.33** |
| *S06* | 0.26 | 0.11 | 0.21 | 0.14 | 0.23 | 0.13 |
| *S07* | 0.42 | 0.13 | 0.36 | 0.20 | 0.40 | 0.17 |
| *S08* | 0.23 | 0.11 | 0.22 | 0.14 | 0.23 | 0.13 |
| *S09* | 0.27 | 0.14 | 0.26 | 0.18 | 0.26 | 0.18 |
| *S10* | 0.23 | 0.10 | 0.22 | 0.12 | 0.22 | 0.13 |
| *S11* | 0.47 | 0.25 | 0.44 | 0.28 | 0.46 | 0.26 |
| *S12* | 0.20 | 0.07 | 0.16 | 0.11 | 0.17 | 0.09 |
| *S13* | 0.50 | 0.22 | 0.44 | 0.26 | 0.47 | 0.24 |
| *S14* | 0.22 | 0.06 | 0.16 | 0.11 | 0.17 | 0.08 |
| *S15* | 0.18 | 0.06 | 0.14 | 0.10 | 0.16 | 0.08 |

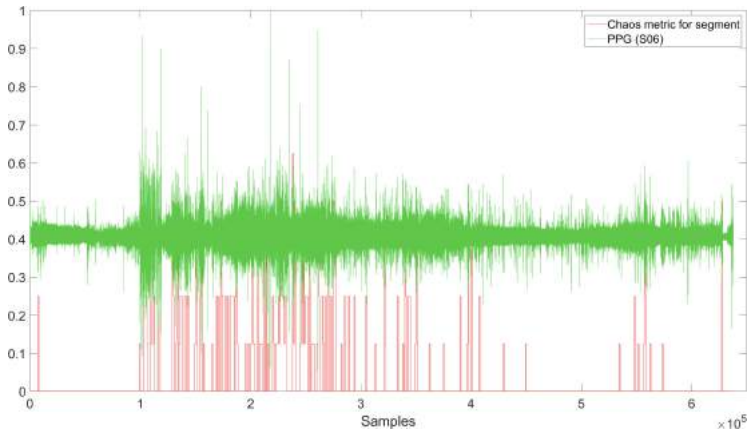



**Fig. 2.** Chaos measure values when using *ModelAll*, for each 1000 sample segment for the *S06* participant.

of the analysis, the signal was split into $W = 8$ scattering windows of 125 samples each. A classification result was generated for each scattering window. The overall classification of a segment is the ratio of the number of windows classified as having chaotic behavior (class 1), to the total number of windows.

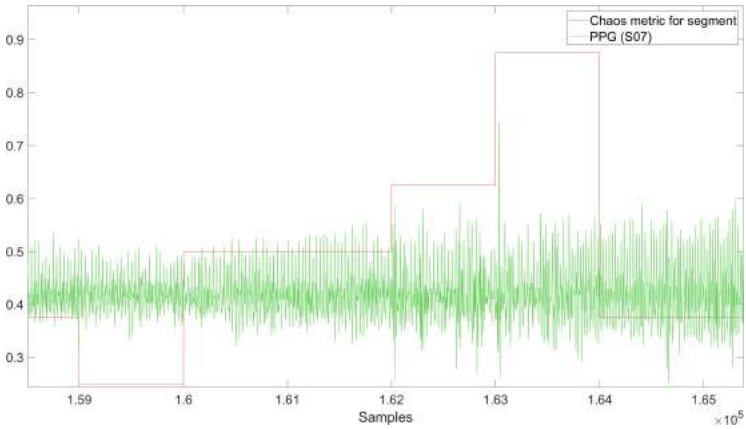



**Fig. 3.** Chaos measure values when using *ModelAll*, for each 1000 sample segment for a fragment of the PPG signal of the *S07* participant.

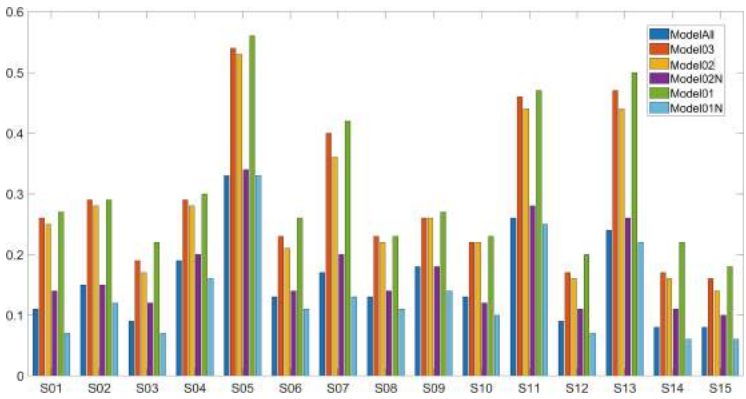



**Fig. 4.** Chaos measure values for each participant.

Hence, a segment that contains eight chaotic windows represents a fully chaotic segment of signal. Within the overall signal, we define the chaotic measure to be the ratio of chaotic windows to total number of windows. Tests were conducted for all models, as shown in Table 2. Figures 2 and 3 present the PPG signal and the value of the chaotic measure for each segment.

The results show that those models trained without additional noise—*Model01*, *Model02*, and *Model03*—display high chaotic measure values (see Fig. 4). The differences between each of these models are not significant. This confirms that noise is present within the data, and is highly relevant to its evaluation.

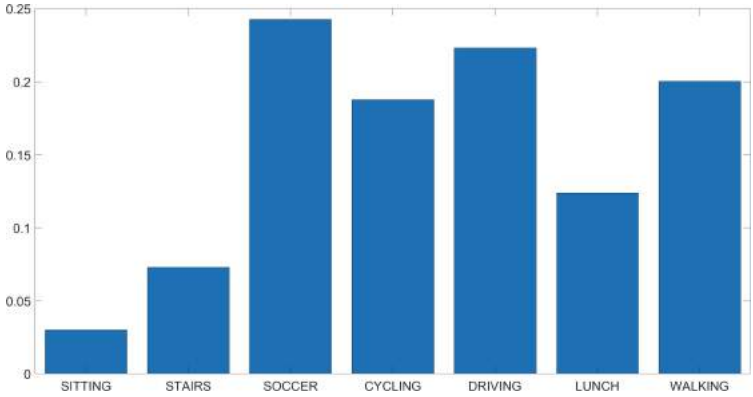



**Fig. 5.** Average chaos measure values for each activity.

Those models that were trained with additional noise—*Model01N*, *Model02N*, and *ModelAll*—display greater differences in the chaotic measure. *Model01N* produced the lowest values of chaotic measure, meaning that fewer segments were classified as exhibiting chaotic behavior. Hence, the PPG signals display some similarity to the Rössler system. *Model02N* produced the highest values of chaotic measure, showing that a model trained without quasi-periodic functions classifies such behavior as chaotic. Moreover, the inclusion of such a class of functions was justified. *ModelAll* produces results between those of *Model01N* and *Model02N*.

The highest values of chaotic measure, independent of the model used, were obtained from the PPG signal of participant *S05*. Interestingly, this participant reported the greatest errors in heart rate estimation using deep neural networks. The mean heart rate was significantly higher than for all other participants [21,31].

Figure 5 shows that the greatest values of chaotic measure are obtained during the cycling, soccer, driving, and walking activities. The values differ between activities, indicating that the measure is influenced by movement or changes in heart rate as a result of physical activity, stress, or the general condition and health of the participant. The relationship is not well defined, as the participants had the highest heart rate when climbing stairs and cycling. Further analysis of this phenomenon is required.

The calculations were performed on the computer with the following parameters: Windows 10, Intel(R) Core(TM) i7-7700HQ CPU 2.80 GHz, 32 GB, Matlab R2021b. The biggest differences in training and classification times are between the models based signals without noise and with additional noisy signals. The average time of determining WST features for models *Model_01*, *Model_02*, *Model_03* is 14.55956667 s (std 0.996971916), the training time is 19.45903333 s (std 2.239389959), and the average classification time of one window is 0.000109 s (std 0.000061). Taking into account the models *Model_01N*,

*Model_02N*, *Model_All* the average times are as follows: WST features calculations - 42.526825 s (std 4.489965561), training - 289.576025 s (std 21.05237103), classifications - 0.000435 s (std 0.000044).

## 4 Conclusions

The results show that signal classification based on system features requires careful preparation of the training set. Chaotic systems create signals that are difficult to predict. Therefore, insufficient data within the training set may cause misclassification. The noise within real measured signals must also be accounted for.

Furthermore, a WST can be used to successfully determine signal features for the purpose of classification. Given that such wavelet analysis is well understood, parameters can be chosen straightforwardly. Moreover, the training model is supposed to be much faster than the use of deep learning methods, which is worth future investigation.

The analysis showed that PPG signals display chaotic features over short time spans. The measure of chaos is dependent upon the activity performed. Given that wearable devices are easily available and are increasingly used for medical diagnosis [19,30], understanding this phenomenon is highly important and the topic requires further detailed analysis.

The aim of this study was to demonstrate the usefulness of WST for the analysis of biomedical signals, and to show the importance of correctly preparing the training set. Interest in the classification of real signals by deep learning methods is increasing; such methods may lead to erroneous conclusions if the training sets are inadequately prepared.

## References

1. Andén, J., Mallat, S.: Deep scattering spectrum. IEEE Trans. Signal Process. **62**(16), 4114–4128 (2014)
2. Biswas, D., Simões-Capela, N., Van Hoof, C., Van Helleputte, N.: Heart rate estimation from wrist-worn photoplethysmography: a review. IEEE Sens. J. **19**(16), 6560–6570 (2019)
3. Boullé, N., Dallas, V., Nakatsukasa, Y., Samaddar, D.: Classification of chaotic time series with deep learning. Physica D **403**, 132261 (2020)
4. Bruna, J., Mallat, S.: Invariant scattering convolution networks. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1872–1886 (2013)
5. Buriro, A.B., et al.: Classification of alcoholic EEG signals using wavelet scattering transform-based features. Comput. Biol. Med. **139**, 104969 (2021)
6. Chen, Z., Xiu, D.: On generalized residual network for deep learning of unknown dynamical systems. J. Comput. Phys. **438**, 110362 (2021)
7. Chung, H., Ko, H., Lee, H., Lee, J.: Feasibility study of deep neural network for heart rate estimation from wearable photoplethysmography and acceleration signals. In: 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pp. 3633–3636. IEEE (2019)

8. Coughlin, S.S., Stewart, J.: Use of consumer wearable devices to promote physical activity: a review of health intervention studies. J. Environ. Health Sci. **2**(6) (2016)
9. De Pedro-Carracedo, J., Fuentes-Jimenez, D., Ugena, A.M., Gonzalez-Marcos, A.P.: Is the PPG signal chaotic? IEEE Access **8**, 107700–107715 (2020)
10. Friel, C.P., Garber, C.E.: An examination of the relationship between motivation, physical activity, and wearable activity monitor use. J. Sport Exerc. Psychol. **42**(2), 153–160 (2020)
11. Glass, L.: Introduction to controversial topics in nonlinear science: is the normal heart rate chaotic? Chaos: Interdisc. J. Nonlinear Sci. **19**(2), 028501 (2009)
12. Hannan, A.L., Harders, M.P., Hing, W., Climstein, M., Coombes, J.S., Furness, J.: Impact of wearable physical activity monitoring devices with exercise prescription or advice in the maintenance phase of cardiac rehabilitation: systematic review and meta-analysis. BMC Sports Sci. Med. Rehabil. **11**(1), 1–21 (2019)
13. Ismail, S., Akram, U., Siddiqi, I.: Heart rate tracking in photoplethysmography signals affected by motion artifacts: a review. EURASIP J. Adv. Sig. Process. **2021**(1), 1–27 (2021). https://doi.org/10.1186/s13634-020-00714-2
14. Jean Effil, N., Rajeswari, R.: Wavelet scattering transform and long short-term memory network-based noninvasive blood pressure estimation from photoplethysmograph signals. SIViP **16**(1), 1–9 (2021). https://doi.org/10.1007/s11760-021-01952-z
15. Kamshilin, A.A., et al.: A new look at the essence of the imaging photoplethysmography. Sci. Rep. **5**(1), 1–9 (2015)
16. Kossi, O., Lacroix, J., Ferry, B., Batcho, C.S., Julien-Vergonjanne, A., Mandigout, S.: Reliability of ActiGraph GT3X+ placement location in the estimation of energy expenditure during moderate and high-intensity physical activities in young and older adults. J. Sports Sci., 1–8 (2021)
17. Kumar, A., Komaragiri, R., Kumar, M., et al.: A review on computation methods used in photoplethysmography signal analysis for heart rate estimation. Arch. Comput. Methods Eng., 1–20 (2021)
18. Mallat, S.: Understanding deep convolutional networks. Philos. Trans. Roy. Soc. A Math. Phys. Eng. Sci. **374**(2065), 20150203 (2016)
19. Manninger, M., et al.: Role of wearable rhythm recordings in clinical decision making-the wEHRAbles project. Clin. Cardiol. **43**(9), 1032–1039 (2020)
20. de Pedro-Carracedo, J., Ugena, A.M., Gonzalez-Marcos, A.P.: Dynamical analysis of biological signals with the 0–1 test: a case study of the photoplethysmographic (PPG) signal. Appl. Sci. **11**(14), 6508 (2021)
21. Reiss, A., Indlekofer, I., Schmidt, P., Van Laerhoven, K.: Deep PPG: large-scale heart rate estimation with convolutional neural networks. Sensors **19**(14), 3079 (2019)
22. Salehizadeh, S., Dao, D., Bolkhovsky, J., Cho, C., Mendelson, Y., Chon, K.H.: A novel time-varying spectral filtering algorithm for reconstruction of motion artifact corrupted heart rate signals during intense physical activities using a wearable photoplethysmogram sensor. Sensors **16**(1), 10 (2016)
23. Schäck, T., Muma, M., Zoubir, A.M.: Computationally efficient heart rate estimation during physical exercise using photoplethysmographic signals. In: 2017 25th European Signal Processing Conference (EUSIPCO), pp. 2478–2481. IEEE (2017)
24. Sprott, J.C.: Chaos and Time-Series Analysis, vol. 69. Oxford University Press (2003)
25. Staszewski, W., Worden, K.: Wavelet analysis of time-series: coherent structures, chaos and noise. Int. J. Bifurcat. Chaos **9**(03), 455–471 (1999)

26. Sviridova, N., Sakai, K.: Human photoplethysmogram: new insight into chaotic characteristics. Chaos Solitons Fractals **77**, 53–63 (2015)
27. Sviridova, N., Zhao, T., Aihara, K., Nakamura, K., Nakano, A.: Photoplethysmogram at green light: where does chaos arise from? Chaos Solitons Fractals **116**, 157–165 (2018)
28. Tamura, T.: Current progress of photoplethysmography and SPO 2 for health monitoring. Biomed. Eng. Lett. **9**(1), 21–36 (2019)
29. Tsuda, I., Tahara, T., Iwanaga, H.: Chaotic pulsation in human capillary vessels and its dependence on the mental and physical conditions. In: Proceedings of the Annual Meeting of Biomedical Fuzzy Systems Association: BMFSA 4, pp. 1–40. Biomedical Fuzzy Systems Association (1992)
30. Vandecasteele, K., et al.: Automated epileptic seizure detection based on wearable ECG and PPG in a hospital environment. Sensors **17**(10), 2338 (2017)
31. Wilkosz, M., Szczęsna, A.: Multi-headed Conv-LSTM network for heart rate estimation during daily living activities. Sensors **21**(15), 5212 (2021)

# Convolutional Neural Network Compression via Tensor-Train Decomposition on Permuted Weight Tensor with Automatic Rank Determination

Mateusz Gabor(✉) and Rafał Zdunek

Faculty of Electronics, Photonics, and Microsystems,
Wroclaw University of Science and Technology, Wybrzeze Wyspianskiego 27,
50-370 Wroclaw, Poland
{mateusz.gabor,rafal.zdunek}@pwr.edu.pl

**Abstract.** Convolutional neural networks (CNNs) are among the most commonly investigated models in computer vision. Deep CNNs yield high computational performance, but their common issue is a large size. For solving this problem, it is necessary to find effective compression methods which can effectively reduce the size of the network, keeping the accuracy on a similar level. This study provides important insights into the field of CNNs compression, introducing a novel low-rank compression method based on tensor-train decomposition on a permuted kernel weight tensor with automatic rank determination. The proposed method is easy to implement, and it allows us to fine-tune neural networks from decomposed factors instead of learning them from scratch. The results of this study examined on various CNN architectures and two datasets demonstrated that the proposed method outperforms other CNNs compression methods with respect to parameter and FLOPS compression at a low drop in the classification accuracy.

**Keywords:** Neural network compression · Convolutional neural network · Tensor decomposition · Tensor train decomposition

## 1 Introduction

The area of convolutional neural networks (CNNs) has attracted growing attention in the field of computer vision for achieving one of the best results in tasks such as image classification [11], segmentation [27] or object detection [26].

However, achieving better results of CNNs is mostly done by designing deeper neural networks, which translates into larger architectures requiring more space and more computing power. Because most of the deep neural networks are overparametrized [5], there exists a possibility of compressing them without reducing


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 654–667, 2022.
https://doi.org/10.1007/978-3-031-08757-8_54

the quality of the network significantly. The neural network compression methods can be classified into weight sharing, pruning, knowledge distillation, quantization and low-rank approximations [1,16,22]. The weight sharing method is the simplest form of compressing a neural network size, in which the weights of the neural network are shared between layers. From this approach, clustering-based weight sharing can be distinguished, in which the clustering is performed on weights, and at the end clustered weights are merged into new compressed weights. In the pruning approach, the redundant connections between neurons are removed, which results in a lower number of parameters and FLOPs. In most cases, the fine-tuning is necessary to recover the original accuracy of the network and often pruning/fine-tuning is alternately repeated in loop to gain larger compression. Quantization is another approach to compress neural network weights. In this method, the neural network weights are represented in a lower-precision format, the most popular is INT8, but the most extreme quantization is based on binary weights. On the other hand, knowledge distillation methods learn a small (student) network from a large one (teacher) using supervision. In short, a student network mimics a teacher network and leverages the knowledge of the teacher, achieving a similar or higher accuracy.

Besides the aforementioned methods, it is possible to compress the neural network using dimensionality reduction techniques such as matrix/tensor decompositions [24] in which the neural network weights are represented in a low-rank format. The low-rank compression methods can be divided into direct decomposition and tensorization. Direct decomposition methods use the factors obtained from the decomposition as new approximated weights, perform all operations on them, and are simple in implementation because they use basic convolutional neural network blocks from deep learning frameworks. The most popular two approaches of using the direct tensor decomposition to compress convolutional layers are the Tucker-2 [15] and CP [18] decomposition. The CP decomposition transforms the original weight tensor into a pipeline of two $1 \times 1$ convolutions and two depthwise separable convolutions, and the Tucker-2 into two $1 \times 1$ convolutions and one standard convolution, which is the same as the Bottleneck block in ResNet networks. Recently, Hameed *et al.* [9] proposed a new direct tensor decomposition method in which the Kronecker product decomposition is generalized to be applied to compress CNN weights. On the other hand, in the tensorization approach, the original weight tensor is tensorized into a higher-order tensor format and new weights are initialized randomly. In this approach, the decomposition algorithm is not used, and therefore the pretrained information from the baseline network is lost. By using tensorization, the achieved compression is relatively high, but the quality of the compressed network is significantly worse than the baseline model. The first tensorization approach to CNN compression was proposed by Garipov *et al.* [8], in which the tensor-train (TT) format was used to matricized weight tensor. The input feature maps tensor was reshaped into a matrix, and the convolution operation was performed as a sequence of tensor contractions. Garipov *et al.* also proposed a *naive* direct TT compression method in which the weight tensor was directly decomposed. All the

decomposed cores were kept in memory, but during the convolution operation, the TT cores were reshaped into the original weight tensor, and the initialization was performed randomly. Among other methods of tensorization, one can mention the tensor ring format [21] or hierarchical Tucker format [31].

In this study, we propose a novel direct low-rank neural network compression method using direct tensor-train decomposition on the permuted kernel weight tensor with automatic rank determination. This method will be referred to as TTPWT. In our approach, each original convolutional layer is replaced and initialized with a sequence of four layers obtained from the decomposed factors, and the original convolution is approximated with four smaller convolutions, which is profitable both with respect to computational and storage complexity. The proposed compression method was applied to four neural networks: TT-conv-CNN [8], VGGnet [28], ResNet-56 [11] and ResNet-110 [11]. The experiments run on the CIFAR-10 and CIFAR-100 datasets showed that the TTPWT considerably outperforms many state-of-the-art compression methods with respect to parameter and FLOPS compression at a low drop in the classification accuracy.

The remainder of this paper is organized as follows. Section 2 presents the notation and the preliminaries to fundamental mathematical operations on tensors. It also contains a short description of the TT decomposition method. The proposed TT-based compression model is presented in Sect. 3. Numerical experiments performed using various CNN architectures tested on the CIFAR-10 and CIFAR-100 datasets are presented and discussed in Sect. 4. The final section provides concluding statements.

## 2 Preliminary

*Notation:* Multi-way arrays, matrices, vectors, and scalars are denoted by calligraphic uppercase letters (e.g., $\mathcal{X}$), boldface uppercase letters (e.g., $\boldsymbol{X}$), lowercase boldface letters (e.g., $\boldsymbol{X}$), and unbolded letters (e.g., $x$), respectively. Multi-way arrays will be equivalently referred to as tensors. We used Kolda's notation [17] for standard mathematical operations on tensors.

**Mode-$n$unfolding**: The mode-$n$ unfolding of the $N$-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ rearranges its entries by placing its mode-$n$ fibers as the columns of matrix $\boldsymbol{X}_{(n)} = [x_{i_n,j}] \in \mathbb{R}^{I_n \times \prod_{p \neq n} I_p}$ for $n \in \{1, \ldots, N\}$, where $j = 1 + \sum_{k=1 k \neq n}^{N} (i_k - 1) j_k$ with $j_k = \prod_{m=1 m \neq n}^{k-1} I_m$, and $i_n = 1, \ldots, I_n$.

**Mode-$\{n\}$ Canonical Matricization**: This matricization reshapes tensor $\mathcal{X}$ into matrix $\boldsymbol{X}_{<n>} \in \mathbb{R}^{\prod_{p=1}^{n} I_p \times \prod_{r=n+1}^{N} I_r}$ by mapping tensor element $x_{i_1,\ldots,i_N}$ to matrix element $x_{i,j}$, where $i = 1 + \sum_{p=1}^{n} (i_p - 1) \prod_{m=1}^{p-1} I_m$ and $j = 1 + \sum_{r>n}^{N} (i_r - 1) \prod_{m=n+1}^{r-1} I_m$. The mode-$n$ unfolding is a particular case of the mode-$\{n\}$ canonical matricization.

**Mode-$n$product** (also known as the tensor-matrix product): The mode-$n$ product of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ with matrix $\boldsymbol{U} \in \mathbb{R}^{J \times I_n}$ is defined by

$$\mathcal{Z} = \mathcal{X} \times_n \boldsymbol{U}, \tag{1}$$

where $\mathcal{Z} = [z_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N}] \in \mathbb{R}^{I_1\times\ldots\times I_{n-1}\times J\times I_{n+1}\times\ldots\times I_N}$, and

$$z_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \bigsqcup_{i_n=1}^{I_n} x_{i_1,i_2,\ldots,i_N} u_{j,i_n}.$$

**Tensor Contraction**: The tensor contraction of tensor $\mathcal{X} = [x_{i_1,\ldots,i_N}] \in \mathbb{R}^{I_1\times\ldots\times I_N}$ across its $n$-th mode with tensor $\mathcal{Y} = [y_{j_1,\ldots,j_M}] \in \mathbb{R}^{J_1\times\ldots\times J_M}$ across its $m$-th mode, provided that $I_n = J_m$, gives tensor $\mathcal{Z} = \mathcal{X} \times_n^m \mathcal{Y}$ whose entries are given by:

$$z_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \bigsqcup_{i_n=1}^{I_n} x_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} y_{j_1,\ldots,j_{m-1},i_n,j_{m+1},\ldots,j_M}. \tag{2}$$

For the matrices: $\boldsymbol{A} \times_2^1 \boldsymbol{B} = \boldsymbol{AB}$. The contraction: $\times_N^1$ will be denoted by the symbol $\bullet$. Thus: $\mathcal{X} \bullet \mathcal{Y} = \mathcal{X} \times_N^1 \mathcal{Y}$.

**Kruskal Convolution**: Let $\mathcal{X} = [x_{i_1,i_2,c}] \in \mathbb{R}^{I_1\times I_2\times C}$ be any activation tensor in any convolutional layer with $C$ input channels, $\mathcal{W} = [w_{t,c,d_1,d_2}] \in \mathbb{R}^{T\times C\times D_1\times D_2}$ be the kernel weight tensor, $\Delta$ be the stride, and $P$ be the zero-padding size. The Kruskal convolution maps input tensor $\mathcal{X}$ to output tensor $\mathcal{Y} = [y_{\tilde{i}_1,\tilde{i}_2,t}] \in \mathbb{R}^{\tilde{I}_1\times\tilde{I}_2\times T}$ by the following linear mapping:

$$y_{\tilde{i}_1,\tilde{i}_2,t} = x_{i_1,i_2,c} \triangleright w_{t,c,d_1,d_2} = \bigsqcup_{c=1}^{C} \bigsqcup_{d_1=1}^{D} \bigsqcup_{d_2=1}^{D} w_{t,c,d_1,d_2} x_{i_1(d_1),i_2(d_2),c}, \tag{3}$$

where $i_1(d_1) = (\tilde{i}_1 - 1)\Delta + i_1 - P$ and $i_2(d_2) = (\tilde{i}_2 - 1)\Delta + i_2 - P$.

$1 \times 1$ **Convolution**: If $D = 1$, $\Delta = 1$, and $P = 0$, then $\mathcal{W} \in \mathbb{R}^{T\times C\times 1\times 1}$, and the Kruskal convolution comes down to the $1 \times 1$ convolution: $y_{i_1,i_2,t} = \int_{c=1}^{C} w_{t,c} x_{i_1,i_2,c}$. Using the notation of the mode-$n$ product in (1), the $1 \times 1$ convolution takes the form:

$$\mathcal{Y} = \mathcal{X} \times_3 \boldsymbol{W}, \tag{4}$$

where $\boldsymbol{W} = [w_{tc}] \in \mathbb{R}^{T\times C}$.

**Tensor Train (TT) Decomposition**: The TT model [23] decomposes tensor $\mathcal{X} = [x_{i_1,\ldots,i_N}] \in \mathbb{R}^{I_1\times\ldots\times I_N}$ to a chain of smaller (3-way) core tensors that are connected by the tensor contraction with operator $\bullet$. It can be formulated as follows:

$$\mathcal{X} = \mathcal{X}^{(1)} \bullet \mathcal{X}^{(2)} \bullet \ldots \bullet \mathcal{X}^{(N)}, \tag{5}$$

where $\mathcal{X}^{(n)}$ is the $n$-th core tensor of size $R_{n-1} \times I_n \times R_n$ for $n = 1, \ldots, N$. The number $\{R_0, \ldots, R_N\}$ determine the TT ranks. Assuming $R_0 = R_N = 1$, we

have $\mathcal{X}^{(1)} = \boldsymbol{X}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$ and $\mathcal{X}^{(N)} = \boldsymbol{X}^{(N)} \in \mathbb{R}^{R_{N-1} \times I_N}$, i.e. the first and the last core tensors become matrices. Model (5) can be expressed equivalently as:

$$x_{i_1,\ldots,i_N} = \bigsqcup_{r_1=1}^{R_1} \bigsqcup_{r_2=1}^{R_2} \cdots \bigsqcup_{r_{N-1}=1}^{R_{N-1}} x^{(1)}_{i_1,r_1} x^{(2)}_{r_1,i_2,r_2} \cdots x^{(N-1)}_{r_{N-2},i_{N-1},r_{N-1}} x^{(N)}_{r_{N-1},i_N}, \quad (6)$$

where $\forall n : \mathcal{X}^{(n)} = [x^{(n)}_{r_{n-1},i_n,r_n}] \in \mathbb{R}^{R_{n-1} \times I_n \times R_n}$.

Assuming $I_1 = \ldots = I_N = I$ and $R_1 = \ldots = R_N = R$, the storage complexities of the CANDECOM/PARAFAC (CP) [2,10], Tucker [29], and TT decomposition models can be approximated by $\mathcal{O}(NIR)$, $\mathcal{O}(NIR + R^N)$, and $\mathcal{O}(NIR^2)$. It is thus obvious that the CP model has the lowest storage complexity, but its flexibility in adapting to the observed data is very low, especially for tensors that have strongly unbalanced modes. Unfortunately, this is the case in the discussed problem because two modes of the decomposed tensor have small dimensions, but the other modes are large. Hence, it is difficult to select the optimal rank. The Tucker decomposition relaxes these problems considerably, but its storage complexity grows up exponentially with the size of the core tensor, which is also not favorable in our case because the ranks for large modes are usually pretty large. The TT model assures the best trade-off between the CP and Tucker decompositions, alleviating the curse of dimensionality and yielding a flexible decomposition with multiple TT ranks. Hence, these advantages of the TT model motivate this study.

## 3 Proposed Method

We assume that each convolutional layer has $C$ input and $T$ output channels, and the size of the filter is $D \times D$. Hence, it can be represented by the kernel weight tensor $\mathcal{W} = [w_{t,c,d_1,d_2}] \in \mathbb{R}^{T \times C \times D \times D}$. The input data is represented by activation tensor $\mathcal{X} = [x_{i_1,i_2,c}] \in \mathbb{R}^{I_1 \times I_2 \times C}$ that consists of $C$ activation maps – each has the resolution of $I_1 \times I_2$ pixels. Each layer performs a linear mapping of tensor $\mathcal{X}$ to output activation tensor $\mathcal{Y} = [y_{\tilde{i}_1,\tilde{i}_2,t}] \in \mathbb{R}^{\tilde{I}_1 \times \tilde{I}_2 \times T}$, where the mapping is determined by the Kruskal convolution in (3). Each output activation map has the resolution of $\tilde{I}_1 \times \tilde{I}_2$ pixels, and there are $T$ output channels.

### 3.1 Model

To reduce the number of parameters and FLOPS in each convolutional layer, the kernel weight tensor $\mathcal{W}$ is decomposed with the TT model.

*Remark 1.* Note that if $\mathcal{W} \in \mathbb{R}^{T \times C \times D \times D}$ is decomposed according to (5), ranks $R_2$ and $R_3$ cannot be greater than $D^2$ and $D$, respectively. This restriction limits the flexibility of compression only to rank $R_1$. Furthermore, the 3D core tensor capturing the second mode of $\mathcal{W}$ could not be processed with a simple $1 \times 1$

convolution. Thus, we propose to apply the circular permutation to $\mathcal{W}$ with one left shift lag. Thus:

$$\tilde{\mathcal{W}} = \texttt{circular\_permutation}\,(\mathcal{W}, -1) \in \mathbb{R}^{C\times D\times D\times T}. \tag{7}$$

Applying the TT decomposition to $\tilde{\mathcal{W}} = [\tilde{w}_{c,d_1,d_2,t}]$, we have:

$$\tilde{w}_{c,d_1,d_2,t} = \bigsqcup_{r_1=1}^{R_1}\bigsqcup_{r_2=1}^{R_2}\bigsqcup_{r_3=1}^{R_3} \tilde{w}^{(1)}_{c,r_1}\tilde{w}^{(2)}_{r_1,d_1,r_2}\tilde{w}^{(3)}_{r_2,d_2,r_3}\tilde{w}^{(4)}_{r_3,t}. \tag{8}$$

Inserting model (8) to mapping (3) and rearranging the summands, we get:

$$\begin{aligned} y_{\tilde{i}_1,\tilde{i}_2,t} &= \bigsqcup_{c=1}^{C}\bigsqcup_{d_1=1}^{D}\bigsqcup_{d_2=1}^{D}\bigsqcup_{r_1=1}^{R_1}\bigsqcup_{r_2=1}^{R_2}\bigsqcup_{r_3=1}^{R_3} \tilde{w}^{(1)}_{c,r_1}\tilde{w}^{(2)}_{r_1,d_1,r_2}\tilde{w}^{(3)}_{r_2,d_2,r_3}\tilde{w}^{(4)}_{r_3,t}x_{i_1(d_1),i_2(d_2),c} \\ &= \bigsqcup_{r_3=1}^{R_3}\tilde{w}^{(4)}_{r_3,t}\Bigg[\bigsqcup_{d_2=1}^{D}\bigsqcup_{r_2=1}^{R_2}\tilde{w}^{(3)}_{r_2,d_2,r_3}\bigsqcup_{d_1=1}^{D}\bigsqcup_{r_1=1}^{R_1}\tilde{w}^{(2)}_{r_1,d_1,r_2} \\ &\quad\times\Bigg(\bigsqcup\Bigg(\underbrace{\bigsqcup_{c=1}^{C}\tilde{w}^{(1)}_{c,r_1}x_{i_1(d_1),i_2(d_2),c}}_{1\times 1\text{ conv.}}\Bigg)\Bigg)\Bigg] \\ &= \bigsqcup_{r_3=1}^{R_3}\tilde{w}^{(4)}_{r_3,t}\Bigg[\bigsqcup_{d_2=1}^{D}\bigsqcup_{r_2=1}^{R_2}\tilde{w}^{(3)}_{r_2,d_2,r_3}\Bigg(\bigsqcup\Bigg(\underbrace{\bigsqcup_{d_1=1}^{D}\bigsqcup_{r_1=1}^{R_1}\tilde{w}^{(2)}_{r_1,d_1,r_2}z_{i_1(d_1),i_2(d_2),r_1}}_{D_1\times 1\text{ conv.}}\Bigg)\Bigg)\Bigg] \\ &= \bigsqcup_{r_3=1}^{R_3}\tilde{w}^{(4)}_{r_3,t}\Bigg[\underbrace{\bigsqcup_{d_2=1}^{D}\bigsqcup_{r_2=1}^{R_2}\tilde{w}^{(3)}_{r_2,d_2,r_3}z^{(V)}_{\tilde{i}_1,i_2(d_2),r_2}}_{1\times D_2\text{ conv.}}\Bigg] = \underbrace{\bigsqcup_{r_3=1}^{R_3}\tilde{w}^{(4)}_{r_3,t}z^{(V,H)}_{\tilde{i}_1,\tilde{i}_2,r_3}}_{1\times 1\text{ conv.}} \end{aligned} \tag{9}$$

It can be easy to note that $z_{i_1,i_2,r_1}$ in (9) can be computed with the $1\times 1$ convolution. According to (4), we have:

$$\mathcal{Z} = \mathcal{X}\times_3 \tilde{\boldsymbol{W}}^{(1)T} \in \mathbb{R}^{I_1\times I_2\times R_1}, \tag{10}$$

where $\tilde{\boldsymbol{W}}^{(1)} = [\tilde{w}^{(1)}_{c,r_1}] \in \mathbb{R}^{C\times R_1}$. Physically, to perform operation (10), the first sublayer with the $1\times 1$ convolutions in the analyzed convolutional layer is created. The activation tensor $\mathcal{Z}$ computed in the first sub-layer is then provided to the second convolutional sublayer represented by $\tilde{\mathcal{W}} = [\tilde{w}^{(2)}_{r_1,d_1,r_2}] \in \mathbb{R}^{R_1\times D\times R_2}$,

which is much smaller than $\mathcal{W}$, and this sublayer computes the 1D convolutions along the 1-st mode (vertically):

$$z^{(V)}_{\tilde{i}_1,i_2(d_2),r_2} = \prod_{d_1=1}^{D} \prod_{r_1=1}^{R_1} \tilde{w}^{(2)}_{r_1,d_1,r_2} z_{i_1(d_1),i_2(d_2),r_1} \tag{11}$$

As a result, we get the second-sublayer output activation tensor

$$\mathcal{Z}^{(V)} = [z^{(V)}_{\tilde{i}_1,i_2(d_2),r_2}] \in \mathbb{R}^{\tilde{I}_1 \times I_2 \times R_2}.$$

Next, the third 1D convolutional sublayer is created to compute the 1D convolutions along the horizontal direction. The output activation tensor obtained from this sublayer has the form: $\mathcal{Z}^{(V,H)} = [z^{(V,H)}_{\tilde{i}_1,\tilde{i}_2,r_3}] \in \mathbb{R}^{\tilde{I}_1 \times \tilde{I}_2 \times R_3}$. Finally, the fourth sublayer is created, which performs $1 \times 1$ convolutions according to the model:

$$\mathcal{Y} = \mathcal{Z}^{(V,H)} \times_3 \tilde{\boldsymbol{W}}^{(4)T} \in \mathbb{R}^{\tilde{I}_1 \times \tilde{I}_2 \times T}, \tag{12}$$

where $\tilde{\boldsymbol{W}}^{(4)} = [\tilde{w}^{(4)}_{r_3,t}] \in \mathbb{R}^{R_3 \times T}$.

### 3.2 TT Decomposition Algorithm

The TT decomposition of $\tilde{\mathcal{W}}$ in (7) can be obtained by using sequential SVD-based projections. In the first step, TSVD with a given precision $\rho_1$ is applied to $\tilde{\mathcal{W}}$ unfolded with respect to its first-mode. Thus:

$$\tilde{\boldsymbol{W}}_{(1)} = \boldsymbol{U}\boldsymbol{\alpha}\boldsymbol{V}^T + \boldsymbol{E}_1, \tag{13}$$

under the assumption the truncation error satisfies the condition $||\boldsymbol{E}_1||_F \leq \rho_1$. Matrix $\tilde{\boldsymbol{W}}^{(1)} \in \mathbb{R}^{C \times R_1}$ is created from $\boldsymbol{U}$ that contains the first $R_1$ left singular vectors (associated with the most significant singular values) of $\tilde{\boldsymbol{W}}_{(1)}$. Note that rank $R_1$ is determined by a given threshold $\rho_1$ for the truncation error. In the second step, $\tilde{\boldsymbol{W}}^{(2)} \in \mathbb{R}^{R_1 I_2 \times R_2}$ is created from the first $R_2$ left singular vectors of the matrix obtained by reshaping matrix $\boldsymbol{\alpha}\boldsymbol{V}^T$ using the mode-2 canonical matricization. In this step, $||\boldsymbol{E}_2||_F \leq \rho_2$ and the core tensor is obtained by reshaping $\tilde{\boldsymbol{W}}^{(2)}$ accordingly. The similar procedure is applied in the third step, where $\tilde{\mathcal{W}}^{(3)} \in \mathbb{R}^{R_2 \times I_3 \times R_3}$ is created from the first $R_3$ singular vectors, and $\tilde{\boldsymbol{W}}^{(4)} \in \mathbb{R}^{R_3 \times T}$ is created from the scaled right singular vectors. Oseledets [23] showed that $||\tilde{\mathcal{W}} - \tilde{\mathcal{W}}^{(1)} \bullet \ldots \bullet \tilde{\mathcal{W}}^{(N)}||_F \leq \sqrt{\sum_{n=1}^{N-1} \rho_n^2}$. Assuming $\rho = \rho_1 = \ldots = \rho_{N-1}$, then the truncation threshold can be set to $\rho = \frac{\epsilon}{\sqrt{N-1}}||\tilde{\mathcal{W}}||_F$, where $\omega > 0$ is a prescribed relative error.

