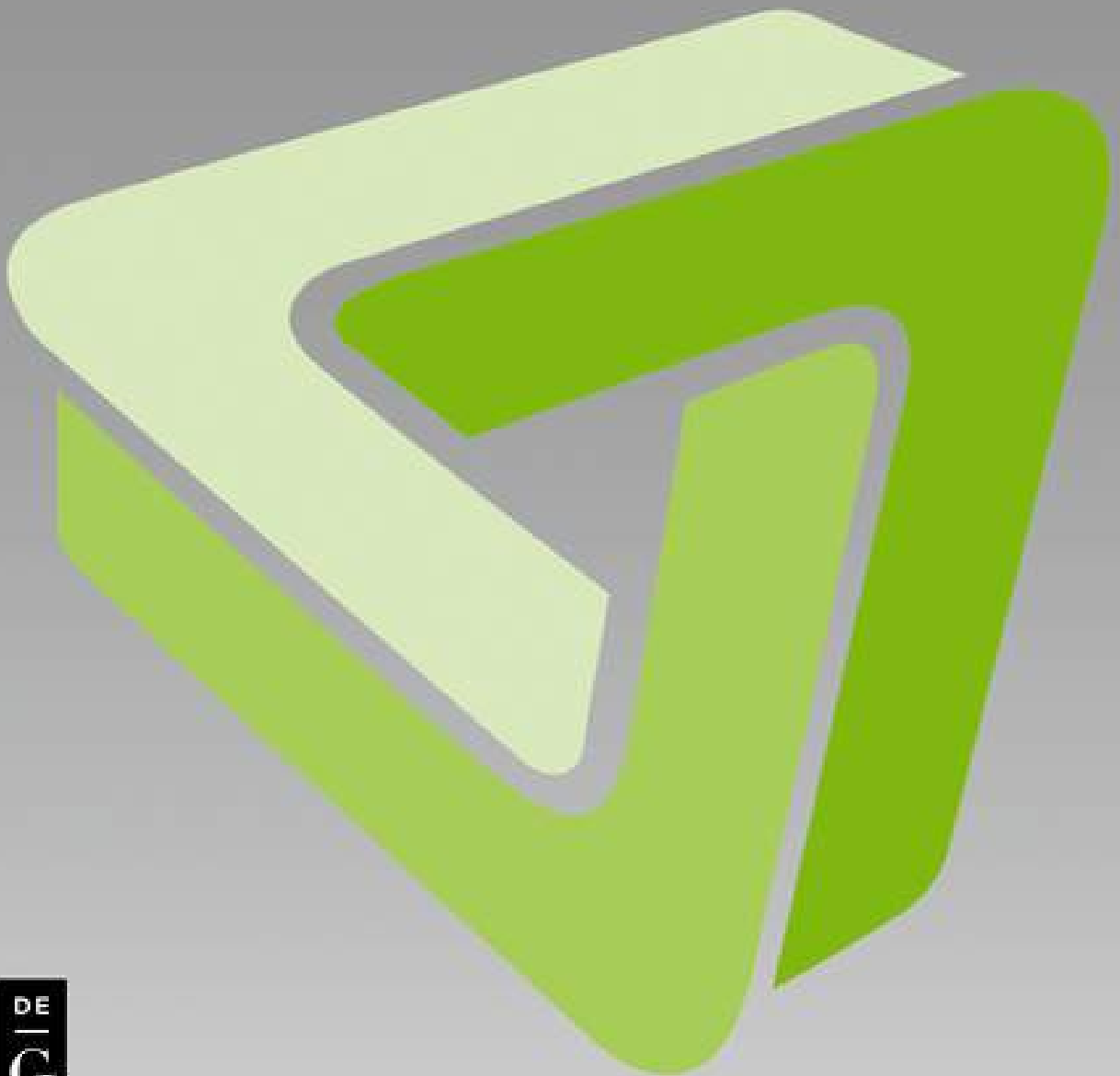
DE GRUYTER STEM

# MACHINE LEARNING UNDER RESOURCE CONSTRAINTS

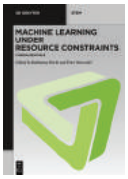
## APPLICATIONS

*Edited by Katharina Morik, Jörg Rahnenführer and Christian Wietfeld*

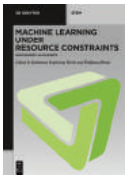


DE G

## Also of interest




Volume 1
*Machine Learning under Resource Constraints. Fundamentals*
Morik, Marwedel (Eds.), 2023
ISBN 978-3-11-078593-7, e-ISBN 978-3-11-078594-4




Volume 2
*Machine Learning under Resource Constraints. Discovery in Physics*
Morik, Rohde (Eds.), 2023
ISBN 978-3-11-078595-1, e-ISBN 978-3-11-078596-8

# Machine Learning under Resource Constraints

Final Report of CRC 876

Editor in Chief
Katharina Morik

## Volume 3/3

DE GRUYTER

# Machine Learning under Resource Constraints

Applications

Edited by
Katharina Morik, Jörg Rahnenführer and
Christian Wietfeld

DE GRUYTER

**Editors**
**Prof. Dr. Katharina Morik**
TU Dortmund University
Department of Computer Sciences
Chair for Artificial Intelligence
Computer Science 8
Otto-Hahn-Str. 12
44221 Dortmund
Germany

**Prof. Dr. Christian Wietfeld**
TU Dortmund University
Department of Electrical Engineering and
Information Technology
Chair for Communication Networks
Otto-Hahn-Straße 6
44221 Dortmund
Germany

**Prof. Dr. Jörg Rahnenführer**
TU Dortmund University
Department of Statistics
Statistical Methods in Genetics and
Chemometrics
Vogelpothsweg 87
44227 Dortmund
Germany

ISBN 978-3-11-078597-5
e-ISBN (PDF) 978-3-11-078598-2
e-ISBN (EPUB) 978-3-11-078614-9
DOI https://doi.org/10.1515/9783110785982



**Library of Congress Control Number: 2022949226**

**Bibliographic information published by the Deutsche Nationalbibliothek**
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at http://dnb.dnb.de.



Cover image: Collaborative Research Center 876
Printing and binding: CPI books GmbH, Leck

www.degruyter.com

# Contents

# 1 Editorial

*Katharina Morik*
*Jörg Rahnenführer*
*Christian Wietfeld*

This is the third book of a series of books dedicated to the results of the DFG Collaborative Research Center 876 on "Machine Learning under Resource Constraints". The first book of the series discusses fundamental innovations in the theory and algorithms of machine learning. The second book covers the use of machine learning in physics. This volume focuses on applications of the machine learning approaches presented in Book 1. The main idea is to demonstrate with specific examples how machine learning has become essential as well as practical in solving real-life problems from diverse application areas, ranging from medicine and robotics to road traffic and communication networks. Various real-life example applications show the significant impact of using tailored machine learning methods to improve the performance of the respective processes and systems. A key boundary condition imposed by the real-life environments is that resources, such as energy, storage, computing power, computing time, etc., are often limited and that the practicability of the proposed machine learning solutions depends on the efficient use of those resources. Therefore, the success of the solutions discussed in this book must not only be measured in terms of performance gains but, at the same time, in terms of their resource efficiency and corresponding sustainability. For many domain experts, the sheer multitude of machine learning approaches makes it difficult to choose the "right" ones for a particular problem. While the availability of software tools lowers the entry barrier to use machine learning methods by non-experts, the application examples contained in this book demonstrate that truly significant impacts can often only be achieved by an interdisciplinary combination of domain knowledge and the appropriate usage of machine learning methods. Accordingly, this book aims to promote proficiency in the use of machine learning methods beyond the quick wins of arbitrarily using whatever approach happens to be in fashion. The applications described in this book will touch upon a multitude of machine learning options covering the complete process chain from data acquisition, feature extraction, model selection via various learning approaches to model verification and model validation. It will show, for example, that while the deep learning approaches popular today can be beneficial for many problems in some areas, alternative methods such as ensemble learning with random forests are more accurate with much less resource utilization in other areas. The first part of the book addresses the application area of health and medicine. After an overview of machine learning in medicine provided by the invited authors Catherine Jutzeler and Karsten Borgwardt, a number of results from the CRC 876 are presented, covering virus detection, protein analysis, and cancer diagnostics and therapy. The


https://doi.org/10.1515/9783110785982-001

second part of the book is dedicated to the application of machine learning for industry use cases. On the one hand, machine learning enables proactive quality assessments; on the other, its application for precise localization, energy harvesting, and swarm control demonstrates the potential of embedding machine learning methods in almost any element of the manufacturing and logistics process of present and future industry environments. In the third part of the book, various examples illustrate the significant potential of machine learning for smart city and traffic use cases, such as the prediction of traffic flows, the privacy-preserving detection of vehicle flows, and resource-efficient crowd sensing in smart cities.