In our approach, the optimal rank of TSVD for matrix $\boldsymbol{M} \in \mathbb{R}^{P \times R}$ in each step was computed by using the energy-threshold criterion. Thus:

$$R_* = \arg\min_j \left\{ \left\{ \frac{\sum_{i=1}^{j} \varphi_i^2}{\sum_{i=1}^{I} \varphi_i^2} > \tau \right\} \right\}, \tag{14}$$

**Algorithm 1. TT-SVD**

**Input** : $\mathcal{W} \in \mathbb{R}^{T \times C \times D \times D}$ – input kernel weight tensor, $\tau$ - threshold
**Output**: $\{\mathcal{W}^{(1)}, ..., \mathcal{W}^{(4)}\}$ - estimated core tensors

Compute $\tilde{\mathcal{W}} \in \mathbb{R}^{C \times D \times D \times T}$ with (7) and set $R_0 = 1$ and $N = 4$,
$\boldsymbol{M} = \tilde{\boldsymbol{W}}_{(1)} = \texttt{unfolding}(\tilde{\mathcal{W}}, 1)$; `// Unfolding`
**for** $n = 1, \ldots, N-1$ **do**
  Compute: $\left[\tilde{\boldsymbol{U}}, \tilde{\boldsymbol{S}}, \tilde{\boldsymbol{V}}, R_n\right] = \texttt{TSVD}_\delta(\boldsymbol{M}, \tau)$; `// TSVD`
  $\tilde{\mathcal{W}}^{(n)} = \texttt{reshape}(\tilde{\boldsymbol{U}}, [R_{n-1}, I_n, R_n])$
  $\boldsymbol{M} = \texttt{reshape}(\tilde{\boldsymbol{S}}\tilde{\boldsymbol{V}}^T, [R_n I_{n+1}, \prod_{p=n+2}^{N} I_p])$; `// Canonical matricization`
**end**
$\tilde{\mathcal{W}}^{(4)} = \texttt{reshape}(\boldsymbol{M}, [R_{N-1}, I_N, 1])$

where $Q = \min\{P, Q\}$, $\varphi_i$ is the $i$-th singular value of $\boldsymbol{M}$, and $\tau = \frac{\epsilon}{\sqrt{N-1}}$ is a given threshold. The energy captured by $i$ components (singular vectors) is expressed in the nominator of (14), the total energy is presented in the denominator.

Due to the low-rank approximation, the TT model always assures the compression [25], i.e.

$$R_n \leq \min\left\{\prod_{i=1}^{n} I_i, \prod_{j=n+1}^{N} I_j\right\}, \quad \text{for} \quad n = 1, \ldots, N-1. \tag{15}$$

The complete sequential routine is presented in Algorithm 1. Function $\texttt{TSVD}_\delta$ performs the $\rho$-truncated SVD at a given threshold $\rho$, where the optimal rank $R_*$ is computed by the energy-based criterion (14).

### 3.3 Implementation

The procedure for training/fine-tuning networks was implemented in the deep learning framework *PyTorch* and the *tensor-train* decomposition in *Matlab*. The convolutional kernel is the main component of the convolutional layer, which is represented as the 4-th order tensor (top block, Fig. 1). After using permutation, the weight tensor can be decomposed into four factors, including two matrices and two 3-rd order tensors. All the factors are used as new weights in a sequence of four sublayers (Tensor-Train model, Fig. 1). Because the basic class of convolutional layer in *PyTorch* accepts only 4-th order tensor as weights, it is necessary to add extra two dimensions to matrices: $\tilde{\boldsymbol{W}}^{(1)} \in \mathbb{R}^{C \times R_1} \rightarrow \tilde{\mathcal{W}}^{(1)} \in \mathbb{R}^{R_1 \times C \times 1 \times 1}$ (Fig. 1, sublayer Conv2D.1), $\tilde{\boldsymbol{W}}^{(4)} \in \mathbb{R}^{R_3 \times T} \rightarrow \tilde{\mathcal{W}}^{(4)} \in \mathbb{R}^{T \times R_3 \times 1 \times 1}$ (Fig. 1, sublayer Conv2D.4) and extra one dimension to 3-rd order tensors $\tilde{\mathcal{W}}^{(2)} \in \mathbb{R}^{R_1 \times D \times R_2} \rightarrow \tilde{\mathcal{W}}^{(2)} \in \mathbb{R}^{R_2 \times R_1 \times D \times 1}$ (Fig. 1, sublayer Conv2D.2), $\tilde{\mathcal{W}}^{(3)} \in \mathbb{R}^{R_3 \times D \times R_2} \rightarrow \tilde{\mathcal{W}}^{(3)} \in \mathbb{R}^{R_3 \times R_2 \times 1 \times D}$ (Fig. 1, sublayer Conv2D.3) and permute the modes accordingly.

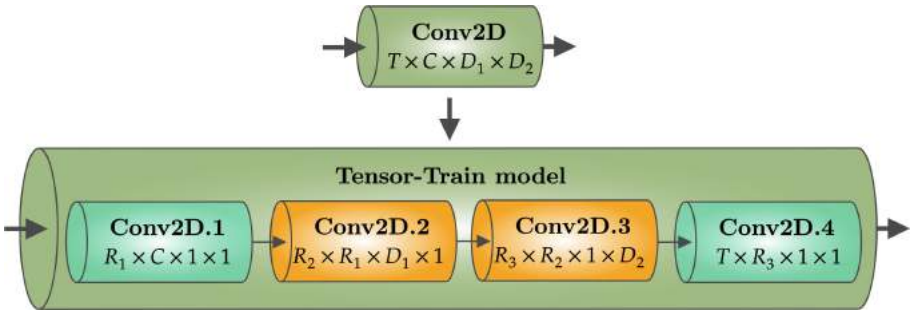



**Fig. 1.** Visual representation of how the decomposed factors are used as new weights in *PyTorch* framework (output channels × input channels × filter height × filter width) for the compressed convolutional layer in the TT model.

### 3.4 Computational Complexity

The space and time complexity of the convolution is defined as $\mathcal{O}(CTD^2)$ and $\mathcal{O}(CTD^2I_1I_2)$. By applying the tensor-train decomposition, the time and space complexity is bounded by $\mathcal{O}(CR_1 + R_2D(R_1 + R_3) + R_3T)$ and $\mathcal{O}(R_1I_1I_2 + R_2D(R_1I_1I_2 + R_3\tilde{I}_1I_2) + R_3T\tilde{I}_1\tilde{I}_2)$ respectively, where $I_1$ and $I_2$ define the height and width of the input image, respectively, and $\tilde{I}_1$ and $\tilde{I}_2$ define the reduced height and width after convolution.

## 4 Results

We evaluated our method on two datasets (CIFAR-10 and CIFAR-100). Each consists of 60,000 examples, including 50,000 in the training dataset, and 10,000 in the validation dataset with 10 and 100 classes respectively. To evaluate effectiveness of our method on networks of various sizes, we selected the following networks: TT-conv-CNN [8], VGGnet [28], ResNet-56 [11] and ResNet-110 [11]. The networks cover the range of models with a medium to a large number of parameters and FLOPS. The total number of FLOPS and the parameters of the mentioned networks are listed in Table 1. The compression experiments were performed with the following scheme:

energy threshold selection ⟶ baseline CNN compression ⟶ fine-tuning.

All the convolutional layers were compressed in each neural network except for the first one whose size is small. All the baseline networks were trained according to the source guidelines. TT-conv-CNN was trained for 100 epochs using stochastic gradient descent (SGD) with a momentum of 0.9, the weight decay was set to 0, the initial learning rate was set to 0.1, and it was decreased by a factor of 0.1 after every 20 epochs. For the fine-tuning process, all the hyperparameters remained the same. VGGnet, ResNet-56, and ResNet-110 were trained for 200 epochs using the SGD with a momentum of 0.9, the weight decay was

set to $10^{-4}$, the initial learning rate was set to 0.1, and it was decreased by a factor of 0.1 after 80 and 120 epochs for VGGnet, and after 100 and 150 for ResNets. In the fine-tuning step, the learning rate was lowered to 0.01 and the weight decay was increased to $10^{-3}$ for the VGGnet, and the hyperparameters were unchanged for ResNet-56 and ResNet-110.

To evaluate the network compression and performance, we used two metrics, such as the parameter compression ratio (PCR) that is defined as $\Downarrow Param = \frac{\text{Param(baseline network)}}{\text{Param(compressed network)}}$, and the FLOPS compression ratio (FCR) defined as $\Downarrow FLOPS = \frac{\text{FLOPS(baseline network)}}{\text{FLOPS(compressed network)}}$. The quality of the network was evaluated with the drop in the classification accuracy of the compressed network with respect to the baseline network, i.e. $\Delta Acc = Acc_{compressed} - Acc_{baseline}$. The error rate is often shown alongside with the accuracy in the literature. However, the error rate may be misleading since we fine-tune the neural networks from decomposed factors. Hence, the accuracy is sufficient to be shown for better interpretability of results. The values of PCR or FCR are not provided in all the papers, which we refer to as the reference results. Hence, the unavailable data are marked with the "–" sign in the tables.

**Table 1.** Total number of FLOPs and parameters for baseline networks.

| Network | Params | FLOPS |
|---|---|---|
| TT-conv-CNN | 558 K | 105 M |
| ResNet-56 | 853 K | 125 M |
| ResNet-110 | 1.73 M | 255 M |
| VGGnet | 20 M | 399 M |

### 4.1 CIFAR-10

**TT-conv-CNN:** Table 2 shows the results obtained for the TT-conv-CNN compression. We compared our method with the tensorized tensor-train version of the matricized weight tensor (TT-conv), direct tensorized tensor-train weight tensor (TT-conv (naive)), and the weight sharing method – Deep $k$-Means [32]. As we can see, our method outperforms both TT-based methods proposed by Garipov *et al.* in terms of PCR, FCR, and the drop in accuracy is at a much lower level. Compared with Deep $k$-Means, our method achieved better accuracy with higher compression.

**VGGnet:** The VGGnet network is a modified VGG-19 neural network adopted for CIFAR datasets. It is the largest neural network analyzed in this study, with 20M of parameters and 399M of FLOPS. Compression of VGGnet using our method was compared with the following pruning approaches: DCP [35], Random-DCP [35], WM+ [35], CP [14] and PFEC [19]. The results given in Table 3 demonstrate that our method outperforms all the compared approaches in terms of PCR and FCR. Our compressed network achieved a positive drop

(gain) in the accuracy compared to the baseline network, and only DCP obtained a higher gain but with worse parameter compression. Moreover, TTPWT reduces FLOPS nearly 2.35 times more than DCP.

**Table 2.** Results of the TT-conv-CNN [8] compression on the CIFAR-10 validation dataset. Different rows of the TT-conv and TT-conv (naive) mean different ranks. The value in parentheses denotes the energy threshold.

| Method | ⋆ Acc | ‖ Param | ‖ FLOPS |
|---|---|---|---|
| TT-conv-1 [8] | −0.80 | 2.02 | – |
| TT-conv-2 [8] | −1.50 | 2.53 | – |
| TT-conv-3 [8] | −1.40 | 3.23 | – |
| TT-conv-4 [8] | −2.00 | 4.02 | – |
| TT-conv-1 (naive) [8] | −2.40 | 2.02 | – |
| TT-conv-2 (naive) [8] | −3.10 | 2.90 | – |
| Deep k-Means [32] | +0.05 | 2.00 | – |
| TTPWT (0.6) | **+0.14** | 3.06 | 2.95 |
| TTPWT (0.5) | −0.25 | **5.03** | **4.73** |

**Table 3.** Results of VGGnet compression on the CIFAR-10 validation dataset. The value in parentheses denotes the energy threshold.

| Method | ⋆ Acc | ‖ Param | ‖ FLOPS |
|---|---|---|---|
| DCP [35] | +0.31 | 1.93 | 2.00 |
| Random-DCP [35] | +0.03 | 1.93 | 2.00 |
| WM+ [35] | −0.10 | 1.93 | 2.00 |
| CP [14,35] | −0.32 | 1.93 | 2.00 |
| PFEC [19,35] | +0.15 | 2.78 | 1.52 |
| TTPWT (0.6) | +0.15 | **3.03** | **4.71** |

### 4.2 CIFAR-100

**ResNet-56:** ResNet-56 was the first network evaluated by us on the CIFAR-100 dataset. We compared the obtained results of our method with pruning approaches. As pruning competitors, we chose the following methods: SFP [12], FPGM [13], DMPP [20], CCPrune [4], FPC [3], and FPDC [36]. As can be seen in Table 4 our method achieved the largest FCR and PCR, and the lowest accuracy drop. It is interesting that TTPWT reduces FLOPS twice as much as SFP, FPGM and CCPrune.

**Table 4.** Results of ResNet-56 compression on the CIFAR-100 validation dataset. The value in parentheses denotes the energy threshold.

| Method | ⋆ Acc | ‖ Param | ‖ FLOPS |
|---|---|---|---|
| SFP [12,20] | −2.61 | 3.20 | 2.11 |
| FPGM [13,20] | −1.75 | 3.30 | 2.11 |
| CCPrune [4] | −0.63 | 1.69 | 2.94 |
| FPDC [36] | −1.43 | 1.93 | 1.99 |
| TTPWT (0.55) | **−0.50** | **3.93** | **4.19** |

**ResNet-110:** As the second network, we selected ResNet-110 that is one of the largest ResNet networks developed for CIFAR datasets. Similar to the previous results, ResNet-110 was compared with different pruning approaches [6,7,12,13,30,33,34,36]. As shown in Table 5, it is clear that TTPWT achieved the lowest drop in accuracy and the largest PCR and FCR over all the compared methods.

**Table 5.** Results of ResNet-110 compression on the CIFAR-100 validation dataset. The value in parentheses denotes the energy threshold.

| Method | ⋆ Acc | ‖ Param | ‖ FLOPS |
|---|---|---|---|
| OED [30] | −3.83 | 2.31 | 3.23 |
| FPDC [36] | −0.61 | 1.93 | 3.24 |
| PKPSMIO [33] | −0.14 | 3.40 | 3.24 |
| PKP [34] | −0.61 | 2.42 | 2.37 |
| TAS [7] | −1.90 | – | 2.11 |
| FPGM [7,13] | −1.59 | – | 2.10 |
| SFP [7,12] | −2.86 | – | 2.10 |
| LCCL [6,7] | −2.01 | – | 1.46 |
| TTPWT (0.55) | **−0.03** | **3.96** | **4.21** |

## 5 Conclusions

This study proposes a new approach to low-rank compression of CNNs. The proposed method is based the tensor train decomposition of a permuted weight tensor with automatic rank determination. The original convolution is approximated with a pipeline of four smaller convolutions, which allows us to significantly reduce a number of parameters and FLOPS at the cost of a low drop in accuracy. The results obtained on two datasets using four networks of different sizes confirm that our method outperforms the other neural network compression methods presented in this study. Further research is needed to investigate the compression of larger CNNs on the ImageNet dataset and to extend the current approach for higher order convolutional neural networks, including 3D CNNs.

## References


1. Alqahtani, A., Xie, X., Jones, M.W.: Literature review of deep network compression. In: Informatics, vol. 8, p. 77. Multidisciplinary Digital Publishing Institute (2021)
2. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition. Psychometrika **35**, 283–319 (1970)
3. Chen, Y., Wen, X., Zhang, Y., He, Q.: FPC: filter pruning via the contribution of output feature map for deep convolutional neural networks acceleration. Knowl.-Based Syst. **238**, 107876 (2022)
4. Chen, Y., Wen, X., Zhang, Y., Shi, W.: CCPrune: collaborative channel pruning for learning compact convolutional networks. Neurocomputing **451**, 35–45 (2021)
5. Denil, M., Shakibi, B., Dinh, L., Ranzato, M., De Freitas, N.: Predicting parameters in deep learning. In: Advances in Neural Information Processing Systems (NIPS), pp. 2148–2156 (2013)
6. Dong, X., Huang, J., Yang, Y., Yan, S.: More is less: a more complicated network with less inference complexity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5840–5848 (2017)
7. Dong, X., Yang, Y.: Network pruning via transformable architecture search. Adv. Neural Inf. Process. Syst. **32**, 760–771 (2019)
8. Garipov, T., Podoprikhin, D., Novikov, A., Vetrov, D.: Ultimate tensorization: compressing convolutional and fc layers alike. [Online] arXiv preprint arXiv:1611.03214 (2016)
9. Hameed, M.G.A., Tahaei, M.S., Mosleh, A., Nia, V.P.: Convolutional neural network compression through generalized Kronecker product decomposition. arXiv preprint arXiv:2109.14710 (2021)
10. Harshman, R.A.: PARAFAC2: mathematical and technical notes. UCLA Work. Papers Phonet. **22**, 30–44 (1972)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
12. He, Y., Kang, G., Dong, X., Fu, Y., Yang, Y.: Soft filter pruning for accelerating deep convolutional neural networks. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 2234–2240 (2018)
13. He, Y., Liu, P., Wang, Z., Hu, Z., Yang, Y.: Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4340–4349 (2019)
14. He, Y., Zhang, X., Sun, J.: Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision (CVPR), pp. 1389–1397 (2017)
15. Kim, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. In: International Conference on Learning Representations (ICLR) (2015)
16. Kirchhoffer, H., et al.: Overview of the neural network compression and representation (NNR) standard. IEEE Trans. Circ. Syst. Video Technol. 1 (2021). https://doi.org/10.1109/TCSVT.2021.3095970
17. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)

18. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned CP-decomposition. In: International Conference on Learning Representations (ICLR) (2014)
19. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: International Conference on Learning Representations (ICLR) (2016)
20. Li, J., Zhao, B., Liu, D.: DMPP: differentiable multi-pruner and predictor for neural network pruning. Neural Netw. **147**, 103–112 (2022)
21. Li, N., Pan, Y., Chen, Y., Ding, Z., Zhao, D., Xu, Z.: Heuristic rank selection with progressively searching tensor ring network. Complex Intell. Syst. 1–15 (2021)
22. Neill, J.O.: An overview of neural network compression. arXiv preprint arXiv:2006.03669 (2020)
23. Oseledets, I.V.: Tensor-train decomposition. SIAM J. Sci. Comput. **33**(5), 2295–2317 (2011)
24. Panagakis, Y., et al.: Tensor methods in computer vision and deep learning. Proc. IEEE **109**(5), 863–890 (2021)
25. Phan, A.H., Cichocki, A., Uschmajew, A., Tichavský, P., Luta, G., Mandic, D.P.: Tensor networks for latent variable analysis: novel algorithms for tensor train approximation. IEEE Trans. Neural Netw. Learn. Syst. **31**(11), 4622–4636 (2020)
26. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779–788 (2016)
27. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
28. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference Learning Representations (ICLR) (2015)
29. Tucker, L.R.: The extension of factor analysis to three-dimensional matrices. In: Gulliksen, H., Frederiksen, N. (eds.) Contributions to mathematical psychology, pp. 110–127. Holt, Rinehart and Winston, New York (1964)
30. Wang, Z., Lin, S., Xie, J., Lin, Y.: Pruning blocks for CNN compression and acceleration via online ensemble distillation. IEEE Access **7**, 175703–175716 (2019)
31. Wu, B., Wang, D., Zhao, G., Deng, L., Li, G.: Hybrid tensor decomposition in neural network compression. Neural Netw. **132**, 309–320 (2020)
32. Wu, J., Wang, Y., Wu, Z., Wang, Z., Veeraraghavan, A., Lin, Y.: Deep k-means: re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In: International Conference on Machine Learning (ICML), pp. 5363–5372. PMLR (2018)
33. Zhu, J., Pei, J.: Progressive Kernel pruning with saliency mapping of input-output channels. Neurocomputing **467**, 360–378 (2022)
34. Zhu, J., Zhao, Y., Pei, J.: Progressive kernel pruning based on the information mapping sparse index for CNN compression. IEEE Access **9**, 10974–10987 (2021)
35. Zhuang, Z., et al.: Discrimination-aware channel pruning for deep neural networks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31 (NeurIPS), pp. 881–892. Curran Associates, Inc. (2018)
36. Zuo, Y., Chen, B., Shi, T., Sun, M.: Filter pruning without damaging networks capacity. IEEE Access **8**, 90924–90930 (2020)

# Comparing Explanations from Glass-Box and Black-Box Machine-Learning Models

Michał Kuk[1(✉)], Szymon Bobek[2], and Grzegorz J. Nalepa[2]

[1] AGH University of Science and Technology, Krakow, Poland
m18.kuk@gmail.com

[2] Jagiellonian Human-Centered Artificial Intelligence Laboratory (JAHCAI) and Institute of Applied Computer Science, Jagiellonian University, Krakow, Poland

**Abstract.** Explainable Artificial Intelligence (XAI) aims at introducing transparency and intelligibility into the decision-making process of AI systems. In recent years, most efforts were made to build XAI algorithms that are able to explain black-box models. However, in many cases, including medical and industrial applications, the explanation of a decision may be worth equally or even more than the decision itself. This imposes a question about the quality of explanations. In this work, we aim at investigating how the explanations derived from black-box models combined with XAI algorithms differ from those obtained from inherently interpretable glass-box models. We also aim at answering the question whether there are justified cases to use less accurate glass-box models instead of complex black-box approaches. We perform our study on publicly available datasets.

**Keywords:** Explainable AI · Machine learning · Artificial intelligence · Data mining

## 1 Introduction

In recent years, the impact of machine learning on our daily life increased significantly, providing invaluable support to the decision making process in many domains. In insensitive areas, such as healthcare, industry, and law, where every decision may have serious consequences, the adoption of AI systems that cannot justify or explain their decisions is difficult and in many cases not desired. Such an observation stays in contradiction to the trend in the AI world, where the most progress is observed in the area of black-box models such as deep neural networks, random forests, etc. This duality led to the development of explainable AI methods, which allows introducing *transparency* and *intelligibility* to the decisions made by not interpretable black-box models. However, this transparency and intelligibility may serve different purposes, depending on the application area and the task that is to be solved with the AI method. In particular, we can define two main goals of XAI methods:


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 668–675, 2022.
https://doi.org/10.1007/978-3-031-08757-8_55

1) understand the mechanics of the ML model in order to debug the model, and possibly the dataset (i.e., what input drives the *model* to classify instance *A* as class *C*,

2) understand the phenomenon that is being modelled with AI methods and to build trust (i.e., what input makes the *instance A* to be classified as *C*).

While these goals might be indistinguishable at first glance, there is a fundamental difference in the assumptions that need to be fulfilled in both cases. In the first case, we assume that the model might be wrong, and we want to fix it, hence the information about the *model* is the most important. The model performance is an objective. In the second case, we assume that the model is correct and we want to use it to obtain information about the *class* or *instance* itself. The explanation itself is the main objective. In this paper, we focus on the second case. We provide a discussion on the performance of the glass-box models and black-box models explained with the use of XAI methods, in the situation when the objective is not to learn about the AI model, but about the phenomenon the model captures. We focused on rule-based explanations as one of the most understandable and widely applicable methods in industrial and medical cases. We performed a comparison of these two approaches on datasets from selected scikit-learn datasets and UCI Machine Learning Repository to see if a simple glass-box model can outperform complex XAI algorithms.

The rest of the paper is organised as follows: In Sect. 2 we describe a few papers which concern the explainable methods and we introduce our motivations. In Sect. 3 we present our approach to the performance comparison mentioned above. Next, Sect. 4 presents and discusses the results we obtained. Finally, in Sect. 5 we summary our work.

## 2 Related Works and Motivation

In this paper, we focused to evaluate what is the difference between the explanations obtained from black-box models combined with XAI algorithms and from those obtained based on interpretable glass-box models. To get such evaluation, firstly we verified the existing researches which concern glass-box and black-box models in the application of Explainable Artificial Intelligence.

In [10] the author makes a comparison of white and black box models. The author outlines that in some cases glass-box models could give as accurate results as the black box models. However, it strongly depends on the application domain and the data delivered. In the case of the black-box models, the author highlights that the experts do not need to understand the mathematical transformations behind them, but they proposed to deliver the output data in a similar form as input.

In [1] the authors pay attention to the fact that nowadays there is the need for XAI application due to commercial benefits, regulatory considerations, or in cases when the users have to effectively manage AI results. They outline that the black-box models do not disclose anything about internal design, structure, or implementation. On the other hand, the glass-box is completely exposed to the user.

In [6] the authors used glass-box and black-box models to predict the ambient black carbon concentration. They used several methods, whereas a neural network with LSTM layers gave the best results. However, they highlight that using black-box models like neural network or random forest complicates explanations.

In [9] the authors used the Anchor algorithm to obtain rules which could be explanations of each cluster of data. As a result, the proposed methodology is able to generate human-understandable rules which could be passed to the experts to support in the explainability process.

In [14] the authors used the black-box model to develop the structured attack (StrAttack), which is able to explore group sparsity in adversarial perturbation by sliding a mask through images. They demonstrate the developed method on datasets consisting of images. Furthermore, they outline that thanks to the sliding masks, they increase the interpretability of the model.

In [15] the authors also concentrate on image classification. They used a deep neural network to assign input to predefined classes. To interpret the models, they considered post-hoc interpretations. More specifically, they focus on the impact of the feature on the predicted result – they tried to uncover the casual relations between input and output.

In [11] the authors created an open-source Python package called InterpretML. They focused in most cases on the feature importance explanations, not on the human-readable rules.

I our work we focus on rule-based explanations, which according to our previous research [3] proves to be one of the most intelligible way of providing explanations to experts. Therefore, we mainly concentrate on the algorithms which can generate explanations which are represented as a logic implication (IF-THEN) by using a conjunction of relational statements. Such explanations can be executed with rule-based engines and verified according to selected metrics such as accuracy, precision, or recall. Having that, a research question arises: If the explanation is as much valuable as the model decisions, what kind of model should be applied to assure good interpretability along with high accuracy of explanations? Should it be 1) a glass-box model to directly generate explainable results, or 2) a complex black-box model and explain its results with Explainable Artificial Intelligence methods? To solve this research problem, we aim to apply XAI methods that are able to generate human-readable rules for complex black-box models and verify if these methods are needed to be applied in the case of simple tabular data or if we should make explanations directly with the use of the glass-box models.

## 3 Experimental Comparison

The scope of this work concentrates on the comparison of using simple glass-box models with complex black-box models explained with the XAI algorithm. In this work, we use a classification task as an exemplary problem to be solved. We considered the most popular classifiers available in the scikit-learn package [12].

For glass-box models, we selected: Decision Tree, Nearest Neighbors. For black-box models, we chose RBF SVM, Gauss Process, Random Forest, Neural Network, AdaBoost, Naive Bayes, QDA. We used a default models' settings as hyperparameter tuning was not the main goal of this paper. Our experiment considers two approaches for solving the classification task: 1) use directly explainable glass-box model, 2) use the black-box model, explain the predictions with XAI method, and solve the main problem based on the explanations obtained.

In the second approach, we took into consideration the XAI algorithms which generate explanations in the form of human-readable rules based on the trained classifier model. In this work, we considered three XAI methods: Anchor [13], Lux [4] and Lore [7]. Each of them generates instance-based explanations (rules), which subsequently were converted into XTT2 format [8]. To allow the results to be compared with glass-box models we used HEARTDROID inference engine [5] to predict the classification target (label) based on the obtained rules and data instances.

The schematic illustration of the considered approaches is presented in the Fig. 1.

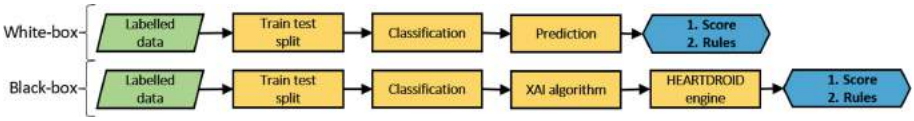



**Fig. 1.** Glass-box and black-box models approaches comparison.

To test the considered approaches and draw reliable conclusions, we chose data from different sources as an input to the experiments. In the work we used the following datasets: banknote and glass[1], cancer and iris[2], and titanic[3].

Each dataset has been divided into train and test datasets. For each considered classifier, we applied the same train instances to train the model and test instances to make predictions to maximize the reliability of the comparison results. For both considered approaches, we computed the accuracy, recall, and precision scores for each considered classifier to compare these two approaches.

## 4 Results and Discussion

In this section, we present the results of our experiments. Firstly, we compared the performance of all considered classifiers used directly to solve the classification problem. Then, we compared scores for the classification problem solved based on the rules generated for the black-box models with XAI methods (only the best results for each XAI method) vs glass-box model. Finally, we also considered the variance of scores that can be obtained for a selected XAI method depending on the black-box model explained. These results are presented in the following figures.

Figure 2 shows the results (scores) which were calculated for all classifiers used directly to predict the classification target (label). As can be seen, there are some datasets for which all classifiers perform with a high score (close to 1.0) such as iris and banknote that suggest that the problem to be solved in their case is relatively simple. For cancer and wine datasets, most classifiers also give high scores, but others (e.g. QDA, RBF, SVM) perform much worse. The most difficult problem to be solved is contained in the glass dataset for which the best score is lower than 0.7. Other difficult dataset is titanic for which scores are not greater than 0.8. Comparing different classifiers across

[1] See: https://archive.ics.uci.edu/ml.
[2] See: https://scikit-learn.org/stable/datasets.html.
[3] See: https://www.kaggle.com/datasets.

all considered datasets, it can be noticed that the glass-box Decision Tree model gives comparable results to black-box models. For the most difficult dataset (glass), it gives the highest precision, keeping recall and accuracy at a competitive level.

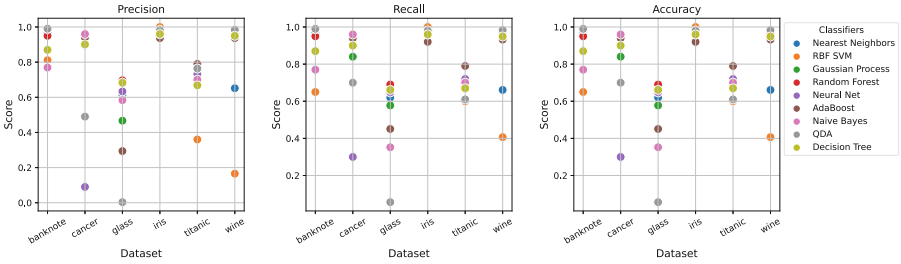



**Fig. 2.** Classifiers score comparison.

Figure 3 presents the comparison of the glass-box model results with the black-box models explanations obtained with XAI methods and executed with the use of HEARTDROID engine. In this figure, only the best scores for each dataset for each of the XAI methods are presented to compare the best possible results that can be obtained with a particular XAI method. In the case of using of XAI methods to generate rules, in Fig. 3 we can observe that the Anchor algorithm gives the best results in most datasets, but the Lux and Lore algorithms give noticeably worse results. In the case of the Lore algorithm, we noticed some of the bugs which resulted in scores equal to 0 which were marked on the charts. Only for the iris dataset (probably the easiest one), the Lux XAI method gives better results than the Anchor algorithm. Comparing the results from the exampled black-box models with Decision Tree, it can be noticed that for simple datasets like banknote, iris, or glass, the glass-box model gives better results. For slightly more difficult but still simple datasets (cancer and wine), slightly better results are obtained with the Anchor algorithm than the Decision Tree. However, the difference is not significant. In the case of more complex datasets like Titanic, glass-box model gives considerably worse results than the black-box model explained with the Anchor method.

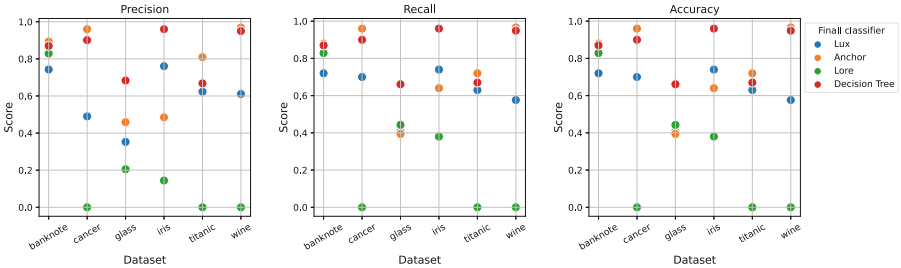



**Fig. 3.** XAI methods score comparison to glass-box model.

Figure 4 shows the variance of the performance obtained with the Anchor algorithm, as it gives the best results from the considered XAI methods, depending on the classification model used. We can observe the biggest score variance for cancer and wine datasets (relatively simple) for which the results strongly depend on the classifier model explained. The lowest variance is observed for glass and iris datasets, so the most and the least difficult datasets.

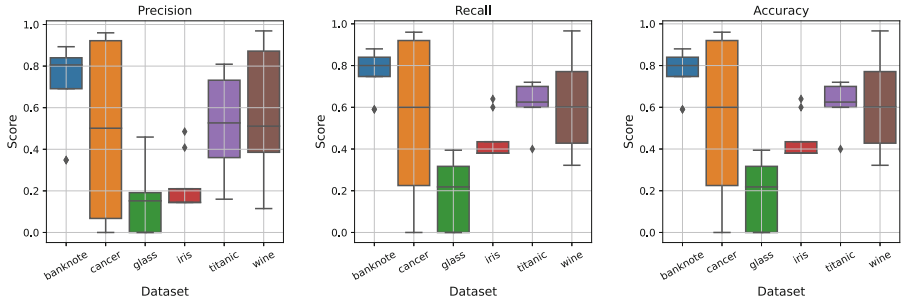



**Fig. 4.** XAI methods dependency on classifiers.

The obtained results allow us to compare how the explanations derived from black-box models combined with XAI algorithms differ from those obtained by interpretable glass-box models. Executed experiment proves that despite the fact that the black-box models are more complex and universal than the simple glass-box models, there is no need to apply them, especially for simple datasets. We found out that Decision Tree classifier gives competitive results and provides the model in an easily understandable format. However, in some examples, even in relatively simple datasets, it can be beneficial to apply more complex explanation methods (black-box model linked with XAI method) than simple glass-box models. Obtained results suggest also that when we need to consider more complex cases, better results can be obtained using the black-box models explained with XAI methods with human-readable rules. However, the final results strongly depend on the selected classifier. Hence, the properly chosen model which is treated as an input to the XAI method is important and has a significant impact on the final result.

## 5 Summary

In this paper, we made a glass-box and black-box classifiers comparison with the application in the explainable artificial intelligence area. The main goal of the work was to investigate if we should use the glass-box models to directly generate explanations or rather use a complex black-box model linked with XAI methods? We compared the classification scores for several classification methods and then we used the same trained models to obtain results with the use of XAI algorithm methods. We conducted our experiments based on the publicly available datasets. The results suggest

that especially in the case of tabular data, it is worth investing resources into research on inherently explainable models, instead of relying on a combination of black-box and XAI algorithms. However, taking into account more complex analyses that concern e.g. embeddings or latent semantic analysis uses of glass-box models could be insufficient and then, black-box models with XAI methods could be applied. However, the choice should be made carefully, with additional evaluation of XAI results to select the most suitable approach. In future work, we plan to extend this analysis, taking into account different types of data, including time series and images and combine it with explanation evaluation methods [2] to provide a comprehensive study on XAI and glass-box models applicability.

**Acknowledgements.** This paper is funded from the XPM (Explainable Predictive Maintenance) project funded by the National Science Center, Poland under CHIST-ERA programme Grant Agreement No. 857925 (NCN UMO-2020/02/Y/ST6/00070)

## References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018). https://doi.org/10.1109/ACCESS.2018.2870052
2. Bobek, S., Bałaga, P., Nalepa, G.J.: Towards model-agnostic ensemble explanations. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12745, pp. 39–51. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77970-2_4
3. Bobek, S., Kuk, M., Brzegowski, J., Brzychczy, E., Nalepa, G.J.: KNAC: an approach for enhancing cluster analysis with background knowledge and explanations. CoRR abs/2112.08759 (2021), https://arxiv.org/abs/2112.08759
4. Bobek, S., Nalepa, G.J.: Introducing uncertainty into explainable AI methods. In: Paszynski, M., Kranzlmüller, D., Krzhizhanovskaya, V.V., Dongarra, J.J., Sloot, P.M.A. (eds.) ICCS 2021. LNCS, vol. 12747, pp. 444–457. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77980-1_34
5. Bobek, S., Nalepa, G.J., Ślażyński, M.: HeaRTDroid - rule engine for mobile and context-aware expert systems. Expert Syst. **36**(1), e12328 (2019)
6. Fung, P.L., et al.: Evaluation of white-box versus black-box machine learning models in estimating ambient black carbon concentration. J. Aerosol Sci. **152**, 105694 (2021)
7. Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., Giannotti, F.: Local rule-based explanations of black box decision systems. ArXiv abs/1805.10820 (2018)
8. Kaczor, K., Nalepa, G.J.: Critical evaluation of the XTT2 rule representation through comparison with CLIPS. In: KESE@ECAI (2012)
9. Kuk, M., Bobek, S., Nalepa, G.J.: Explainable clustering with multidimensional bounding boxes, pp. 1–10 (2021). https://doi.org/10.1109/DSAA53316.2021.9564220
10. Loyola-González, O.: Black-box vs. white-box: understanding their advantages and weaknesses from a practical point of view. IEEE Access **7**, 154096–154113 (2019). https://doi.org/10.1109/ACCESS.2019.2949286
11. Nori, H., Jenkins, S., Koch, P., Caruana, R.: Interpretml: a unified framework for machine learning interpretability (2019)
12. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)

13. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: high-precision model-agnostic explanations. In: AAAI (2018)
14. Xu, K., et al.: Structured adversarial attack: towards general implementation and better interpretability (2019)
15. Zhang, X., Wang, N., Shen, H., Ji, S., Luo, X., Wang, T.: Interpretable deep learning under fire (2019)

# Virtual Reality Prototype of a Linear Accelerator Simulator for Oncological Radiotherapy Training

Vei S. Chan[1(✉)], Andrés Iglesias[2,3(✉)], Habibollah Haron[1], Pedro J. Prada[4], Samuel Ruiz[4], Akemi Gálvez[2,3], Lihua You[5], Faezah M. Salleh[6], and Farhan Mohamed[1,7]

[1] School of Computing, Faculty of Engineering, University of Technology Malaysia, 81310 Johor Bahru, Malaysia
vschan2@live.utm.my, habib@utm.my
[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, 39005 Santander, Spain
{iglesias,galveza}@unican.es
[3] Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan
[4] Hospital Universitario Marqués de Valdecilla, 39008 Santander, Spain
jpedraja@hvvaldecilla.es, samuel.ruiz@scsalud.es
[5] National Center for Computer Animation, Faculty of Media and Communication, Bournemouth University, Poole BH12 5BB, UK
lyou@bournemouth.ac.uk
[6] Department of Biosciences, Faculty of Science, University of Technology Malaysia, 81310 Johor Bahru, Malaysia
faezah@utm.my
[7] Media and Game Innovation Centre of Excellence, 81310 Johor Bahru, Malaysia
farhan@utm.my

**Abstract.** Learning to operate medical equipment is one of the essential skills for providing efficient treatment to patients. One of the current problems faced by many medical institutions is the lack or shortage of specialized infrastructure for medical practitioners to conduct hands-on training. Medical equipment is mostly used for patients, limiting training time drastically. Virtual simulation can help alleviate this problem by providing the virtual embodiment of the medical facility in an affordable manner. This paper reports the current results of an ongoing project aimed at providing virtual reality-based technical training on various medical equipment to radiophysicist trainees. In particular, we introduce a virtual reality (VR) prototype of a linear accelerator simulator for oncological radiotherapy training. The paper discusses the main challenges and features of the VR prototype, including the system design and implementation. A key factor for trainees' access and usability is the user interface, particularly tailored in our prototype to provide a powerful and versatile yet friendly user interaction.

Supported by European Union Horizon 2020 project PDE-GIR (Ref. MSCA-RISE-778035), Malaysia's Fundamental Research Grant Scheme (No. 5F395) and project of Ref. TIN2017-89275-R, of the MCIN/AEI/10.13039/501100011033/FEDER, Spain.


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 676–689, 2022.
https://doi.org/10.1007/978-3-031-08757-8_56

**Keywords:** Virtual reality · Linear accelerator · Medical simulator training · Oncological radiotherapy · Head-mounted display · User interface

## 1 Introduction

With the increasing number of cancer cases worldwide, there is urging need for highly-skilled specialists from the various disciplines involved in the prevention, diagnosis, monitoring, and treatment of cancer. An illustrative example is given by oncological radiotherapy (ORT), a field where different types of oncologists work in close cooperation with radiophysicists, a type of medical physicists with the technical ability to operate medical radiation equipment efficiently [1,4].

Radiophysicists typically work with linear accelerators (LINAC), sophisticated devices used to speed up charged subatomic particles or ions through a series of oscillating electric potentials. Oncological radiotherapy linear accelerators (ORTLINAC) are used for procedures such as intensity-modulated radiation therapy (IMRT), a level-3 high-precision technique that combines the use of computer tomography (CT) imaging and multileaf collimators. In IMRT, CT is used to get a volumetric representation of the tumor, while the collimators use a set of individual "leaves" equipped with independent linear in/out movement (orthogonal to the radiotherapy beam) to fit the treatment volume to the boundary shape to the tumor and vary the radiation signal intensity accordingly. In this way, IMRT is used to deliver precise radiation doses at targeted areas within the tumor, thus reducing the radiation impact on healthy organs and tissues.

Unfortunately, IMRT requires a lot of expertise and considerable experience for optimal performance. For instance, an individual radiation treatment planning (RTP) must be set up for each patient before the therapy sessions. The plan needs to consider several factors and parameters, such as the region of interest (ROI) where the radiation beam will focus, the most suitable beam type, the energy to be applied, the appointment schedule of therapy sessions and many others. The RTP is intended at maximizing the treatment effectiveness while minimizing the physical strain upon the patients. Once all details of the RTP are agreed, radiation sessions are set on place. During the radiotherapy session, the practitioners arrange the patients on a motorised table with six degrees of freedom. Then, they use remote controls to move the table and match the tumor location and orientation to the radiation beam's focal point of ORTLINAC. The whole procedure is highly-demanding in terms of concentration and skills to get the precise position. As a result, intensive practising is required for the medical trainees to master the positioning of the patients to the precise radiation beam's focal point by using the remote controls. However, the ORTLINAC therapy schedule is often full due to its high demand [1]. The medical trainees have scarce time to access the facility for practising [5]. Moreover, it is not affordable to allocate ORTLINAC rooms for training purposes owing to their high costs [1,4]. In this context, virtual reality (VR) emerges as a suitable technology to simulate the real medical environment in the virtual world, allowing the trainees to conduct hands-on practice and let them accustomed to the environment.

Previous studies explored the usage of VR in training and education in various domains [14,19,20]. These works show that VR technology contributes to psychomotor or technical skills development, knowledge transfer, and social skills. In the radiotherapy field, there is also research work using VR technology for training the novice to operate the medical equipment [24]. VR is also used to inform the patients about the therapy session and reduce their anxiety [15,23]. Nevertheless, realism is still an issue because most studies visualise the virtual operation in the ORTLINAC room. In contrast, the practitioners in real-life situations are also involved in operating the equipment remotely in another control room. Therefore, both the ORTLINAC room and the control room must be fully integrated and coordinated in the VR simulator for realistic training. In addition, the user interaction in the VR simulator must be as similar as possible to that in the real-world setting. These are the goals of the present contribution.

This paper introduces a VR prototype of a unified system comprised of the ORTLINAC room and the control room along with their interactions. The paper describes the main tasks of the system design and implementation, including the research workflow to determine the user requirements and to address the user interaction issues. The structure of this paper is as follows: Sect. 2 reports previous work regarding VR for medical science. Section 3 describes the workflow for the design and implementation of the VR prototype introduced in this paper. Then, Sect. 4 shows the main results of the implementation. Lastly, Sect. 5 discusses the conclusions and future work in the field.

## 2 Previous Work

### 2.1 Virtual Simulation for Medical Science

The presence of virtual simulation is soundly significant in many areas of the medical science, such as radiation therapy [5], radiography [13], and surgery [24]. The 3D visualisation of a patient's body and internal organs is helpful for medical practitioners to make a treatment plan and discuss it with their peers. The large-size wall display can also help them present and collaborate effectively. In radiotherapy, most institutions use the virtual environment for radiotherapy training (VERT) system for such purposes [21]. Besides, the 3D view and virtual simulation can also benefit the teaching and learning process [9].

### 2.2 Oncology Radiotherapy Training Issues

To provide efficient treatment to the patients, quality and effective medical education are of utmost importance. Training is also essential in solving the shortage of qualified staff operating the medical equipment for radiotherapy [6,21]. Here we discuss some issues found in conventional ORTLINAC radiotherapy training.

**Need to Learn Diverse Skills.** Radiotherapy workflow involves many medical knowledge and skills [10]. According to [2,4], the workflow includes: (a) CT scanning to obtain the imaging data of patient's anatomy; (b) segmentation

to extract the region of interest (organ, tumor); (c) treatment planning and evaluation; (d) quality assurance to avoid patient injury; (e) image-guided to place the patient on the ORTLINAC; and (f) perform the radiotherapy. Aside from the technical skills, the novice also needs to learn to communicate with patients and provide patient-care service [4]. These factors require the trainees to spend much time mastering these skills. However, the ORTLINAC equipment is often in high demand, giving the trainees sparse access for practising [1,5,13].

**Patient Care Issues.** The need to ensure patient safety and well-being during the ORT sessions is a big concern, as it may cause psychological pressure for the trainees [5,16]. Using phantoms might help minimize this issue [8]. However, the limited access to the ORTLINAC still affects the training progress.

**Limitations of the 2D Medium.** Conventional treatment planning and demonstration use 2D media to explain the concept, such as 2D imaging slice view and printed medium. Previous studies found it challenging to visualise and explain the spatial relationship between organs and anatomy and how the radiation beam affects these organs [9]. Therefore, 3D and immersive techniques can help effectively explain these spatial concepts to the trainees.

### 2.3 VR for Radiotherapy Education and Training

Several research works addressed the use of VR technology for RT training, including skin apposition application [3], medical imaging [13], breast cancer [15], prostate cancer [18], medical dosimetry [9,17], and brachytherapy [24].

**The VERT System.** Most of the VR training approaches make use of the VERT system, consisting of a wall-size display and an actual hand pendant controller to operate the virtual ORTLINAC. The studies show that VERT provides an optimal environment for hands-on practice [4,21]. The virtual simulation can offer a safe working environment where the trainees can practise by trial and error without the fear of injuring the actual patients [4,12]. The work in [5] reported that most of the trainees utilised VR simulation training significantly whenever it is available. This study is consistent with the high level of satisfaction and enjoyment among the trainees, as reported by [3,9,10,22]. These results showed that VR simulation could provide a conducive environment and a valuable opportunity for practice, which can help the trainees to master the skills effectively and in shorter time [9]. Besides, the large screen display of VERT helps the educators to demonstrate and explain therapy concepts in a classroom setting [17,21]. This display can help the trainees understand the spatial relationship between the radiation beam and the target organs [21] and allow more engagement and discussion of learned knowledge into the professional conversation [10]. These results provide evidence that VR-based training fosters the trainees' development and confidence in operating the radiotherapy equipment.

However, previous studies reported several limitations of the VERT system. First is the realism issue. Most users stated that the VERT lacks immersion and does not provide tactile feedback when the collision occurs [3,12]. A few

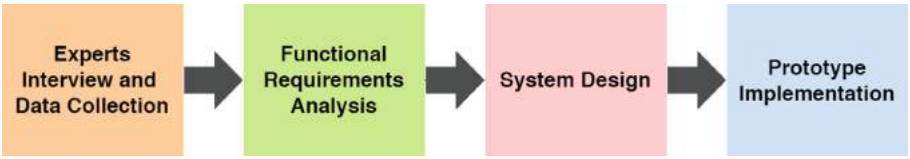



**Fig. 1.** Our design and implementation flowchart.

studies showed that the students obtained less performance and task accuracy in actual treatment planning after practice using virtual simulation units compared to conventional simulation practice by using the actual unit or the treatment planning software [12,17]. Some users also criticised the complex control of the VERT system as a limiting factor for their training and skill development [4].

Secondly, [10] revealed the lack of autonomous and self-directed learning in their blended learning framework. This issue is possibly caused by the large display unit where the student has less opportunity to conduct the practice by him/herself. Moreover, the COVID-19 pandemic also caused the cancellation of many on-site clinical practices [5], which further exacerbates the usual access limitation to the learning facility. Clearly, there is a need to explore an alternative immersive technology that can allow distance learning but without these issues.

**Head-Mounted Displays.** The advancement of VR technology allows the increased affordability of small-size equipment. The head-mounted display (HMD) is one such VR equipment that can solve the issues found in large 3D displays. Authors in [1] created an HMD VR application for ORTLINAC training, resulting in better learnability and effectiveness in training radiotherapy compared to VERT. The work in [24] utilised the room-scale VR headset HTC Vive for brachytherapy training to improve the trainees' technical skills. With the recent research trend in collaborative VR [11], HMDs can improve both autonomous and group learning environments.

Based on [10], medical practitioners and experts' involvement can help design the software and education curriculum to fulfil the real-life situation. Since the practitioners spend most of the time in the control room, there is a requirement to simulate the virtual embodiment in the control room where they have limited view and need to depend on the camera to operate the ORTLINAC. Accordingly, this paper emphasises the development of a virtual control room to train hand-eye coordination skills and spatial awareness in a limited viewing condition.

## 3 System Design and Implementation

This research work is part of an ongoing project aimed at providing VR-based technical training on various medical equipment to radiophysicist trainees. Although the full project is still a work in progress, we think that it has already reached significant results to justify publication. The work described in this paper concerns the development of a workable medical simulation training system to

simulate the real-life working condition of ORTLINAC based on user requirements. In this context, this section focuses on the design and implementation workflow plan for creating a VR prototype of the ORTLINAC simulator for ORT training. Figure 1 shows the main steps of the design and implementation flowchart. They are described in detail in next subsections.

### 3.1 On-site Medical Facility Visits and Meetings with Experts

The first step of the process involves visits to medical facilities and meetings with medical practitioners to elicit the functional requirements of the system. Some authors visited the medical facilities in the oncology department of Hospital Universitario Marqués de Valdecilla, Santander, Spain, where they were presented the daily operation in both the ORTLINAC therapy room and the remote control room, including the features of ORTLINAC, how to control the ORTLINAC and some standard procedures in radiotherapy. The authors collected photos, videos and other materials to analyse and design the VR training system.

### 3.2 Functional Requirements

After the visits to the medical facilities and meetings with experts, the authors analysed the collected materials (video transcripts, photos, printed materials, and others) to extract the functional requirements for the VR ORTLINAC training system. Figure 2 shows the use case diagram of the VR medical simulation training system.

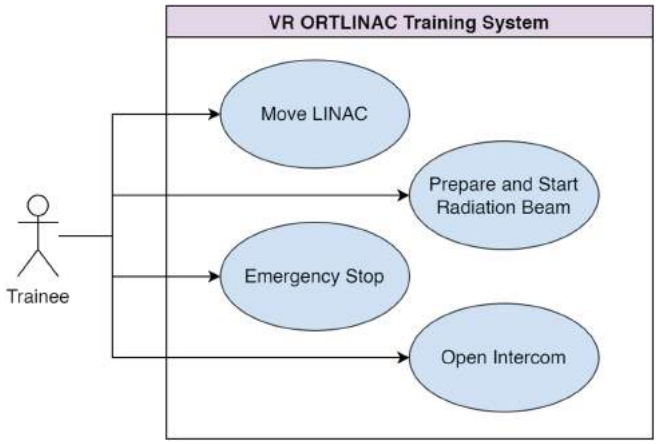



**Fig. 2.** The use case diagram of the VR ORTLINAC training system.

The trainee is the primary use case actor to use the VR ORTLINAC training system for practising. Table 1 shows the analysed functional requirements and their description.

**Table 1.** Functional requirements and their description.

| ID | Requirement | Description |
|---|---|---|
| RQ1 | Move LINAC | Trainee can move the position of table and gantry by pressing the button in the remote control panel |
| RQ2 | Prepare and Start Radiation Beam | Trainee can choose, prepare, start and stop the radiation beam |
| RQ3 | Emergency Stop | Trainee can perform an emergency stop to shut down the ORTLINAC immediately |
| RQ4 | Open Intercom | Trainee can open the intercom and give instructions to the patient to adjust his/her position remotely during the radiotherapy session |

### 3.3 System Design

As indicated above, there are two rooms for our VR-based training system: the ORTLINAC radiotherapy room and the control room. During the radiotherapy session, the practitioners in the radiotherapy room place and fasten the patient on the motorised table of ORTLINAC to stabilise and fix the patient' position. In some cases, an individually customized plastic mask is provided to the patient to wear during treatment. After this, any other person than the patient should leave the room to avoid the harmful effects of the radiation. The LINAC machine is operated from the control room, where the patient can be tracked through a window and/or one or several cameras. There is also an intercom for oral communication with the patient. The practitioners in the control room use different controls to guide the motion of the table and gantry and align the tumor's region of interest to the centre of the radiation beam's focal point. This external radiotherapy procedure is typically applied in several sessions distributed over days and weeks according to the patient's RTP.

In this paper, we will focus on the design and development of the virtual control room. Firstly, we designed and created the control room simulation according to the real-life situation. According to the experts' feedback, the practitioners spent most of their time in the control room. Therefore, this simulation can provide more exposure for the novices to the environment. To furnish the virtual scene, the authors utilised Blender and SketchUp software to create the 3D model of the furniture, electronic devices and medical equipment, such as the camera display of the ORTLINAC device room and control panels.

The user interface (UI) is also an essential element for interacting with the virtual world. For example, to operate the ORTLINAC using the control panel and view the camera display. We identified several design considerations, leading to different versions throughout the design process. The first design version relied on virtual buttons for user interaction, allowing the users to click on the buttons of the 3D model to perform different actions. However, this feature may cause navigation difficulties for the trainees because they may accidentally click on another nearby button. The alternative solution of increasing the size or changing

the buttons' orientation may reduce the realism and familiarity to the actual control panel. Therefore, this work proposes sign-posting and annotation above the 3D models to attract the users to click on the button in the VR world. Once clicked, it will open a larger user interface panel that displays the camera view and control panel layout for easy viewing and selection, respectively. In this way, the 3D medical equipment can be displayed in VR at its original real-life size scale, thus improving the realism of the system.

### 3.4 Implementation

The VR ORTLINAC training system was developed in Unity3D with the Oculus Integration package. This package provides various templates and prefabs to develop a VR application in Unity3D, including an avatar framework and customised configurations. This work also used the Oculus Quest as the VR head-mounted display (HMD) with two Oculus Touch controllers for user interaction. The reason to use Oculus Quest is that it is a standalone system, requires fewer set-up procedures, and is very ubiquitous to carry around. In addition, Blender and SketchUp were used to create the 3D models of the medical equipment, electronic devices, furniture and room. Blender supports texture mapping on the 3D models to improve their visual realism. Table 2 shows the hardware and software specifications used in this work along with their versions.

**Table 2.** Hardware and software specifications.

| Name | Category | Specification |
|---|---|---|
| Unity3D | Software -> Game Engine | Version: 2020.3.25f1 |
| Oculus Integration | Software -> Unity Asset | Version: 37.0 |
| Blender | Software -> 3D Modelling | Version: 3.0.0 |
| SketchUp | Software -> 3D Modelling | Version: Pro 2022 |
| Oculus Quest | Hardware -> HMD | Generation: 1 |
| | | Software version: 37.0 |

After creating the 3D models in Blender and SketchUp, they were imported in Unity3D to build the virtual control room scene according to the sketch and requirements. The Unity UI can implement the UI design and user interaction based on our design considerations. Furthermore, the VR system included the room-scale locomotion feature to let the users physically walk around the virtual room. Another VR feature is to show the 3D models of users' hands with a Touch controller and cast a laser pointer from the right-hand controller to allow the users to point and click on the virtual button. The inclusion of virtual hands can also improve the users' perception of presence in the VR world. Lastly, the VR training system was built as an Android application package (APK) file and deployed in the Oculus Quest. Additional visualization on smartphones and tablets has also been developed and is fully supported.

## 4 Results

This section presents the results of the VR ORTLINAC training prototype. There are three main components: the virtual remote control room, the ORTLINAC radiotherapy room, and the proposed UI and user interaction.

### 4.1 Remote Control Room

Figure 3 shows a scene comparison between the real (top) and virtual (bottom) environments. As the reader can see, the virtual simulation was created as similar as possible to the actual control room, including the furniture, electronic devices, and interior layout. Real-world textures were extracted from the photos and applied on the virtual surfaces to improve the overall realism of the scene.

**Fig. 3.** Comparison of actual (top) and virtual (bottom) control rooms.

Figure 4 shows the comparison of the 3D models of the camera display (left) and control panel (right). We also mapped the button icon and text annotation textures on the 3D models, based on their appearance in the actual equipment.

### 4.2 ORTLINAC Radiotherapy Room

For the ORTLINAC room, the current work focused on the 3D modelling of ORTLINAC equipment and the position of cameras. The authors edited and

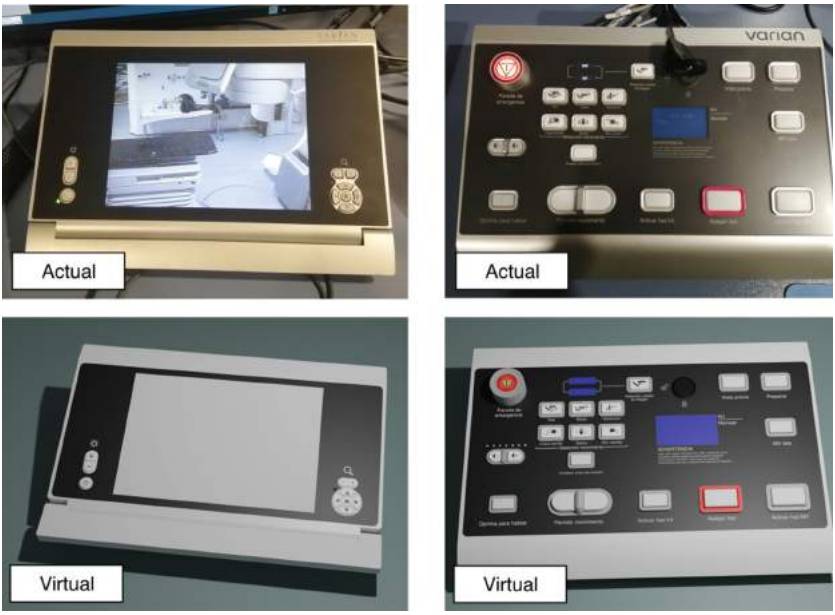



**Fig. 4.** The actual (top) and virtual (bottom) models of the camera display (left) and control panel (right).

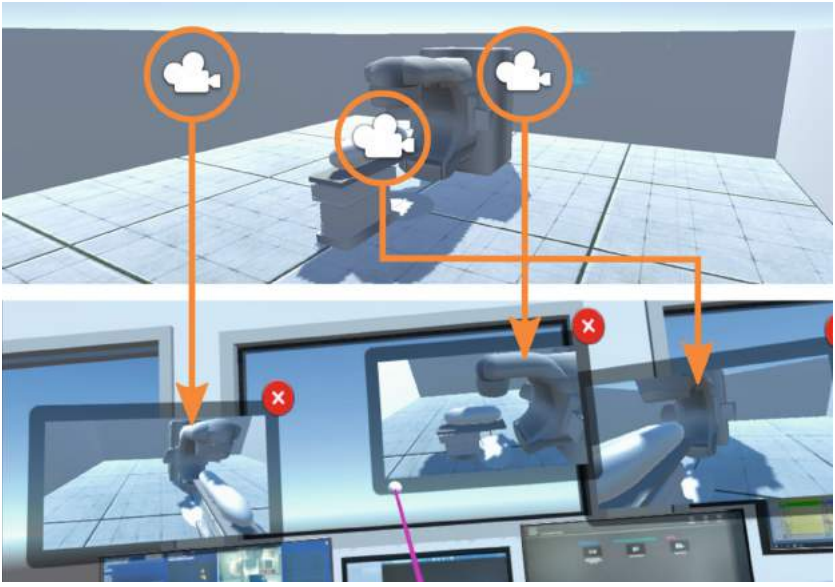


**Fig. 5.** The camera positioning in the ORTLINAC radiotherapy room (top) and their displays in the virtual control room (bottom).

modified the 3D LINAC model created by [7] in SketchUp and Blender, and shown in Fig. 5(top). Meanwhile, we set the positioning of cameras in the radiotherapy room in order to project the camera view into the camera displays in the virtual control room by using render texture mapping in Unity3D. Based on these features, this work simulates the remote control room successfully. Figure 5 shows the multiple camera positions and their displays in the control room.

### 4.3 User Interface and Interaction

As mentioned above, this work implements the sign-posting UI displayed above the 3D models to attract the users to point the controller's laser to the button, as shown in Fig. 6(left). After pressing the "A" button, this action opens a larger UI panel that displays the camera view or control panel layout for the users to interact, as shown in Fig. 6(right).

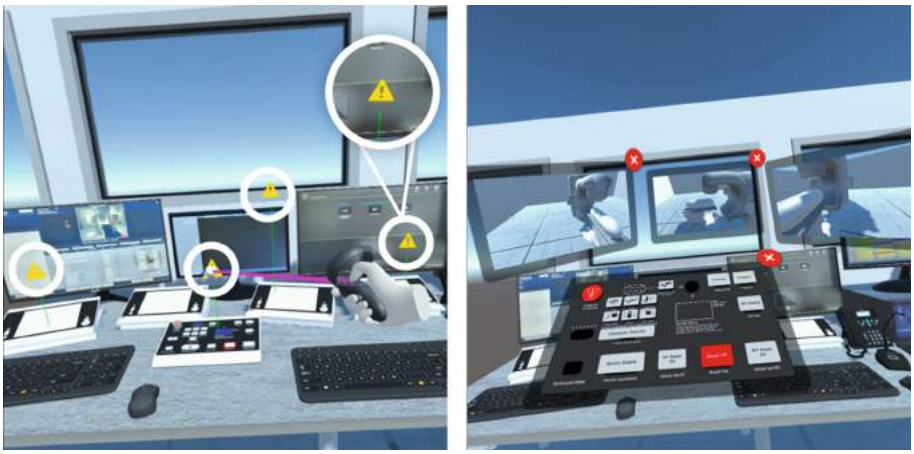


**Fig. 6.** The UIs presented in the VR ORTLINAC training system: sign-posting UIs (left); camera view and control panel layout UIs (right).

Besides, the users can point and click on the buttons in the control panel layout UI to control the movement of the ORTLINAC. Currently, this prototype only includes the functionalities to rotate the gantry and move the table linearly. The movement of ORTLINAC is reflected in the camera displays, as shown in Fig. 7. Hence, the users can observe the ORTLINAC position remotely. We also included the grab interaction by using the grip button in the left-hand controller when the virtual hand touches the UI. The users can grab any UI panel and place it in the desired location to customise their workspace, as shown in Fig. 8. The video demonstration can be found in this link: https://youtu.be/5YmY_0EsiLQ.

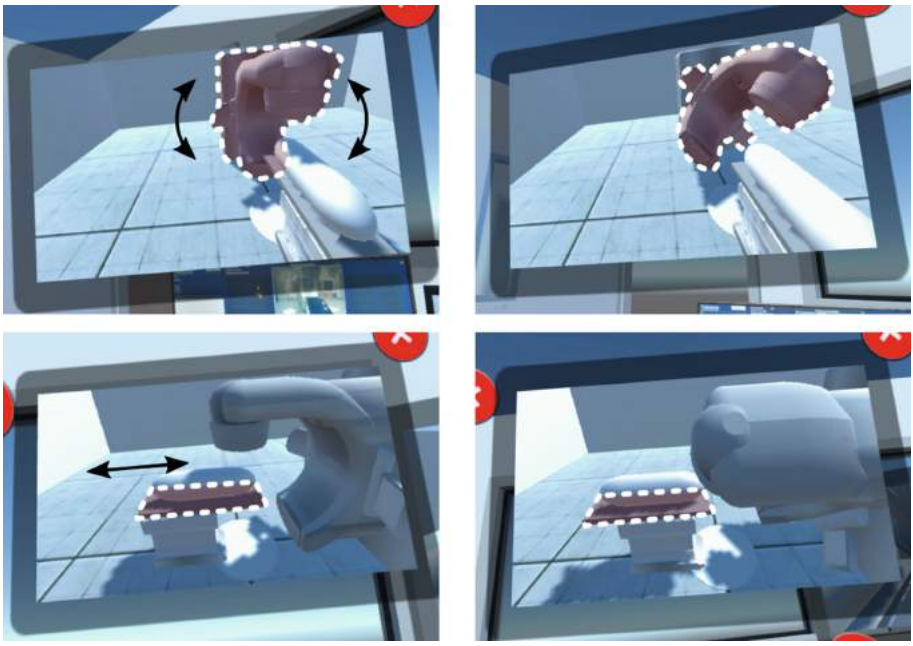


**Fig. 7.** The movement function for the rotation of the gantry and translation of the table before movement (left) and result after movement (right).

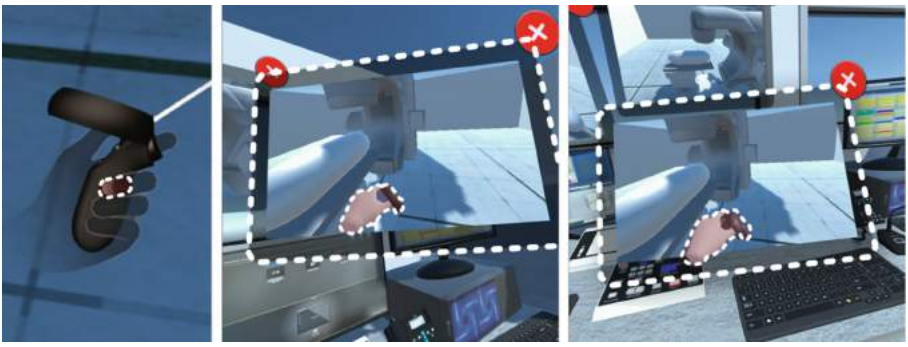


**Fig. 8.** The grab interaction to move the UI location: the grip button (left); the user activates his/her left hand on the UI panel (middle); using this feature to grab and drag the UI panel to other location (right).

## 5 Conclusions and Future Work

This paper presents a VR prototype of an ORTLINAC system for oncological radiotherapy training. Based on the experts' feedback, the ORTLINAC room and the control room are now integrated within a unified framework. Unity3D's render texture mapping functionalities are used to achieve the effect of the remote

camera display for effective synchronization between both rooms. Also, several UI design considerations are proposed for the VR world, including sign-posting and displaying a larger UI button layout for easier user interaction.

We will continue improving the prototype in terms of functionalities, graphics, and user experience (UX) for the next step. Furthermore, 3D reconstruction of imaging data that shows a patient's body with internal organs and tumor regions should be included in this system. This feature can challenge the trainees to practice operating the ORTLINAC correctly and avoid collision between the patient and the gantry. In addition, we wish to evaluate the efficiency and effectiveness of using this VR system for training medical practitioners and compare it with the traditional pedagogical approach.

**Acknowledgements.** This research is supported by the PDE-GIR project from the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No. 778035. The Fundamental Research Grant Scheme (FRGS/1/2020/ICT06/UTM/02/2) under the Malaysias Ministry of Higher Education (MOHE) with vote No. 5F395 supported C.V. Siang for the doctoral study. A. Iglesias and A. Galvez also thank the projects TIN2017-89275-R and PID2021-127073OB-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER "Una manera de hacer Europa".

## References

1. Bannister, H., et al.: LINACVR: VR simulation for radiation therapy education. In: 25th ACM Symposium on Virtual Reality Software and Technology, pp. 1–10 (2019)
2. Boejen, A., Grau, C.: Virtual reality in radiation therapy training. Surg. Oncol. **20**(3), 185–188 (2011)
3. Bridge, P., Appleyard, R.M., Ward, J.W., Philips, R., Beavis, A.W.: The development and evaluation of a virtual radiotherapy treatment machine using an immersive visualisation environment. Comput. Educ. **49**(2), 481–494 (2007)
4. Bridge, P., Crowe, S.B., Gibson, G., Ellemor, N.J., Hargrave, C., Carmichael, M.: A virtual radiation therapy workflow training simulation. Radiography **22**(1), e59–e63 (2016)
5. Bridge, P., et al.: International audit of simulation use in pre-registration medical radiation science training. Radiography **27**(4), 1172–1178 (2021)
6. Burger, H., et al.: Bridging the radiotherapy education gap in Africa: lessons learnt from the cape town access to care training programme over the past 5 Years (2015–2019). J. Cancer Educ. 1–7 (2021). https://doi.org/10.1007/s13187-021-02010-5
7. Carina, C.: Linear Accelerator (2021). https://3dwarehouse.sketchup.com/model/u9d9c8922-fce6-4f94-8483-0c0b7bad6c39/Linear-Accelerator?hl=en
8. Chamunyonga, C., Burbery, J., Caldwell, P., Rutledge, P., Fielding, A., Crowe, S.: Utilising the virtual environment for radiotherapy training system to support undergraduate teaching of IMRT, VMAT, DCAT treatment planning, and QA concepts. J. Med. Imaging Radiat. Sci. **49**(1), 31–38 (2018)
9. Cheung, E.Y.W., Law, M.Y.Y., Cheung, F.: The Role of Virtual Environment for Radiotherapy Training (VERT) in Medical Dosimetry Education. J. Cancer Educ. **36**(2), 271–277 (2019). https://doi.org/10.1007/s13187-019-01622-2

10. Czaplinski, I., Fielding, A.L.: Developing a contextualised blended learning framework to enhance medical physics student learning and engagement. Physica Med. **72**, 22–29 (2020)
11. Ens, B., Bach, B., Cordeil, M., Engelke, U.: Grand challenges in immersive analytics. In: Conference on Human Factors in Computing Systems - Proceedings (2021)
12. Flinton, D.: Competency based assessment using a virtual environment for radiotherapy. Procedia Comput. Sci. **25**, 399–401 (2013)
13. Gunn, T., Rowntree, P., Starkey, D., Nissen, L.: The use of virtual reality computed tomography simulation within a medical imaging and a radiation therapy undergraduate programme. J. Med. Radiat. Sci. **68**(1), 28–36 (2021)
14. Isham, M.I.M., Haron, H.N.H., bin Mohamed, F., Siang, C.V.: VR welding kit: accuracy comparison between smartphone VR and standalone VR using RMSE. In: 2021 IEEE International Conference on Computing (ICOCO), pp. 341–346. IEEE (2021)
15. Jimenez, Y.A., Cumming, S., Wang, W., Stuart, K., Thwaites, D.I., Lewis, S.J.: Patient education using virtual reality increases knowledge and positive experience for breast cancer patients undergoing radiation therapy. Support. Care Cancer **26**(8), 2879–2888 (2018). https://doi.org/10.1007/s00520-018-4114-4
16. Kane, P.: Simulation-based education: a narrative review of the use of VERT in radiation therapy education (2018)
17. Leong, A., Herst, P., Kane, P.: VERT, a virtual clinical environment, enhances understanding of radiation therapy planning concepts. J. Med. Radiat. Sci. **65**(2), 97–105 (2018)
18. Marquess, M., et al.: A pilot study to determine if the use of a virtual reality education module reduces anxiety and increases comprehension in patients receiving radiation therapy. J. Radiat. Oncol. **6**(3), 317–322 (2017). https://doi.org/10.1007/s13566-017-0298-3
19. Na, K.S., Mohamed, F., Isham, M.I.M., Siang, C.V., Tasir, Z., Abas, M.A.: Virtual reality application integrated with learning analytics for enhancing english pronunciation: A conceptual framework. In: 2020 IEEE Conference on e-Learning, e-Management and e-Services (IC3e), pp. 82–87. IEEE (2020)
20. Novotny, J., et al.: Developing virtual reality visualizations for unsteady flow analysis of dinosaur track formation using scientific sketching. IEEE Trans. Vis. Comput. Graph. **25**(5), 2145–2154 (2019)
21. Phillips, R., et al.: Virtual reality training for radiotherapy becomes a reality. Stud. Health Technol. Inform. **132**, 366–371 (2008)
22. Ryan, E., Poole, C.: Impact of virtual learning environment on students' satisfaction, engagement, recall, and retention. J. Med. Imaging Radiat. Sci. **50**(3), 408–415 (2019)
23. Wang, L.J., Casto, B., Luh, J.Y., Wang, S.J.: Virtual reality-based education for patients undergoing radiation therapy. J. Cancer Educ. (3), 1–7 (2020). https://doi.org/10.1007/s13187-020-01870-7
24. Zhou, Z., Jiang, S., Yang, Z., Zhou, L.: Personalized planning and training system for brachytherapy based on virtual reality. Virt. Real. **23**(4), 347–361 (2018). https://doi.org/10.1007/s10055-018-0350-7

# A Review of 3D Point Clouds Parameterization Methods

Zaiping Zhu[1(✉)], Andres Iglesias[2,3], Lihua You[1], and Jian Jun Zhang[1]

[1] The National Center for Computer Animation, Bournemouth University, Poole, UK
s5319266@bournemouth.ac.uk
[2] Department of Applied Mathematics and Computational Sciences, University of Cantabria, 39005 Cantabria, Spain
[3] Department of Information Science, Faculty of Sciences, Toho University, 2-2-1 Miyama, Funabashi 274-8510, Japan

**Abstract.** 3D point clouds parameterization is a very important research topic in the fields of computer graphics and computer vision, which has many applications such as texturing, remeshing and morphing, etc. Different from mesh parameterization, point clouds parameterization is a more challenging task in general as there is normally no connectivity information between points. Due to this challenge, the papers on point clouds parameterization are not as many as those on mesh parameterization. To the best of our knowledge, there are no review papers about point clouds parameterization. In this paper, we present a survey of existing methods for parameterizing 3D point clouds. We start by introducing the applications and importance of point clouds parameterization before explaining some relevant concepts. According to the organization of the point clouds, we first divide point cloud parameterization methods into two groups: organized and unorganized ones. Since various methods for unorganized point cloud parameterization have been proposed, we further divide the group of unorganized point cloud parameterization methods into some subgroups based on the technique used for parameterization. The main ideas and properties of each method are discussed aiming to provide an overview of various methods and help with the selection of different methods for various applications.

**Keywords:** Parameterization · Organized point clouds · Unorganized point clouds · Mesh reconstruction

## 1 Introduction

3D point clouds parameterization, also called point clouds mapping, is the process of mapping a 3D point cloud onto a suitable (usually simpler) domain. It has many applications such as object classification, texture mapping and surface reconstruction [1–3]. In many situations, it is computationally expensive or difficult to work with 3D point clouds directly. Therefore, projecting them onto a lower-dimensional space without distorting their shape is necessary. Compared to mesh parameterization, 3D point clouds parameterization is more challenging in general because there is no connectivity information


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 690–703, 2022.
https://doi.org/10.1007/978-3-031-08757-8_57

between points, which hinders the direct extension of well-established mesh parameterization algorithms to point cloud parameterization. There are some survey papers on mesh parameterization [4, 5]. However, to the best of our knowledge, there are no survey papers about point clouds parameterization. In this paper, we will review the methods of parameterizing point clouds. Notice there are also some works on 2D point clouds parameterization. Since 2D point clouds parameterization is different from 3D point clouds parameterization in most cases, this paper will only focus on the methods of 3D point clouds parameterization.

Some methods have been proposed to parameterize point clouds. In this paper, we roughly divide them into two main groups according to whether point clouds are organized or not. For each of the two groups, we further divide it into some subgroups based on the property of the mapping process and review each of the methods.

## 2 Some Concepts

In this section, some concepts related to point clouds will be introduced to help readers understand the problem of point clouds parameterization. Since mesh parameterization has been well investigated in existing work and some ideas of mesh parameterization can be adopted by or adapted to point cloud parameterization, we will also introduce some concepts about mesh parameterization in this section.

1) **Organized and unorganized point clouds:** Generally, point clouds can be divided into organized and unorganized ones. Organized and unorganized point clouds are also called structured and unstructured point clouds, respectively. The division is determined by the way of storing point cloud data. For organized point clouds, the data are stored in a structured manner, while unorganized point cloud data are stored arbitrarily. Specifically, an organized point cloud is similar to a 2-D matrix and its data are divided into rows and columns according to the spatial relationships between the points. Accordingly, the spatial layout represented by the *xyz*-coordinates of the points in a point cloud decides the memory layout of the organized point cloud. Contrary to organized point clouds, unorganized point clouds are just a collection of 3-D coordinates, each of which denotes a single point.
2) **Global and local parameterization:** To parameterize point clouds, some methods map the whole point set of an underlying structure to a parameterization domain. In contrast, some other methods split the problem into several subproblems, each of which is called a local parameterization. The choice between global and local parameterization has impacts on mapping processes and results. Globally parameterizing the whole point set can guarantee the reconstructed mesh is a perfect manifold, meaning there are no seams, which may exist if the point cloud is partitioned and locally parameterized. However, processing the whole point cloud at the same time may be computationally expensive, especially for large structures.
3) **Topological shapes:** Topological shapes can be grouped based on the number of holes they own. Shapes with no holes such as spheres and bowls are treated as genus-0 shapes. Similarly, genus-1, genus-2 and genus-3 shapes have one, two and three holes in them, respectively, and so on.