The fourth part of the book is about improving the performance of communication networks through machine learning. This includes new approaches for highly resource-efficient vehicle-to-cloud communications as well as machine learning-enabled mobile data network analytics and proper dimensioning of 5G network slices. As many applications of machine learning involve personal data and may affect privacy concerns, this book also includes a chapter on a general methodology to classify and handle privacy aspects of data management as part of the machine learning process chain. This focus on the data handling complements the privacy-preserving machine learning techniques. With this broad spectrum of application and practical implementation examples, we hope that this book will serve domain experts from diverse application areas as inspiration for the use of machine learning for their application-specific problems. To maximize the impact, many of the presented solutions are provided as open source published together with open data sets, allowing for reproducibility and sustainable transfer. At the same time, machine learning experts are expected to be motivated by the impressive impact of their work on real-life problems to further expand the machine learning solution space in terms of accuracy and resource efficiency.

Dortmund, 14.10.2022
Katharina Morik, Jörg Rahnenführer and Christian Wietfeld

# 2 Health / Medicine

## 2.1 Machine Learning in Medicine

*Catherine Jutzeler*
*Karsten Borgwardt*

**Abstract:** The combination of machine learning and population-scale health data holds the potential to revolutionize disease diagnosis and prognosis as well as to enable personalized predictions of therapy responses. The foundation for this unique opportunity is the ever-increasing amount of complex high-dimensional health data, from the molecular level of genome sequences to the level of image phenotypes and health history, that is available in digital form and at high resolution for cohorts of hundreds of thousands—and soon millions—of individuals. Machine learning promises to be a key technology in the generation of new knowledge from this big health data, by detecting new statistical dependencies in large and multisource medical datasets. These new data-driven insights may drastically improve our abilities to predict disease onset early, define sub-types of diseases, and model disease progression and patient heterogeneity at an unprecedented level of detail, thereby supporting clinical decisions. Nevertheless, the practical implementation of the vision of machine learning-guided precision medicine faces considerable clinical challenges that have to be addressed in the future. In this contribution, we will describe the envisioned role of machine learning in the context of healthcare, critically discuss the challenges faced in terms of the implementation in the clinical routine, and outline future directions of this growing field.

### 2.1.1 Introduction: The Envisioned Role of Machine Learning in Precision Medicine

Precision medicine envisions that medical diagnosis, prognosis, and interventions can be tailored to the clinical, molecular, and genetic signature of individual patients [278]. One promising path towards precision medicine is to exploit the explosion of health data with modern computational approaches, in particular machine learning. Machine learning can be leveraged to detect hidden signals in digital health data (e.g., risk factors), uncover patterns or associations with certain diseases, and evaluate the outcome of treatments or interventions. Applications of machine learning have also been proposed to facilitate early disease recognition, refine diagnosis and prognosis, support therapy decisions, and improve biomedical data management. An ever-increasing amount of data, from the molecular level of genome sequences to the level of image


https://doi.org/10.1515/9783110785982-002

phenotypes and health history, is available for rapidly growing cohorts of individuals. A prime example is the UK Biobank [658], which makes health data (genetic, molecular, imaging, clinical data) of more than half a million healthy people and patients available to the global research community. Exploring population-scale health data presents enormous opportunities for understanding disease mechanisms, ameliorating therapy outcomes, and ultimately improving healthcare. Machine learning and artificial intelligence offer the methods to mine and analyse the vast amounts of high-dimensional digital health data. For instance, designing computational models of diseases opens new opportunities to refine our understanding of diseases and their subtypes, discover novel biomarkers for early disease detection, and guide clinical decisions. A crucial step towards realizing the vision of precision and eventually personalized medicine, will be the ability of machine learning algorithms to simultaneously compute and consider a multitude of patient characteristics. The problem is exacerbated by the fact that current machine learning applications are often restricted by (1) a lack of patient data, let alone patient data with temporally-resolved clinical phenotypes; by (2) massive missingness in longitudinal, multi-modal patient data; by (3) the enormous effort required to combine data from different hospitals, with legal, information technology (IT), and data engineering challenges.

The remainder of this contribution will provide an overview of machine learning applications in the field of medicine, with a special emphasis on early disease recognition, diagnosis, prognosis, and therapy decisions. Moreover, we will critically discuss the challenges of machine learning-guided applications in the context of medicine and health care in general. Lastly, we conclude with an outlook of what the future may hold for machine learning-driven applications in the different areas of health care, such as diagnosis, prognosis, and therapy development.

### 2.1.2 Overview of Common Topics in Machine Learning in Medicine

The notion of advancing medicine by means of computation is almost as old as digital computers [438]. When deployed into the clinical routine, machine learning-guided approaches can facilitate early disease recognition, refine diagnosis and prognosis, support therapy decisions, and improve biomedical data management [161]. In this section, we discuss studies that illustrate the potential role of machine learning to tackle these tasks.

### 2.1.3 Automation of Diagnoses and Treatment

Depending on the disease type, time is often a limiting factor for the diagnosis and initiation of effective treatment. This is particularly true for patients facing serious medical conditions (e.g., cardiovascular complications, sepsis, cancer), which require

immediate attention and timely clinical decisions. A delay in the diagnostic work-up puts the patient at risk because the medical condition can get worse the longer it remains undiagnosed and untreated. To accelerate the diagnostic workup the medical field has increasingly used automation in diagnostics, surgical planning, and therapy selection (Table 2.1). Nowadays, machine learning models play an important role in the development and implementation of automation processes in the clinical routine. Large clinical datasets provide ample amounts of 'raw' data from which machine learning algorithms can derive clinically relevant insights that can inform the diagnosis or treatment selection of a patient. Specific examples, which will be discussed in detail, are the timely identification of circulatory failure [293], automatic antimicrobial resistance prediction [722], cancer tumor recognition in radiology images [28, 123], and automation of treatment planning in oncology [713].

A prominent example for automated diagnosis is circulatory failure, which occurs when the arterial pressure and capillary stream are reduced for a prolonged period of time [89]. Subsequently, the functions of supplied organs are impaired or in the worst case even lost. As circulatory failure is common in critically ill patients, monitoring of circulatory function is an indispensable aspect of the patient management in the Intensive Care Unit (ICU). Short-term effects of circulatory failure are usually reversible, whereas repeated or extended episodes of low arterial pressure adversely affect outcomes and worsen the prognosis [184, 537]. Therefore, the early recognition of circulatory failure is of highest priority. Conventional alarm systems to detect circulatory failure do not utilize comprehensive patient information, which often lead to alarms that are non-specific or false [553, 619]. False or unspecific alarms can trigger "alarm fatigue" among intensive care practitioners [97]. In the ICU, large quantities of measurements from multiple monitoring systems are generated that carry clinically relevant information. While too complex to analyze for a human brain, machine learning applications thrive in data-rich environments, such as the ICU. Leveraging clinical and ICU monitoring data, Hyland and colleagues show that a machine learning algorithm based on an array of demographic, physiological, and clinical information is able to predict the circulatory failure of ICU patients several hours prior to its onset [293]. Their early-warning system outperforms current conventional threshold-based systems and has a significantly lower false-alarm rate. In order to learn to detect deterioration events from monitoring data, which are indicative of circulatory failure, different state-of-the-art supervised machine learning techniques were employed, including Light Gradient Boosting Machine (lightGBM) [315], Logistic Regression [283], and Long Short-Term Memory (LSTM) based recurrent neural network model [277]. When implemented in the clinic, such machine learning guided multi-modal early-warning systems are a first step towards (semi-) automation of the identification of patients at risk for the development of circulatory failure in the ICU, while avoiding "alarm fatigue" among the clinical staff.