4) **Bijective function:** also called bijection, invertible function, or one-to-one correspondence, pairs each element in one set exactly to one element in the other set, and vice versa.
5) **Isometric, conformal, and equiareal mappings:** Suppose $f$ is a bijective function between a mesh $S$ or a point cloud and a mapping domain $S^*$, then $f$ is isometric (length preserving) if the length of any arcs on $S$ is preserved on $S^*$; $f$ is conformal (angle preserving) if the angle of intersection of every pair of intersecting arcs on $S$ is preserved on $S^*$; $f$ is equiareal (area preserving) if the area of an area element on $S$ is preserved on $S^*$. Isometric mappings are equiareal and conformal. Any mappings that are equiareal and conformal are isometric mapping.

## 3 Parameterization Methods of Organized Point Clouds

To parameterize an organized point cloud, many methods iteratively obtain a topologically identical 2D triangulation from the underlying 3D triangulation of the point cloud, and the 2D triangulation determines the parameter values of the vertices in the domain plane. Depending on the ways of transforming from 3D to 2D, there are several methods, including Harmonic parameterization [6], Floater's barycentric mappings [7] and the most Isometric parameterization [3]. For Harmonic parameterization in [6], the arc length is regarded as the parameter value of a spline curve, which is used to minimize the integral of the squared curvature with respect to the arc length for fairing the spline curve. With regard to barycentric mappings in [7], a shape-preserving parameterization method is applied for smooth surface fitting; the parameterization that is equivalent to a planar triangulation can be obtained by solving a linear system based on the convex combination. In [3], Hormann and Greiner propose a method to parameterize triangulated point clouds globally, the way of parameterizing inner point set is the same as that of parameterizing boundary point set. However, they ignore the problem of parameterizing triangulated point clouds with holes.

Energy function has also been defined to minimize the metric distortion in the transformation process from 3D to 2D. The methods described in [7, 8] follow the shared approach, which firstly parameterizes the boundary points, and then minimizes the following edge-based energy function for the parameterization of inner points [3]:

$$E = \frac{1}{2} \sum c_{ij} ||P_i - P_j||^2 \tag{1}$$

where $c_{ij}$ is the edge coefficient that can be chosen in various ways, $P_i$ and $P_j$ are two points at the same edge.

In order to reconstruct a tensor product B-spline surface from scattered 3D data with specified topology, choosing a suitable way to parameterize the points is crucial in the reconstruction process. The method adopted by Greiner and Hormann in [8] is called the spring model. With this method, the edge of the 3D triangulation is replaced by a spring. Then the boundary points are mapped first onto a plane and stay unchanged. Next, the inner points are mapped onto this plane by minimizing the spring energy. The procedure is repeated to improve the parameters until certain conditions are satisfied.

The above methods are mainly applicable to structured point clouds. They are not efficient when the number of points increases, and are likely to fail when holes and concave sections exist in the point clouds.

## 4 Parameterization Methods of Unorganized Point Clouds

In comparison with the parameterization of organized point clouds, many more methods have been proposed to parameterize unorganized point clouds. Table 1 lists these methods and gives the information about the category, parameter domain, local or global parameterization, topology, applications and publication year.

**Table 1.** Methods to parameterize unorganized point clouds.

| Methods | Category | Parameter domain | Local/global parameterization | Topology | Applications | Year |
|---|---|---|---|---|---|---|
| "Simplicial" surface [10] | Base surfaces-based methods | Base surfaces | / | Arbitrary topology | Surface reconstruction | 1992 |
| Manually define [9] | | | Global | / | Least square fitting of B-spline curves and surfaces | 1995 |
| Minimizing quadratic function [11] | | | / | / | B-spline curves and surfaces approximation | 2002 |
| Recursive DBS [12] | | | Global/local | Disk | Efficient parameterization | 2005 |
| Recursive subdivision technique [13] | | | Global/local | Disk (With hole is ok) | Parameterizing point clouds | 2007 |
| Floater meshless parameterization [14–17] | Meshless parameterization | Plane | Global | Disk | Surface reconstruction | 2000 |
| Meshless parameterization for spherical topology [18] | | Planes | Local | Genus-0 | Surface reconstruction | 2002 |
| As-rigid-as-possible meshless parameterization [19] | | Plane | Global | Disk | Denoising and parameterizing point clouds, mesh reconstruction | 2010 |
| Meshless quadrangulation by global parameterization [20] | | Plane | Global | Arbitrary genus | Meshless quadrangulation | 2011 |
| Spherical embedding [23] | Spherical mapping | Sphere | Global | Genus-0 | Mesh reconstruction | 2004 |
| 3D point clouds parameterization algorithm [22] | | Sphere | Global | Relatively simple models | Parameterizing point clouds | 2008 |
| Spherical conformal parameterization [21] | | Sphere | Global | Genus-0 | Mesh reconstruction | 2016 |

(*continued*)

**Table 1.** (*continued*)

| Methods | Category | Parameter domain | Local/global parameterization | Topology | Applications | Year |
|---|---|---|---|---|---|---|
| Discrete one-forms [24] | Adapt from mesh parameterization | Planes | Local | Genus-1 | Mesh reconstruction | 2006 |
| Periodic global parameterization [25] | | Plane | Global | Arbitrary genus | Direct quad-dominant meshing of point cloud | 2011 |
| PDE & SOM [26] | Neural networks-based methods | Adaptive base surface | Global | Complex sculptured surfaces | Surface reconstruction | 2001 |
| Adaptive sequential learning RBFnetworks [27] | | / | Global | Freeform | Point-cloud surface parameterization | 2013 |
| Residual neural network [28] | | / | Local | Fixed degree curve | Polynomial curve fitting | 2021 |
| A new parameterization method [29] | Other | / | / | / | NURBS surface interpolation | 2000 |
| Pointshop 3D [31] | | / | / | / | Point-based surface editing | 2002 |
| Free-boundary conformal parameterization [30] | | / | Global/local | / | Parameterizing point clouds for meshing | 2022 |

According to the property of the mapping process, we divide the parameterization methods of unorganized point clouds into base surfaces-based methods, meshless parameterization, spherical mapping, methods adapted from mesh parameterization, neural networks-based methods, and other methods.

### 4.1 Base Surfaces-Based Methods

For parameterization of unorganized point clouds, base surfaces, which approximate the underlying structure of point clouds, have been widely applied to parameterize point clouds. Base surfaces can be a plane, a Coons patch, or a cylinder [2]. The parameter values of each point in a point cloud can be obtained by projecting the point cloud onto a base surface. The projection direction can either be perpendicular to the surface or based on a determined projection vector. According to [9], a base surface should own the following properties:

a) Unique local mapping: The uniqueness implies that any two different points on the underlying surface should be mapped onto two different locations on the mapping domain.

b) Smoothness and closeness of base surface: This indicates that a base surface should be as smooth and simple as possible, while still approximating the underlying surface as much as possible. The balance between these properties should be carefully considered.
c) Parameterization of base surface: This implies that how we parameterize a base surface has a direct effect on the parameterization of the fitting surface. We can choose a more suitable way to parameterize a base surface by referring to the underlying structure of the fitting surface.

To get access to such base surfaces, some approaches have been proposed. For example, Hoppe et al. [10] propose a method to produce so-called "simplicial" surfaces. They first define a function to estimate the signed geometric distance to the underlying surface of the point clouds, then a contouring algorithm is applied to approximate the underlying surface by a "simplicial" surface. Their method is capable of reconstructing a surface with or without boundary from an unorganized point set. However, there is no formal guarantee that the reconstructed result is correct and the space required to store the reconstruction is relatively large. In [9], users can also manually define some section curves and four boundary curves to get a base surface of a point cloud, as some characteristic curves approximating the underlying structure of the point cloud are sufficient in defining a base surface. But it is also necessary to take advantage of the interior characteristic curves when the geometry is complex, even though just four corner points can be used to create a base surface in some cases. A base surface can also be obtained by iteratively minimizing a quadratic objective function [11]. With this method, a linear system of equations is solved in each step. To parameterize unstructured point clouds, Dynamic Base Surfaces (DBS) are also proposed by Azariadis [2]. As its name implies, a BDS is gradually improved regarding its approximation to the underlying structure of a point cloud, and the parameter value of each point in the point cloud is obtained by projecting it orthogonally to the DBS. Different from existing methods, no restrictions are required for the density and the homogeneity of point clouds. The limitation of this method is that it is only applicable to the point clouds where a closed boundary consisting of four curves exists. Azariadis and Sapidis [12] present a method to parameterize a point cloud globally and/or locally using recursive dynamic base surfaces. Their method can handle arbitrary point clouds of disk topology. Figure 1 shows the local parameterization of one subset of several point clouds using this method. The same authors [13] extend the DBS concept and use a recursive subdivision method to improve the accuracy of point clouds parameterization, especially for some small regions of the point clouds, where the approximation error by the DBS is not acceptable. They divide such regions into smaller parts and the points on these parts are approximated by $c^0$ composite surface based on recursive DBS subdivision to increase the approximation error, then to make the point clouds parameterization more accurately.

### 4.2 Meshless Parameterization

Meshless parameterization, first proposed by Floater and Reimers in [14], is also a widely used method to parameterize and mesh point clouds. As shown in Fig. 2, the main idea of meshless parameterization is to map the points in a point cloud onto a plane, where the

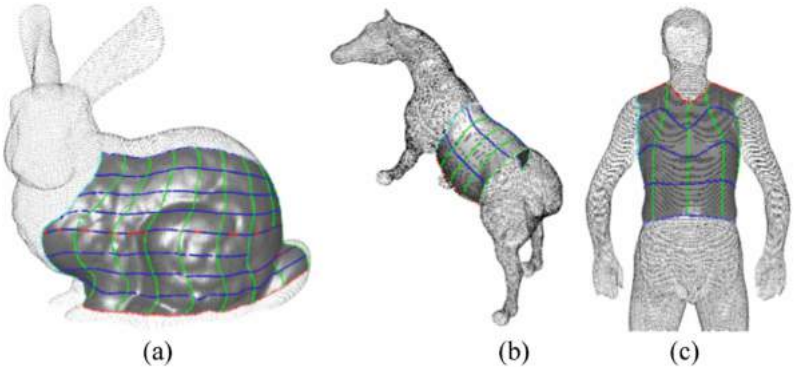



**Fig. 1.** Local parameterization of: (a) "bunny" point cloud, (b) "horse" point cloud, and (c) "human" point cloud [12].

mapping points are triangulated using an appropriate triangulation method, and then the original point cloud is meshed with the same triangulation edge structure as the mapping points. In order to make sure the reconstructed mesh has high quality, the mapping points should preserve the local structure of the original point clouds as much as possible. Therefore, the shape distortion ought to be minimized in the parameterization process. This is formulated as the problem of solving a sparse linear system [14, 15]. Since the mapping does not depend on the topological structure of point clouds, this method is called meshless parameterization. After the projection, the corresponding triangulation of the point clouds before mapping can be obtained by triangulating the projecting points in the planar parameter domain. This method has some limitations. First of all, solving a large linear system using their method is not efficient. Secondly, the reconstructed 3D triangles may distort and intersect each other due to the artificial convex boundary, which is also a problem when there are concave holes and the convex combination is not well defined along the concave parts of the hole boundary. To improve the efficiency of solving the linear system more efficiently, Volodine et al. [16] show that it can be done by an appropriate reordering of the matrix, which enables the linear system to be solved efficiently by deploying a direct sparse solver. To overcome the second problem, the same authors [17] extend the method to avoid distortion in the vicinity of concave boundaries by inserting virtual points to the concave neighbourhood, which can make sure the convex combination mapping is always defined. The methods described in [17] are only applicable to disk shape point clouds. To make the method presented in [17] more general, Hormann and Reimers [18] present an algorithm that can handle genus-0 topology as well by dividing the problem of triangulating point clouds into subproblems, each of which can be solved using the method in [17]. To improve the reconstructed result, Zhang et al. [19] apply an "as-rigid-as-possible" meshless parameterization method to parameterize a disk topology point cloud onto a plane while denoising the point cloud. Since their method can preserve local distances in the point cloud, a more regular 3D mesh can be obtained. Li et al. [20] present a meshless global parameterization method to parameterize point clouds and use the obtained parameterization to mesh the point clouds automatically.

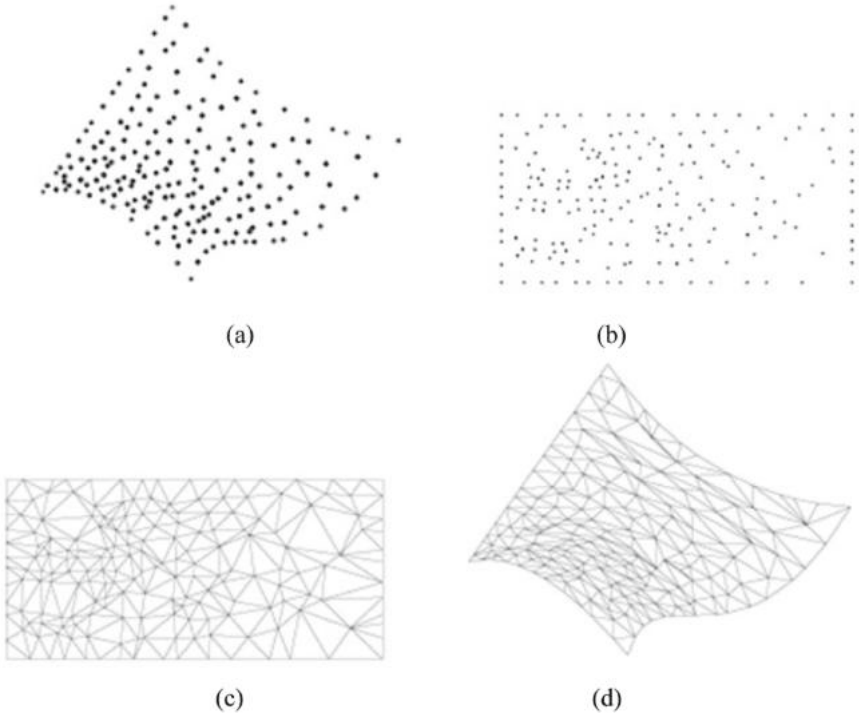



**Fig. 2.** (a) Point set. (b) meshless parameterization. (c) Delaunay triangulation of the mapping points. (d) surface triangulation [15].

### 4.3 Spherical Mapping

When the underlying structure of the point clouds is closed, which means there are no boundaries of the structure, "spherical mapping" is normally applied to parameterize the point clouds. The reason why "spherical mapping" is applied under such conditions can be partly explained by the uniform theorem [21], which states that every genus-0 closed surface is conformally equivalent to $S^2$. Thus, mapping from a genus-0 surface to the unit sphere is natural. The same idea is also applied to genus-0 point clouds. One such example is shown in Fig. 3. The problem of forming a spherical mapping given a point cloud model P can be formulated as [22]:

$$s = o + r_s \frac{p - o}{||p - o||} \tag{2}$$

where $s$ are the spherical mapping points, $p$ is the original point set, $o$ is the centre of the original point set and $r_s$ represents the largest distance between the original point set and the centre with the radius of the sphere.

Spherical parameterization is mostly used to mesh point clouds. For example, Zwicker and Gotsman [23] present a method to reconstruct a manifold genus-0 mesh from a 3D point cloud by using spherical embedding of a k-nearest neighbourhood graph

of a point cloud. Then the embedded points are triangulated and the reconstructed mesh structure is used to mesh the original point cloud. The main advantage of this method is that it can guarantee a closed manifold genus-0 mesh, even the input point cloud is noisy. However, its drawbacks are that pre-processing and post-processing may be required for the input point clouds and the output mesh, respectively. In [21], Choi et al. extend a state-of-the-art spherical conformal parameterization algorithm used to parameterize genus-0 meshes to the case of point clouds, which are achieved by using an improved approximation of the Laplace-Beltrami operator on the point cloud and a scheme named the north-south reiteration for the meshing of point clouds. The reason why they apply the method of spherical conformal parameterization method to reconstruct meshes from point clouds is mainly that directly triangulating a point cloud is challenging, especially for complex geometry, which can be achieved more easily with the aid of spherical conformal parameterization. Specifically, instead of directly triangulating a point cloud, the points on the unit sphere after mapping are triangulated using the spherical Delaunay triangulation algorithm. Then triangulation of the original point cloud can be obtained from the triangulation on the spherical point cloud as these two point clouds have a one-to-one correspondence.

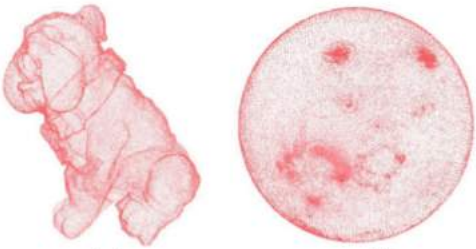



**Fig. 3.** (a) A bulldog point cloud. (b) the spherical conformal parameterization of the bulldog point cloud [21].

### 4.4 Methods Adapted from Mesh Parameterization

There are also some methods that are adapted from parameterizing meshes to parameterizing point clouds. For example, Tewari et al. modify the harmonic one-form method used in parameterizing manifold meshes to parameterize genus-1 point clouds that are sampled from such meshes [24]. They locally parameterize the subsets of a point cloud and the way they parameterize the point cloud can guarantee the consistency between the pieces. Even though the reconstructed results using their method are not much better than other reconstruction techniques, their method presents some new tools to the surface reconstruction problem and is very simple to implement. Li et al. [25] present a new method to reconstruct quad-dominant mesh from unorganized point clouds using the adapted periodic global parameterization method, which is modified from the periodic global parameterization method that is used to parameterize a triangle mesh. The local

Delaunay triangulation is used to design the parameterization of the point cloud. Their method can be used to deal with noisy point clouds without global connectivity. But it suffers from close-by structures because topological errors may be raised from the local Delaunay triangulation method by connecting two nearby surfaces.

### 4.5 Neural Networks-Based Methods

With the rapid development of neural network techniques, they have been applied to three main tasks of point cloud processing, i.e., 3D shape classification, 3D object detection and tracking, and 3D point cloud segmentation [26]. Besides their applications in the three main tasks, some researchers have investigated neural network-based point cloud parameterization. For example, Barhak and Fischer [27] adopt a self-organizing map (SOM) for the parameterization of small sets of clean points with low-frequency spatial variations, which can be used to reconstruct smooth surfaces. There are mainly two steps in the parameterization process: In the first step, Partial Differential Equation (PDE) and SOM are applied where the former technique can yield a parametric grid without self-intersection and the latter one makes sure all the sampled points have an impact on the grid, which guarantees the uniformity and smoothness of the reconstructed surface. In the second step, an adaptively modified 3D base surface is created for point clouds parameterization. Meng et al. [28] proposed a method to parameterize larger, noisy and unoriented point clouds by using adaptive sequential learning RBFnetworks. The network adopts a dynamic structure by adaptive learning and the neurons are adjustable regarding their locations, widths and weights, thus making it more powerful compared to other methods that apply RBFs at determined locations and scales. What is more, multi-level parameterization and multiple level-of-details (LODs) can be achieved in two ways. When multiple LODs meshes are required, parameterizing the point clouds with the best resolution and the points and surfaces can be computed at degrading sampling level to get the required LODs. In the second case where only one downgraded LOD is required, downgraded parameterization can be applied to obtain the result. Scholz and Juttler [29] apply residual deep neural networks to parameterize point clouds for polynomial curve fitting. Since the network approximates the function that assigns a suitable parameter value to a sequence of data points, optimal curve reconstruction from point clouds can be obtained. However, their method is only applicable to a small number of sample points and the proposed neural networks do not consider discrete surface point data. Figure 4 shows the layout of their proposed residual neural network.

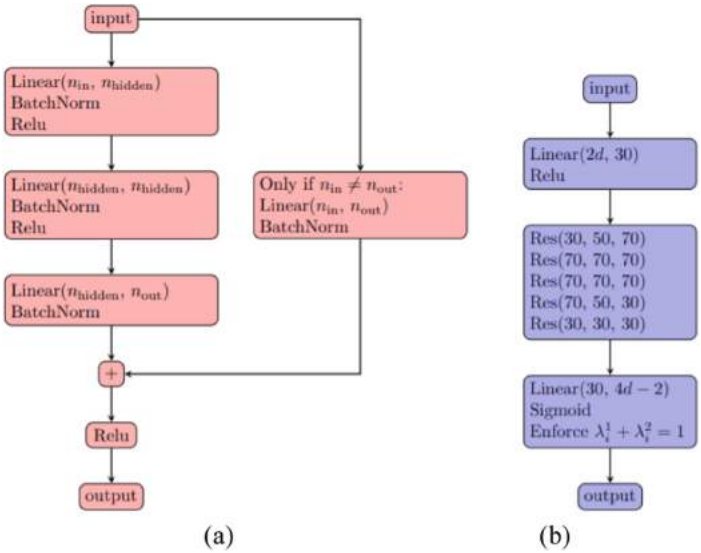



**Fig. 4.** (a) The layout of a building block. (b) the layout of the whole residual neural network [29].

### 4.6 Other Methods

Some other methods cannot be easily grouped. Therefore, we refer to them as other methods in this subsection and review them below.

As Ma and Kruth discuss in [9], three methods are usually adopted to parameterize digitized points for performing least squares fitting of B-spline curves and surfaces. These three methods are uniform parameters, cumulative chord length parameters and centripetal parameterization parameters. Since all these methods assume that the points are scattered in a special pattern, like chain points for curves and grid points for surfaces, these methods are very likely to fail when the points are irregularly spaced. To address this issue, Ma and Kruth [9] propose a simple technique, which parameterizes the irregularly spaced points by projecting them onto a base surface and obtaining their parameters from the parameters of the projected points. Jung and Kim [30] propose a new method to parameterize data points for NURBS surface interpolation, which is more powerful than the existing point clouds parameterization methods. With this method, the parameter value at the maximum of each rational B-spline basis function is treated as the parameter value of the corresponding data point. The empirical results show that their method outperforms the other methods as aforementioned in [10] regarding interpolation surfaces. In addition, many works consider mapping them onto a simple domain with a fixed boundary shape such as a sphere, a circle or a rectangle. However, some undesirable distortion may occur during the parameterization process due to the fixed boundary shape. To overcome such a problem, Choi et al. [31] develop a free-boundary conformal parameterization technique to parameterize disk-shape point clouds, which leads to high quality of the reconstructed mesh. By free boundary, it means that the positions of only two boundary points are fixed, and the left boundary points are parameterized

to a suitable location automatically based on the structure of the original point clouds. To make the parameterization of point clouds more flexible, Zwicker et al. [32] present a system in which interactively parameterizing point clouds can be done. During the mapping process, an objective function is applied to minimize distortions automatically. Furthermore, the user can adjust the mapping intuitively at the same time.

## 5 Conclusion

In this paper, we have reviewed various methods used to parameterize 3D point clouds. These methods are grouped into organized point parameterization and unorganized point cloud parameterization ones and unorganized point cloud parameterization methods are further divided into some subgroups according to the property of the point clouds and the mapping technique. We discussed each of these methods.

It should be pointed out that there is no "best" parameterization method applicable to all point clouds, as one method may succeed in parameterizing some point clouds but fail in parameterizing other point clouds. Therefore, for a given point cloud, it is necessary to choose a suitable method to parameterize the 3D point cloud according to the desirable properties of low distortion and high computing efficiency in parameterizing the point cloud.

**Acknowledgements.** This research is supported by the PDE-GIR project which has received funding from the European Union Horizon 2020 research and innovation programme under the Marie Skodowska-Curie grant agreement No 778035. Zaiping Zhu is also sponsored by China Scholarship Council.

## References

1. Meng, T.W., Choi, G.P.-T., Lui, L.M.: Tempo: feature-endowed teichmuller extremal mappings of point clouds. SIAM J. Imag. Sci. **9**, 1922–1962 (2016)
2. Azariadis, P.N.: Parameterization of clouds of unorganized points using dynamic base surfaces. Comput. Aided Des. **36**, 607–623 (2004)
3. Hormann, K., Greiner, G.: MIPS: an efficient global parametrization method. Curve and Surface Design: Saint-Malo, pp. 153–162 (2000)
4. Floater, M.S., Hormann, K.: Surface parameterization: a tutorial and survey. Adv. Multiresolution Geometric Modelling **2005**, 157–186 (2005)
5. Sheffer, A., Hormann, K., Levy, B., Desbrun, M., Zhou, K., Praun, E., Hoppe, H.: Mesh parameterization: theory and practice. ACM SIGGRAPPH, course notes **2007**, 10
6. Eck, M., Hadenfeld, J.: Local energy fairing of B-spline curves. In: Hagen, H., Farin, G., Noltemeier, H., (eds.) Proceedings of the Geometric Modelling, pp. 129–147. Springer, Vienna, (1995)
7. Floater, M.S.: Parametrization and smooth approximation of surface triangulations. Comput. Aided Geometric Design **14**(3), 231–250 (1997)
8. Greiner, G., Hormann, K.: Interpolating and approximating scattered 3D-data with hierarchical tensor product B-splines. In: Proceedings of the in Surface Fitting and Multiresolution Methods, pp. 163–172. Vanderbilt University Press (1997)

9. Ma, W., Kruth, J.-P.: Parameterization of randomly measured points for least squares fitting of B-spline curves and surfaces. Comput. Aided Des. **27**, 663–675 (1995)
10. Surface Reconstruction from Unorganized Points | Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques Available online: https://dl.acm.org/doi/abs/https://doi.org/10.1145/133994.134011. Accessed 19 Feb 2022
11. Pottmann, H., Leopoldseder, S., Hofer, M.: Approximation with active B-spline curves and surfaces. In: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings, pp. 8–25. IEEE (2002)
12. Azariadis, P., Sapidis, N.: Efficient parameterization of 3D point-sets using recursive dynamic base surfaces. In: Panhellenic Conference on Informatics, pp. 296–306. Springer, Heidelberg (2005)
13. Azariadis, P., Sapidis, N.: Product design using point-cloud surfaces: a recursive subdivision technique for point parameterization. Comput. Ind. **58**, 832–843 (2007). https://doi.org/10.1016/j.compind.2007.03.001
14. Floater, M.S., Reimers, M.: Meshless parameterization and surface reconstruction. Comput. Aided Geometric Design **18**, 77–92
15. Floater, M.S.: Meshless parameterization and B-spline surface approximation. In: Proceedings of the Mathematics of Surfaces IX, pp. 1–18. Springer (2000)
16. Volodine, T., Roose, D., Vanderstraeten, D., Volodine, T., Roose, D., Vanderstraeten, D.: 65F05, 65M50. Efficient Triangulation of Point Clouds Using Floater Parameterization 2004
17. Volodine, T., Vanderstraeten, D., Roose, D.: Experiments on the Parameterization of Point Clouds with Holes. TW Reports, volume TW432 **2005**, 12
18. Hormann, K., Reimers, M.: Triangulating point clouds with spherical topology. Curve and Surface Design: Saint-Malo (2002), pp. 215–224 (2002)
19. Zhang, L., Liu, L., Gotsman, C., Huang, H.: Mesh reconstruction by meshless denoising and parameterization. Comput. Graph. **34**, 198–208 (2010)
20. Li, E.: Meshless quadrangulation by global parameterization. Comput. Graph. **35**(5), 992–1000 (2011)
21. Choi, G.P.-T., Ho, K.T., Lui, L.M.: Spherical conformal parameterization of genus-0 point clouds for meshing. arXiv:1508.07569 [cs, math] **2016**. https://doi.org/10.1137/15M1037561
22. 3D Point Clouds Parameterization Alogrithm | IEEE Conference Publication | IEEE Xplore Available online: https://ieeexplore.ieee.org/abstract/document/4697396. Accessed 19 Feb 2022
23. Zwicker, M., Gotsman, C.: Meshing point clouds using spherical parameterization. In: PBG, pp. 173–180 (2004)
24. Tewari, G., Gotsman, C., Gortler, S.J.: Meshing genus-1 point clouds using discrete one-forms. Comput. Graph. **30**, 917–926 (2006). https://doi.org/10.1016/j.cag.2006.08.019
25. Li, E., Che, W., Zhang, X., Zhang, Y.-K., Xu, B.: Direct quad-dominant meshing of point cloud via global parameterization. Comput. Graph. **35**, 452–460 (2011). https://doi.org/10.1016/j.cag.2011.03.021
26. Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., Bennamoun, M.: Deep learning for 3D point clouds: a survey. IEEE Trans. Pattern Anal. Mach. Intell. **43**, 4338–4364 (2021)
27. Barhak, J., Fischer, A.: Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques. IEEE Trans. Visual Comput. Graphics **7**, 1–16 (2001)
28. Meng, Q., Li, B., Holstein, H., Liu, Y.: Parameterization of point-cloud freeform surfaces using adaptive sequential learning RBFnetworks. Pattern Recogn. **46**, 2361–2375 (2013). https://doi.org/10.1016/j.patcog.2013.01.017
29. Scholz, F., Jüttler, B.: Parameterization for polynomial curve approximation via residual deep neural networks. Comput. Aided Geometric Design **85**, 101977 (2021). https://doi.org/10.1016/j.cagd.2021.101977

30. Jung, H.B., Kim, K.: A new parameterisation method for NURBS surface interpolation. Int. J. Adv. Manuf. Technol. **16**, 784–790 (2000). https://doi.org/10.1007/s001700070012
31. Choi, G.P.T., Liu, Y., Lui, L.M.: Free-Boundary Conformal Parameterization of Point Clouds. arXiv:2010.15399 [cs, math] 2021
32. Zwicker, M., Pauly, M., Knoll, O., Gross, M.: Pointshop 3D: an interactive system for point-based surface editing. ACM Trans. Graph. (TOG) **21**, 322–329 (2002)

# Machine Learning and Data Assimilation for Dynamical Systems

# Statistical Prediction of Extreme Events from Small Datasets

Alberto Racca[1(✉)] and Luca Magri[1,2,3]

[1] Department of Engineering, University of Cambridge, Cambridge, UK
ar994@cam.ac.uk
[2] Aeronautics Department, Imperial College London, London, UK
l.magri@imperial.ac.uk
[3] The Alan Turing Institute, London, UK

**Abstract.** We propose Echo State Networks (ESNs) to predict the statistics of extreme events in a turbulent flow. We train the ESNs on small datasets that lack information about the extreme events. We asses whether the networks are able to extrapolate from the small imperfect datasets and predict the heavy-tail statistics that describe the events. We find that the networks correctly predict the events and improve the statistics of the system with respect to the training data in almost all cases analysed. This opens up new possibilities for the statistical prediction of extreme events in turbulence.

**Keywords:** Extreme events · Reservoir computing · Heavy tail distribution

## 1 Introduction

Extreme events arise in multiple natural systems, such as oceanic rogue waves, weather events and earthquakes [1]. A way to tackle extreme events is by computing their statistics to predict the probability of their occurrence. Because extreme events are typically rare, information about the heavy tail of the distribution that describes the events is seldom available. This hinders the performance of data-driven methods, which struggle to predict the events when extrapolating from imperfect datasets [8]. In this work, we assess the capability of a form of reservoir computing, the Echo State Network [5], to predict the statistics of extreme events in a turbulent flow [6]. In particular, we analyse the ability of the networks to improve the prediction of the statistics of the system with respect to the available training data. The paper is organised as follow. Section 2 introduces the turbulent flow model. Section 3 describes the Echo State Network. Section 4 analyses the statistical prediction of extreme events. We summarize the work and present future developments in Sect. 5.

A. Racca is supported by the EPSRC-DTP and the Cambridge Commonwealth, European & International Trust under a Cambridge European Scholarship. L. Magri is supported by the ERC Starting Grant PhyCo 949388.


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 707–713, 2022.
https://doi.org/10.1007/978-3-031-08757-8_58

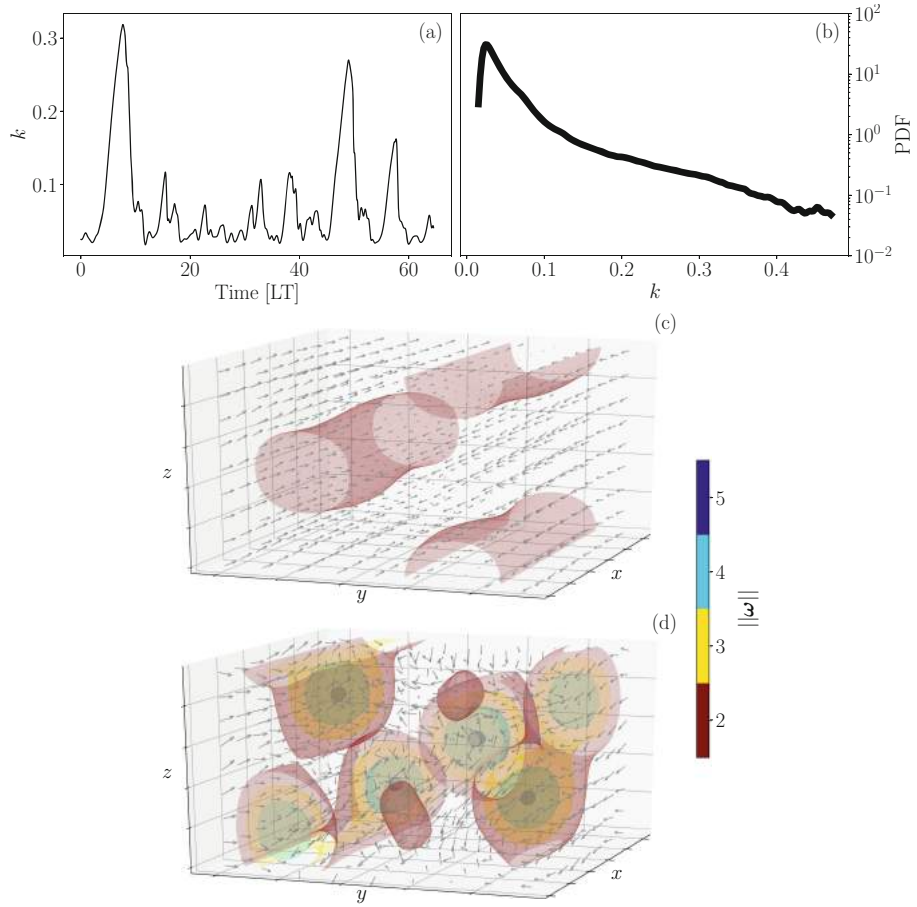



**Fig. 1.** One time series of the kinetic energy, (a), and Probability Density Function of the kinetic energy computed from the entire dataset, (b). The time in panel (a) is normalized by the Lyapunov time. Vorticity isosurfaces, $\boldsymbol{\omega} = \nabla \times \mathbf{v}$, and velocity flowfield before, (c), and after, (d), an extreme event. The laminar structure, (c), breaks down into vortices, (d).

## 2 A Low-Dimensional Model for Turbulent Shear Flow

We study a nine-equation model of a shear flow between infinite plates subjected to sinusoidal body forcing [6]. The incompressible Navier-Stokes equations are

$$\frac{d\mathbf{v}}{dt} = -(\mathbf{v} \cdot \in)\mathbf{v} - \in p + \frac{1}{\mathrm{Re}}\mathrm{ffi}\mathbf{v} + \mathbf{F}(y), \tag{1}$$

where $\mathbf{v} \leq (u, v, w)$ is the velocity, $p$ is the pressure, Re is the Reynolds number, $\mathbf{F}(y) = \overline{2}\pi^2/(4\mathrm{Re})\sin(\pi y/2)\mathbf{e}_x$ is the body forcing along $x$, $y$ is the direction of the shear between the plates and $z$ is the spanwise direction. We solve the flow in

the domain $L_x \times L_y \times L_z$, where the boundary conditions are free slip at $y \pm L_y/2$, and periodic at $x = [0; L_x]$ and $z = [0; L_z]$. Here, we set $L_x = 4\pi, L_y = 2, L_z = 2\pi$ and Re $= 400$ [9]. We project (1) on compositions of Fourier modes, $\hat{\mathbf{v}}_i(\mathbf{x})$, so that the velocity is $\mathbf{v}(\mathbf{x}, t) = \sum_{i=1}^{9} a_i(t)\hat{\mathbf{v}}_i(\mathbf{x})$ . The projection generates nine nonlinear ordinary differential equations for the amplitudes, $a_i(t)$, which are the state of the system [6]. The system displays a chaotic transient that converges to the laminar solution $a_1 = 1, a_2 = \cdots = a_9 = 0$. In the turbulent transient, the kinetic energy,

$$k = 0.5 \sum_{i=1}^{9} a_i^2, \tag{2}$$

shows intermittent large bursts, i.e. extreme events, panel (a) in Fig. 1, which generate the heavy tail of the distribution [8], panel (b). In the figure, time is expressed in Lyapunov Times (LT), where a LT is the inverse of the Lyapunov exponent, ffi $*$ 0.0163. The Lyapunov exponent is the average exponential rate at which arbitrarily close trajectories diverge, which is computed with the QR algorithm [2,3]. Each extreme event is an attempt of the system to reach the laminar solution. During an extreme event, the flow slowly laminarizes, panel (c), but the laminar structure violently breaks down into vortices, panel (d). To study only the transient, we (i) generate 2000 time series series of length of 4000 time units through a 4th order Runge-Kutta scheme with $dt = 0.25$, (ii) discard all the time series that laminarized, i.e. the ones with $k \odot 0.48$, and (iii) use the remaining time series as data. The different time series are obtained by randomly perturbing a fixed initial condition [9].

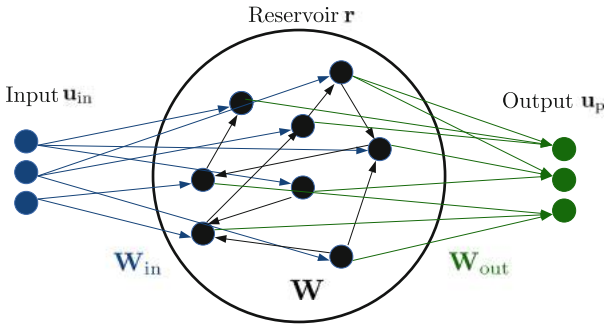



**Fig. 2.** Schematic representation of the echo state network.

## 3 Echo State Networks

As shown in Fig. 2, in an Echo State Network [5], at the $i$-th time step the high-dimensional reservoir state, $\mathbf{r}(t_i) \in \mathbb{R}^{N_r}$, is a function of its previous value and

the current input, $\mathbf{u}_{\text{in}}(t_i) \in \mathbb{R}^{N_u}$. The output, $\mathbf{u}_{\text{p}}(t_{i+1})$, which is the predicted state at the next time step, is a linear combination of $\mathbf{r}(t_i)$:

$$\mathbf{r}(t_i) = \tanh\left(\mathbf{W}_{\text{in}}[\tilde{\mathbf{u}}_{\text{in}}(t_i); b_{\text{in}}] + \mathbf{W}\mathbf{r}(t_{i-1})\right); \quad \mathbf{u}_{\text{p}}(t_{i+1}) = \mathbf{W}_{\text{out}}[\mathbf{r}(t_i); 1] \tag{3}$$

where $\tilde{(\,)}$ indicates normalization by the range component-wise, $\mathbf{W} \in \mathbb{R}^{N_r \times N_r}$ is the state matrix, $\mathbf{W}_{\text{in}} \in \mathbb{R}^{N_r \times (N_u+1)}$ is the input matrix, $\mathbf{W}_{\text{out}} \in \mathbb{R}^{N_u \times (N_r+1)}$ is the output matrix, $b_{\text{in}}$ is the input bias and $[\,;\,]$ indicates vertical concatenation. $\mathbf{W}_{\text{in}}$ and $\mathbf{W}$ are sparse, randomly generated and fixed. These are constructed in order for the network to satisfy the echo state property [5]. The input matrix, $\mathbf{W}_{\text{in}}$, has only one element different from zero per row, which is sampled from a uniform distribution in $[-\sigma_{\text{in}}, \sigma_{\text{in}}]$, where $\sigma_{\text{in}}$ is the input scaling. The state matrix, $\mathbf{W}$, is an Erdős-Renyi matrix with average connectivity $\langle d \rangle$. This means that each neuron (each row of $\mathbf{W}$) has on average only $\langle d \rangle$ connections (non-zero elements). The value of the non-zero elements is obtained by sampling from an uniform distribution in $[-1, 1]$; the entire matrix is then scaled by a multiplication factor to set its spectral radius, $\rho$. The only trainable weights are those in the output matrix, $\mathbf{W}_{\text{out}}$. Thanks to the architecture of the ESN, training the network by minimizing the Mean Square Error (MSE) on $N_{\text{t}} + 1$ points consists of solving the linear system

$$(\mathbf{R}\mathbf{R}^T + \beta\mathbf{I})\mathbf{W}_{\text{out}}^T = \mathbf{R}\mathbf{U}_{\text{d}}^T, \tag{4}$$

where $\mathbf{R} \in \mathbb{R}^{(N_r+1)\times N_{\text{t}}}$ and $\mathbf{U}_{\text{d}} \in \mathbb{R}^{N_u \times N_{\text{t}}}$ are the horizontal concatenation of the reservoir states with bias, $[\mathbf{r}; 1]$, and of the output data, respectively; $\mathbf{I}$ is the identity matrix and $\beta$ is the Tikhonov regularization parameter [5].

The input scaling, $\sigma_{\text{in}}$, spectral radius, $\rho$, and Tikhonov parameter, $\beta$, are selected using Recycle Validation [7] to minimize the MSE of the kinetic energy. The Recycle Validation is a recent advance in hyperparameter selection in Recurrent Neural Networks, which is able to exploit the entire dataset while keeping a small computation cost. To minimize the function provided by the validation strategy, we use Bayesian Optimization for $\sigma_{\text{in}}$ and $\rho$ in the interval $[0.1, 10] \times [0.1, 1]$ seen in logarithmic scale and perform a grid search in each $[\sigma_{\text{in}}, \rho]$ point to select $\beta$ from $[10^{-6}, 10^{-9}, 10^{-12}]$. We set $b_{\text{in}} = 0.1$, $d = 20$ and add gaussian noise with zero mean and standard deviation, $\sigma_n = 0.01\sigma_u$, where $\sigma_u$ is the standard deviation of the data, to the training data [10].

## 4 Statistical Prediction of Extreme Events

We study the capability of the networks to predict the statistics of the system through long-term predictions. Long-term predictions are closed-loop predictions, i.e. predictions where we feed the output of the ESN as an input for the next time step, which lasts several tens of Lyapunov Times. These predictions diverge from the true trajectory due to the chaotic nature of the signal, but remain in the region of phase space of the chaotic transient. In doing so, they

replicate the statistics of the true signal. The long-term predictions are generated in the following way: (i) from 500 different starting points in the training set, we generate 500 different time series by letting the ESN evolve each time for 4000 time units ($*$ 65LTs); (ii) we discard the laminarized time series and (iii) use the remaining ones to compute the statistics as done for the data, see Sect. 2. To quantitatively assess the prediction of the statistics, we use the Kantorovich metric [4], $\mathcal{K}$, also known as Earth mover's distance, and the Mean Logarithmic Error (MLE) with respect to the true Probability Density Function of the kinetic energy, $\mathrm{PDF}_{\mathrm{True}}(k)$,

$$\mathcal{K} = \int_{-\infty}^{\infty} |\mathrm{CDF}_{\mathrm{True}}(k) - \mathrm{CDF}_j(k)| dk, \tag{5}$$

$$\mathrm{MLE} = \sum_{i=1}^{n_b} n_b^{-1} |\log_{10}(\mathrm{PDF}_{\mathrm{True}}(k)_i - \log_{10}(\mathrm{PDF}_j(k)_i)|, \tag{6}$$

where CDF is the Cumulative Distribution Function, $j$ indicates the PDF we are comparing with the true data and $n_b$ is the number of bins used in the PDF. When a bin has a value equal to zero and the logarithm is undefined, we saturate the logarithmic error in the bin to be equal to 1. On the one hand, we use the Kantorovich metric to assess the overall prediction of the PDF of the kinetic energy. On the other hand, we use the MLE to assess the prediction of the extreme events, as the logarithm highlights the errors in the small values of the tail.

In Fig. 3, we compare the statistics of the training data and an ensemble of 10 networks of 2000 neurons. We do so because the objective of predicting the statistics is to improve our knowledge, by employing the networks, with respect to the already available knowledge, the training data. Panel (a) shows the PDF of the kinetic energy in the training set for different sizes of the training set, from 1 time series to the entire data (1440). The prediction of the PDF improves with the size of the datasets, and values of the tail up to laminarization are observed only after 100 time series. The unresolved tail due to lack of data is a signature problem of data-driven analysis of extreme events [8]. Panels (b)–(c) show the Kantorovich metric and the MLE of the training sets and networks as a function of the training set size. The networks improve the prediction of the PDF with respect to the available data in all figures of merits analyzed, except for one outlier. The MLE of the training set improves more than the Kantorovich metric as the dataset becomes larger. This happens because a small amount of data is needed to accurately describe the peak of the PDF, which affects more the Kantorovich metric, while many time series are needed to describe the tail, which affects more the Mean Logarithmic Error. The results indicate that the networks are able to extrapolate from an imperfect dataset and improve the prediction of the overall dynamics of the system.

Fig. 4 shows the statistics of the square of the normal vorticity to the midplane, $\omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}$. We plot the square of the vorticity, $\omega_y^2$, because the symmetry of the problem causes the time-average of the vorticity to be equal to zero. Panel (a) shows the flowfield of the time-average, $\overline{(\,)}$, for the entire data,

while panels (b) and (c) show the error with respect to (a) for an Echo State Network and the ten time series training set, respectively. All networks in the ensemble decrease the average error, up to values 7 times smaller than the training data (results not shown). This means that the Echo State Networks are able to extrapolate the statistics of the flowfield in addition to the statistics of the kinetic energy.

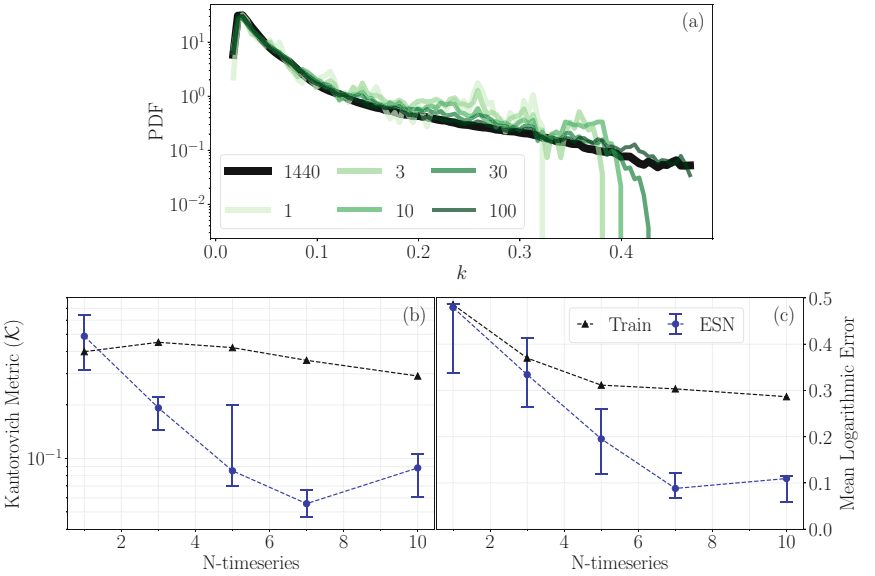



**Fig. 3.** PDF of the kinetic energy for different number of time series up to the entire data of 1440 time series, (a). For example, 3 means that the PDF is computed from 3 out of 1440 time series. 25th, 50th and 75th percentiles of Kantorovich Metric, (b), and MLE, (c), for the training set (Train), and the networks (ESN) as a function of the number of time series in the training set (N-timeseries). For example, N-timeseries means that the training set consists of N out 1440 time series.

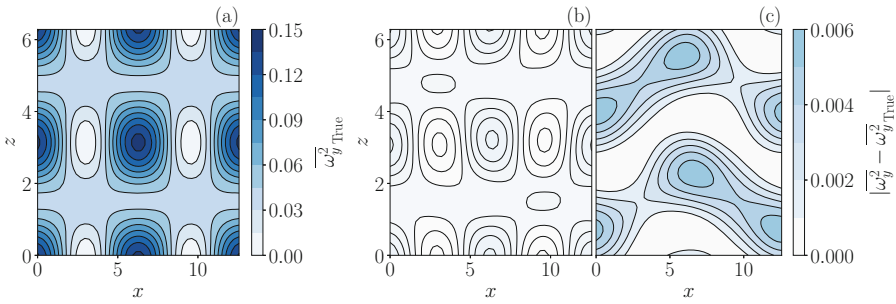



**Fig. 4.** Time average of the square of the midplane vorticity, $\epsilon_y^2(x, 0, z)$ for the entire data, (a), error for a 2000 neurons network, (b), and training set, (c).

## 5 Conclusions and Future Directions

We propose Echo State Networks to predict the statistics of a reduced-order model of turbulent shear flow that exhibits extreme events. We train fully data-driven ESNs on multiple small datasets and compare the statistics predicted by the networks with the statistics available during training. We find that the networks improve the prediction of the statistics of the kinetic energy and of the vorticity flowfield, sometimes by up to one order of magnitude. This means that the networks are able to extrapolate the statistics of the system when trained on small imperfect datasets. Future work will consist of extending the present results to higher-dimensional turbulent systems through the combination of Echo State Networks and autoencoders.

The code is available on the github repository MagriLab/ESN-MFE.

## References

1. Farazmand, M., Sapsis, T.P.: Extreme events: mechanisms and prediction. Appl. Mech. Rev. **71**(5) (08 2019). https://doi.org/10.1115/1.4042065, 050801
2. Ginelli, F., Poggi, P., Turchi, A., Chaté, H., Livi, R., Politi, A.: Characterizing dynamics with covariant lyapunov vectors. Phys. Rev. Lett. **99**(13), 130601 (2007)
3. Huhn, F., Magri, L.: Stability, sensitivity and optimisation of chaotic acoustic oscillations. J. Fluid Mech. **882**, A24 (2020). https://doi.org/10.1017/jfm.2019.828
4. Kantorovich, L.V.: On the translocation of masses. In: Dokl. Akad. Nauk. USSR (NS), vol. 37, pp. 199–201 (1942)
5. Lukoševičius, M.: A practical guide to applying echo state networks. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 659–686. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_36
6. Moehlis, J., Faisst, H., Eckhardt, B.: A low-dimensional model for turbulent shear flows. New J. Phys. **6**(1), 56 (2004)
7. Racca, A., Magri, L.: Robust optimization and validation of echo state networks for learning chaotic dynamics. Neural Netw. **142**, 252–268 (2021)
8. Sapsis, T.P.: Statistics of extreme events in fluid flows and waves. Ann. Rev. Fluid Mech. **53**, 85–111 (2021)
9. Srinivasan, P.A., Guastoni, L., Azizpour, H., Schlatter, P., Vinuesa, R.: Predictions of turbulent shear flows using deep neural networks. Phys. Rev. Fluids **4**, 054603 (2019)
10. Vlachas, P.R., Pathak, J., Hunt, B.R., Sapsis, T.P., Girvan, M., Ott, E., Koumoutsakos, P.: Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. Neural Netw. **126**, 191–217 (2020)

# Outlier Detection for Categorial Data Using Clustering Algorithms

Agnieszka Nowak-Brzezińska(✉) and Weronika Łazarz

Institute of Computer Science, Faculty of Science and Technology, University of Silesia, Bankowa 12, Katowice 40-007, Poland
agnieszka.nowak-brzezinska@us.edu.pl

**Abstract.** Detecting outliers is a widely studied problem in many disciplines, including statistics, data mining and machine learning. All anomaly detection activities are aimed at identifying cases of unusual behavior when compared to the remaining set. There are many methods to deal with this issue, which are applicable depending on the size of the dataset, the way it is stored and the type of attributes and their values. Most of them focus on traditional datasets with a large number of quantitative attributes. While there are many solutions available for quantitative data, it remains problematic to find efficient methods for qualitative data. The main idea behind this article was to compare categorical data clustering algorithms: *K-modes* and $ROCK$. In the course of the research, the authors analyzed the clusters detected by the indicated algorithms, using several datasets different in terms of the number of objects and variables, and conducted experiments on the parameters of the algorithms. The presented study has made it possible to check whether the algorithms detect the same outliers in the data and how much they depend on individual parameters such as the number of variables, tuples and categories of a qualitative variable.

**Keywords:** Qualitative data · Outlier detection · Data clustering · *K-modes* · $ROCK$

## 1 Introduction

The article deals with the clustering of qualitative data to detect outliers in these data. Thus, in the paper, we encounter two research problems: clustering qualitative data and detecting outliers in such data. We look at outliers as atypical (rare) data. If we use clustering algorithms for this purpose, outliers are data that are much more difficult to include in any group than the typical (normal) data. Clustering qualitative data is a more extensive research problem than clustering quantitative data. We count the distance between the numeric values on each attribute that describes the objects. Quantitative data can be normalized which allows us to interpret the differences between the compared objects properly. Assessing the similarity between two objects described by qualitative attributes is a challenging task. Let us take *eye color* as an example of a qualitative attribute. Now, let us take into account three persons: $A$ with *blue* eyes, $B$ with *brown* and $C$ with *gray* eyes. There are various methods to measure their similarity. We may say that *blue* is more similar to *gray* than *brown*. In fact, we know that *gray* is much


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 714–727, 2022.
https://doi.org/10.1007/978-3-031-08757-8_59

more similar to *blue* than *brown*. But we may also want to compare them as a plain text and then *blue* and *brown* share the same initial letter which makes them more similar than pairs *blue-gray* or *brown-gray*. It all depends on the method we use to compare the objects. It is also worth remembering that the comparing of the objects in the set will significantly impact the structure of the groups that we create.

By default, clustering algorithms, known in the literature for years, are based on the concept of data distances in a metric space, e.g., in Euclidean space. The smaller the distance between the objects, the greater the probability that they will form one group. If the distance between a given object from all created groups is too great, then we should consider the object as an *outlier* in the data. This idea seems logical. In the context of qualitative data: when a given object shows no similarity to the created groups, then it can be considered an outlier in the data.

In the study, we have made use of real datasets from various fields. This type of data very often contains some unusual pieces of data. They are not the result of a measurement error, but they actually differ from most of the data in the set. It is not always the case that one or more objects stand out significantly from the rest, and we can easily see it. Sometimes, it is also the case that certain subsets of objects differ to the same extent from the majority of data. The problem becomes even more complicated when we take into account the fact that these objects in the sets may be more or less differentiated by the specificity of the domain they come from, but also by the method of describing these data (the number of attributes, the number of possible values of these attributes, the number of objects). When objects are described on a categorical scale, the effectiveness of their correct clustering and outlier detection is necessary for a deeper study. In this paper, we analyze clustering algorithms from two types of clustering: hierarchical ($ROCK$) and non-hierarchical ($K-modes$). In case of quantitative data, the clustering process works as follows. Hierarchical algorithms in each iteration look for a pair or groups of objects with a smallest distance and combine them into a group. The process is repeated until an expected number of clusters is reached or until all groups have merged into one group. On the other hand, non-hierarchical algorithms (like the most popular clustering algorithm $K-means$), search for the best partition for a predetermined number of groups so that the distances inside the clusters are small and the clusters are as large as possible. In qualitative data, we should modify the algorithms to be suitable for operating on data for which we cannot explicitly measure distances. In case of non-hierarchical algorithms, we cannot use the $K-means$ algorithm because it forms its representative by determining the value of the so-called center of gravity of the group. For quantitative data, it is simply an arithmetic mean of the attribute values describing the features that make up the group. For qualitative data, we cannot derive a mean value. However, we can find a most common value. And this is the concept behind the $K-modes$ algorithm we chose for our research. In case of hierarchical algorithms, where two objects with the shortest distance are combined into a group iteratively, for datasets with qualitative data we cannot rely on the notion of distance. Instead, we use measures to determine the similarity of objects and, at each step of the algorithm, we connect the objects or groups of objects with the greatest similarity. This is the main idea of the $ROCK$ algorithm - a hierarchical clustering algorithm for qualitative data. We group the data to explore it better. Exploration has to do with the fact

that apart from its obvious task, which is discovering patterns or rules in data, we can also discover unusual data, outliers in the data.

Therefore, in this study, we decided to investigate the effectiveness of the two selected clustering algorithms: $K-modes$ and $ROCK$, in outlier detecting. We want to compare how consistent the algorithms are in this respect. If they are consistent, then they should designate the same objects for potential outliers. In the research, we will change the clustering parameters to find the optimal results. We will repeat the experiments for $5\%$, $10\%$, and $15\%$ outliers in the dataset. We expect that the more outliers we identify, the greater the coverage of the analyzed methods may be. We present the results in the section on experiments and research results.

## 2 State of Art

The methods of outlier detecting in datasets can be divided into formal and informal. Most formal tests require test statistics to test hypotheses and usually rely on some well-behaved distribution to check whether the extreme target value is out of range. However, real-world data distributions may be unknown or may not follow specific distributions. That is why it is worth considering other solutions, for example, clustering algorithms. In addition to the distribution-based methods, cluster-based approaches are also welcome. These approaches can effectively identify outliers as points that do not belong to the created clusters or the clusters distinguished by a small number of elements [6,9]. So far, numerous works have been published focusing on detecting outliers and good data clusters in a quantitative dataset. The most well-known algorithm is the $LOF$ (Local Outlier Factor) algorithm proposed by Breunig in [2], in which local outliers are detected. Based on the ratio of the local density of a given object and the local density of its nearest neighbors, the $LOF$ factor is calculated. Then, the objects with the highest $LOF$ values are considered as outliers. Another method that isolates outliers and normal objects is the $IsolationForest$ method based on the construction of a forest of binary isolation trees. Then outliers are observations with shortest average path lengths from the root to the leaf [8]. The indicated algorithms are widely used in IT systems, both to clean datasets from noise so that they do not interfere with the system operation, and to detect unusual observations in the data for a further analysis. The presence of outliers in qualitative data can significantly disrupt the effectiveness of machine learning algorithms that try to find patterns in the data, such as rules, decision rules or association rules. Dividing the objects into groups in which the objects are as similar to each other as possible and thus detecting objects that do not match the groups is a very efficient solution to explore the outliers. We decided to choose two clustering algorithms, $K-modes$ and $ROCK$ - as they are the representatives of both hierarchical and non-hierarchical clustering algorithms. We found them very simple to interpret and implement on real data. So far, no papers describing the application of the indicated algorithms on a large scale or comparing the results with the distinction as to the type of data processed and the time of execution have been published. This has become the direct motivation of the authors of this paper to analyze those two selected clustering algorithms $K-modes$ and $ROCK$ in the context of their efficiency in detecting outliers in the qualitative data.

## 3 Data Clustering

The problem of clustering is one of the most researched issues in social sciences, psychology, medicine, machine learning, and data science. In addition to the standard benefits of data clustering, it has found a wide application in dataset processing with categorical domains, both in the course of preparation for mining and in the modeling process itself. Here, data clustering was used to find outliers in qualitative datasets. The two algorithms described in this section differ in terms of data clustering and outliers detection. The $K-modes$ algorithm, most frequently used in research and real IT systems, creates groups of clusters from objects closest to selected centroids and defines outliers as objects farthest from the cluster center. The $ROCK$ algorithm calculates the similarity measures between objects and groups of objects, creating data clusters containing objects that should not belong to any other cluster.

When dealing with quantitative data, we can easily use descriptive statistics, using quantities such as mean, median, standard deviation, and variance. When we handle qualitative data, it is not possible. We only know the most common value - a dominant. In such a case, clustering algorithms will cluster objects with the same value of a given attribute into groups. Of course, large clusters will be created by objects with a value equal to the dominant for a given attribute. For the clusters to be of good quality, we must effectively detect unusual data not to disturb the coherence and separation of the created data structures. We do not make assumptions that our sets contain outliers. We want our model to deal with any given dataset. If there are no outliers in the set, the cluster quality indicators will be very close to the values expected for the sets without outliers.

### 3.1 K-modes Clustering

The $K-modes$ clustering algorithm was proposed as an alternative to the popular $K-means$ algorithm, the most used centroid-based non-hierarchical algorithm [5]. The modifications made to the $K-means$ algorithm include using a simple measure of matching dissimilarity for qualitative features, replacing the group averages with vectors composed of the most common values at individual coordinates of the objects (modes), and using a frequency-based method to modes update. Let $X = \{x_1, \ldots, x_n\}$ be a set of $n$-objects $x$, such that $x = (x_1, \ldots, x_m)$.The dissimilarity measure of $x_1, x_2$ objects is defined as $d(x_1, x_2) = \sum_{i=1}^{m} \sigma(x_{1i}, x_{2i})$, where $\sigma(x_{1i}, x_{2i}) = 0$ if $x_{1i} = x_{2i}$ and 1 otherwise. Having $A = \{A_1, ..., A_m\}$ - set of the attributes of the objects in $X$ it is possible to define $S \approx X$ - a cluster of data. The mode of $S = \{x_1, ..., x_p\}, 1 \parallel p \parallel n$ is the vector $q = (q_1, ..., q_m)$ which minimizes the function $D(S, q) = \sum_{i=1}^{n} d(x_i, q)$ called the cost function. A cluster center is called a mode and is defined by considering those values of the attributes that appear most frequently in the data points which belong to that cluster. The $K-modes$ (Algorithm 1) algorithm begins with a random selection of $k$ objects (centroids) which are the central objects of $k$ clusters. Then, the dissimilarity measure is calculated and the closest centroid is determined for each object. When all objects are assigned to individual clusters, the centroids are updated by creating new modes from objects present in the cluster. The calculations are repeated until the differences in the generated clusters in the following steps cease to exist.

**Algorithm 1.** $K - modes$ algorithm

$input$: $X$-dataset, $k$-expected number of clusters
$output$: a set of $k$ clusters

1. Randomly select $k$ items (modes) from the dataset.
2. For each pair (mode, object), calculate the dissimilarity measure.
3. For each object that is not a mode, find the mode closest to the object.
4. Join objects with the corresponding modes to create clusters.
5. For all clusters, recalculate the modal vectors containing in successive coordinates the most common values on attributes of cluster objects.
6. Perform steps 3-5 until the generated clusters do not repeat themselves.

The $K - modes$ algorithm is the easiest to implement and the most popular among the categorical data clustering algorithms because it is linearly scalable concerning the size of the dataset. The disadvantage of the algorithm is that it selects random initial modes, leading to unique structures around objects that are undesirable in the set. A method to prevent such situations is to draw the initial set of modes multiple times and assign each object to the cluster with the greatest number of times. The output clusters generated by the $K - modes$ algorithm have a similar cardinality, which does not have to reflect the actual data clusters on the sets having atypical distributions of variables. As with most categorical clusters, clusters containing a tiny number of elements or a single element can be considered outliers. The specifics of $K - modes$ clustering show that we will create single-element clusters only if the initially drawn object is an outlier. If we want to obtain a reliable mapping in small individual clusters, we can run the algorithm multiple times, each time randomizing a different set of initial $K - modes$ and finish the work when the variability is low in the final set of clusters. Finding the similarity between a data object and a cluster requires $n$ operations, which for all $k$ clusters is $nk$. Assigning objects to the appropriate $k$ clusters and updating mods also require $nk$ operations. Assuming the algorithm is run $I$ times for different starting objects, the algorithm will have a linear complexity of $O(nkI)$.

### 3.2 ROCK Clustering

The $ROCK$ algorithm (RObust Clustering using linKs) [4], is a hierarchical clustering algorithm for categorical data. The algorithm introduces notions of neighbors and links. A point's neighbours are those points that are considerably similar to it. A similarity function between points defines the closeness between pairs of points. A user defines the threshold for which the pairs of points with a similarity function value greater than or equal to this value are considered to be neighbors. The number of links between pairs of points is defined to be the number of common neighbors for the points. The larger the number of links between a pair of points, the greater the likelihood is that they belong in the same cluster. Starting with each point in its own cluster, the algorithm repeatedly merges the two closest clusters till a desired number of clusters remain or when a situation arises in which no two clusters can be merged.

**Algorithm 2.** $ROCK$ algorithm

$input$: sample set of objects. Number of $k$ clusters to be found. The similarity threshold: $\theta \geq 0.4$
$output$: A group of objects - a cluster
Do for All Data {

1. Initially, place each object into a separate cluster.
2. Construction of a Similarity Matrix with similarity for each pair of objects (A,B) using measure $Similarity(A, B) = \frac{|A \cap B|}{|A \cup B|}$
3. Computation of an Adjacency Matrix (A) using a similarity threshold $\theta \geq 0.4$ if $similarity(A, B) \geq \theta$ then 1; else 0
4. Compute a Link Matrix by multiplying an Adjacency Matrix by itself to find the number of links.
5. Calculation of a Goodness Measure for each pair of objects by using the $g$ function
6. Merge the two objects with the highest similarity (goodness measure).
7. When no more entry exists in the goodness measure table then stop the algorithm which by now should have returned $k$ number of clusters and outliers (if any), otherwise go to step 4.

}

The following features of this algorithm are necessary to define:

- Links - the number of common neighbors between two objects.
- Neighbors - if a similarity between two points exceeds certain similarity threshold, they are neighbors: if $similarity(A, B) \in \theta$ then two points $A$, $B$ are neighbors, for $\theta$ being a user-specified threshold.
- Criterion Function - the objective is to achieve a good cluster quality by maximizing the sum of links of intra cluster point pairs and minimizing the sum of links of inter cluster point pairs.
- Goodness Measure to maximize the criterion function and identify the best pair of clusters to be merged at each step of the $ROCK$ clustering algorithm.

$ROCK$ is a unique algorithm because it assumes that an attribute value, in addition to its frequency, must be examined based on the number of other attribute values with which it occurs. Due to its high computational complexity, $ROCK$ is good at detecting outliers in small datasets, and its computational time increases as the records in the set increase. This is because each record must be treated as a unique data cluster. If the user does not have a comprehensive knowledge about the dataset, the appropriate selection of the $\theta$ value and the minimum number of clusters generated on the output is a challenging task. The $ROCK$ algorithm is very resistant to outliers and can successfully identify outliers that are relatively isolated from the rest of the points. The ones with very few or no neighbors in one- or several-member clusters will be considered outliers. The overall computational complexity will depend on the number of neighbors of each facility. In most cases, the order of complexity will be $O(n^2 \log n)$. If a maximum and an average number of neighbors are close to $n$, then the algorithm's complexity increases to $O(n^3)$.

## 4 Conducted Research

The algorithms described in Sect. 3 were implemented in the Python language (version 3.8.8). We used the JupyterHub (version 6.3.0) environment available at *https://jupyter.org/hub* for the implementation and visualization of the data. JupyterHub runs in the cloud or on hardware locally and supports a preconfigured data science environment for each user. We used Anaconda package containing most of the libraries, enabling machine learning models and visualization of results. The existing models of the Scikit-Learn library were used to implement the $K-modes$ algorithm. The $ROCK$ algorithm due to a lack of previous implementation was implemented by the authors. We used the Matplotlib library and the Pandas Dataframe structure for data visualization. Most of the computation is based on the Pandas data structures that hold the results.

The computer program described by the authors has been divided into sections containing:

- Importing Python libraries SciPy (1.6.2), Scikit-learn (0.24.1), NumPy, Pandas (1.2.4), Matplotlib (3.3.4) and libraries to perform operations related to time.
- Implementing algorithms: $ROCK$ with the parameters: $k$ denoting the expected number of clusters and *theta* being a parameter of a function that returns an estimated number of neighbors and $K-modes$ with $k$ parameter denoting the expected number of clusters and *threshold* parameter denoting the percentage of expected outliers.
- Data preprocessing: dealing with missing values (function that completes missing fields with the most common value in a column and removes columns that contain more than 60 empty values), coding the variables (encoding text values into numerical values), decoding encoded text variables.
- Uploading all datasets (reading, calculating the descriptive statistics, encoding text variables for the selected dataset to visualize the result).
- Execution of $ROCK$ and $K-modes$ algorithms on datasets. Presentation of the algorithms' computation time in relation to the type of the algorithm.
- Presentation of the algorithms' computation time in relation to the number of variables, the number of records, and data diversity.
- Listing the numbers of individual clusters obtained by the $ROCK$, $K-modes$ algorithms.
- Showing the selected dataset with assigned cluster numbers for the $ROCK$ and $K-modes$ algorithms and flags that indicate whether a record has been classified as an outlier. If the flag is $-1$, the object is an outlier. If it is 1, the object is considered normal.
- Presentation of the matrix of similarities and differences in classifying values as outliers for the $ROCK$ and $K-modes$ algorithms when compared in pairs.
- Identification of common outliers generated by the $ROCK$ and $K-modes$ algorithms.

The source of the software was placed in the GitHub repository: https://github.com/wlazarz/outliers2. It contains the implementation of the $K-modes$ and $ROCK$ algorithms and six datasets on which the experiments were conducted. The sequence of steps performed to compare the clustering and outliers detection algorithms is presented in

Fig. 1. The equipment specification on which we conduct our research is as follows: MacBook Pro Retina (15-inch, Mid 2015), macOS Catalina (10.15.7), 2,2 GHz processor four-core Intel Core I7, RAM 16 GB 1600 MHz MHz DDR3, GPU Intel Iris Pro 1536 MB. GPU acceleration and XAMPP were not used.

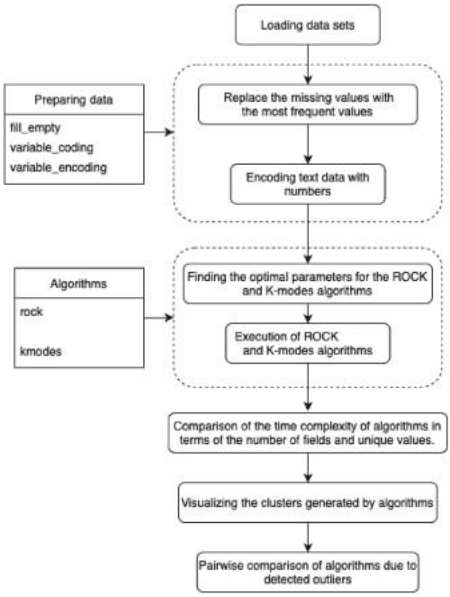



**Fig. 1.** Scheme of the program comparing algorithms clustering data and detecting outliers.

### 4.1 Data Description

We used six qualitative datasets to compare the algorithms that detect outliers in the data, each with a different structure of the variables matched to the clustering-based algorithms which support the detection of outliers in the qualitative datasets. The sets have different sizes and consist of a different number of categorical variables. The characteristics of the selected datasets are presented in Table 1. All analyzed datasets are real datasets, four of which relate to the domain of medicine ($Primary\ Tumor$ [10], $Lymphography$ [11], $SPECT\ Heart$ [12], $Covid-19$ [13]). In addition to the medical databases, two others were also analyzed: $BM\_attack$ [16] and $wiki$ [15]. The set $wiki$ contains the highest number of objects (913) and attribute values (285 unique values).

The first step in the project was to load datasets and prepare them properly before clustering commences. In all datasets, we filled empty fields with the most common value on a given variable. Categorical variables were encoded into numeric variables on *Primary Tumor Dataset* and *Lymphography Dataset*. Despite reducing the dataset to a numerical form, algorithms working on qualitative sets treat numbers as categories of

variables. The process of numerical encoding of the test values was intended to reduce a long execution time of the algorithms resulting from the need to compare each sign of the test value.

### 4.2 Methodology

We conducted the experiments empirically. Initially, we tried to automate the experiments by launching the execution of the algorithms: $K - modes$ and $ROCK$, and changing the parameter values of these algorithms iteratively. However, several lenghtly multi-hour processes were interrupted by an excessive memory consumption. As a result, the experiments were finally carried out empirically for the gradually and consciously changed parameter values (e.g., number of clusters). The *elbow method* was used while looking for parameters for the $K - modes$ algorithm [14]. If the number of clusters selected with this method generated substantial outliers (many objects were on the border of $5\%$, $10\%$, $15\%$ of outliers), the number of clusters was increased or decreased, still oscillating around the threshold point. The authors checked a cluster relevance using the *Silhouette method*, but the structure of created clusters was not always satisfactory [17]. In case of the $ROCK$ algorithm we took into account the number of clusters (already established during the execution of the $K - modes$ algorithm) and and initial epsilon value (a maximum distance at which elements can be in one cluster) = $0.6$. Most of the sets we dealt with had a reasonable number of outliers within the epsilon value of $0.6$. If too many outliers were obtained, the epsilon value was increased. If increasing this value results in even more outliers, the number of clusters was decreased. Conversely, for too few outliers obtained, the epsilon was reduced, or the number of clusters was increased.

## 5 Experiments

This section covers the results of the comparison of the two algorithms described in the previous section: $ROCK$ and $K - modes$. We compared the algorithms in terms of their time complexity. At the very beginning, it is worth emphasizing that in this paper, we present the results obtained as a result of optimization of clustering parameters. Thus, by diligently changing the clustering parameters of both algorithms, we checked which combination of the values of these parameters gives optimal results. These optimal results (as one of many obtained) are presented below.

### 5.1 Time Complexities of Clustering Algorithms

Based on the sets described in Sect. 4.1, we performed an analysis of time complexity of the algorithms described in this work. The execution time of the algorithms is given in seconds. The study was conducted in the $JupyterHub$ environment installed locally on $MacBookPro$ hardware with $IntelCorei7$ quad-core processor and 16 GB RAM. The datasets are characterized by a different number of objects and variables and represent different types of data. The results are included in Table 1.

**Table 1.** Time complexity for $ROCK$, $K-modes$ and $K-means$ clustering algorithms

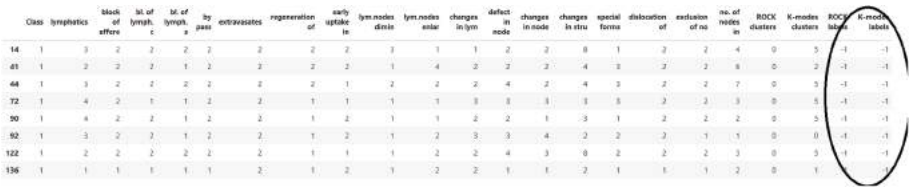
| Dataset | Rows | Columns | Values | Time Complexity [s] | | |
|---|---|---|---|---|---|---|
| | | | | ROCK | K-modes | K-means |
| BM_attack | 322 | 6 | 20 | 5,81 | 1,4 | 0,11 |
| SPECT | 267 | 23 | 46 | 3,67 | 2,91 | 0,47 |
| primary-tumor | 339 | 18 | 58 | 6,91 | 3,18 | 0,77 |
| lymphography | 148 | 19 | 62 | 0,72 | 1,52 | 0,22 |
| covid | 204 | 16 | 91 | 1,64 | 1,84 | 0,29 |
| wiki | 913 | 53 | 285 | **141,96** | **26,57** | **3,06** |

The $K-modes$ algorithm has an average linear or near-square complexity when diagnosed with many clusters. Regardless of the number of records, variables, and values, the execution time for the $K-modes$ algorithm is the lowest for each dataset. We can observe that the complexity of the $ROCK$ algorithm increases rapidly with the increase in the number of data.

### 5.2 Outlier Detection for Clustering Results

Algorithms working on qualitative datasets require the indication of individual parameters for the dataset: the number of generated clusters in case of the $K-modes$ algorithm and a minimum number of generated clusters and in case of the $ROCK$ algorithm the estimated number of neighbors between objects in the clusters. Implementing the $ROCK$ algorithm became a tough challenge due to a very high computational complexity and unusual parameters. We selected the $ROCK$ algorithm parameters on a trial and error basis. While the $ROCK$ algorithm analyzes the similarities not only between objects but also between clusters that should be merged into a single cluster, the $K-modes$ algorithm arranges objects from a dataset between clusters so that each cluster contains a similar amount of data and focuses only on the similarities between individual objects in the data. As mentioned earlier, the definition of an outlier generated by the $ROCK$ algorithm, taken from [4] indicates one-element classes. The records marked as anomalies by the $K-modes$ algorithm are the records from the farthest neighborhood of the centroid in which cluster the object is located. All datasets used in this research were taken from the *UCI Machine Learning Repository* database and represent real data collected during research on real data objects with different distributions, possibly containing a small number of deviations, which results in significantly different sizes of clusters generated by the $ROCK$ algorithm. The results of the outlier detection analysis for the lymphography set are presented in Fig. 2.