In addition to the early recognition of circulatory failure or other serious conditions (e.g., sepsis), a major challenge faced by intensive care practitioners is the escalating rates of antibiotic resistance in ICU patients. The administration of antibiotics is up to

**Tab. 2.1:** Selected examples of proposed machine learning approaches that could guide the automation of early detection, disease diagnosis, and treatment planning.

| Disease | Automated task | Input data | Analytical methods used |
|---|---|---|---|
| Antibiotic resistance [722] | Diagnostic and treatment support | MALDI-TOF mass spectra and antimicrobial resistance profiles, demographics | Logistic Regression, Light Gradient Boosting Machine, Multilayer Perceptron Deep Neural Network |
| Circulatory failure [293] | Early detection | Physiological parameters, blood values, vitals, monitoring data, demographics | Light Gradient Boosting Machine, Logistic Regression, and Long Short-Term Memory based Recurrent Neural Network model |
| COVID-19 [487] | Detection | X-ray images, demographics, clinical data | Deep Learning, Convolutional Neural Network |
| Diabetic retinopathy [364] | Early detection | Fundoscope images, diabetic retinopathy images | Deep Learning, Convolutional Neural Network |
| Hyperlipidemia [760] | Diagnosis | Blood parameters, urine parameters, biochemical test parameters, blood sugar parameters, and glycosylated hemoglobin parameters | 1-D Convolutional Neural Network |
| Laparoscopic robotic surgery [35] | Surgical path planning | Gall bladder images | Reinforcement Learning |
| Multiple sclerosis [659] | Detection | Brain MRI images, clinical data | Deep Learning, Convolutional Neural Network |
| Plastic and reconstructive surgery [329] | Diagnosis and surgical planning | 3D face surface scans, demographics | Linear Regression, Ridge Regression, Least-Angle Regression, and Least Absolute Shrinkage and Selection Operator Regression, Support Vector Machine |
| Prostate cancer [569] | Therapy selection | Prostate cancer images, clinical data | Deep Learning Neural Networks |
| Prostate cancer [464] | Therapy selection, dose optimization | Prostate cancer computed tomography images, clinical data | Generative Adversarial Network |

ten times higher in ICU patients compared with non-ICU patients [542]. Moreover, the proportion of antimicrobial resistant isolates was found to be considerably higher in ICU patients than in patients on general medical floors [27]. Antibiotic resistance substantially adds to the morbidity, mortality, and healthcare cost related to infections in the ICU [118]. Timely initiation of effective antimicrobial treatment has emerged as a critical determinant of outcome in patients with bacterial infections. The selection of optimal treatment warrants an exact characterization of the underlying pathogen, including the determining of resistance profiles. As time is of the essence, streamlining the antimicrobial resistance profiling is imperative. Matrix-Assisted Laser Desorption/Ionization Time-of-Flight (MALDI-TOF) mass spectrometry (MS) has become the standard rapid technology for the identification of microbial species, at least in specialised centers. Multiple studies have suggested that machine learning algorithms could be used to thoroughly exploit the information contained in MALDI-TOF MS spectra in order to expedite species identification and antimicrobial resistance determination [723]. However, there is a lack of comprehensive information on marker mass for all existing pathogens and antibiotics, impeding the interpretation of MALDI-TOF spectra. In a seminal study, Weis and colleagues used machine learning to harness the full potential of MALDI-TOF MS of microbial isolates to predict antimicrobial resistance [722]. Both of the implemented machine learning algorithms, Light Gradient Boosting Machine and a multilayer perceptron deep neural network, could reliably identify antimicrobial resistance of clinically important pathogens, including *Staphylococcus aureus* (*S. aureus*), *Escherichia coli* (*E. coli*), and *Klebsiella pneumoniae* (*K. pneumoniae*). Moreover, high predictive performance was observed for individual species–antibiotic combinations, such as ceftriaxone resistance in *E. coli* and *K. pneumoniae* and oxacillin resistance in *S. aureus* [722]. A retrospective clinical trial further demonstrated the clinical benefit of machine learning guided antibiotic resistance profiling. Based on the results provided by machine learning algorithms, the empiric antibiotic regimes would have changed for a subset of patients (≈ 15 %). Importantly, such a change would have been beneficial for the majority of patients. This study constitutes the first step of automatic phenotype determination, which promises to accelerate the diagnostic work up and guide treatment selection. Consequently, this will reduce the time from diagnosis to initiation of antibiotic treatment for ICU patients.

Another medical condition that will likely benefit from machine learning guided automated diagnosis is lung cancer, which is the world's the leading cause of cancer-related deaths [45]. Despite recent advances in the treatment of lung cancer, the overall 5-year survival is still low for advanced stages with distant metastases [616]. Initial symptoms of lung cancer tend to be unspecific (e.g., coughing, fatigue) and thus, are easy to dismiss as inconsequential. Consequently, the majority of patients present at an advanced disease stage when curative treatment is out of reach. Early diagnosis of lung cancer is thus imperative to increase the likelihood of survival and treatment success [45]. A successful strategy to significantly reduce the mortality is the regular screening of at-risk populations [338]. Yet, up to one third of lung nodules are missed

at the initial screening, likely owing to low the sensitivity and specificity of current screening methods as well as the limits of human vision. Imaging (e.g., X-ray, computer tomography [CT], or positron emission tomography-computerized tomography [PET-CT]) is an integral part of the diagnostic workup for lung cancer. Evaluation of the images is based on a number of imaging attributes, including the nodule size, density, and growth [511]. The detection of informative imaging features is a prime example of an area in which machine learning and artificial intelligence can excel and be of great value to the clinicians in terms of precision and time effectiveness. In particular, approaches based on deep learning [236], a branch of artificial intelligence, are an intriguing option for automating the complex image analysis to detect subtle alterations that specialists might overlook. In a seminal study, Ardila and colleagues developed three-dimensional Convolutional Neural Network (CNN) models that perform end-to-end analysis of CT images of pathology-confirmed lung cancer images [28]. Importantly, the model learns the features of interest as opposed to previous models that use hand-engineered features. Learning features have been repeatedly shown to be superior to hand-engineered features [415, 554]. The developed model was demonstrated to generate highly accurate patient-level malignancy risk predictions, which has important potential for clinical relevance. If clinically validated, the results of this study may represent a step toward automated image evaluation for risk malignancy estimation by means of deep learning. Importantly, though deep learning systems might outperform human specialists on some diagnostic tasks, they will not replace the radiologists but provide diagnostic guidance. When making a clinical decision, clinicians take into account a variety of factors that are not necessarily captured in the input data used by the machine learning model.

Along with early disease detection and phenotype detection, machine learning based automation has been demonstrated to be useful in the context of treatment planning. One striking example is Automated Therapy Planning (ATP) in patients with cancer that require radiotherapy [713]. The treatment success is highly dependent on the quality of the treatment plan. Inverse planning, a trial-and-error iterative process [512], is conventionally used to tailor radiotherapy treatment to the individual patients–a strategy that is strenuous and burdensome for patients. Machine learning and deep learning have gained momentum in the field as they are thought to improve the quality and efficiency of radiotherapy treatment planning. Specifically, the learning capability of these techniques enable oncologists to tailor the treatment plans to individual patients based on patient-specific anatomical features and by incorporating knowledge from optimization methods or physicians' behaviors. A variety of machine learning and deep learning techniques, from Artificial Neural Networks (ANN) [716], Convolutional Neural Networks (CNNs) [480], to Generative Adversarial Networks (GANs) [236], have been explored and incorporated in the different stages of radiotherapy treatment planning [44, 395, 570, 603]. While these methods promise to refine the therapy planning of various types of cancer, there are some issues relating to patient safety as well as legal

and ethical responsibilities that have to be considered before deep learning-based ATP can be implemented in the clinical routine.

### 2.1.4 Biomarker Discovery

Biomarker discovery, the search for measurable and reproducible indicators of specific clinical states, has been a major research avenue in recent years. A biomarker constitutes a measurable and reproducible indicator of specific clinical state or response to an intervention. In addition to refining disease diagnosis and prognosis [103, 139, 286], biomarkers of all sorts (e.g., molecular, digital, imaging) are instrumental in discovering and defining therapeutic targets [211, 601]. Data from various sources, including electronic health records, patient monitoring, and imaging, can be leveraged for biomarker discovery. With the emergence of affordable and time-efficient high-throughput molecular and gene expression profiling technologies (e.g., DNA microarrays and RNA sequencing) [282], the search space for biomarkers has reached unprecedented dimensions and complexity. The challenging nature of these datasets (e.g., high dimensionality with large number of noisy features and low sample size) require suitable computational models that thrive in these complex, data-rich environments. In light of that, a variety of machine learning-guided biomarker discovery strategies have gained great popularity across different fields of medicine [112, 362, 456, 667] (Table 2.2).

At the core of biomarker discovery is the search for markers that can discriminate between samples or clinical characteristics of diseased patients and those of healthy subjects. Biomarker discovery is equivalent to feature selection in machine learning [595]. Feature selection algorithms are intended to reduced the dimensionality of the feature space by removing non-informative and redundant features, while retaining the informative features [595]. In general, feature selection algorithms can be used to (i) modify representations of data by extracting informative variables (i.e., feature extraction), (ii) create probabilistic models of disease progression, and (iii) determine what specific piece of (unknown) information, for instance laboratory tests, will be most valuable in optimizing the predictive ability of a model. Broadly speaking, there are two major strategies of feature selection. The first is univariate feature selection that investigates each feature one by one to determine the strength of the relationship with the outcome variable. Popular univariate feature selection methods include linear mixed models [416], support vector machine [764], and kernel-based measures [142]. Variants of these models tackle the challenging problem of feature selection from time series data [91, 92, 320] and can account for covariates or confounders [23, 419]. The second major strategy is multivariate feature selection, which considers whole groups of features together. Common multivariate prediction models are Lasso regression models , tree ensembles (i.e., gradient boosting trees) [315], support vector machines, and Gaussian processes with kernels for comparing time series [430], neural networks from