Data clustering algorithms do not have a natural definition of outliers and do not return points considered as variances in the data. The problem of marking objects that differ the most from the others due to the calculations characteristic of the algorithm was solved by generating an additional column for the dataset containing the values $-1$ or $1$, where the value $-1$ means that the object was considered an outlier and $1$ means that the object is normal. In most cases, the analyzed algorithms returned completely



| | Class | lymphatics | block of affere | bl. of lymph. c | bl. of lymph. s | by pass | extravasates | regeneration of | early uptake in | lym.nodes dimin | lym.nodes enlar | changes in lym | defect in node | changes in node | changes in stru | special forms | dislocation of | exclusion of no | no. of nodes in | ROCK clusters | K-modes clusters | ROCK labels | K-modes labels |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 1 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 3 | 1 | 1 | 2 | 2 | 8 | 1 | 2 | 2 | 4 | 0 | 5 | -1 | -1 |
| 41 | 1 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 4 | 2 | 2 | 2 | 4 | 3 | 2 | 2 | 8 | 0 | 2 | -1 | -1 |
| 44 | 1 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 4 | 2 | 4 | 3 | 2 | 2 | 7 | 0 | 5 | -1 | -1 |
| 72 | 1 | 4 | 2 | 1 | 1 | 2 | 2 | 1 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 0 | 1 | -1 | -1 |
| 90 | 1 | 4 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 3 | 1 | 2 | 2 | 2 | 0 | 5 | -1 | -1 |
| 92 | 1 | 3 | 2 | 2 | 1 | 2 | 2 | 1 | 2 | 1 | 2 | 3 | 3 | 4 | 2 | 2 | 2 | 1 | 1 | 0 | 0 | -1 | -1 |
| 122 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 2 | 2 | 4 | 3 | 8 | 2 | 2 | 2 | 3 | 0 | 3 | -1 | -1 |
| 136 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 2 | 1 | 2 | 2 | 1 | 1 | 2 | 1 | 1 | 1 | 2 | 0 | 1 | -1 | -1 |

**Fig. 2.** The results of the outlier detection analysis for the lymphography set

different results. Large differences in outliers selection are the results of the different nature of those algorithms. The $ROCK$ algorithm is the most diligent in detecting outliers. It focuses on inter-object and inter-cluster connections, tying them together until well-defined clusters are obtained with the number of common neighbors below a certain threshold. Thus, single-member clusters contain far-away objects from every other cluster and every data object. In case of the $K - modes$ algorithm, due to randomness during the selection of an initial set of cluster centroids, outliers are considered as the objects whose distance from the centroids in the clusters they belong to, is the greatest. Due to a very different approach to determining good clusters and detecting outliers by these two algorithms, the anomaly classification result will also be different for each of the algorithms. We can design the anomaly search process in a qualitative set in two steps. Initially, all algorithms for the low anomaly threshold can search for common anomalies. If the process does not return results, you can increase the threshold and see if there are common outliers in the set this time.

### 5.3 Detection of Common Outliers

We should notice the relationship between the number of outliers and the degree of coverage of clustering algorithms in the context of outliers detection. Table 2 presents some interesting results. For each of the analyzed knowledge bases and the three analyzed levels of the number of outliers ($5\%$, $10\%$, and $15\%$, respectively), the table presents the number of clusters for each of the algorithms: $ROCK$ and $K - modes$, number of outliers detected by each of these algorithms separately, and then the number of common outliers detected by these algorithms and a percentage that these common outliers represent concerning the entire analyzed set. One of the more essential conclusions is that, the more outliers we look for ($5\%$, $10\%$, or $15\%$), by running each of the two analyzed algorithms separately, the more common outliers are found by these algorithms. For example, we found 3, 6, and 8 common outliers in the lymphography dataset, respectively, for the $5\%$, $10\%$, and $15\%$ outliers we searched. There are also interesting results in the *BM_attack* dataset. In regard to the number of outliers we searched for, the number of actually found outliers and common outliers did not change (2 common outliers no matter how many outliers we were looking for). It is worth looking at the structure of this data set. It contains the fewest attributes and possible values of these attributes when compared to the rest of the sets, which brings about difficulties with regards to distinguishing objects from each other and detecting a greater or lesser number of outliers. In general, when analyzing all sets, one can notice a specific influence the number of

attributes and their values have on the efficiency of outlier detection. The more attribute values there are, the greater the coverage of commonly detected outliers. This is easily explained. With a greater number of features describing the objects, we achieve a greater differentiation, so it is easier for us to correctly (not accidentally) determine the outliers.

**Table 2.** The results of % of common outliers obtained for 5%, 10%, and 15% of outliers in each of the datasets

| | | Clusters | | Outliers | | common | % of common |
|---|---|---|---|---|---|---|---|
| Dataset | % | ROCK | K-modes | ROCK | K-modes | outliers | outliers |
| lymphography | 5% | 2 | 5 | 5 | 5 | 3 | 0,020000 |
| | 10% | 3 | 5 | 16 | 11 | 6 | 0,040500 |
| | 15% | 3 | 5 | 16 | 21 | 8 | 0,054100 |
| covid | 5% | 3 | 6 | 7 | 6 | 1 | 0,004900 |
| | 10% | 5 | 6 | 34 | 25 | 13 | 0,063700 |
| | 15% | 5 | 6 | 34 | 25 | 13 | 0,063700 |
| SPECT | 5% | 6 | 4 | 8 | 10 | 4 | 0,014980 |
| | 10% | 1 | 4 | 23 | 21 | 10 | 0,037450 |
| | 15% | 6 | 4 | 50 | 47 | 31 | 0,116100 |
| BM_attack | 5% | 20 | 3 | 3 | 44 | 2 | 0,006200 |
| | 10% | 20 | 3 | 3 | 44 | 2 | 0,006200 |
| | 15% | 20 | 3 | 3 | 44 | 2 | 0,006200 |
| primary-tumor | 5% | 3 | 5 | 12 | 24 | 8 | 0,023599 |
| | 10% | 6 | 5 | 40 | 24 | 19 | 0,056000 |
| | 15% | 6 | 5 | 40 | 53 | 28 | 0,082596 |
| wiki | 5% | 2 | 9 | 28 | 24 | 13 | 0,014240 |
| | 10% | 1 | 9 | 78 | 96 | 44 | 0,048193 |
| | 15% | 5 | 9 | 143 | 143 | 74 | 0,081100 |

### 5.4 Evaluation of the Proposed Methods

As part of this work, a vast number of experiments were performed. We changed the values of individual parameters to observe changes in the cluster structure, the number of generated outliers, and most importantly, in assessing whether the analyzed clustering algorithms return similar results in terms of outliers. In the study, we considered real datasets which frequently contain unusual data. They are not the result of a measurement error, but they differ from most data in the set. It is not always the case that one or more objects stand out significantly from the rest, and we can easily see it. Sometimes, it is also the case that specific subsets of objects differ to the same extent from most of the data. The problem becomes even more complicated when we take into account the fact that these objects in the sets may be more or less differentiated by the specificity of the domain they come from, but also by the method of describing these data

(the number of attributes, the number of possible values of these attributes, the number of objects). When objects are described on a categorical (qualitative) scale, the effectiveness of their proper clustering and outlier detection is necessary for a deeper study. Hence, in this paper, we analyze selected clustering algorithms which exemplify two types of clustering: hierarchical ($ROCK$) and non-hierarchical ($K-modes$). Analysis of the results allows us to conclude that if we care about the speed of calculations or have a large dataset, a good choice will be to use the $K-modes$ algorithm. The algorithm is recommended to be used in datasets that we know are divided into a small number of large clusters. Then the initially drawn centroids will have less influence on clustering quality. In most cases, the most reasonable approach is to use the $ROCK$ method because it performs an exhaustive analysis of the dataset in search of outliers - it approaches object variables individually. It looks for relationships between objects and variables (attributes and their values). The main disadvantage of this algorithm is a very high computational complexity, which in extreme cases may be close to the cube of the number of objects in the set. For this reason, the algorithm is a good choice if we have small datasets, up to 1000 records. Another difficulty is the selection of the distance between the clusters and the minimum number of clusters. The algorithm execution time and clustering quality are improved by knowing an estimated number of clusters in the set and how far the elements should be apart from each other to not be included in a common cluster. Let us suppose that we do not have an exhaustive knowledge about the dataset. In that case, it is worth running the algorithm many times and analyzing the generated clusters to assess the quality of the parameters.

## 6 Conclusions

This paper focuses on searching for outliers in qualitative data sets depending on the type and the number of variables. Section 3 describes relatively novel approaches to qualitative clustering data. The results presented in this paper are based on six datasets characterized by a different structure. While there is a multitude of solutions related to quantitative data, clustering data containing only qualitative variables remains a challenge for data scientists. The authors attempted to compare the effectiveness of cluster and outlier detection in qualitative datasets, between which there is no explicit comparison so far. Algorithms based on quantitative data generally tend to have better mathematical properties. This does not apply to qualitative sets, so it is difficult to determine which algorithm works better on the data, and it is difficult to detect natural groups. We define the performance of algorithms in terms of their scalability and cluster generation time. We can draw a primary conclusion from the research that the data structure significantly impacts the algorithm's time complexity. The $K-modes$ algorithm defines clusters and outliers as objects far away from modes if we have visible modes in a data set. Otherwise, the optimal number of clusters can be very large or very small, and objects that should be in separate clusters will be in one due to a small distance from central modes. Then, it is better to use the $ROCK$ algorithm, which is less efficient and has a much greater computation complexity but is not sensitive to unusual data distribution. We should adequately select the algorithm for a dataset. Each algorithm classifies outliers differently and the results will differ. Algorithms based on categorical data clustering are relatively new methods of detecting outliers in data, having no

implementation in commonly used programming languages. The discussed $ROCK$ and *K-modes* algorithms introduce different methods to solve this problem and give different solutions in terms of their performance concerning the time needed to execute the algorithms when the number of records and dimensions change. The quality of the created clusters is measured by the user's knowledge and the examination of the results. The user sets basic parameters of clustering, which require an extensive knowledge of the data [1].

## References


1. Carletti, M., Terzi, M., Susto, G.A.: Interpretable anomaly detection with DIFFI: depth-based feature importance for the isolation forest, 1–12. IEEE, US (2000). arXiv preprint arXiv:2007.11117
2. Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 93–104 (2000)
3. Gibson, D. and Kleinberg, J. and Raghavan, P.: Clustering categorical data: an approach based on dynamical systems. In: Proceedings of the 24th International Conference on Very Large Data Bases, the VLDB Journal, pp. 222–236 (2000)
4. Guha, S., Rastogi, R., Shim, K.: ROCK: a robust custering algorithm for categorical attributes. Inf. Syst. **25**(5), 345–366 (2000)
5. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)
6. Jiang, M.F., Tseng, S.S., Su, C.M.: Two-phase clustering process for outliers detection. Pattern Recogn. Lett. **22**(6–7), 691–700 (2001)
7. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons (2005)
8. Liu, F.T., Ting, K.M., Zhou, Z.-H.: Isolation forest. In: IEEE International Conference on Data Mining, pp. 413–422 (2009)
9. Loureiro, A., Torgo L., Soares, C.: Outlier detection using clustering methods: a data cleaning application. In: Proceedings of KDNet Symposium on Knowledge-Based Systems for the Public Sector (2004)
10. Primary Tumor, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Primary+Tumor. Accessed 23 May 2020
11. Lymphography, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Lymphography. Accessed 23 May 2020
12. SPECT, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/spect+heart. Accessed 9 June 2021
13. COVID, UCI Machine Learning Repository. https://www.kaggle.com/anushiagrawal/effects-on-personality-due-to-covid19. Accessed 9 June 2021
14. Elbow method. https://en.wikipedia.org/wiki/Elbow_method_(clustering). Accessed 9 June 2021
15. Wiki, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/wiki4HE. Accessed 9 June 2021
16. BM_attack, UCI Machine Learning Repository. https://archive.ics.uci.edu/ml/datasets/Dishonest+Internet+users+Dataset. Accessed 9 June 2021
17. Sillhouette method. https://en.wikipedia.org/wiki/Silhouette_(clustering). Accessed 9 June 2021

# Augmenting Graph Inductive Learning Model with Topographical Features

Kalyani Selvarajah(✉) and Jae Muzzin

School of Computer Science, University of Windsor, Windsor, ON, Canada
{kalyanis,muzzin4}@uwindsor.ca

**Abstract.** Knowledge Graph (KG) completion aims to find the missing entities or relationships in a knowledge graph. Although many approaches have been proposed to construct complete KGs, graph embedding methods have recently gained massive attention. These methods performed well in transductive settings, where the entire collection of entities must be known during training. However, it is still unclear how effectively the embedding methods capture the relational semantics when new entities are added to KGs over time. This paper proposes a method, AGIL, for learning relational semantics in knowledge graphs to address this issue. Given a pair of nodes in a knowledge graph, our proposed method extracts a subgraph that contains common neighbors of the two nodes. The subgraph nodes are then labeled based on their distance from the two input nodes. Some heuristic features are computed and given along with the adjacency matrix of the subgraph as input to a graph neural network. The GNN predicts the likelihood of a relationship between the two nodes. We conducted experiments on five real datasets to demonstrate the effectiveness of the proposed framework. The AGIL in relation prediction outperforms the baselines both in the inductive and transductive setting.

**Keywords:** Knowledge graphs · Graph neural networks · Subgraph

## 1 Introduction

In recent years, significant progress has been made in the construction and deployment of knowledge graphs (KGs) [29]. KGs represent structured relational information in the form of subject-predicate-object (SPO) triples, e.g.,$\langle Justin\ Trudeau, fatherOf, XavierJames\rangle$. Freebase [4], YAGO [23], DBPedia [1], ConceptNet [22], and Never-ending language learning (NELL) [6] are a few prominent examples of large KGs. Recently, KGs have gained widespread attention because of their benefits in a variety of applications, including question answering [14], dialogue generation [13], information retrieval [30], entity linking [10] and recommendation systems [37].

Despite their usefulness and popularity, KGs are often noisy and incomplete because it is challenging to incorporate all information in the real world, and these data are typically dynamic and evolving, making it difficult to generate accurate and complete KGs [28]. Therefore, automating the construction


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 728–741, 2022.
https://doi.org/10.1007/978-3-031-08757-8_60

of a complete KG is a tedious process. Various techniques have been proposed for knowledge graph completion, such as the traditional Statistical Relational Learning (SRL) methods and Knowledge graph embedding methods. Building a complete KG is possible by predicting objects (known as link prediction) and relations.

A relation or logical induction prediction problem discovers probabilistic logical rules from a given KG. Induction can be learned in several ways such as from examples [16] and from interpretations [7]. For example, let's say, "the 23rd prime minister of Canada, Justin Trudeau lives in Ottawa, and is married to Sophie Trudeau." The first-order logic of the above sentence would be $LivesIn(Justin\ Trudeau, Ottawa) \wedge MarriedTo(JustinTrudeau, SophieTrudeau)$. Therefore, a logical rule can be derived based on the concept that a married couple lives together (generally); $LivesIn(X, Y) \wedge MarriedTo(X, Z) \rightarrow LivesIn(Z, Y)$. This rule can be used to find the relation or possible hypothesis $LivesIn(SophieTrudeau, Ottawa)$. Here, the known logical rules have been generalized to derive a new rule or relationship which is true most of the time. Additionally, this rule predicts the relation for the entities which did not exist when KGs were trained. In reality, KGs evolve with time and new entities will join. Most of the existing embedding-based methods are highly successful in predicting the relations if the entities were seen when KGs were trained, which is transductive reasoning. Generalizing relational semantics is a challenging task and important to see the relationships in unseen entities, which is inductive reasoning. However, these embedding methods have some limitations in explicitly capturing the relational semantics when new entities are added to KGs over time.

This paper proposes an Augmenting Graph Inductive Learning (AGIL) framework to learn relational semantics in a given KG, and predict relations $(s, ?, o)$. Since much of the existing machine learning methods suffer from scalability issues, recently, PLACN [17] and GraIL [25] applied subgraph-based methods in link prediction and relation prediction, respectively, to overcome this problem. GraIL used a Graph Neural Network (GNN) based relations prediction method to learn relational semantics even if the entities were unseen during training. However, GraIL operated strictly on subgraphs and utilized no additional information. PLACN, on the other hand, successfully used local features as additional information for link prediction. So, our proposed model exploits both PLACN and GraIL to derive AGIL, which includes three primary steps. First, the subgraph is extracted with common neighbors of the target link between nodes $i$ and $j$. The common neighbors in the enclosed subgraph are collected till $k$ number of hops. Then the subgraph is labeled using the Double-Radius Node Labeling [35] method. In the final steps, the heuristic features of nodes for the entire subgraph are extracted and fed into GNN along with the adjacency matrix of the subgraph, which aggregates the feature vectors into a scoring function for the prediction.

**Our Contribution:** The followings are the summary of our contributions:

1. We propose an Augmenting Graph Inductive Learning (AGIL) framework based on common neighbors-based subgraphs for relations prediction in both transductive and inductive settings.
2. We extract heuristic features of nodes from the entire subgraph and model a new prediction framework based on Graph Neural Networks (GNN);

The rest of the paper is organized as follows. Section 2 discusses related existing work. Our framework is presented in Sect. 3. Following that, Sect. 4 presents the experimental setup and the corresponding results. Finally, Sect. 5 concludes the research idea of this paper with directions for future work.

## 2 Related Work

Multiple methods have been proposed to construct a complete knowledge graph. Graph Embedding is one of the most broadly used solutions for Knowledge-Graph Completion challenges. Translation-based approach [5,5,24] Bilinear-based approach [27,33] and Neural-Network-based approach [2,8] are well-known graph embedding approaches.

Traditional approaches on the KG embedding methods are in a transductive manner. They require all entities during training. However, many real-world KGs are ever-evolving by adding new entities and relationships. Several inductive KG embedding approaches are proposed to address the issue of emergent entities. Graph2Gauss [3] is an approach to generalize to unseen nodes efficiently on large-scale attributed graphs using node features. Then Hamilton et.al. [11] proposed a generic inductive framework, GraphSAGE, that efficiently generates node embeddings for previously unseen data in a graph by leveraging node feature information. Node features are, however, not available in many KGs. In addition to these inductive embedding methods, DRUM [18], NeuralLP [34], and RuleN [15] are few models which learn logical rule and predict relations in KGs.

Recently, GraIL [25] was proposed to generalize inductive relation based on subgraph reasoning. Since GraIL shows comparatively better performance than the state-of-the-art methods, we consider extending it. Additionally, SEAL [35], PLACN [17] and DLP-LES [21] are few recent approaches that successfully extracted subgraphs from a given networks and applied heuristic features to train the model. Motivated by their high performance, we incorporate these heuristic features with our model.

## 3 Problem Definition and Proposed Approach:

Given a KG, $G = \langle V, E, R \rangle$ is a directed graph, where $V$ is the set of vertices, $E$ is the set of edges and $R$ represents the set of relations. The edges in $E$ connect two vertices to form triplets $(h, r, t)$, where $h$ is a head entity in $V$, $t$ is a tail entity in $E$ and $r$ is a relation in $R$, i.e., $E = \{(h, r, t)|h \in V, r \in R, t \in V\}$.

In a given KG, there is a high chance of missing relations $(h, ?, t)$, head entity $(?, r, t)$ and tail entity $(h, r, ?)$. Knowledge graph completion in a given KG, $G$ is defined as the task of predicting missing triplets, $E' = \{(h, r, t)|h \in V, r \in R, t \in V, (h, r, t) \notin E\}$ in both transductive and inductive settings. In the transductive setting, the entities in a test triple are considered to be in the set of training entities. Predicting missing triplets in the transductive setting is defined as $E'' = \{(h, r, t)|h \in V, r \in R, t \in V, (h, r, t) \notin E\}$. In the inductive setting, the entities in a test triple are never seen in the set of training entities. Predicting missing triplets in the inductive setting is defined as $E''' = \{(h, r, t)|h \in V' \text{ or } t \in V', r \in R, (h, r, t) \notin E\}$, where $V' \cap V = \emptyset$ and $V' \neq \emptyset$.

Our primary objective is to predict the relation between two nodes. We employ Graph Neural Network (GNN) [19] to learn the knowledge graph's structural semantics. The proposed model has the following steps;

1. Subgraph extraction.
2. Node labeling.
3. Feature matrix construction.
4. Scoring the subgraph using GNN.

### 3.1 Subgraph Extraction

For each triple in the knowledge graph, the subgraph is extracted with the goal of isolating the connecting nodes between the two target nodes $u$ and $v$. We wish to isolate only the nodes which are found along every possible path between the head and tail of the knowledge triple, referred to as the target nodes of the subgraph. A few approaches in the existing literature [17,35] have been proposed for subgraph extraction from a given graph. AGIL extracts subgraphs using common neighbors of any targeted nodes $u$ and $v$ because sufficient information of entire nodes of subgraphs can be taken for the training process [17]. Moreover, having additional information about the shared neighbours of both nodes $u$ and $v$ allows to determine the future existence of a relationship between them. We set a number $k$ for the number of hops to collect the nodes in the subgraph, which can be defined as given below.

**Definition 1.** ***Subgraph:*** *For a given knowledge graph $G = \langle V, E, R \rangle$, let $\pi_k(x)$ be the neighbors of $x$ within $k$ hops. The subgraph of a target link between nodes $u$ and $v$ is given by the function $S : V^2 \rightarrow 2^V$, the function that returns the set of common neighbor nodes connecting $u$ and $v$,*

$$S = \bigcup_{i=1}^{k} (\pi_i(u) \cap \pi_i(v)); \ for\ some\ m > 1 \tag{1}$$

*where $\{u, v\} \in V'$, $V' \subseteq V$ and $V'$ is a set of common neighbors for the targeted nodes, and $|V'| = \emptyset$.*

### 3.2 Subgraph Node Labeling

Generally, GNN takes both feature matrix $X$ and adjacency matrix $A$ as input, $(A, X)$. To construct a feature matrix $X$ of a subgraph, the position of nodes are really important to maintain the consistency of the structural information. GNN learns the existence of target links for prediction. So, we exploit the Double-Radius Node Labeling method, which was proposed by SEAL [35] to label the subgraphs.

Each label is a 2-tuple. The first element is the distance from the first target node, the second element is the distance from the second. The target node labels are always $(0, 1)$ and $(1, 0)$. Figure 1 is an example of a subgraph for target nodes $\langle University, ?, ComputerScience \rangle$, and the labeled subgraph.

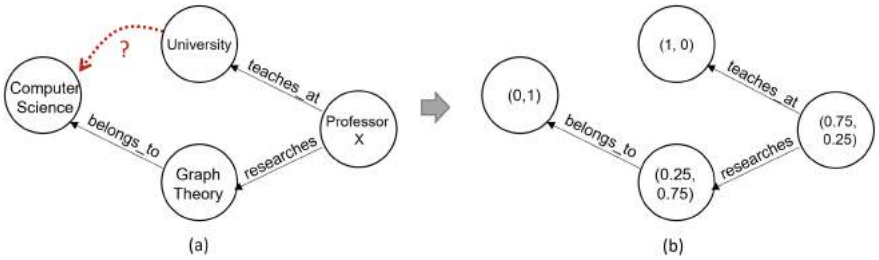



**Fig. 1.** (a) Subgraph of target nodes. (b)Labeled Subgraph.

### 3.3 Feature Matrix Construction

The GraIL [25] graph neural network architecture considers only the structural node feature $X$ for predicting triplets of a given KG. However, we believe that in addition to the structural node feature, incorporating explicit features of the subgraph to the feature matrix $X$ send additional information to the graph neural network training model. Since a knowledge graph does not always have any explicit feature information about a node, we decided to use the topological feature of the subgraph to see the importance of topological features in relation prediction. We belive that topological heuristics are useful in knowledge graph completion because entities are less likely to form relationships with entities that are farther away. Similarly with social networks, people tend to create new relationships with people that are closer to them. The motivation to apply topological heuristics to knowledge graphs was inspired by research in social networks. Our model uses the proximity measures taken from the topology as a heuristic in link prediction.

The specific heuristics used in this research were chosen to give a precise measurement of the notion of proximity of entities within the knowledge graph. In research done by Liben-Nowell and Kleinberg [12] with neighbour-based proximity measures, it was found that predictions outperformed chance by factors of

40 to 50, which led the authors to concluded that topology does indeed contain latent information which can be used to predict missing or future relationships.

We chose to use multiple proximity measurements as each has their own characteristics. AGIL uses the following five simple heuristics as used in PLACN model. Here $\pi(v)$ and $\pi(u)$ specify the set of neighbors within $k$ hops for nodes $v$ and $u$ respectively.

**Common Neighbors (CN)** counts how many neighbours any two vertices $v$ and $u$ have in common.

$$CN_{u,v} = |\pi_k(v) \cap \pi_k(u)| \tag{2}$$

**Jaccard Coefficient (JC)** produces the normalized form of CN.

$$JC_{u,v} = \frac{|\pi_k(v) \cap \pi_k(u)|}{|\pi_k(v) \cup \pi_k(u)|} \tag{3}$$

**Adamic-Adar (AA)** is a modified version of JC, which gives a higher priority to the common neighbors with lower degree.

$$AA_{u,v} = \sum_{w \in |\Gamma_k(v) \cup \Gamma_k(u)|} \frac{1}{log|\pi_k(w)|} \tag{4}$$

**Preferential Attachment (PA)** The idea behind PA is that a node with a higher degree has a better probability of forming new connections.

$$PA_{u,v} = |\pi_k(v).\pi_k(u)| \tag{5}$$

**Resource Allocation (RA)** is much more similar to AA, but gives higher priority to low-degree common neighbors.

$$RA_{u,v} = \sum_{w \in |\Gamma_k(i) \cup \Gamma_k(j)|} \frac{1}{|\pi_k(w)|} \tag{6}$$

Let $f : V^2 \rightarrow R^5$ be the function which returns the set of above five heuristic features for the pair of nodes $u, v$. So, $f(u, v)$ returns a vector of five components, each holding the CN, JC, AA, PA, and RA Value. Let $S$ be the set of nodes in the enclosing subgraph of target nodes $u$ and $v$, and $\{u, v\} \in S$ and $S \subseteq V$. Then for every node $i \in S$, we can evaluate the heuristic features of $x$ and each of $x$'s neighbors $y \in \pi(x)$ using $f(x, y)$.

Here, we discuss how we calculate the feature vector of node $x$. Let $P_x$ be the matrix whose columns are label of nodes in $S$ and rows are five feature vectors, i.e., $P_x = [R_0, R_1, \ldots, R_n]$, where $R_y = f(x, y)\ \forall y \in S$ and $n = |S|$. The matrix $P_x$ contains all five heuristic features of every possible pair of nodes in the subgraph. The number of rows and columns of $P_x$ are 5 and $|S|$ respectively.

**Example 1.** *Consider a subgraph* $S = \{u, v, w, x, y, z\}$*, and* $u$ *and* $v$ *are the target nodes of the subgraph* $S$*, then*

$$P_x = \begin{bmatrix} CN_u & CN_v & CN_w & 0 & CN_y & CN_z \\ AA_u & AA_v & AA_w & 0 & AA_y & AA_z \\ JC_u & JC_v & JC_w & 0 & JC_y & JC_z \\ RA_u & RA_v & RA_w & 0 & RA_y & RA_z \\ PA_u & PA_v & PA_w & 0 & PA_y & PA_z \end{bmatrix}_{5\times n}$$

*where the column for* $x$ *is zero, because it is pointless to compare* $x$*'s topology to itself.*

Since the size of $P_x$ depends on the size of the subgraph, $|S|$, which is variable, this is not suitable for scaling in training on when the node degree of the graph is very high. A very large feature vector can cause critical performance issues in the model. This is where the Fixed Sized Subgraph and Variable Sized Subgraph models diverge. Each take a different approach in deriving a feature vector $F_x$ from the topology matrix $P_x$.

PLACN uses a constant value $k$, which is the absolute maximum size a subgraph may reach. $k$ is derived in a way to be large enough for most node pairs. The value of $k$ is a function of the number of edges and nodes in the complete graph.

$$k \approx \left\lceil \frac{2|E|}{|V|} \left(1 + \frac{2|E|}{|V|(|V|-1)}\right) \right\rceil \tag{7}$$

The theoretical analysis of GraIL determined that any logical rule $R$ derived from the topology of a knowledge graph uniquely corresponds to a set of nodes connected through a sequence of relations, and that GraIL can learn this rule if the nodes and relations are present in the graph neural network.

To examine the differences between fixed and variable size sub graph, we constructed our feature matrix and sent it to GNN.

**Topology Information in Variable Sized Subgraphs:** In order to have a feature vector of constant size, we take a statistical analysis of each heuristic feature, across all of the nodes in the subgraph. For each heuristic feature $R \in R^5$, we can take the mean, median, standard deviation, minimum, maximum and variance across all of the nodes in the subgraph. In other words, we can apply the statistical functions to the rows of the topology matrix $P_x$.

Let $F_x = Stat(P_x)$, where $Stat(P_x)$ replaces each row of $P_x$ with $\langle Mean(r), Median(r), Variance(r), Min(r), Max(r), Std(r)\rangle$, So the resulting feature matrix has 30 elements,

$$F_x = \begin{bmatrix} Mean(CN) & Med(CN) & Var(CN) & Min(CN) & Max(CN) & STD(CN) \\ Mean(AA) & Med(AA) & Var(AA) & Min(AA) & Max(AA) & STD(AA) \\ Mean(JC) & Med(JC) & Var(JC) & Min(JC) & Max(JC) & STD(JC) \\ Mean(RA) & Med(RA) & Var(RA) & Min(RA) & Max(RA) & STD(RA) \\ Mean(PA) & Med(PA) & Var(PA) & Min(PA) & Max(PA) & STD(PA) \end{bmatrix}_{5\times 6}$$

Therefore, $F_x$ sees the rows of $P_x$ replaced by the statistical results which are rows of fixed size 5, since we consider five heuristic values and columns of fixed size 6 since there are 6 statistical functions. $F_x$ will always have a total of 30 elements, suitable to be encoded into a node $x$'s feature vector for training in the Graph Neural Network. We simply list all 30 elements as components of the final feature vector.

**Topology Information in Fixed Sized Subgraphs:** As PLACN used in it's architecture, the topological feature matrix $P_x$ of subgraphs is fixed for a given KG. The columns correspond to the fixed subgraph size and the rows correspond to each heuristic function. Therefore the size of $P_x$ is always $5 \times |S|$. Thus, for fixed sized subgraphs, we can directly encode $P_x$,

$$F_x = P_x \tag{8}$$

In practice, this has led to very large vectors, when the fixed size of the subgraphs is large.

### 3.4 Scoring Subgraph Using GNN

This section explains the importance of GNN in our framework.

**GNN Message Passing:** In a GNN, a hidden embedding $h_u^k$ for each node $u \in V$ is updated on each message-passing iteration based on information gathered from $u$'s graph neighbor $\pi(u)$. In other terms, the representation of the node $u$ is iteratively updated by aggregating its neighbors' representations [32]. So basically, GNN works based on two functions: Aggregation function passes information from $\pi(u)$ to $u$, and update function update features of $u$ based on the information to form an embedded representation.

In AGIL model, each enclosed subgraph has a network of $k$-hop neighborhood nodes. So, after aggregating for $k$ iteration, the $k$th layer of GNN is represented as,

$$m_u^k = \text{AGGREGATE}^k(\{h_v^{k-1} : v \in \pi(u)\} \tag{9}$$

$$h_u^k = \text{UPDATE}^k(h_u^{k-1}, m_u^k) \tag{10}$$

where $h_u^k$ is the feature vector of node $u$ at $k$th iteration, Initially, $h_u^0 = X_u$, and $m_u^k$ is the message aggregated from $\pi(u)$ at $k$th iteration.

In Eq. 9, there are various approaches proposed for message AGGREGATE function. Motivated by these architectures, GraIL adopts the method proposed by [20]. The following function defines the message aggregated function in a relational multi-graph:

$$h_u^k = \sigma\left(\sum_{r \in R} \sum_{v \in \Gamma_i^r} \rho_{rr_uvu} W_r^{k-1} h_v^{k-1} + W_0^{k-1} h_u^{k-1}\right) \tag{11}$$

where $k$ is the current layer of the neural network, $u$ is the node being aggregated, $R$ is the set of relationship types, $\rho_{rr_uvu}$ is the attention value for layer $k$, $r$ is

**Table 1.** The statistical information of datasets for inductive setting, where R, V and E are relations, vertices and edges respectively.

| | | WN18RR | | | FB15k-237 | | | NELL-995 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | # R | # V | # E | # R | # V | # E | # R | # V | # E |
| v1 | Train | 9 | 2746 | 6678 | 183 | 2000 | 5226 | 14 | 10915 | 5540 |
| | ind-test | 9 | 992 | 1991 | 146 | 1500 | 2404 | 14 | 225 | 1034 |
| v2 | Train | 10 | 6954 | 18968 | 203 | 3000 | 12085 | 88 | 2564 | 10109 |
| | ind-test | 10 | 2923 | 4863 | 176 | 2000 | 5092 | 79 | 4937 | 5521 |
| v3 | Train | 11 | 12078 | 32150 | 218 | 4000 | 22394 | 142 | 4647 | 20117 |
| | ind-test | 11 | 5084 | 7470 | 187 | 3000 | 9137 | 122 | 4921 | 9668 |
| v4 | Train | 9 | 3861 | 9842 | 222 | 5000 | 33916 | 77 | 2092 | 9289 |
| | ind-test | 9 | 7208 | 15157 | 204 | 3500 | 14554 | 61 | 3294 | 8520 |

any relationship , $r_t$ is a target relationship between nodes $v$ and $u$, $W_r^k$ is the transformation matrix for $r$ and layer $k$, and $h_v^k$ is the feature vector of the node $v$.

The GNN uses an aggregation function to distribute features of nodes into their neighbors, for each layer of the neural network. The aggregation function used by GraIL uses the node's labels as the feature vector. We append the elements of the feature matrix $F_x$ to the $h$ vector used in formulation 11. The feature vector $h$ in the GraIL model uses only the node labels $(L1, L2)$, for example $(1, 0)$, or $(25.75)$. However, the node structured information (i.e., node labels) are limited information for training GNN. Therefore, we extend the feature vector with the 30 elements from $F_x$. So, in AGIL model, the feature vector $h$ for node $x$ would incorporate the statistical analysis of heuristic features as below;
$\langle L1, L2, Mean(CN), Med(CN), Var(CN), Min(CN), Max(CN), STD(CN), Mean(AA), Med(AA), Var(AA), Min(AA), Max(AA), STD(AA), Mean(JC), Med(JC), Var(JC), Min(JC), Max(JC), STD(JC), Mean(RA), Med(RA), Var(RA), Min(RA), Max(RA), STD(RA), Mean(PA), Med(PA), Var(PA), Min(PA), Max(PA), STD(PA)\rangle$.

At each layer, the graph neural network continuously combines feature vectors of nodes with the aggregates of their 1-hop neighborhoods.

## 4 Experiments

We perform experiments to demonstrate the efficiency and effectiveness of our framework, AGIL. Experiments are carried out on benchmark datasets, WN18RR [9], FB15k-237 [26], and NELL-995 [31] which were originally developed for transductive settings. To conduct inductive relation prediction, we use 4 versions of inductive datasets and 2 versions of transductive datasets, which are prepared by the GraIL authors and identical to the data used in their experiments. They constructed fully-inductive benchmark datasets by sampling

disjoint subgraphs from the KGs. These datasets consist of two set of graphs: Train-graph and Ind-test-graph. Table 1 represents the statistical information on how benchmark datasets are split for inductive setting.

All the experiments are performed on a Intel(R) Core(TM) i7-3770 CPU computer @3.40 GHZ speed and 24 GB of RAM.

**Table 2.** Inductive Setting Experimental Result (AUC-PR)

| | WN18RR | | | | FB15k-237 | | | | NELL-995 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | v1 | v2 | v3 | v4 | v1 | v2 | v3 | v4 | v1 | v2 | v3 | v4 |
| Neural-LP | 86.02 | 83.78 | 62.90 | 82.06 | 69.64 | 76.55 | 73.95 | 75.74 | 64.66 | 83.61 | 87.58 | 85.69 |
| DRUM | 86.02 | 84.05 | 63.20 | 82.06 | 69.71 | 76.44 | 74.03 | 76.20 | 59.86 | 83.99 | 87.71 | 85.94 |
| RuleN | 90.26 | 89.01 | 76.46 | 85.75 | 75.24 | 88.70 | 91.24 | 91.79 | 84.99 | 88.40 | 87.20 | 80.52 |
| GraIL | 94.32 | 94.18 | 85.80 | 92.72 | 84.69 | 90.57 | 91.68 | **94.46** | 86.05 | 92.62 | 93.34 | 87.50 |
| AGIL (F-Subgraph) | **96.38** | **95.77** | **89.28** | **95.66** | 73.5 | 84.56 | 76.4 | NA | 90.56 | 93.7 | 94.18 | NA |
| AGIL (V-Subgraph) | 94.76 | 94.92 | 86.46 | 93.65 | **87.42** | **91.20** | **93.44** | 93.52 | **91.21** | **96.84** | **97.04** | **95.42** |

### 4.1 Inductive Relation Prediction

We test our model, AGIL on inductive datasets to determine if it can generalise relations when the entities aren't visible during GNN training. AGIL is trained on Train-graph and tested on Ind-test-graph.

To evaluate the performance, we compare AGIL against the following state-of-the-art methods.

1. NeuralLP [34]: an end-to-end differentiable model for inductive relation prediction.
2. DRUM [18]: a scalable and differentiable approach for mining first-order logical rules from KG.
3. RuleN [15]: statistical rule mining method, and the current state-of-the-art in inductive relation prediction on KGs.
4. GraIL [25]: inductive relation prediction by subgraph reasoning, and highly similar to AGIL.

We use the original source code by the authors for the implementation of above methods, NeuralLP[1], DRUM[2], RuleN[3] and GraIL[4]. For AGIL framework, the implementation is built upon the Python code base provided by [25] in their GraIL implementation. It uses the Deep Graph Learning library to implement a graph neural network.

**Results and Discussion:** The performance of the experimental setup for AGIL is represented in Table 2 against baseline methods. The Precision Recall

[1] https://github.com/fanyangxyz/Neural-LP.
[2] https://github.com/alisadeghian/DRUM.
[3] https://web.informatik.uni-mannheim.de/RuleN/.
[4] https://github.com/kkteru/grail.

Area Under Curve (AUC-PR) is used to evaluate the model's accuracy. The AGIL model is tested based on fixed sized subgraph (F-Subgraph) as proposed in PLACN, and variable sized subgraph (V-Subgraph). We observed that the model with fixed sized subgraph fails to perform better in some dataset such as v4-FB15k-237 and v4-NELL-995. The poor performance might be due to the absence of critical nodes and relations in the subgraph with fixed size neighbours. But, AGIL model with variable sized subgraph outperforms most of the standard baseline methods. In the NELL-995 dataset, the improvement is most significant compared to the other datasets. In WN18RR dataset, AGIL performs significantly better when we use fixed sized subgraph extraction. This indicates that for any knowledge graph of realistic size, fixed sized subgraphs are not always suitable for Inductive Graph Neural Network models.

If $k$ value is sufficiently large enough, it may include all connecting paths. The recommended calculation to derive $k$ by PLACN was shown to be too low for certain data sets, such as FB15k-237. If a knowledge graph has a high number of cycles, there may be many alternative paths between target nodes. Due to the truncation of the subgraph size to $k$, only a subset of possible paths will be analyzed by the neural network. Therefore, only a subset of the possible inductive rules will be learned by the GNN. When those inductive rules are applied to link prediction, they fail to produce accurate results. Both AGIL and GraIL provide a limiting factor to prevent excessively large subgraphs. It limits the number of hops from each node to a maximum, in all experiments, this maximum was 3 hops.

Moreover, GraIL outperforms on v4 of FB15k-237. However, the performance of AGIL is still close to GraIL on this dataset.

### 4.2 Transductive Relation Prediction

Most existing embedding based KG completion methods consider transductive setting for the prediction. Basically, all the existing KGs including WN18RR, FB15k-237, and NELL-995 are originally developed for the transductive setting. We test AGIL on transductive setting to determine it can predict the links accurately. We then compare AGIL against GraIL and RuleN.