**Tab. 2.2:** Selected examples of machine learning applications in the context of biomarker discovery.

| Disease | Use of biomarker | Input data | Analytical methods used |
|---|---|---|---|
| Alzheimer's disease [456] | Disease progression | MRI brain images, clinical data | Logistic Regression, Support Vector Machine |
| Autism [173] | Disease diagnosis | Functional connectivity, structural connectivity, behavioral data, and brain activation measures | Recursive Cluster Elimination based Support Vector Machine |
| COVID-19 [428] | Mortality prediction | Laboratory values, demographics, medical history | Support Vector Machine |
| COVID-19 [590] | Mortality prediction | Laboratory data, clinical data, demographics | Cox Proportional Hazard Model |
| Dermatitis [211] | Disease type discrimination | Transcriptomics, skin biopsies, clinical data | Multi-island Adaptive Genetic Algorithm, Principle Component Analysis |
| Diabetes [261] | Prediction of disease progression | Physiological, biochemical, and sequencing data | Decision Trees, Logistic Regression, Linear Discriminant Analysis, K-Nearest Neighbors Classifier, Gaussian Naïve Bayes, and Support Vector Machine |
| Huntington's disease [538] | Early detection | Structural and functional MRI, diffusion weighted-imaging scans | Linear Discriminant Analysis, Support Vector Machine |
| Lung cancer [734] | Early detection | Plasma metabolomic data | AdaBoost, K-nearest neighbor, Naïve Bayes, Neural Network, Random Forest, Support Vector Machine |
| Prostate cancer [284] | Screening and diagnosis | Microarray data, cancer tissue | Artificial neural network |
| Prostate cancer [139] | Diagnosis and prognosis | Proteomic data | Random Forest, Brute Force |
| Sepsis [454] | Early detection | Physiological parameters, blood values, vitals, monitoring data, demographics | Gaussian Process Temporal Convolutional Networks and Dynamic Time Warping |
| Traumatic brain injury [448] | Diagnosis | Structural MRI data, clinical data, demographics | Principle Component Analysis, Random Forest |
| Tuberculosis [363] | Disease detection | Chest X-Ray images, radiology reports, clinical data | Convolutional Neural Networks |

deep learning (long short-term memory [277], gated recurrent units [135], temporal convolutional networks [38], and attention models [702]. Several of these techniques have been successfully applied to clinical outcome prediction over recent years, especially in the area of intensive care research [216, 293, 454, 455].

The ongoing coronavirus disease (COVID-19) pandemic has been a shining example of how machine learning can support and even guide the biomarker discovery. At the beginning of the pandemic, there were no recommendations or guidelines in place on how to manage COVID-19 patients or identify patients at risk. As a consequence, physicians in emergency and intensive care units were deemed to improvise on an individual patient level and administer off-label treatments. This was particularly difficult for patients who appeared to be on a disease trajectory towards recovery, but suddenly deteriorate at a speed that does not allow for timely targeted management. COVID-19 has been associated with a high 'failure-to-rescue' rate (i.e., number of deaths of a patient following a complication) [618]. Further complicating COVID-19 disease management was the limited understanding of the different clinical phenotypes associated with COVID-19 [672] and how forthcoming mutations of SARS-CoV-2 will modify the clinical manifestation and/or responses to current off-label treatments. In combatting the COVID-19 pandemic, massive global efforts have been undertaken to determine factors that are associated with the clinical presentation of the disease and its progression [88, 205, 575], in-hospital mortality risk [428, 742], and treatment response [396]. With the availability of COVID-19-related clinical, imaging, and multi-omics data, clinically relevant biomarker signatures can be determined by means of computational modeling [428, 430, 590]. A recent study leveraged electronic clinical trial data from 69 hospitals to develop a risk-scoring system for assessing COVID-19 related in-hospital mortality risk [590]. A wide range of biomarkers (age, pre-existing cardiovascular issues, blood markers) were found to be significantly associated with mortality outcomes.

In conclusion, machine learning-driven applications can be found across various medical disciplines and bear great potential to advance health care as a whole. Nevertheless, it is important to mention that there are many challenges (e.g., data privacy, quality of the data, generalizability of the models) that have to be tackled on the road to the clinical implementation.

### 2.1.5 Biomedical Data Management

As data collection and volume surges, machine learning has emerged as a key player in the data management, easing the burden of querying data source, as well as the curation and governance of data. In the context of healthcare, machine learning-guided data management ranges from genome assembly to managing national electronic health record systems. In general, data management is a labor- and time-intensive task that often involves repetitive steps, which can be (partially) automated by means of machine learning (Table 2.3). Machine learning algorithms pursue three major data management

goals: automation of time-consuming and iterative development tasks (cataloging data, mapping sources to targets, data preprocessing), optimization of system performance (table-join approaches), and capacity management (workload-aware autoscaling).

For instance, preparing and cleaning the raw data and making it suitable for subsequent analysis is an integral part of creating any statistical or machine learning model. Data preprocessing entails different steps: datasets requests, data fusion, and data anomaly detection. Defining the quality of the data is an important step as it will directly impact the performance and reliability of machine learning algorithms. Anomaly detection aims to identify observations or data elements that raise suspicions as they significantly deviate from the majority of the data. Anomalous data can be indicative of the data-quality issues, nonstandard data, or outliers. Machine learning algorithms have the potential to automatically detect and remediate data-quality issues [138]. Specific examples of machine learning applied to anomaly detection and data cleaning are clustering [641], classification [707, 739], autoregression [758], and Bayesian statistics [162].

**Tab. 2.3:** Data management tasks that can be optimized by machine learning algorithms.

| Task | Explanation |
|---|---|
| Data cataloging and curation | To override manual data labeling by using automatic labeling [202] |
| Data preprocessing, anomaly detection | To identify missing data, help identify and fix incorrect labels [620, 641, 758]; to identify observations or data elements that raise suspicions because they significantly deviate from the majority of the data [641] |
| Data mapping | To match fields from multiple datasets into a manageable and harmonized system [66] |
| Feature engineering | To create candidate features out of a dataset from which the best can be selected and used for training [221] |

### 2.1.6 Challenges for Machine Learning Methods in the Context of Health Care

Computational innovations are bound to transform health care. Nevertheless, there are a number of challenges that have to be tackled in order to successfully adopt machine learning in the clinical setting. The challenges concern data quality (e.g., missing data, outliers), learning while preserving privacy, the interpretability and generalizability of machine learning models, and clinical implementation. In the following section, we

will elaborate on some of these challenges and potential mitigation strategies.

**Missing Data** One of the most common machine learning challenges faced is the occurrence of missing data in digital health datasets. There is a multitude of reasons why missing data occur, ranging from (human) data-entry error, missing measurements, dropouts in clinical studies, and merging unrelated data, to software errors in the data processing pipeline [285, 646, 663]. Missing data can have a significant effect on the data quality, lead to application performance degradation, cause analytical issues, and bias outcomes. In the context of medicine, the latter has been previously associated with misdiagnosis, wrong treatment decisions, and even discrimination of marginalized groups [656, 744]. Moreover, most state-of-the-art machine learning models require complete input variables. Missing data is typically handled by either the deletion of all data for an observation that has one or more missing values or the replacement with estimated values (i.e., imputation) [177]. A variety of methods have been developed that can account for different levels of sparsity in the data as well as efficiently handle missing information (Figure 2.1). Popular machine learning algorithms include k-nearest neighbors [50], multi-task Gaussian processes [729], random forest–based approaches [647, 670], matrix factorization [341, 692], discriminative deep learning methods [64], and generative deep learning methods [469, 596, 749]. Another elegant strategy of handling missing data is the use of end-to-end models that impute and predict jointly, such as Gaussian process adapter [393] and interpolation-prediction networks [610], and models that do not require imputation and can act on irregular data directly, including attention models and gated recurrent unit-decay [702, 725]. For a comprehensive review on the problem of missing values and strategies for handling missing data, see [188].