**Table 3.** Transductive Setting Experimental Result (AUC-PR)

| | WN18RR | | FB15k-237 | | NELL-995 | |
|---|---|---|---|---|---|---|
| | v1 | v2 | v1 | v2 | v1 | v2 |
| RuleN | 81.79 | 83.97 | 87.07 | 92.49 | 80.16 | 87.87 |
| GraIL | 89.00 | 90.66 | 88.97 | 93.78 | 83.95 | 92.73 |
| AGIL | **92.77** | **92.80** | **90.03** | **95.56** | **92.44** | **93.84** |

**Results and Discussion:** The experimental results on transductive setting is represented in Table 3, which compares AGIL with GraIL and state-of-the-art method RuleN. In all the cases, AGIL outperforms the other two methods.

For the time being, we could not compare AGIL with other embedded-based methods. We will compare this in the future.

During the experiments it was shown that use of feature vectors would cause the model to overfit the training data, and loose some generality when applied to the test triples. To resolve this, we utilized the NodeNorm function [36] to normalize the feature vector. This gives the effect of making each feature vector have the save variance. Zhou et.al [36] have observed that GNNs perform poorly when the variance of features of nodes is very high. The normalization replaces each component in the feature vector with the difference from the mean divided by the variance.
The code is available in the GitHub link:
https://anonymous.4open.science/r/agil2021/README.md.

## 5 Conclusions

This paper examines an augmenting graph inductive learning framework based on GNN, named AGIL. Since many real-world KGs evolve with time, training very large networks with GNN is a challenging task. Therefore, we used a common neighbor-based subgraph to solve the scalability issue. Although AGIL is highly similar to the recently proposed model GraIL, AGIL incorporates topological heuristic features as additional information when GNN trains. Experimentally, we can see that the additional feature information gives better accuracy in both transductive and inductive settings. We also proved experimentally that fixed-sized subgraphs are not always suitable for Inductive Graph Neural Network models. Overall, our model, AGIL, outperforms most of the baseline methods. In the future, we are planning to examine the importance of individual topological features for the relation prediction.

## References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
2. Balažević, I., Allen, C., Hospedales, T.M.: Hypernetwork knowledge graph embeddings. In: Tetko, I.V., Kůrková, V., Karpov, P., Theis, F. (eds.) ICANN 2019. LNCS, vol. 11731, pp. 553–565. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30493-5_52
3. Bojchevski, A., Günnemann, S.: Deep gaussian embedding of graphs: unsupervised inductive learning via ranking. arXiv preprint arXiv:1707.03815 (2017)
4. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250 (2008)
5. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems 26 (2013)

6. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
7. De Raedt, L., Džeroski, S.: First-order JK-clausal theories are PAC-learnable. Artif. Intell. **70**(1–2), 375–392 (1994)
8. Demir, C., Ngomo, A.-C.N.: Convolutional complex knowledge graph embeddings. In: Verborgh, R., Hose, K., Paulheim, H., Champin, P.-A., Maleshkova, M., Corcho, O., Ristoski, P., Alam, M. (eds.) ESWC 2021. LNCS, vol. 12731, pp. 409–424. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-77385-4_24
9. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2D knowledge graph embeddings. In: Thirty-second AAAI Conference on Artificial Intelligence (2018)
10. Hachey, B., Radford, W., Nothman, J., Honnibal, M., Curran, J.R.: Evaluating entity linking with wikipedia. Artif. Intell. **194**, 130–150 (2013)
11. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1025–1035 (2017)
12. Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. J. Am. Soc. Inf. Sci. Technol. **58**(7), 1019–1031 (2007)
13. Liu, S., Chen, H., Ren, Z., Feng, Y., Liu, Q., Yin, D.: Knowledge diffusion for neural dialogue generation. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1489–1498 (2018)
14. Lukovnikov, D., Fischer, A., Lehmann, J., Auer, S.: Neural network-based question answering over knowledge graphs on word and character level. In: Proceedings of the 26th International Conference on World Wide Web, pp. 1211–1220 (2017)
15. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 3–20. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_1
16. Muggleton, S.: Inductive logic programming. New Gene. Comput. **8**(4), 295–318 (1991)
17. Ragunathan, K., Selvarajah, K., Kobti, Z.: Link prediction by analyzing common neighbors based subgraphs using convolutional neural network. In: ECAI 2020, pp. 1906–1913. IOS Press (2020)
18. Sadeghian, A., Armandpour, M., Ding, P., Wang, D.Z.: Drum: end-to-end differentiable rule mining on knowledge graphs. arXiv preprint arXiv:1911.00055 (2019)
19. Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Trans. Neural Netw. **20**(1), 61–80 (2008)
20. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
21. Selvarajah, K., Ragunathan, K., Kobti, Z., Kargar, M.: Dynamic network link prediction by learning effective subgraphs using CNN-LSTM. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
22. Speer, R., Chin, J., Havasi, C.: Conceptnet 5.5: An open multilingual graph of general knowledge. In: Thirty-first AAAI Conference on Artificial Intelligence (2017)
23. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706 (2007)

24. Sun, Z., Deng, Z.H., Nie, J.Y., Tang, J.: Rotate: knowledge graph embedding by relational rotation in complex space. arXiv preprint arXiv:1902.10197 (2019)
25. Teru, K., Denis, E., Hamilton, W.: Inductive relation prediction by subgraph reasoning. In: International Conference on Machine Learning, pp. 9448–9457. PMLR (2020)
26. Toutanova, K., Chen, D., Pantel, P., Poon, H., Choudhury, P., Gamon, M.: Representing text for joint embedding of text and knowledge bases. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1499–1509 (2015)
27. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning, pp. 2071–2080. PMLR (2016)
28. Wang, M., Qiu, L., Wang, X.: A survey on knowledge graph embeddings for link prediction. Symmetry **13**(3), 485 (2021)
29. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
30. Xiong, C., Callan, J.: Esdrank: connecting query and documents through external semi-structured data. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 951–960 (2015)
31. Xiong, W., Hoang, T., Wang, W.Y.: Deeppath: a reinforcement learning method for knowledge graph reasoning. arXiv preprint arXiv:1707.06690 (2017)
32. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
33. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
34. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. arXiv preprint arXiv:1702.08367 (2017)
35. Zhang, M., Chen, Y.: Link prediction based on graph neural networks. Adv. Neural. Inf. Process. Syst. **31**, 5165–5175 (2018)
36. Zhou, K., Dong, Y., Wang, K., Lee, W.S., Hooi, B., Xu, H., Feng, J.: Understanding and resolving performance degradation in graph convolutional networks. arXiv preprint arXiv:2006.07107 (2020)
37. Zhu, F., Wang, Y., Chen, C., Liu, G., Orgun, M., Wu, J.: A deep framework for cross-domain and cross-system recommendations. arXiv preprint arXiv:2009.06215 (2020)

# Generative Networks Applied to Model Fluid Flows

Mustapha Jolaade(✉), Vinicius L. S. Silva, Claire E. Heaney, and Christopher C. Pain

Applied Modelling and Computation Group, Imperial College London, London, UK
moj20@ic.ac.uk

**Abstract.** The production of numerous high fidelity simulations has been a key aspect of research for many-query problems in fluid dynamics. The computational resources and time required to generate these simulations can be so large and impractical. With several successes of generative models, we explore the performance and powerful generative capabilities of both generative adversarial network (GAN) and adversarial autoencoder (AAE) to predict the evolution in time of a highly nonlinear fluid flow. These generative models are incorporated within a reduced-order model framework. The test case comprises two-dimensional Gaussian vortices governed by the time-dependent Navier-Stokes equation. We show that both the GAN and AAE are able to predict the evolution of the positions of the vortices forward in time, generating new samples that have never before been seen by the neural networks.

**Keywords:** Generative adversarial networks · Adversarial autoencoder · Two-dimensional turbulence · Spatial-temporal predictions · Deep learning

## 1 Introduction

The study of fluid dynamics has involved massive amounts of data generated either from controlled experiments, field measurements or large-scale numerical simulations. The high volume of data, amongst other reasons, means these methods can be relatively slow and require a great deal of computational power to be able to model the underlying physics. While advancements in high performance computing research has boosted speed and accuracy of numerical simulation, obstacles still remain [2]. Thus, the development of computational frameworks that are accurate, robust, cheap and fast enough to model fluid dynamics remains a key aspect of computational science and engineering research.

In this paper, generative models, a branch of machine learning, is applied to a two-dimensional turbulent fluid problem for the purposes of rapidly predicting forward in time while avoiding the high computational cost of traditional numerical methods.

Generative models have garnered a huge amount of interest in recent years [11]. The main idea behind generative models is to build a statistical model


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 742–755, 2022.
https://doi.org/10.1007/978-3-031-08757-8_61

around a given dataset that is capable of generating new sample instances that appear to be taken from the original dataset. These new samples can further be used for tackling problems related to the case under study. When the building process is based on deep networks (artificial neural networks such as convolutional neural networks (CNN) [12]) that use multiple layers to capture how patterns/features of the dataset are organised or clustered, the resulting model is termed a deep generative model. Once a deep generative model has learned the structure of the training dataset, by being fed a random vector as input, its networks can generate desired samples from complex probability distributions in high-dimensional spaces [8]. In building deep generative networks two main methods have been widely used. The first is a variational autoencoder that uses stochastic variational inference to minimize the lower bound of the data likelihood [11]. The second is a generative adversarial network (GAN) whereby two players (neural network) play a zero-sum game. The game seeks to minimize the distribution divergence between the model output and the real/training dataset by using real samples as a proxy for optimization. A novel third method born out of the amalgamation of these two methods is the use of adversarial autoencoder (AAE) [14]. In this project, attention is given to both GAN and AAE as data-driven methods for prediction and modelling of spatial-temporal turbulent fluid flow.

Although reduced order models have been used for time-dependent turbulent fluid modelling in areas such as subsurface flow [3] and for the solution of the Navier-stokes equation [19]. In this project, for the first time, we use generative models in a reduced-order model framework to carry out efficient predictions in time of a two-dimensional turbulent fluid flow problem.

The rest of this paper is structured as follows: the next section provides a description of the methodology adopted from [16] for spatial-temporal prediction with GANs. Here, we also include the methodology for prediction using the AAE. Section 3, introduces the test fluid system and a relevant discussion about the transformation carried out to make the data suitable for use. The obtained results from predicting single and multiple time levels are also presented in Sect. 3. Finally, conclusion and remarks about possible future work are provided in Sect. 4.

## 2 Methodology

The use of GANs for time series prediction and data assimilation of real world dynamical systems has been proven to be successful for the spatial-temporal spread of COVID-19 using SEIRS type models [15,16]. Particularly, the method in [16] has been shown to be independent of the underlying system, thus this project will apply the same method for the two-dimensional turbulent fluid model.

In this project, we start by building a reduced model of the turbulent fluid flow, going from a high-fidelity spatial domain to a lower dimensional representation. Then, a generative model is built and trained to learn a mapping between

a input latent vector and the lower dimensional representation. Finally, we apply the processes of simulating forward in time using the capabilities of the generative models. The aim is then for the generative networks to serve as surrogate models that can reproduce the high-fidelity numerical model.

### 2.1 POD-based Non-Intrusive Reduced Order Modelling

The connection between physics-based machine learning and dimensionality reduction has been substantially studied and well-documented [18]. Results of these studies have shown that many methods used to obtain a low-dimensional subspace of a system are related to machine learning methods. In modern computational research, Reduced Order Modelling (ROM) is a well-known technique for dimensionality reduction [3]. By constructing reduced-order models that encapsulate the original features of the fluid systems while maintaining its underlying physics, it is possible to seek solutions to a model in an efficient and much less expensive way [20]. The Non-Intrusive Reduced Order Modelling (NIROM) is a type of ROM so named due to its non-dependent on the system under study. This model reduction approach can use proper orthogonal decomposition (POD) [17] to derive a physics-inspired low-dimensional parameterization that represents the high dimension of the high-fidelity spatial domain of the fluid model (i.e. state of snapshots). POD is closely related to the principal component analysis (PCA) method in statistics and was first used for turbulent flows by [13].

In this project, the dimensionality reduction aspect of our methodology is set within a NIROM framework that involved computing the POD basis vectors (via PCA) using the snapshots of the input data [17].

Consider a three-dimensional field $\boldsymbol{\alpha}$, which is dependent on some input parameter and varies in space and time. We can define its function as $\boldsymbol{\alpha} : \mathcal{X} \times \mathcal{T} \times \partial \approx \mathbb{R}$ where $\mathcal{X}$ is the spatial domain, $\mathcal{T}$ is the time domain, and an input domain $\partial$ of initial parameters/condition. The aim of data-driven/non-intrusive dimensionality reduction is to find an approximate model for $\boldsymbol{\alpha}$ from the data

$$\mathcal{D} \parallel \{\Omega(\boldsymbol{x}, t, \boldsymbol{z}) \mid \boldsymbol{x} \in \mathcal{X}, t \in \mathcal{T}, \boldsymbol{z} \in \partial\} \tag{1}$$

which, in this case, are snapshots in time of the field. The desired approximate model of the field can be expressed as a linear expansion in the POD basis. This POD basis would be computed from many snapshots data developed as solutions of a high-fidelity model that describes the field. To compute the POD basis, we consider a snapshot data to be $\boldsymbol{\alpha}(t; \boldsymbol{z}) \in \mathbb{R}^{n_x}$ where $n_x$ is the dimension of the spatial domain (from finite discretization). Thus, the set $\{\boldsymbol{\alpha}(t_i; \boldsymbol{z}_j) \mid i = 1, \cdots, n_t;\ j = 1, \cdots, n_z\}$ of snapshots at $n_t$ different time levels/steps of $t_1,\ t_2,\ \cdots,\ t_{n_t} \in \mathcal{T}$ and $n_z$ different initial input conditions of $\boldsymbol{z_1}, \boldsymbol{z_2}, \cdots, \boldsymbol{z_{n_z}} \in \partial$ comprises of $n_s = n_t n_z$ snapshots. The snapshot matrix can be defined as $\boldsymbol{S} \in \mathbb{R}^{n_x \times n_s}$ with each row corresponding to a spatial location and each column representing a snapshot in the set. At this stage, PCA can then be introduced for dimensionality reduction.PCA seeks a transformation $\boldsymbol{T}$ that

maps each vector $\{\boldsymbol{\alpha}(t_i; \boldsymbol{z}_j)$ in $\boldsymbol{S}$ (i.e. each snapshot) from the original dimensional space of $n_x$ to a new space that only keeps the first $r$ principal components using the first $r$ eigenvectors of the transformed matrix [10].

The idea is to maximize the variance of the original data while minimising the total least squared errors in the representation of the snapshots. The size of the POD basis/principal component $r$ is chosen by specifying a tolerance in this error calculation. This user-specified tolerance, $k$ also indicates how much information/energy of the data is captured by the resulting snapshot representation. We chose $r$ such that:

$$\frac{\sum_{k=1}^{r} \pi_k^2}{\sum_{k=1}^{n_s} \pi_k^2} > k, k = 0.999 \tag{2}$$

This means given a snapshot field we can compute its original state, using the POD coefficients, with 99.9% reconstruction accuracy. Hence, once the dimension reduction is completed, the POD expansion coefficients $\boldsymbol{\beta_{k=1,\ldots,r}(t;\ z)}$ denote the model approximation and parameterization of a snapshot field $\boldsymbol{\alpha}(t;\ \boldsymbol{z})$ at time $t$ and input conditions $\boldsymbol{z}$. The coefficients are then employed in the training of generative models for the time series prediction. Results of the POD-based compression are shown and discussed in Sect. 3.2.

### 2.2 Generative Models

Generative modelling is the process of training a machine learning model with specific data to produce 'fake' data from a distribution that mimics the probability distribution of the original training set. Here, we produce two generative models to perform time series prediction of a turbulent fluid flow. The two models utilized are: a generative adversarial network (GAN) [7] and an adversarial autoencoder (AAE) [14]. The choice of these models was based on their proven successes in the use of nonlinear fluid modelling. In the result section, a comparison between outputs of the two models is presented.

**Generative Adversarial Network:** A GAN is an artificial learning technology that is composed of two neural networks as shown in Fig. 1. GANs have been adopted widely in several research areas, showing huge successes in practical applications including simulating fluid models [5]. The training process is essentially a game between two models competing as adversaries. While the generator module (G) generates fake samples from an input random distribution, a discriminator module (D) tries to distinguish between real samples drawn from the original distribution and the sample output from the generator. D does this by estimating a score which serves as the probability that a particular sample came from the original distribution i.e. $D(G(\boldsymbol{\beta_r})) = 1$. The training process of a GAN is a minimization-maximization problem that is based on a cross-entropy loss function

$$\boldsymbol{J}(D, G) : \min_G \max_D E_{\boldsymbol{\beta}_r \sim p_{data}(\boldsymbol{\beta}_r)}[logD(\theta_r)] + E_{\mathbf{z} \sim p_z(\mathbf{z})}[log(1 - D(G(\theta)))] \tag{3}$$

where $p_{data}(\theta_r)$ is the probability data distribution of the target output of real samples $\theta_r$ and $p_z(z)$ is the prior distribution for the random latent vector $\mathbf{z}$. The training process involves:

- Updating $D$ with gradients that maximize the discriminator function by differentiating with respect to parameters of the discriminator.
- Updating $G$ with gradients that minimize the generator function by differentiating with respect to parameters of the generator.

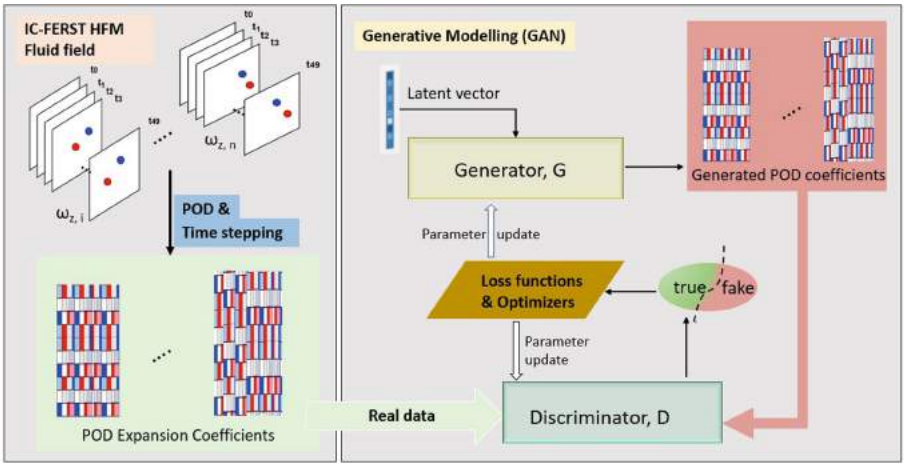



**Fig. 1.** Generative modelling using GAN. In this workflow, real samples obtained from POD-based NIROM are utilized as training data for the discriminator module of a GAN. Fake data produced by the generator, G from an input latent vector is simultaneously used in the training process, with loss back propagated through both neural network modules.

A common problem in the use of GANs for sample generation is mode collapse. Typically, a GAN is trained to produce a wide variety of outputs that mimic the training data distribution. For example, if a GAN is trained with pictures of different dog breeds, we want a different dog for every random input to the dog generator. However, it is possible that the generator only produces a small set of realistic outputs and learns to generate only that seemingly credible output (or small set of outputs) to the discriminator.

The Wasserstein GAN (WGAN) [1] is a type of GAN that avoids this problem of mode collapse by circumventing the issue of vanishing gradients. This implies that the discriminator is trained to optimality, learning to reject any output/set of outputs the generator tries to stabilize on. The WGAN method introduces a new loss function that alternatively minimizes an approximation of the Earth Mover distance between the distributions completely avoiding mode collapse.

In developing a GAN for the generative modelling of this project, a typical Deep Convolutional GAN (DCGAN) was developed and trained to produce the target output. Following evidence of mode collapse however, an Improved WGAN [9] was also developed by altering the loss functions of the original DCGAN. The WGAN also included a gradient penalty term that led to more diverse output from the generator.

The WGAN loss function uses a Earth mover distance criteria to enforce match of a prior data distribution. The loss function for this type of GAN is given as follows

$$L = \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_g} \left[ D\left(\tilde{\mathbf{x}}\right) \right] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}_r} \left[ D\left(\mathbf{x}\right) \right] + \lambda \mathbb{E}_{\hat{\mathbf{x}} \sim \mathbb{P}_{\hat{\mathbf{x}}}} \left( (||_{\sim \hat{\mathbf{x}}} D\left(\tilde{\mathbf{x}}\right) ||_2 - 1)^2 \right) \tag{4}$$

where the second term is a gradient penalty that replaces weight clipping to achieve Lipschitz continuity (gradient with norm at most 1 everywhere). The discriminator in this GAN works as a critic.

The generative model (GAN and/or WGAN) developed and trained using the presented workflow can be used for time-series/forward prediction without any changes to its structure. This is also the case when the model is utilized for the assimilation of given observation/sensor data [16].

**Adversarial Autoencoder:** A second type of generative model built and implemented in this project is an AAE (Fig. 2). Similar to a GAN, the AAE was proposed as a generative model that seeks to match an aggregated posterior distribution of its hidden latent vector with a prior distribution. To be able to function as a deep generative model, the AAE is trained to perform variational inference that enables its decoder to learn a statistical model that maps between the imposed prior and the data distribution. The AAE has a wide range of applications including semi-supervised classification, unsupervised clustering and data visualization [14]. In the field of computational fluid dynamics, Cheng et al. [4] studied the capability of an advanced deep-AAE for parameterizing nonlinear fluid flow and utilized it in the prediction of a water collapse test case. Here, we develop an AAE and test it for prediction of nonlinear turbulent flow.

### 2.3 Space-time Predictions Using Generative Models

The goal of this project is to show that generative models such as GANs and AAEs can be utilized for the time-series prediction of nonlinear turbulent fluid models. This section discusses the time-series prediction and an algorithm for its implementation. The methodology proposed by [16] is further tested on a two-dimensional turbulent fluid model to obtain a surrogate model that is accurate and computationally cheap. The next subsections discuss the this method and its components.

**Prediction Using GANs:** The ability of a GAN to produce realistic samples that seem to belong to a prior distribution is leveraged in this project. To

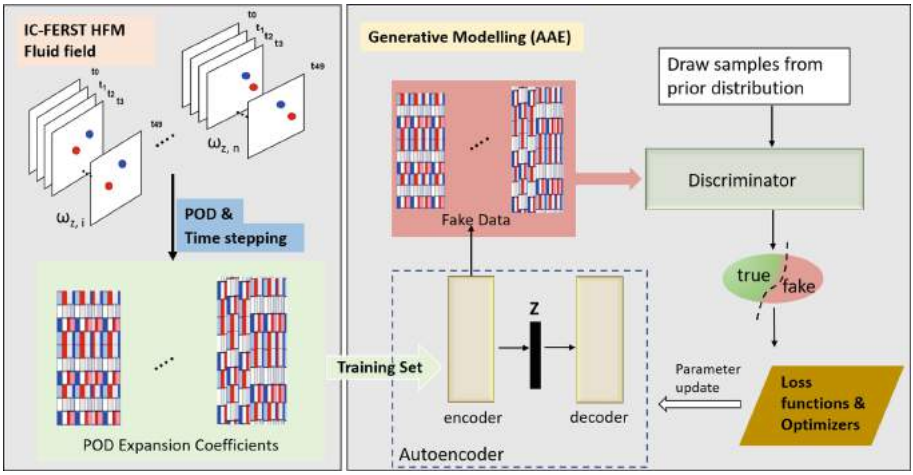



**Fig. 2.** Generative modelling using AAE. The training process of this workflow attempts to match output of the autoencoder with a prior distribution. While the encoder generates fake samples that matches this distribution, the discriminator attempts to critic against the generated samples.

predict forward in time, an algorithm, Predictive GAN (PredGAN) algorithm [16] is implemented in this project on two-dimensional turbulent flow data. The PredGAN algorithm begins with training a GAN to generate a data sequence of $p+1$ time levels from an input latent vector. To achieve this, the GAN is trained with $p+1$ consecutive time levels of compressed variables/POD coefficients concatenated to form a trajectory. Once the training is completed, the generator of the GAN is capable of producing fake snapshots at multiple time levels, $n-p$ to $n$ where $n \geq p$. In order to complete prediction with the trained GAN, the first $p$ time levels of a known trajectory/given solution is matched with corresponding time levels of the output of the GAN through loss optimization. Once convergence has been reached, the additional time step $p+1$, in the output of the generator serves as the forward prediction of the trajectory. This process can be repeated by using the predicted $p+1$ solution as a known solution while similar optimization is carried out to predict time level $p+2$. Ultimately, all time steps can be predicted by replicating the process and obtaining a new time step for each iteration of the PredGAN algorithm.

**Prediction Using Adversarial Autoencoder:** To predict with an autoencoder, the following steps were followed:

1. Since the autoencoder does not require a latent variable as input, we use the first $p-1$ time levels of the known solution as input.
2. To predict forward, the $p-1$ time level is used as an initial guess for the desired $p$ time level. and passed into the autoencoder to give a prediction.

Following successful training, the autoencoder then attempts to match the true snapshot at time $p$ from the input initial guess.
3. The output prediction from a single iteration through the autoencoder is further re-used as input guess for the time level $p$ and the time series is passed through the autoencoder till convergence is reached.
4. The final output is the snapshot prediction at time $p$.
5. For multiple time level predictions, the process is repeated from steps 1–4 using the last $p$ time levels as initial guesses for subsequent time levels.

## 3 Implementation and Results

### 3.1 Case Study: Parameter-varying Flow in a Periodic Box

In order to train a GAN capable of time-series prediction of the two-dimensional turbulent fluid problem, a dataset comprising two velocities component (x, y) and the pressure for each discretized node of a two-dimensional incompressible Navier-Stokes simulations has been obtained. Figure 3 shows the spatial properties of each snapshot. This dataset represents a parameter-varying flow in a fixed-wall box. Given that the convolutional layers of a neural network are designed to detect object/features anywhere in an image, it can be used in this project since the large-parameter variations implicit in the dataset generation is of a similar nature as object randomly located in an image [6].

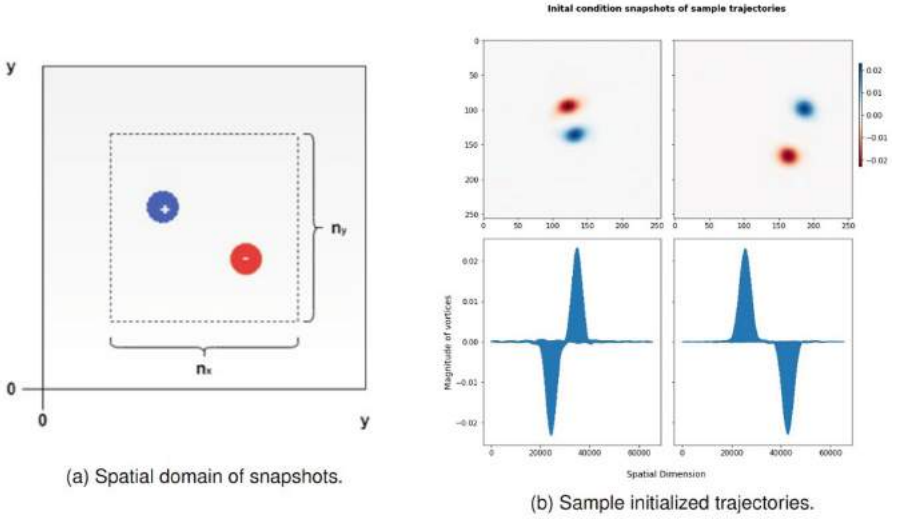



**Fig. 3.** Two-dimensional Gaussian vortices in a square domain. The positive and negative vortices are of equal strength and each snapshot $S \in \mathbb{R}^{y \times y}$ where $y = 256$, are randomly initialized within a predefined subdomain $n_x \times n_y$. The images on the bottom right show the magnitude of the vortices projected over a 1-D domain.

The simulations were run on Imperial College-Finite Element Reservoir Simulator (IC-FERST) using the following criteria: turbulent flow with $Re$=5000, constant viscosity and no slip walls boundary conditions. The first step in the project is to transform the velocity dataset into vorticity data so it represents initial Gaussian (randomly initialized) vortices that decay due to viscosity changes. Following this transformation, the vorticity data are then compressed by carrying out a POD.

The training set for this project included snapshots from 300 separate trajectories. Snapshots from trajectories were obtained such that each trajectory included 50 snapshots - a total of 15K snapshots. Prior to actual training, the data is prepared for time series prediction by concatenating successive snapshots, 5 s apart, into a time series of 7 instances (i.e. each time series represents vortices' evolution over a period of 30 s). This sums up to 6K distinct time series - one trajectory can be split into a maximum of 20 time series of 7 snapshots. In predicting with generative models post-training, we were able to forecast 1–3 additional instances (evolution over a period of $\leq$15 s) for never before seen time series (30 trajectories). See 3.3 for more details.

### 3.2 POD Compression and Order Reduction

Each snapshot is no longer a 256 by 256 array but now represented using 292 features (POD coefficients). The cumulative information/energy retained measured using explained variance is over 99.99% as shown in Fig. 4. A visual comparison of the compression is shown in Fig. 5.

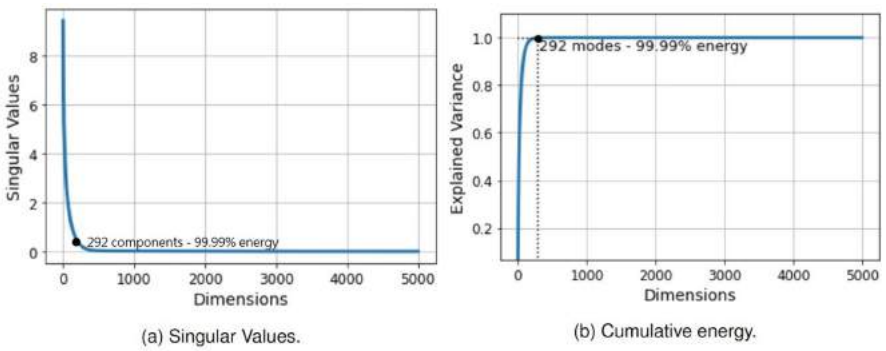



**Fig. 4.** POD singular values and relative cumulative energy for the two-dimensional Gaussian vorticity filed snapshot set.

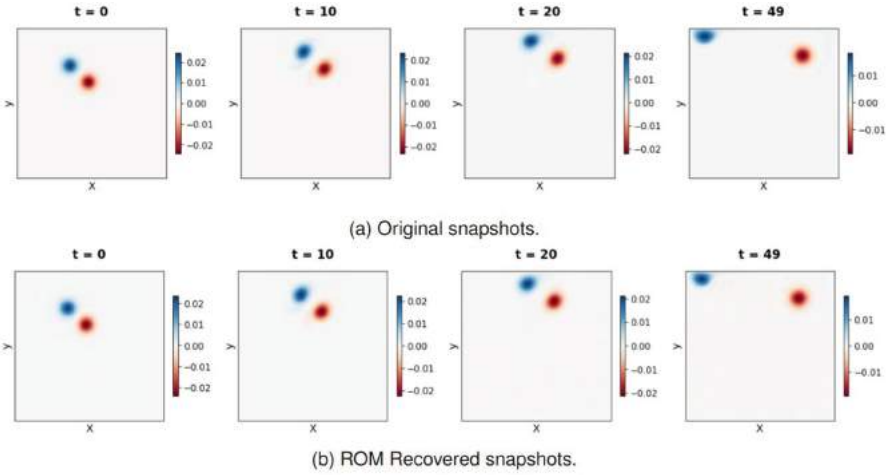



**Fig. 5.** Original and recovered snapshots following POD-based NIROM. The reduction was specified to retain 99.99% information from the original snapshots. The reduction decreased the dimension from 256-by-256 to 292 POD coefficients.

### 3.3 Prediction Using GAN and AAE

**Single Time Level Prediction:** In this section, we apply the PredGAN algorithm to a sample trajectory to predict a single time level forward. Following the training of a WGAN-GP and an AAE with 7 time levels from sample trajectories, we proceed to predict a single step forward in the test set using the PredGAN and PredAAE algorithms. In this application, the first 6 time levels (t=0 to t=25) are considered known while the 7th time level (t=30) is the predicted time step. Results of the single time level prediction can be found in Fig. 6. A sample trajectory (Fig. 6a) serves as the input to the generative models (WGAN-GP and AAE). Snapshots of each generative model shows output following convergence of the loss between generated sample and known solution (Fig. 6bc). The second row visualizes the mismatch between the magnitude and location of the true data (in blue) and generated prediction (in orange). The vertical axis represents the magnitude of the vortices while the horizontal axis is a one-dimensional projection of the two-dimensional domain. Given these results, AAE is shown to have a better performance both for predicting forward and matching known solutions with samples generated from a random input latent vector.

**Multiple Time Levels Prediction:** The results from predicting multiple time levels, shown in Fig. 7, follows a similar pattern as that of the single time level prediction. Following results for the single time level prediction (t=30), we proceed to predict multiple time levels from t=35 to t=45. It is worth mentioning that this data was not present in the training set. The first row of snapshots

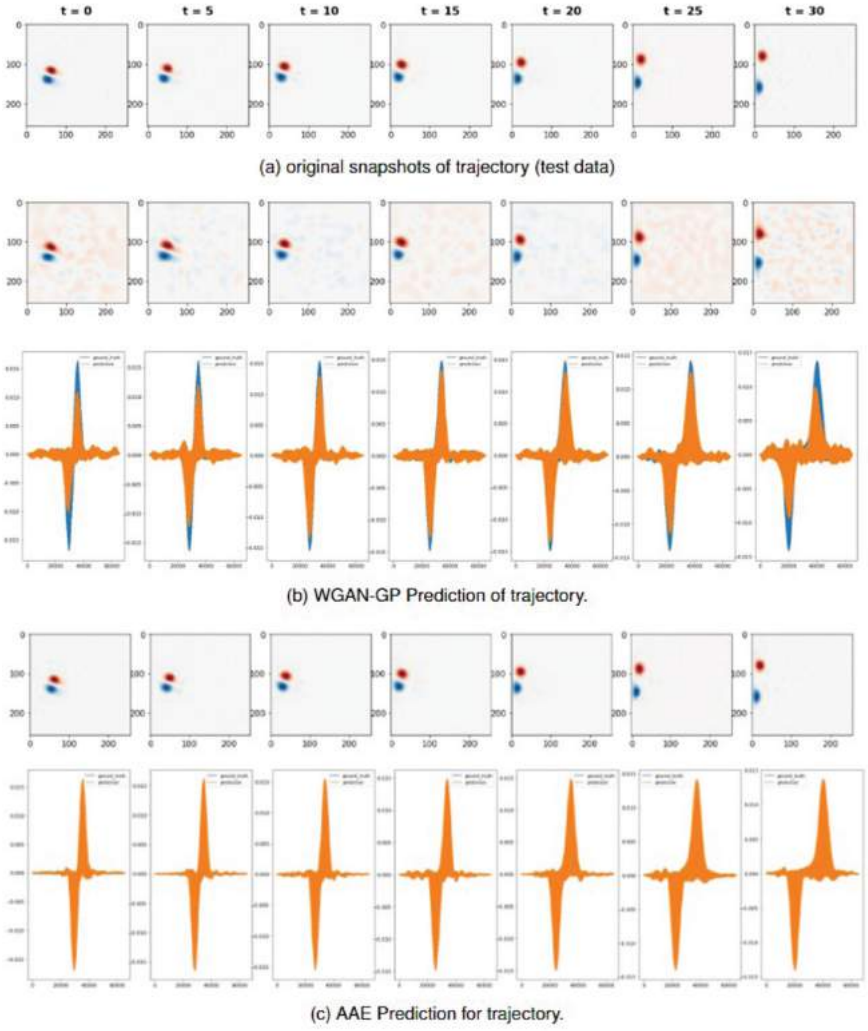



**Fig. 6.** Prediction of one time level (t=30) using WGAN-GP(b) and AAE(c) on a sample trajectory from train dataset.

(Fig. 7a.) shows the true snapshot of the trajectory at times t=35 to t=45. This is the known/given solution form the high fidelity simulation. Figure 7b. shows predicted output for these time levels using the same WGAN-GP. Here, we see that while the the WGAN-GP is able to predict the spatio-temporal distribution, the prediction ability reduces with forward time. The AAE (Fig. 7c), however, shows no such sign.

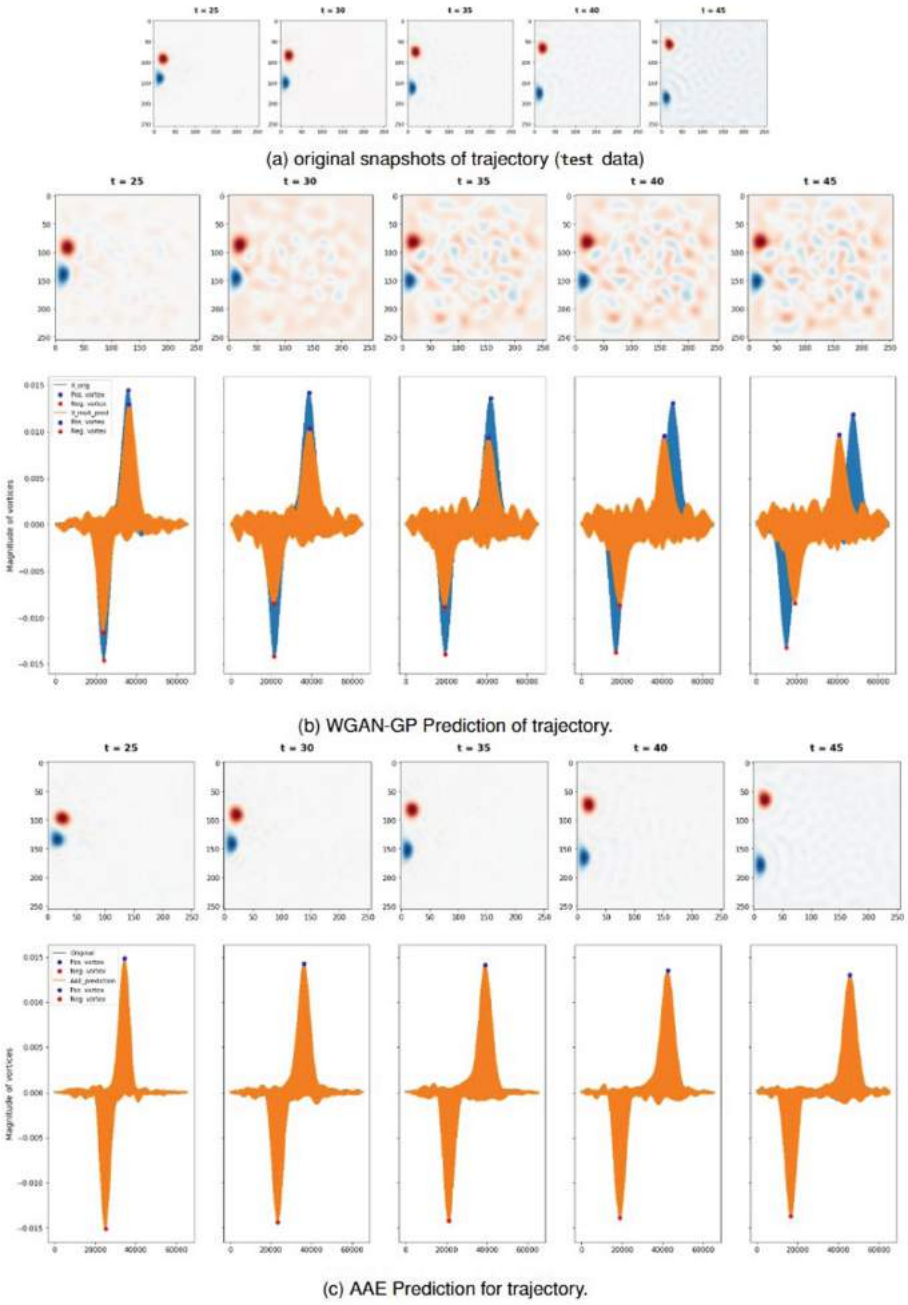



**Fig. 7.** Multiple time level prediction (t=35 to t=45) using WGAN-GP(b) and AAE(c) on a sample trajectory.

## 4 Discussion and Conclusion

The use of machine learning techniques for fluid modelling problems is very promising. The low cost, speedup and relative accuracy provided by machine learning tools, specially generative models, are attractive features in the study of forward modelling. In this project, an exploratory study is done to understand the capabilities of two generative models - generative adversarial network (GAN) and adversarial autoencoder (AAE) - for predicting the evolution in time of a highly nonlinear turbulent fluid flow. We use the capabilities of the generative models within a non-intrusive reduced order model framework. The results demonstrate that both generative models are capable of predicting the evolution of the vortice positions in time, although the AAE has generate more accurate predictions than the WGAN-GP. Furthermore, with the event of mode collapse, we conclude that a 'vanilla' DCGAN may be insufficient for the turbulent flow prediction. We also show that the WGAN-GP and AAE can generalise and generate solutions not present in the training set.