**Outlier Detection** Another noteworthy challenge is how to detect and handle outliers in a dataset. Outliers are defined as observations that raise suspicion because they deviate markedly from other observations in the given dataset. Common causes of the occurrence of outliers in digital health datasets include, measurement error, data entry error, sampling error, and natural outliers. A special category of outliers are the intentional outliers, which are dummy outliers created to assess the efficiency of detection methods. It is important to note that outliers are inherently different from noise, which is commonly defined as a random error or variance. The outlier is part of the data and can even carry (clinically) important information, while noise is simply a random error (e.g., mislabeled data, missing data). Detecting outliers in a dataset is a highly relevant task as outliers can impact the distribution of the data, increase the error variance, reduce the power of statistical tests, introduce bias, influence estimates, and impact key assumptions of statistical tests. A multitude of statistical and

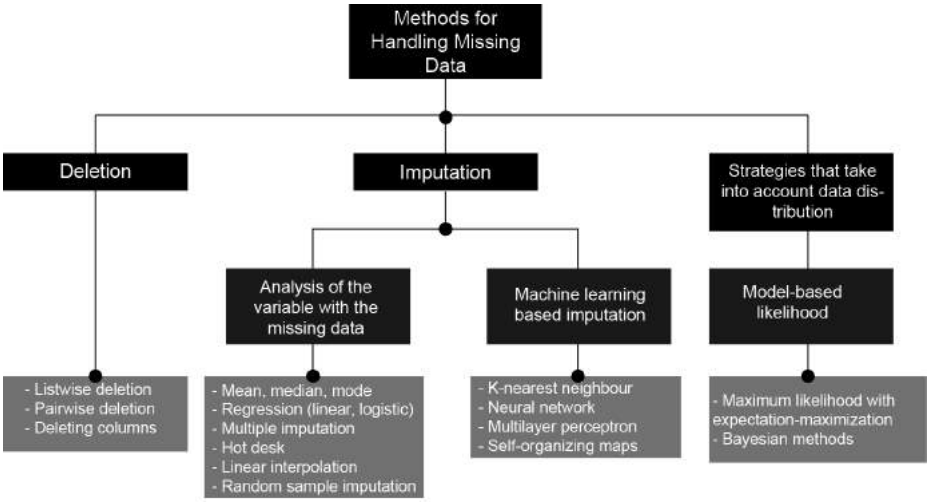



**Fig. 2.1:** Strategies for handling missing data.

machine learning methods exists for the task of outlier detection, including k-nearest neighbor [290], linear regression, naive Bayes [541], decision trees [620], and support vector machines [764]. The classic distance-based methods are empirically highly successful [104, 330, 759]. For instance, they might deem certain patients outliers if they are distant from other patients in the dataset. A patient is deemed an outlier if they are distant from a randomly drawn subset of historic patients (Figure 2.2). This scheme is extremely scalable, as it requires only the computation of distances between the patient and the small subset of historic patients, even for large clinical data warehouses. The size of the sample can even be explicitly optimized.

**Learning while Preserving Privacy** Data privacy has become one of the most important issues of our time. A breach of personal information can infringe fundamental rights and freedoms of an individual, including the risk of being identified and disclosure of personal (health) data. Data privacy breaches, such as the Facebook Cambridge Analytica scandal [287], have made patients and their caregivers reluctant to share sensitive and personal information. In response, data-privacy concerns have taken center stage and countries around the world have implemented legislation, such as the European Union's General Data Protection Regulation (GDPR) [706] and the California Consumer Privacy Act (CCPA) [488]. Medical questions that are tackled by a data-driven approach often require access to sensitive, personal information.

Major efforts have been undertaken to develop privacy-preserving machine learning algorithms that keep the patients details secure without compromising the model's performance [533, 719, 736]. Federated learning [394, 745] addresses some of these

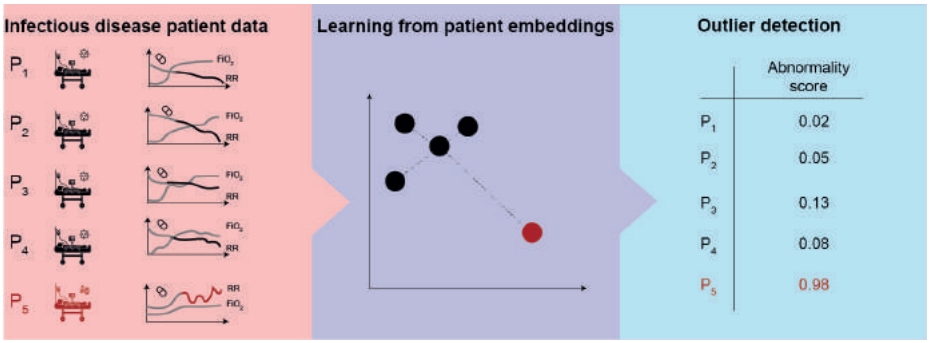



**Fig. 2.2:** Illustration of the goal of abnormality detection in COVID-19 patients (left panel). The electronic health record of a new patient P5 is checked for being abnormal relative to all patients in the clinical data warehouse (here patients P1-P4) (center panel). Relevant features are extracted to compare the similarity of the new patient to previous patients. The features describing a patient are referred to as patient embeddings (right panel). Based on the (dis)similarity of the new patient to previous patients, an abnormality score is computed. In this example, the abnormality score is the distance to the most similar patient in the database, which is largest for P5 (0.97).

concerns through training algorithms collaboratively without requiring exchange of raw data. In federated learning, the model parameters are handled centrally. To overcome this "concentration of power", swarm learning was recently introduced [719]. The principle of swarm learning is to build the machine learning models independently on data from individual sites (e.g., hospitals) and share the model parameters via a so-called swarm network. With this approach, swarm learning secures data sovereignty while preserving privacy and confidentiality. Data mining is another popular discipline in medical data science concerned with preservation of privacy [12, 18]. Also known as knowledge discovery in data, data mining is the process of automatically uncovering novel patterns and trends in big data that would otherwise remain hidden [137]. In the recent years, data mining has been successfully applied in a variety of medical disciplines: detection of diabetes [36], cancer prognosis prediction [572], biomarker discovery [244], sepsis [91], and prediction of stroke mortality [186]. As data privacy should be preserved at all costs, numerous privacy-preserving data-mining methods have been developed. These include randomization [183], classification [130], clustering [229], association rule [539], K-anonymity [134], L-diverse [425], distributed privacy preservation [182], condensation [10], and cryptographic [376, 501]. A comprehensive survey on the contributions of privacy preserving data mining techniques can be found in [16, 567].

Access to data generated within the healthcare systems is often restricted due to privacy and confidentially concerns. A strategy to make sensitive health data available is to de-identify or anonymize the data by deleting or encoding identifiers that link individuals (e.g., names and patient identifier), by perturbating the data (e.g., applying round-numbering methods and adding random noise), by swapping data (e.g., shuf-

fling dataset attribute values), or by generalizing information (e.g., grouping variables). Even with these de-identifcation efforts, it remains extremely difficult to guarantee that the re-identification of individual patients is not possible. A promising mitigation strategy is to generate synthetic, representative, data that can be safely shared. Synthetic data can both be used to augment datasets and generate artificial, but realistic patient data that can be shared, even across geographical and political borders. For this form of synthetic data generation, Generative Adversarial Networks (GANs) [235] are well suited. Briefly, GANs are a type of deep learning model that consists of two networks one called the generator and the other called the discriminator. These two networks are simultaneously trained competitively, as in a zero-sum game framework. The generator learns how to map from a latent space to a data distribution of interest, i.e., generate candidates of synthetic patients, and the discriminator evaluates the candidates distinguishing from the true distribution. In the context of clinical data, medical GAN (medGAN) [117] is a recent approach that can generate high-dimensional discrete variables via a combination of variational autoencoders (VAEs) and GAN. Furthermore, medical Wasserstein (medWGAN) and boundary-seeking GAN (medBGAN) improved the performance of medGAN to generate synthetic data from the "Medical Information Mart for Intensive Care" database and the Taiwan National Health Insurance Research Database [144]. Instead of a general GAN, medWGAN uses an improved generative network named WGAN-GP, where the model overcomes the issue of fails to converge in some settings owing to the use of the weight-clipping technique using gradient penalty [43]. Finally, medBGAN improves GAN training to create new samples that lie on the decision boundary of the discriminator at each update [43].

**Interpretability and Generalizability of Models** In addition to being applied in complex high-stakes settings such as medicine, machine learning algorithms are also becoming increasingly complex in terms of their architecture. At times, these algorithms become so complex that the humans forfeit comprehension of the underlying models or how variables are jointly related to make predictions. Modern machine learning algorithms are commonly referred to as "black boxes". The alleged black box nature constitutes a major barrier to the adoption of machine learning in the clinical routine. But what is the reason for not trusting a machine learning model that has been proven to perform well and can accurately diagnose patients? "The problem is that a single metric, such as classification accuracy, is an incomplete description of most real-world tasks" [179]. The interpretability of machine learning models is critical to understand the accuracy of findings, identify variables that drive the predictions, improve model performance, guarding against embedded bias, and debug models. Medicine is among the domains where scientists are often compelled to implement simpler and interpretable machine learning models (e.g., linear models or decision trees) as every decision being taken by the model has to be interpretable. Clinical staff needs to understand why a

machine learning algorithm generates the results it does (i.e., interpretability) and ideally how it arrives at its conclusions (i.e., explainability) [190]. Better interpretability might come at the expense of model performance. As biological associations are hardly ever of a linear nature, complex models, including ensembles and neural networks, typically result in more accurate performance. Incomplete interpretability is often compensated with judgement, knowledge provided by domain experts (e.g., clinicians), rigorous monitoring, and diligent understanding of the data used. The development of methods to enhance the interpretability of machine learning models is a vivid area of investigations. As a matter of fact, numerous model-agnostic interpretation tools exist that can be applied to any supervised machine learning model [529]. There are two major categories of model-agnostic methods. First, local methods that describe individual predictions and secondly, global methods that explain how features affect the prediction as a whole. Table 2.4 provides an overview of common local and global model-agnostic methods. These model-agnostic interpretability methods allow researchers to interpret the results of (complex) machine learning models and can pave the way for the implementation of such models in the clinical routine.