**Acknowledgements.** The authors would like to acknowledge the following EPSRC grants: RELIANT, Risk EvaLuatIon fAst iNtelligent Tool for COVID19 (EP/V036777/1); MAGIC, Managing Air for Green Inner Cities (EP/N010221/1); MUFFINS, MUltiphase Flow-induced Fluid-flexible structure InteractioN in Subsea applications (EP/P033180/1); the PREMIERE programme grant (EP/T00 0414/1); and INHALE, Health assessment across biological length scales (EP/T00 3189/1). Most sincere appreciation goes out to the Department of Earth Science and Engineering at Imperial College London for support and resources provided over the period of this project. We would also like to extend gratitude to everyone who was engaged in discussions during the project. Thank you for giving your time and sharing your ideas.

## References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein GAN (2017), http://arxiv.org/abs/1701.07875
2. Brunton, S.L., Noack, B.R., Koumoutsakos, P.: Machine learning for fluid mechanics. Annu. Rev. Fluid Mech. **52**, 477–508 (2020). https://doi.org/10.1146/annurev-fluid-010719-060214
3. Cardos, M., Durlofsky, L., Sarma, P.: Development and application of reduced-order modeling procedures for subsurface flow simulation. Int. J. Numerical Methods Eng. **77**(9), 1322–1350 (2009)
4. Cheng, M., Fang, F., Pain, C.C., Navon, I.M.: An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. Comput. Methods Appl. Mech. Eng. 372 (2020). https://doi.org/10.1016/j.cma.2020.113375
5. Cheng, M., Fang, F., Pain, C.C., Navon, I.M.: Data-driven modelling of nonlinear spatio-temporal fluid flows using a deep convolutional generative adversarial network. Comput. Methods Appl. Mech. Eng. 365 (2020)
6. Gonzalez, F.J., Balajewicz, M.: Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems (2018), http://arxiv.org/abs/1808.01346

7. Goodfellow, I.: NIPS 2016 Tutorial: Generative Adversarial Networks (2016), http://arxiv.org/abs/1701.00160
8. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
9. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.: Improved training of wasserstein GANs. Adv. Neural Inf. Process. Syst. 2017-Decem, 5768–5778 (2017)
10. Hotelling, H.: Analysis of a complex of statistical variables into principal components. J. Educ. Psychol. **24**(6), 417 (1933)
11. Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2014)
12. Lecun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
13. Lumley, J.: Atmospheric turbulence and radio wave propagation. In: Atmospheric Turbulence and Radio Wave Propagation, pp. 166–178 (1967)
14. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., Frey, B.: Adversarial Autoencoders (2015), http://arxiv.org/abs/1511.05644
15. Quilodrán-Casas, C., Silva, V.L., Arcucci, R., Heaney, C.E., Guo, Y., Pain, C.C.: Digital twins based on bidirectional LSTM and GAN for modelling the covid-19 pandemic. Neurocomputing **470**, 11–28 (2022)
16. Silva, V.L.S., Heaney, C.E., Li, Y., Pain, C.C.: Data Assimilation Predictive GAN (DA-PredGAN): applied to determine the spread of COVID-19 (2021), http://arxiv.org/abs/2105.07729
17. Sirovich, L.: Turbulence and the dynamics of coherent structures. III. Dynamics and scaling. Q. Appl. Math. **45**(3), 583–590 (1987). https://doi.org/10.1090/qam/910464
18. Swischuk, R., Mainini, L., Peherstorfer, B., Willcox, K.: Projection-based model reduction: formulations for physics-based machine learning. Comput. Fluids **179**, 704–717 (2019)
19. Xiao, D., Fang, F., Buchan, A.G., Pain, C.C., Navon, I.M., Muggeridge, A.: Non-intrusive reduced order modelling of the Navier-Stokes equations. Comput. Methods Appl. Mech. Eng. **293**, 522–541 (2015). https://doi.org/10.1016/j.cma.2015.05.015
20. Xiao, D., Yang, P., Fang, F., Xiang, J., Pain, C.C., Navon, I.M., Chen, M.: A non-intrusive reduced-order model for compressible fluid and fractured solid coupling and its application to blasting. J. Comput. Phys. **330**, 221–244 (2017). https://doi.org/10.1016/j.jcp.2016.10.068

# Towards Social Machine Learning for Natural Disasters

Jake Lever[1,2,3(✉)] and Rossella Arcucci[1,2,3]

[1] Department of Earth Science and Engineering, Imperial College London, Exhibition Rd, London SW7 2BX, UK
{j.lever20,r.arcucci}@imperial.ac.uk
[2] Data Science Institute, Imperial College London William Penney Laboratory, South Kensington, London SW7 2AZ, UK
[3] Leverhulme Centre for Wildfires, Environment and Society, Imperial College Department of Physics, South Kensington, London SW7 2BW, UK

**Abstract.** We propose an approach for integrating social media data with physical data from satellites for the prediction of natural disasters. We show that this integration can improve accuracy in disaster management models, and propose a modular system for disaster instance and severity prediction using social media as a data source. The system is designed to be extensible to cover many disaster domains, social media platform streams, and machine learning methods. We additionally present a test case in the domain context of wildfires, using Twitter as a social data source and physical satellite data from the Global Fire Atlas. We show as a proof of concept for the system how this model can accurate predict wildfire attributes based on social media analysis, and also model social media sentiment dynamics over the course of the wildfire event. We outline how this system can be extended to cover wider disaster domains using different types of social media data as an input source, maximising the generalisability of the system.

**Keywords:** Natural disasters · Machine learning · Sentiment analysis

## 1 Introduction

As the climate begins to change, the severity and frequency of natural disasters is increasing yearly. Between 1998 and 2017, climate-related/geophysical disasters killed 1.3 million people, and left a further 4.4 billion injured, homeless, displaced or in need of emergency assistance [7]. In the last 30 years, the number of climate related disasters has tripled, and this increase is almost certain to continue into the future as climate records continue to be broken all over the world.

This increase in natural disaster activity due to climate change is additionally being compounded by human activity. Modern agricultural practices increase the risk of natural disaster through deforestation, and air pollution and the emissions of water soluble particles into the atmosphere also increase the risk of extreme weather. These disasters are also often highly linked, in ways which are still being


D. Groen et al. (Eds.): ICCS 2022, LNCS 13352, pp. 756–769, 2022.
https://doi.org/10.1007/978-3-031-08757-8_62

studied; such as the recent arctic heatwave in 2020, which destabilized the polar vortex and allowed a cold front of air to move down over North America, causing sub-zero temperatures in Texas, freezing the power grid and leaving 210 people dead. Increasing world population and housing, water, food and health crises in many highly populated areas are forcing more people into living at the wildland urban interface, which is at a raised risk of disasters due to these reasons. As a result, more individuals & infrastructure will be increasingly exposed to more frequent and intense natural disasters, raising the overall potential cost to society of these destructive events. Since 1980, the US has sustained 310 weather events and climate disasters where the overall damages & costs reached or exceeded $1 billion USD (adjusted for CPI to 2021). 20 of these events occurred in 2021 alone, leading to the deaths of 688 people in this year. The total cost of these 310 events alone exceeds $2.155 trillion USD [17].

Advancements in computational models in the past 20 years have allowed humans to predict localised environmental conditions with increasing accuracy. This is due to increased computational resources, advancements in understanding of these systems, and the inclusion of better data sources. However, numerical computational models during natural disasters often suffer in terms of accuracy due to their extreme and unpredictable nature, meaning often not enough useful training data is available. One massive source of data which is increasingly being used in models is social media.

Social media plays an ever increasing role in society. In 2020, 23% of US adults reported getting their news from social media often [19], up from 18% in 2016 [19]. Additionally, 82% of US adults are using social media as of 2021 [6], representing a massive audience and huge amounts of shared information. This paper proposes a more simple, socially focused model which aims to avoid the common pitfalls of modern commonly used models by combining social media data as an input source. In the advent of social media in the last decade, more studies developing social and physical models are including these types of data sources as an input [1,9,14,22]. These studies introduce the concept of the 'human sensor', where social media users are considered to be noisy sensors, posting a subjective account of their localised conditions. By analysing the response from these sensors, we can infer a model of the disaster as it unfolds in real time. We propose a system for collecting and analysing social media data to improve current natural disaster models, by monitoring online discussions and sentiments with the aim of ultimately identifying areas of actionable interest to disaster management teams. We show that online social discussions and sentiment are often linked to disaster activity, and that models can be trained to predict these shifts in sentiment over the course of the disaster. We also propose methods for information extraction & analysis of textual tweet data published during natural disasters, and discuss how this could be incorporated into a real-time model for public alerts.

The paper is structured as follows; Sect. 2 outlines the need for more socially conscious natural disaster models, and discusses the benefits of this. Following this, Sect. 3 defines the domain specific, modular architecture of the system we

are proposing, describing the function of each of these modules. In Sect. 4 we present a test case of the system in the context of north American wildfires from 2016. Finally, Sect. 5 summarises our contribution and further work.

## 2 Background

Extensive work has been put into climate modelling with increasing success, and managing and mitigating climactic effects has been achieved in the past with good results [13], such as flood mitigation with dams or drought management using reservoirs [16]. However, it has been shown that during natural disasters, the coordination of teams from a crisis management perspective becomes challenging [15]. Often emergency operations are given from a centre and coordinate multiple organisations including local government, police, fire, hospital, utility, and Red Cross representatives. These teams are often 'flown in' ad hoc, and are expected to collaborate to deliver optimal crisis management at often very short timescales. This can very easily lead to poor communication and coordination of disaster management teams at a time when quick, coordinated and effective action is often key to saving lives.

Inherent properties of social media have been shown to lend themselves towards crisis management during natural disasters from both sides of the public / disaster management coin [23]. On the one hand, social media allows people in different locations to post a subjective description of their surroundings & immediate dangers as a disaster unfolds, which again employs the concept of the human sensor. By in-taking and analysing this data during disasters, management teams could build up a geographic picture of these noisy accounts, and use predictions to quickly identify areas of interest/danger from public accounts posted in real-time.

Conversely, social media can also be effectively used by these management teams to relay operational messages, warnings, and updates back to the public once they have been authorized. An example of this has already been implemented by Google using satellite data for a number of crisis alerts including flood forecasting, wildfire boundary lines, earthquakes, and more [10]. Operational updates posted on social media by local authorities could also be used in models to coordinate information/response strategies between the different organisations, such as the ones mentioned previously, involved in crisis management. This could ease the communication strains between organisations and help coordinate a quicker, more direct response.

Social media data is increasingly being analysed using Natural Language Processing (NLP) methods. NLP is a set of broad analytical techniques for computationally interpreting human language [12]. The field has undergone rapid recent development, with increasingly more information such as topics, intent, themes, entities, and sentiments being able to be inferred by models from text. Due to the vast amounts of text information generated by social media posts, it has been shown that analysis of these posts show insights [1,2,5,9,22] into processes and events. This paper aims to yield similar insights in the context

of natural disasters by performing information extraction & sentiment analysis (SA). APIs exist for accessing data from all main social media platforms, including Twitter [20], Facebook [8], Instagram [8], and Reddit [18].

By implementing a machine learning (ML) based information system such as the one described in this paper, we can improve the streamlining of information at emergency operations centres, hereby allowing disaster management teams to make real-time, bottom-up decision as an event unfolds. Previous work has shown that there is a link between social media expression/sentiment, and natural disaster activity in the context of hurricanes [22].

This paper proposes a system, shown in Fig. 1, for collecting and analysing social media data in real time, and uses models trained on historical data to predict instances and characteristics of natural disasters. We show that gathering information on social media given the current state of the art of language models is feasible and efficient, and discuss the domain specific training using a test case in the context of wildfires for forecasting. We outline a system which adds a social aspect to disaster models, and contributes to the advancement of their capabilities by providing additional information to the overall system.

## 3 System Overview

The main concept behind the system is the live extraction of important information and sentiments from social media channels regarding natural disasters as they unfold. This system could be implemented initially in a predictive manner, predicting whether there is a natural disaster unfolding, and subsequently monitor and track the disaster by analysing social media discussions and extracting information from this live text data.

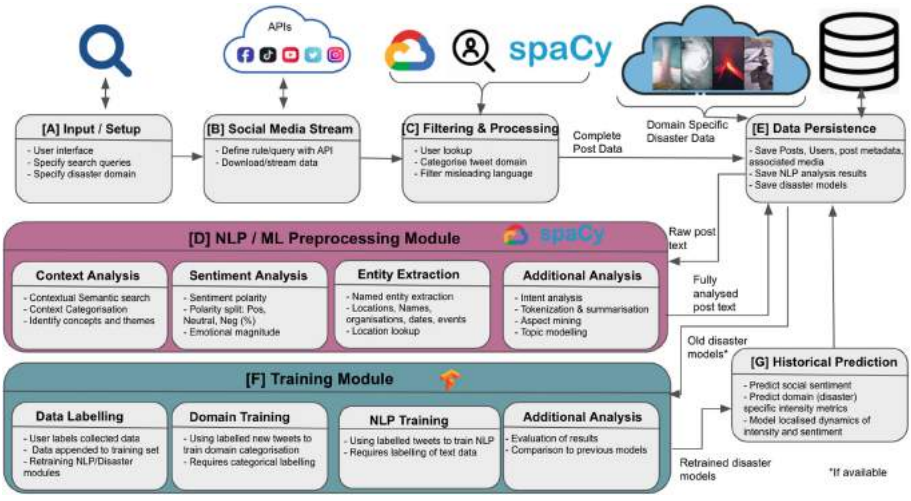



**Fig. 1.** Modular system diagram outlining flow of processes.

The system implements a live data stream based on search queries and using a prediction model which is specific to the disaster domain for the model. Here we mean the domain to be the category of natural disaster in the scope of this study, for example; Wildfires (this domain will be the demonstration case presented in Sect. 4), Earthquakes, Flood forecasting, or Hurricanes. With more work, the system could also further still be extended to include modelling and prediction on a wider range of types of events such as protests and political unrest, as well as political events and elections, health and food crises, and crime. This would require further development, primarily a more complex language model.

As mentioned, the prediction model would need to be domain specific to the type of disaster it is making predictions on. A model is made to be domain specific by training on a historical social media dataset which is again domain/disaster specific. For example, a model specific to the domain of wildfires would need to be trained on wildfire specific social media data. The entire system is designed to be modular and so can be adapted to be domain specific for each type of natural disaster, as well as different platforms of social media. This section will outline the systems component modules, and how they differ in terms of domains. The operational modular system is shown in Fig. 1. Each modules function is now briefly explained.

### 3.1 [A] System Input/Setup

The system is designed to take a series of input queries, or rules on which to set up a live real time stream for social media posts which are published satisfying one of the given rules. This is shown in Module A of Fig. 1. The rules must be formatted conforming to the social media platforms query syntax, e.g. Twitter's syntax rules [20]. The set of queries must be domain specific to the disaster, and include keywords specifically used in language specific to this domain.

### 3.2 [B] Social Media Data Streaming

This system implements real time social media streaming for the use of this live data in predictions. The function of this module is the implementation of a social media platform specific streaming function to download live data from the given website, as shown in Module B of Fig. 1. A number of different APIs exist to facilitate this for Twitter [20], Facebook [8], Instagram [8], and Reddit [18]. The function sends a set of queries, outlined with the domain specific language in the syntax outlined by the individual APIs documentation, as outlined in Sect. 3.1. This starts the live data stream. Here, we can also search on accounts which are publishing posts, and part of the training phase for this system will be the creation of a list of reputable accounts to monitor. These accounts will have been shown to be useful in providing information about disaster instance and development. These may include local authority service accounts, local and environmental journalists, rescue workers, and local government and authorities.

### 3.3 [C] Filtering and Text Pre-processing

Following collection of the social media data, filtering of results removes false positives (posts which do not mention the natural disaster domain when the model detects that they have) from the collected data. Filtering by misuse/misleading phrases, e.g. 'Corruption is spreading like wildfire' makes the dataset cleaner, resulting in more accurate models. This is represented by Module C of Fig. 1. Text is also pre-processed in this module, removing punctuation, hyperlinks, correcting spelling mistakes and formalising words etc., reducing noise in the text.

### 3.4 [D] Information Extraction and Sentimental Analysis

The function of this module is for the generation of social sentiment data from posts on natural disasters, and the extraction of published information which may be of help to the prediction module of the system. This is shown in Module D of Fig. 1. Sentiment Analysis is the process of computationally extracting a numerical value corresponding to the overall emotional leaning of the text. This is achieved using analysis of words used, word patterns and part-of-speech (POS). This is useful because it allows us to convert the qualitative text data into quantitative metrics which can be used for further analysis. The system can then be trained to predict the aggregated sentimental values for each natural disaster in Sect. 3.7. SA was performed using Google's NLP API for SA [11], and yields two metrics:

- Firstly, the **Sentiment Score** ranges between -1.0 (negative) and 1.0 (positive) and corresponds to the overall emotional leaning of the text.
- Secondly, tweet **Magnitude** indicates the overall strength of emotion (both positive and negative) within the given text, between 0.0 and +inf. [11]

This defines two numerical sentimental variables; $S$ and $M$, which are the social sentiment variables for each natural disaster, observing the constraints $S \in \mathbb{R} : 0 \leq S \leq 1$ and $M \in \mathbb{R} : 0 \leq M$ respectively.

A domain specific language model could be used to infer greater insight into the text data collected by the collection module outlined in Sect. 3.2. Models like this can be trained using domain specific language datasets, which can be the textual social media data previously collected as part of the system setup in Sect. 3.6. The NLP model will be similar to the one implemented for hurricanes in [22], and will additionally aim to extract domain specific disaster characteristics from the noisy text data. These may include but are not limited to; disaster duration, total area damage, and number of people displaced, in need of emergency assistance, injured or killed. This is achieved through the sentimental & linguistic analysis of the words used in the posts, for example; contextual, entity, and intent analysis, results of which are also saved to the database. Tokenisation of the text is also stored.

### 3.5 [E] Data Persistence

Data persistence and model iteration & integration are a key aspect of the evolutionary aspect of this system. Due to the systems online data collection module outlined in Sect. 3.2, data is constantly collected and analysed surrounding different types of natural disaster domains. The function of this module is to save analysed social media data, including metadata on posts, users etc. and from Sect. 3.2, NLP and sentimental analysis results from Sect. 3.4. Additionally stored will be the models trained in Sect. 3.6 and the downloaded natural disaster data, as shown in Module E of Fig. 1.

### 3.6 [F] System Training (Pre-Operational)

Before the system can be used in real time for operational use, a model needs to be trained in order to make the model domain specific for the type of natural disaster it will be used for. Each type of disaster discussed in Sect. 3 will have different types of expression on social media and the dynamics and relationship between social media activity and disaster activity will vary from disaster to disaster. Thus, it is necessary to implement individual models for each type of disaster.

To train a domain specific model, we need a similarly domain specific historical social media dataset related to this type of disaster. The system is set up using historical natural disaster data for the domain specific disaster data in order to implement a base model.

This module also supports the labelling of training data gathered from the data collection and analysis modules of the system. The reason for this is for the iterative improvement of the ML models in Modules D & G, described in Sects. 3.4 & 3.7 respectively.

An important aspect of this training phase is that it can be achieved off ine. That is, models can be iteratively trained while the system is online, and swapped out as they are improved. Due to the system constantly collecting and saving tweet data surrounding these types of natural disasters, the database used to train these models will only get larger and more diverse, both geographically and between disaster domains, as the system is used more. This new data can be labelled and then used to iteratively train improved systems in a supervised manner, or remain unlabelled for models to be trained using unsupervised methods. To summarise, the system is designed to have an off ine training programme which iteratively retrains and improves the model as it runs, leading to evolutionary improvement and in turn greater accuracy, and the rapid development & deployment of improved predictive models.

### 3.7 [G] ML Prediction Model

This part of the system utilises the model trained historical data outlined in Sect. 3.6. We take the model trained on historical, domain specific disaster data, and use this to make predictions about the occurrences of new disaster instances. The aims of this module are as follows;

– To detect instances and categorise types of new disasters, and predict the times and locations of these new events.
– To predict the severity of disasters over the course of the crisis period. By severity we mean the physical domain specific disaster variables from the historical data supplied in the training dataset outlined in Sect. 3.6, e.g. size of area affected, number of injuries etc. We hypothesise that there exists some function of $f$ which allows us to predict disaster intensity from social media post data.

The ML method which we utilise for this system is the Gradient Boosted Random Forest, implemented in python with the *XGBoost* package [21]. Random Forests are an ensemble method extension of Classification and Regression Trees (CART) [4]. In the training phase, these methods start with a simple model, often a single tree (or weak learner), and then additive training occurs where trees are generated and added to build up a forest of trees which is used for the final prediction. For new data, predictions are then made by averaging the majority vote of all trees in the forest. The GBRFs are built and evaluated using the MAE, and the Gini coefficient [4]. The Gini coefficient is a scoring metric which is a measure of the degree to which a particular element is wrongly classified when randomly chosen and it is expressed by the formula:

$$\sum_{i=1}^{n} p_i(1 - p_i) \tag{1}$$

where $p_i$ denotes that the physical event $i$ happens (or that the sentiment $i \in S$ is evaluated) and $n$ is the number of possible events (or possible sentiments). Gradient boosting Random Forest algorithms are particular Random Forests which begin with a base (weak) learner (tree), and consecutively add more weak learners to the ensemble with the goal of minimising a loss function [4]. Given a training sample $\{\mathbf{x}_i, \mathbf{y}_i\}_1^n$, the goal is to find function of $F^{\circ}(\mathbf{x})$ such that the expected value of the loss function $\alpha(\mathbf{y}, F(\mathbf{x}))$ is minimised [4]. The gradient boosting problem is then expressed as:

$$F^{\circ}(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \mathbb{E}_{\mathbf{x},\mathbf{y}} \alpha(\mathbf{y}, F(\mathbf{x})) \tag{2}$$

where $\mathbb{E}$ denotes the expected value. The boosting algorithm then approximates $F^{\circ}(\mathbf{x})$ by additive expansion, which can be summarised as:

$$F(\mathbf{x}) = \sum_{m=0}^{s} \sigma_m h(\mathbf{x}, a_m) \tag{3}$$

where the functions $h(\mathbf{x}, a)$ are known as the 'weak learners'.

Over a series of $s$ steps, the weak learners are sequentially added, and the expansion coefficients $a = \{a_1, a_2, ...\}$ and $\{\sigma\}_0^s$ are jointly fit to the current models pseudo-residuals. For $m = 1, 2, \ldots, s$, this gives

$$(\sigma_m, a_m) = \arg\min_{\Delta a} \sum_{i=1}^{n} \alpha(\mathbf{y}_i, F_{m-1}(\mathbf{x}_i) + \sigma h(\mathbf{x}_i; a)) \tag{4}$$

and

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \sigma_m h(\mathbf{x}; a_m) \tag{5}$$

which shows the step-wise optimisation of $F$ . Equation (4) is then optimised to solve the loss function $\alpha$ by fitting least squares on $h(\mathbf{x}, a)$. This replaces the optimisation problem presented in Eq. (2) with one based on reducing least squares in Eq. (4).

The models were trained on a wide grid search covering the hyper-parameters denoted by:

- $ETA$ (Learning Rate): Step size shrinkage used in update.
- $maxDepth$: Maximum depth of a tree.
- $minChildWeight$: Minimum sum of instance weight needed in a child.
- $subsample$: Subsample ratio of the training instances.
- $colSampleByTree$: Subsample ratio of columns when constructing each tree.
- $nEstimators$: Number of trees in the ensemble.

The output of this module is the overall system output: predictions of domain, instances and severity of the natural disaster. This represents a socialised model of the disaster, from detection to evolution and finally resolution. Once a disaster has been detected and registered in the application database, it follows the process flow shown in Fig. 1, being continuously updated as more information on this suspected disaster is streamed through by the data collection stream. A disaster is monitored until information is passed through that this crisis is resolved. We now demonstrate in Sect. 4 a prototype of the model described in Sect. 3, applied with a wildfire domain.

## 4 Experimental Test Case: Wildfires - Satellites and Twitter

We now present a prototype version of the domain specific model applied to satellite wildfire data using twitter as the social media data source. We choose wildfire events occurring in Australia and North America (United States and Canada) in 2016 for the creation of our training dataset. We chose to use this type of disaster domain in this geographic area due to the abundance of both wildfire activity in this area and active Twitter users. We now outline the implementation of the system for this particular context.

*Satellite & Twitter Data Collection:* For the combination of social media data with wildfire data in our model component of the system, we chose to use twitter for the historical social media source, and wildfire data from the Global Fire Atlas [3] for the 2016 wildfire data. The physical wildfire characteristics taken from this data are: Latitude ($*$), Longitude ($*$), Size (km2), Perimeter ($Per$) (km), Duration ($d$) (days), Speed(km/day), Expansion ($Exp$) (km2/day), and Start ($S_{DOY}$) and End ($E_{DOY}$) . This defines in the physical vector $\mathbf{x}$ of our model:

$$\mathbf{x} = [Lat, Lon, Size, Per, d, Speed, Exp, S_{DOY}, E_{DOY}, Pop_{Density}] \tag{6}$$

Twitter's V2 API [20] with an academic research product track was used for the collection of the twitter data associated with historical wildfires, using a query implementation in line with Twitter's query syntax [20]. Queries were designed to search for tweets mentioning locations of the burn as well as a set of wildfire domain specific keywords to search on; Fires OR Wildfires OR Bushfires OR "Landscape Burn" OR "Wildland Burn" , as well as generated hashtags included in the query. Tweets which contained certain misuse phrases such as "like wildfire" were removed to reduce noise. Tweets were saved to a database with meta and user data and media, along with a Fire ID. The result is a dataset of US wildfires in 2016 and tweets associated with these events.

*Twitter Data Analysis:* We are now able to analyse this text data for wildfires and generate social sentiment variables from this data using SA. Recalling Sect. 3.4 where the two numerical sentimental variables $S$ and $M$ are defined, Tweets are grouped by day for each fire and social sentiment values from the $S$ and $M$ are averaged and summed to generate the following social sentiment variables for each wildfire:

- $S_{mean}$ : Average Daily Sentiment Score
- $M_{mean}$ : Average Daily Magnitude Score
- $S_{ovr}$: Overall Sentiment Score
- $M_{ovr}$: Overall Magnitude Score
- $Tot_{tweets}$: Total number of Tweets for Wildfire

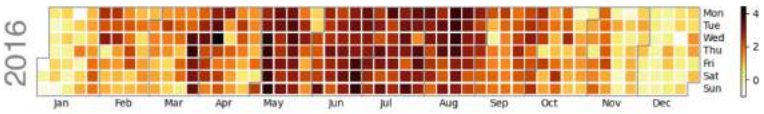
These variables constitute the sentimental vector

$$\mathbf{y} = [S_{mean}, M_{mean}, S_{ovr}, M_{ovr}, Tot_{tweets}]. \tag{7}$$

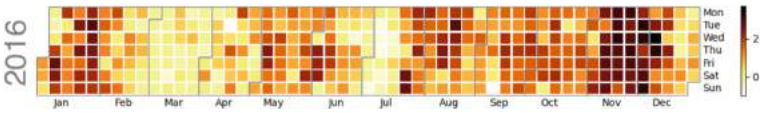
The generation of these social sentiment variables for each wildfire represents the completion of our two datasets in both the Australian (AUS) and North American (US) domains.

The data can now be viewed from a temporal perspective, by plotting heatmaps for the online social sentiment across the year of 2016. Fig. 2 shows how online sentimental activity matches the fire seasons in the two geographic domains, which is a positive indication of the quality of the data.

*Wildfire & Social Modelling Results:* Two types of ML models were implemented using the *XGBoost* package [21] in python as outlined in Sect. 3.7. The first type of model takes as an input the physical vector $\mathbf{x}$, and is trained using this data on the target vector $\mathbf{y}$. The 5 variables which were described in this section were used to train 5 models, one predicting each social sentiment variable. These models are called the sentimental prediction models. The second type of model predicted the 10 physical wildfire characteristics vector $\mathbf{x}$, using the social sentiment vector $\mathbf{y}$, predicting in the opposite direction to the sentimental prediction model. This type of model was called the physics prediction model, and there were 10 of these models trained, one for each variable, meaning total of 15 models were trained on the combined AUS + US dataset. The results are shown in Tables 1




(a) Heatmap of US online social sentiment for 2016




(b) Heatmap of AUS online social sentiment for 2016

**Fig. 2.** US (a) and AUS (b) online social Sentiment heatmaps for 2016

& 2 below. As the last column in Tables 1 & 2 shows, the execution time ($Exe_T$) for running the model on a 2020 Macbook Pro 2.3GHz 8 core i7 Intel processor is very low, whch is an important condition for real time operational predictions.

**Table 1.** Results from predicting social sentiment variables from physical wildfire characteristics.

| Variable | MAE | RMSE | Gini | $Gini_N$ | Score | $Exe_T$ (msecs) |
|---|---|---|---|---|---|---|
| $S_{mean}$ | 6.34 | 17.83 | 0.367 | 0.846 | 38.7% | 2.94 |
| $M_{mean}$ | 10.88 | 40.85 | 0.316 | 0.822 | 38.3% | 5.75 |
| $M_{ovr}$ | 107.27 | 407.36 | 0.338 | 0.872 | 24.4% | 3.37 |
| $S_{ovr}$ | 54.75 | 364.76 | 0.381 | 0.865 | 15.3% | 2.28 |
| $tot_{tweets}$ | 458.927 | 2777.28 | 0.323 | 0.863 | 10.6% | 2.94 |

Table 1 demonstrates that the models for predicting average Sentiment and Magnitude ($S_{mean}$ and $M_{mean}$) show good results when attempting to predict these variables, with low MAEs of 6–10. Overall Sentiment and Magnitude ($S_{ovr}$ and $M_{ovr}$) models also preformed well. The most notable result shown in Table 2 is the model predicting fire Duration ($d$). This model showed very positive results, with an MAE of 0.84. This shows that the resulting model was able to predict wildfire duration to within one day from social sentiment values/social media data alone. Additionally, *Speed* and *Exp* both performed well.

**Table 2.** Results from predicting physical variables from social sentiment data.

| Variable | MAE | RMSE | Gini | $Gini_N$ | Score | $Exe_T$ (msecs) |
|---|---|---|---|---|---|---|
| $Lat$ | 3.72 | 5.2 | 0.344 | 0.868 | 0.75 | 4.90 |
| $Lon$ | 6.185 | 7.40 | 1.73 | 0.929 | 0.79 | 6.11 |
| $Size$ | 9.66 | 33.89 | 0.354 | 0.835 | 0.14 | 1.95 |
| $Per$ | 7.22 | 15.89 | 0.227 | 0.764 | 0.24 | 3.69 |
| $d$ | 0.84 | 2.01 | 0.207 | 0.961 | 0.87 | 4.22 |
| $Speed$ | 0.533 | 0.86 | 0.143 | 0.659 | 0.25 | 2.32 |
| $Exp$ | 0.88 | 3.24 | 0.226 | 0.652 | 0.16 | 2.09 |
| $Pop_{Density}$ | 69.05 | 575.18 | 0.397 | 0.424 | 0.02 | 2.22 |
| $S_{DOY}$ | 40.73 | 60.90 | 0.100 | 0.736 | 0.56 | 2.83 |
| $E_{DOY}$ | 39.62 | 61.15 | 0.095 | 0.734 | 0.56 | 5.48 |

## 5 Conclusion and Future Work

This paper outlines a system to facilitate the integration of social media data into physical models, discusses the benefit of this, and demonstrates a prototype test case with current wildfire models. As shown, social media data is being adopted increasingly in scientific studies and specifically disaster management, yielding benefits which could be transferred to natural disaster relief efforts. This work attempts to bridge this gap by developing a modular social media alarm system for natural disasters which predicts instances and localised severity of the event based on analysis of social media posts.

Having successfully trained and implemented a retrospective historical social media wildfire model using Twitter as a data source, the next step for testing this type of model would be the integration/coupling of the system with a real time wildfire model. The modular design of the system allows for rapid updating of the system as future work allows. This will primarily be focused on the development of ML methods for information extraction of the post text data and modelling disaster activity. The benefit of this will be improved accuracy of these models which will allow localised modelling of disaster conditions.

Social media data represents a near limitless supply of real time data on almost any large event. If disaster models do not consider this data, then this represents a loss of information, as there is data available via these networks which is not accessible elsewhere. Systems which analyse this data in real time are able to recoup this loss, which ultimately leads to the development of improved models.

## References


1. Alkouz, B., Aghbari, Z.A., Abawajy, J.H.: Tweetluenza: predicting flu trends from twitter data. Big Data Mining Anal. **2**(4), 273–287 (2019). https://doi.org/10.26599/BDMA.2019.9020012
2. Alrashdi, R., O'Keefe, S.: Automatic Labeling of Tweets for Crisis Response Using Distant Supervision, p. 418–425. Association for Computing Machinery, New York, NY, USA (2020), https://doi.org/10.1145/3366424.3383757
3. Andela, N., et al.: The global fire atlas of individual fire size, duration, speed and direction. Earth Syst. Sci. Data **11**(2), 529–552 (2019)
4. Bishop, C.M.: Pattern recognition and machine learning (2006)
5. Cui, R., Gallino, S., Moreno, A., Zhang, D.J.: The operational value of social media information. Prod. Oper. Manage. **27**(10), 1749–1769 (2018)
6. Department, S.R.: Social media usage in U.S. November 2021, https://www.statista.com/statistics/273476/percentage-of-us-population-with-a-social-network-profile/
7. for Research on the Epidemiology of Disasters & The UN Office for Disaster Risk Reduction, T.C.: Economic Losses, Poverty & Disasters 1998–2017. Tech. rep., The Centre for Research on the Epidemiology of Disasters & The UN Office for Disaster Risk Reduction (2018)
8. Facebook: Facebook graph API, https://developers.facebook.com/docs/graph-api/
9. Gallagher, R.J., Reagan, A.J., Danforth, C.M., Dodds, P.S.: Divergent discourse between protests and counter-protests: #blacklivesmatter and #alllivesmatter. PLOS ONE **13**, 1–23 (2018). https://doi.org/10.1371/journal.pone.0195644
10. Google: Forecasting & alerts: Google's crisis alerts provide access to trusted safety information across search, maps, and android (2017). https://crisisresponse.google/forecasting-and-alerts/. Accessed 17 Jan 2021
11. Google: Analyzing sentiment (2021), https://cloud.google.com/natural-language/docs/analyzing-sentiment
12. Hirschberg, J., Manning, C.D.: Advances in natural language processing. Science **349**(6245), 261–266 (2015). https://www.science.org/doi/abs/10.1126/science.aaa8685, https://doi.org/10.1126/science.aaa8685
13. Dullaart, J.C.M., Muis, S., Bloemendaal, N., Aerts, J.C.J.H.: Advancing global storm surge modelling using the new ERA5 climate reanalysis. Climate Dyn. 1007–1021 (2019). https://doi.org/10.1007/s00382-019-05044-0
14. Kryvasheyeu, Y., et al.: Rapid assessment of disaster damage using social media activity. Sci. Adv. **2**(3) (2016). https://advances.sciencemag.org/content/2/3/e1500779, https://doi.org/10.1126/sciadv.1500779
15. Laura G. Militello, Emily S. Patterson, L.B.R.W.: Information flow during crisis management: challenges to coordination in the emergency operations center. Cogn. Technol. Work **9**(1), 25–31 (2007). https://doi.org/10.1007/s10111-006-0059-3
16. Lempérière, F.: Dams and floods. Engineering **3**(1), 144–149 (2017)
17. National Centers for Environmental Information: U.s. billion-dollar weather and climate disasters, 1980 - present (NCEI accession 0209268) (2021). https://www.ncdc.noaa.gov/billions/overview. Accessed 17 Jan 2021
18. Reddit: Reddit API documentation, https://www.reddit.com/dev/api/
19. Shearer, E.: 86% of Americans get news online from smartphone, computer or tablet, January 2021, https://www.pewresearch.org/fact-tank/2021/01/12/more-than-eight-in-ten-americans-get-news-from-digital-devices/

20. Twitter: Twitter API v2, https://developer.twitter.com/en/docs/twitter-api
21. XGBoost: Dart booster (2020), https://xgboost.readthedocs.io/en/latest/tutorials/dart.html
22. Yao, F., Wang, Y.: Domain-specific sentiment analysis for tweets during hurricanes (DSSA-H): a domain-adversarial neural-network-based approach. Comput. Environ. Urban Syst. **83**, 101522 (2020)
23. Young, C., Kuligowski, E., Pradhan, A.: A review of social media use during disaster response and recovery phases. https://doi.org/10.6028/NIST.TN.2086. Accessed 31 Jan 2020

# Author Index