**Tab. 2.4:** Common local and global model-agnostic interpretation methods of machine learning models. For a comprehensive overview on interpretable machine learning, see [453].

| Local methods | Global methods |
|---|---|
| Individual Conditional Expectation (ICE) [233] | Partial Dependence Plot (PDP) [213] |
| Local Surrogate (LIME) [527] | Accumulated Local Effects (ALE) Plot [25] |
| Counterfactual Explanations [711] | Feature Interaction [214] |
| Scoped Rules (Anchors) [528] | Functional Decompositon [281] |
| Shapley Values [661] | Permutation Feature Importance [102] |
| SHAP (SHapley Additive exPlanations) [661] | Global Surrogate [152] |

Apart from being interpretable, generalizability is a desired attribute of machine learning algorithms. Generalizable refers to the ability of a trained algorithm to perform well on unseen data. One particularly elusive challenge regards generalizability across different patient groups [489]. There are numerous examples of promising machine learning applications that struggled when applied to diverse populations. Google, for instance, introduced a machine learning algorithm for the diagnosis of diabetic retinopathy that performed poorly in India [7]. This is likely attributable to the fact that the algorithm was developed, trained, and evaluated on a dataset that lacked the necessary ethnoracial and demographic diversity. Another example of unintended effects of artificial intelligence is an algorithm that was developed to detect skin cancer on images of skins. While the algorithm performed well on fair skin, it was not able to reliably diagnose lesions on darker skin [240].

These examples highlight the importance of data from diverse groups (i.e., in terms of sex, ethnic background, and race) are fundamental to realize universal precision medicine. The reality, however, is that data is often derived from a worryingly small and homogeneous sample of the population (e.g., white male individuals). As a result, ethnoracial disparities are evident in many patient populations and include differences in access to care, time to diagnosis, treatment, and mortality [545, 566, 748]. In order to eliminate bias and create fairness and equity, scientists have to be conscious of bias that can occur at different levels, namely data collection and selection, model development and evaluation, as well as model deployment and clinical implementation. Data is driving force behind any machine learning and artificial intelligence algorithm. That is why, the data underlying the development and evaluation of algorithms must be unbiased and representative of the target populations to avoid generating or perpetuating biases that may worsen patient outcomes. Often bias is rooted in systematically skewed data collection, e.g., through clinical trials predominantly carried out with white male participants, or the reliance on historical data that might have been subject to biased data generation or clinical practices. To mitigate bias in the data, diverse and well balanced study populations are crucial for any collection and/or selection of data (e.g., clinical trials, registry, electronic health records). Particular attention should be paid to ethnoracial diversity, sex/gender balance, socioeconomic equity, and other social, and ethical, determinants of disease and access to healthcare. Assuming that the available data is unbiased, researchers have to carefully select the data variables to avoid introducing a bias in the phase of algorithm development.

If possible, algorithms can be tested on different patient populations for both scientific and ethical performance. Ideally, the development, evaluation, and clinical implementation of algorithms is done in liaison with clinicians to ensure that the algorithms do not exhibit bias in the clinical setting. Lastly, the healthcare system, in which machine learning tools are implemented, is an important entry point of bias. Awareness of inherent biases of machine learning assisted tools among healthcare professionals is pivotal to mitigate bias. This starts by ensuring equitable patient access to the technology, and then diligently observing how it performs in diverse populations and underrepresented communities. Any bias noticed should immediately be reported to an appropriate committee at the hospital, which can then communicate with the developers. Understanding bias inherent in medical technology allows clinicians to question the accuracy of the technology if the results do not meet the expectations from their clinical expertise.

**Clinical Implementation and Validation** Considerable technological progress has been made over the last decades with machine learning applications transforming the clinical decision making and how health resources (e.g., data) are managed. Nevertheless, the greatest challenge of machine learning applications is not the medical utility,

but rather the implementation in the clinical routine. While these developments are crucial to advance health care, they raise a number of ethical and legal ethical concerns that machine learning and artificial intelligence might harm patients and/or clinicians. Technological and diagnostic failures can lead to adverse events or seriously harm patients. It is not yet clear who is liable for malfunctions in, or erroneous decisions made by artificial intelligence-based clinical tools that result in inaccurate or delayed diagnosis [510]. The attribution of accountability becomes even more complicated when an artificial intelligence-based clinical tool gives a wrong treatment recommendation, yet the clinician makes the final decision. Is the clinician liable or can the liability be delegated to a company or person that engineered the tool? In [510], Price et al. provide an overview of potential scenarios and associated probable legal outcomes related to AI use in clinical practice. Under the current law, clinicians are shielded from liability as long as they do not deviate from the standard of care [532]. As a consequence, clinicians are advised to utilize machine learning-guided tools to support and confirm existing decision-making processes as opposed to solely relying on computational algorithm output for diagnosis or treatment selection [441]. The complexity further increases when considering that there are multiple stakeholders in the ecosystem of liability, including the healthcare institutions that purchase and implement computational algorithms.

### 2.1.7 Future Directions of Machine Learning in Medicine

The accelerating generation of unparalleled amounts of health data will lead to fundamental changes in medicine and health care. Machine learning applications are poised to play an increasingly prominent role in medicine. Specifically, they will facilitate early disease recognition, refine diagnosis and prognosis, support therapy decisions, and streamline biomedical data management. Importantly, machine learning-guided systems will not replace clinicians or therapists, but will augment their efforts and time to care for patients thanks to guidance for clinical decision-making and the automation of time-consuming and repetitive tasks. As of yet, many barriers exist to the adoption of such applications in the clinical routine. In the coming years, the data explosion will continue and reach unparalleled dimensions, including the number of patients (e.g., UK Biobank), the length of time series (e.g., ICU monitoring data, wearable devices), and the breadth of data type (from molecular to higher-level phenotypes). This affluence of rich datasets will open new avenues for machine learning-driven applications to assist in clinical decision-making. In addition to refining clinical processes, machine learning and artificial intelligence will also play a pivotal role in other important areas of biomedical research, such as protein structure prediction, molecule design, drug discovery, or single-cell research. A groundbreaking example is the recently introduced machine learning approach *AlphaFold* [308], which performs predictions of protein structure with unprecedented accuracy, by incorporating physical and biological knowl-

edge. Besides the technological advances and the availability of vast amounts of data, overcoming the challenges of handling missing data and outliers, preserving privacy, interpretability, and clinical application will be critical for the adoption of machine learning-supported guidance tools in clinical routine.

## 2.2 Virus Detection

*Roland Hergenröder*
*Frank Weichert*
*Konstantin Wüstefeld*
*Victoria Shpacovitch*

**Abstract:** Amid the accelerating spread of viral diseases throughout the world, the rapid detection of pathogens is of essential importance. Viruses can be transmitted very quickly via contacts in public transportation or at large social events. Therefore, a rapid virus test system for such crowded locations is highly desirable. The plasmon-assisted microscopy of nano-objects (PAMONO) sensor is one such analytical instrument. The sensor required the development of software and optomechanical parts to detect viruses in complex biological liquids. While the focus lies on viral particles, other nanoparticles can also be analyzed by employing a similar principle. The latter issue vastly expands the potential application field of the PAMONO sensor. The developed methods are tailored to the spatiotemporal characteristics of the underlying sensor system, making use of the adaptivity of machine learning approaches. As a result, 80 nm to 300 nm particles can be detected in signals with different types of imaging artifacts and different resolutions, reaching accuracies of over 80 % with respect to the expected particle counts of test samples. For mobile use as a rapid test system, resource-saving and real-time capability are of similar importance to make the device accessible in as many application areas as possible. Multi-objective optimization in terms of detection quality and energy consumption was applied to demonstrate that the usage as a mobile system is feasible.

### 2.2.1 Introduction

For the detection of nanoparticles such as virus particles a device that can make them perceptible is required. Such a device is the PAMONO sensor [385, 607], which pumps a liquid or air sample through a flow cell to reveal the particles of interest contained in it. This is achieved by making use of what is known as the Surface Plasmon Resonance (SPR) phenomenon [340].

Special proteins—antibodies—immobilized on a gold sensor film help to accomplish the specificity of viral particle detection. Particles that bind to the antibodies cause local changes in the reflection conditions near the surface of the film. Due to locally increased reflection, particle binding events become detectable by a charge-coupled device (CCD) [389] integrated into the PAMONO sensor.

The limits of a manual analysis are reached quickly, when trying to identify particle signals. Even after enhancing the visibility of particles in the recorded images by preprocessing, finding particle regions is a tedious and time-consuming task for a human observer. Test analyses determined that it takes an expert approximately two days to accurately analyze a dataset for an experiment of around 4000 images with varying results for different particle sizes, different levels of disturbances, and different human observers [613]. This time range, the need for visually trained experts, and the deviations in the subjective perception of different persons predestine this task to become the subject of automated analysis to enable the PAMONO sensor to be used as a rapid test.

To reduce manual interactions and to enable quick testing by non-expert users, more adaptive solutions from the field of deep learning were developed to adapt to specific signal characteristics while tolerating deviations that inevitably occur between different recordings when operating outside a controlled environment.

While manual interactions are shifted from on-site usage to training time, these approaches make actual on-site testing faster. The challenge which arises in return is the high amount of manually annotated training data to learn the patterns of interest. At the same time, the recording and annotation of new experiments cause material and time costs. This is a problem that is worth addressing since it can be observed in various tasks of medical data analysis. Dealing with the limited availability of training data while leveraging the generalization of machine learning is, therefore, a key aspect in this area.

There are three major challenges to be addressed when detecting nanoparticles in samples: dealing with varying artifact characteristics and intensities, real-time capability, and the feasibility of mobile usage. Considering the on-site operation, the concept of mobility again contains the aspect of resource optimization in terms of computing power and energy consumption. We present an approach for multi-objective optimization of parameters in the employed algorithms. This optimization can target high detection accuracy or low energy consumption. Alternative approaches based on deep learning overcome the need for defining specific operators by learning them from more general functions. The aspects of the application under natural conditions, including the analysis of physical particle sizes, are viewed with particular attention. The analysis of physical particle sizes is also described in that context. It can provide a more accurate classification of the contained particles and enables plausibility checks by taking domain knowledge about the specific types of particles into account.

Here is a short overview of the sections below: Section 2.2.2 provides details on the different types of particles. The setup of the PAMONO sensor is detailed in Section 2.2.3. Section 2.2.4 describes the underlying data characteristics and the classic and deep learning-based methods for the detection of nanoparticles. Section 2.2.5 presents an approach developed for the multi-objective optimization of parameters used in an operator. Section 2.2.6 describes aspects of the application in natural environments with particular emphasis on determining the size of the analyzed particles and an ap-

proach that increases the robustness of image analyses based on generative adversarial networks. Finally, Section 2.2.7 provides an outlook on potential approaches to improve the hardware and software of the PAMONO sensor.

### 2.2.2 Types of Detectable Nanoparticles

It is first worth noting which types of particles the PAMONO sensor can analyze. Different physical and biological characteristics can be spotted, that are attributable to biological Nanoparticles (bio-NPs). Besides particles of interest such as viruses, Virus-Like Particles (VLPs) [754], and Extracellular Vesicles (EVs) [504, 505], there are interfering objects can also be observed, such as lipid and protein agglomerates, which are often considered contaminating substances hampering the bio-analytical examination of samples.

While viruses are well-recognized as the smallest infectious agents containing only one type of nucleic acid, Ribonucleic Acid (RNA), or Deoxyribonucleic Acid (DNA) [332], VLPs, and extracellular vesicles are significantly less analyzed. It is important to highlight the key difference between VLPs and viruses: VLPs do not possess any nucleic acids and, thus, lack a principal opportunity to reproduce themselves in the host organism. On the other hand, VLPs carry the same antigens (molecules considered foreign by the immune system) on their surface as the corresponding native viruses [754]. Thus, VLPs in science can serve as a safe and reliable model of dangerous viruses since VLPs efficiently mimic the structural properties of corresponding viruses but cannot replicate. In practice, VLPs are well known as commercial medical products serving as a basis for vaccines [754].

Another group of bio-NPs, Extracellular Vesicles (EVs), have recently started to attract the attention of scientists and physicians. EVs are submicro- and nano-sized vesicles released by the majority of cells [505]. Another principal feature of EVs is their ability to carry different molecules inside as well as on their surface. Among such active molecules are hormones, growth factors, active peptides, and nucleic acids [505]. Their cargo makes EVs active messengers participating in intercellular communication and reflecting cellular status under normal conditions or during the pathological processes. Moreover, the abundance of EVs in body fluids such as blood or saliva drew the attention of clinicians and medical researchers, who harness EVs as a means of drug delivery or to estimate their potential as biomarkers of the progression of a disease.

However, any analysis of bio-NP samples for scientific and practical needs requires the selection of reliable techniques and instruments. Certainly, bio-NPs have to be swiftly characterized for their abundance in a sample and their size. From a different perspective, biochemical information regarding their surface antigens (proteins) and their content is of interest as well. A simultaneous quantification and determination of the sizes of bio-NPs can be achieved by the principle of surface plasmon resonance (SPR). The plasmon-assisted microscopy of nano-objects (PAMONO) sensor, which exploits

this principle, is an instrument for label-free and specific detection of individual bio-NPs in solutions [245, 772]. In the following section, the sensor and the underlying principle for visualizing particle signals are introduced in more detail.

### 2.2.3 PAMONO Sensor

Surface plasmons can be thought of as propagating electron density waves. These waves can be excited by incident light (usually by an incident laser beam) in a thin metal film at a dielectric-metal interface. It is precisely this thin metal film that serves as a sensor surface. Surface Plasmon Resonance (SPR) is a physical phenomenon that served as a basis for the development of the PAMONO sensor.

Conventional SPR biosensors deal with measurements of the layers of bio-molecules formed onto the sensor surface, and thus, conventional SPR sensors are not applicable for the detection of individual Nanoparticles (NPs). By contrast, the special quality of the PAMONO sensor is exactly the ability to detect the binding of individual NPs to the gold sensor surface [772]. Kretschmann's scheme of plasmon excitation is utilized in the PAMONO sensor as well as in the majority of conventional commercial SPR-based biosensors [350]. However, there are specific issues that distinguish the PAMONO sensor from known conventional SPR biosensors. In Kretschmann's scheme, shown in Figure 2.3, a p-polarized light (polarization of the electric field occurs in the plane of incidence) illuminates a glass prism with a very thin (tens of nanometers) noble metal film deposited on the base of the prism. Often a superluminescent diode or a diode laser is used as a source of light [350]. Surface Plasmons (SPs) are excited as propagating electron density waves at the metal-dielectric interface in the presence of p-polarized incidence light at a particular angle [328, 577, 693]. This event occurs when the energy of an incidence beam transforms into electron-polaritons within the thin metal film deposited on the prism. SPs excited along the metal-dielectric interface result in a substantial reduction of the reflection intensity. In turn, this fact leads to changed reflection conditions, which are extremely sensitive to any refractive index changes occurring close to the metal-dielectric interface [577]. Such changes can be caused by the adsorption of molecules onto the metal film surface. SPR-based biosensors harness this trait and enable the analysis of interactions between bio-molecules immobilized onto the metal film surface and their counterparts in the analyzed liquid sample. Such analysis can be performed in real time and without labeling the target molecules. Thus, it is not surprising that conventional SPR biosensors are actively used to measure binding constants and the kinetics of bio-molecular interactions, and to perform concentration measurements [577].

The PAMONO sensor also harnesses the most convenient scheme of plasmon excitation: Kretschmann's configuration [350]. Figure 2.4 shows a photograph of the device setup and the flow cell as the core of the apparatus. The entire device fits into a suitcase-sized enclosure.

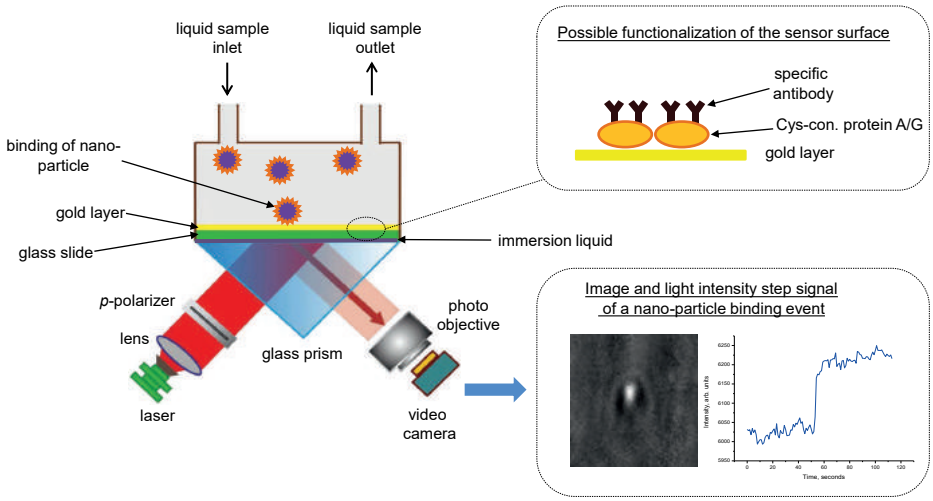
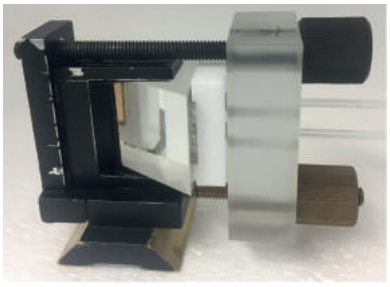



**Fig. 2.3:** Schematic setup of the PAMONO sensor (left), abstract view of a used antibody coating (top right), preprocessed image of an attached particle clearly visible as a bright elliptic region (bottom right, left image), and the average pixel intensities of this area over time (bottom right, right image); modified from [669].

Moreover, the events preceding the substantial reduction of the reflected light as well as the events leading to the restoration of reflection are similar for the PAMONO sensor and conventional SPR-based sensors. In the case of the PAMONO sensor, such events occur locally, in the spot of NP binding, not on the entire sensor surface as it happens in the case of classic SPR sensors [773]. A developed model [773] explains key differences in physics between the detection of bio-molecule layer formation (classic SPR sensor) and individual NPs (PAMONO sensor). Polystyrene NPs were employed as a model system [245]. The use of these particles helped to demonstrate linear dependency between the number of signals detected by the PAMONO sensor and the concentration of particles in liquid samples. In turn, this fact confirmed the applicability of the PAMONO sensor for the concentration measurements of NPs, in which NP concentration is expressed as a number of particles in a volume unit [245]. The work of Shpacovitch and colleagues [608] focused on the bio-analytical features of the PAMONO sensor and demonstrated the ability of the PAMONO sensor to detect not only HIV-VLPs (100 —-140 nm) but also influenza A viral particles (80 —-120 nm).

The selectivity studies were performed in phosphate buffered saline (PBS buffer) employing specially engineered HIV-VLPs of two types: one containing target protein on the surface and one lacking it [608]. Under these conditions, the selectivity of HIV-VLPs binding to the PAMONO gold sensor surface reached 90 % without special treatment of the sensor surface with substances preventing the binding of non-target VLPs [608]. Moreover, the ability of the PAMONO sensor to work with biological samples containing serum was also demonstrated [608].

**(a)** The real setup of the PAMONO sensor corresponding to the schematic setup shown in Figure 2.3. The case of the instrument is approximately the size of a suitcase.



**(b)** The flow cell with mounted gold-coated glass plate attached to a prism base. The tubing system serves to guide a liquid sample into and out of the flow cell.

**Fig. 2.4:** Photos of the PAMONO sensor (a) as a whole and of the flow cell (b) as the heart of the device individually.

Further work [609] proved the power of the PAMONO sensor for the detection of the other type of bio-NPs: extracellular vesicles. The authors employed cysteine-conjugated protein A/G for the functionalization of the PAMONO sensor surface. This was done to allow for the elution of bio-NPs captured on the sensor surface and, thus, enable a post-PAMONO analysis [609]. Moreover, the PAMONO sensor was capable of supplying sufficient information for the sizing of studied polystyrene nanoparticles. Such information could be extracted from the intensity step signal caused by NP binding. It is important to mention that the Nanoparticle Tracking Analysis (NTA) instrument Malvern Panalytical NanoSight LM10[1] was used as a reference method in the studies performed with the PAMONO sensor. Thus, it was also necessary to verify the accuracy of the LM10 instrument before its use in the studies. This work was performed by Usfoor and colleagues [696]. It was demonstrated that NP size measurements performed by the LM10 device are quite accurate, but concentrations were not determined precisely. Moreover, the NTA analysis of bio-NPs requires the labeling procedure of target particles, while the PAMONO sensor provides results employing a label-free approach. In detail, the drawbacks and advantages of the PAMONO sensor and other SPR-based platforms for the sizing, quantification, and biochemical analysis of extracellular vesicles are given in a review work [606]. One of the advantages of the PAMONO sensor is the possibility of NP quantification without prior calibration, as as shown by Kuzmichev and colleagues [361].

The analysis of sensor data requires the use of robust detection algorithms that can adapt to data variations in real use cases. Approaches that incorporate this criterion are presented in the following sections.

**1** Malvern Panalytical NanoSight LM10, https://www.malvernpanalytical.com/en/products/product-range/nanosight-range/nanosight-lm10, accessed 31 March 2022.

### 2.2.4 Automated Nanoparticle Detection

Based on the challenges of nanoparticle detection, a signal model was developed to represent the individual components of the total signal

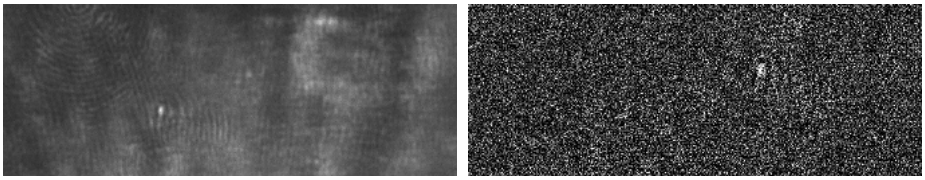
$$I(x, y, t) = B(x, y) \cdot A(x, y, t) \cdot P(x, y, t) + R(x, y, t) \tag{2.1}$$

which consists of the background signal $B$, which is constant within a recorded image, an artifact signal $A$, the signal of interest $P$, and a residuum $R$, which includes random noise [614]. Figure 2.5 shows an example of an unprocessed recording and the corresponding image in which the contained particle signal is made visible in a *preprocessing* step by removing the background $B$ and reducing the noise $R$. This is achieved using a sliding-window approach that averages signal values, amplifies pixel intensities increasing over time, and weakens constant or falling intensities. Subsequently, a dynamic contrast enhancement is applied in some approaches to further emphasize particle signals [733]. After that, the regions of interest can be spotted as elliptical areas that are brighter than their environments. Based on the presented signal model, the goal of the automated detection is to make the pixel values of signals of interest $P$ more distinguishable from their surroundings. Typically this is done by employing approaches that attempt to highlight particles directly and, in some cases, by weakening artifact signals beforehand.

From the algorithmic point of view, the detection of nanoparticle signals on preprocessed images can be seen as a combined task consisting of blob detection [639] and time series analysis as particles can only be reliably recognized when their spatial features like region size and shape as well as their temporal behavior are taken into account. The characteristic temporal behavior, which can be described as a step-like curve, is shown in Figure 2.6 for two images from sets with different particle sizes. From the algorithmic point of view, the detection of nanoparticle signals on preprocessed images can be seen as a combined task consisting of blob detection [639] and time series analysis. This is because particles can only be reliably recognized when their spatial features like region size and shape as well as their temporal behavior are taken into account. The characteristic temporal behavior, which can be described as a step-like curve, is shown in Figure 2.6 for two images from sets with different particle sizes. The origin of the temporal characteristics lies in the way a particle of interest interacts with the antibody layer. When such a particle attaches to it, it remains in contact for a prolonged time. It is assumed that, in the time of one recording, particles of interest stay attached permanently while other particles only cause short-time peaks in the measured intensities as their physical shape does not match the specific antibodies applied to the gold film (see Section 2.2.3).

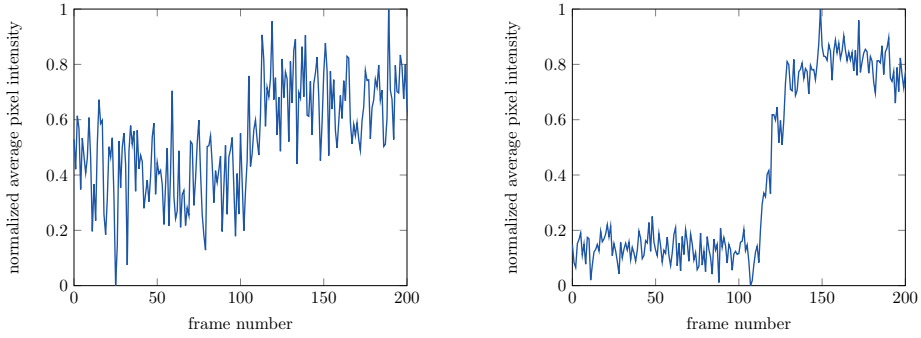
A closer look at intensity curves belonging to particles of different sizes reveals that a larger particle causes a higher intensity difference

$$I(x, y, t_j) - I(x, y, t_i), t_i < t_j$$



**(a)** Unprocessed **(b)** Preprocessed

**Fig. 2.5:** Comparison of (a) an unprocessed, recorded image and (b) the same image after preprocessing, which enables detecting particles as ellipsoids with high pixel intensities compared with the background. The particle that can be seen in the preprocessed image can hardly be detected in the unprocessed image data. To improve the visibility of particles, the preprocessing has to make use of temporal information from a time window around the current frame.




**(a)** Intensity of an 80 nm particle over time. **(b)** Intensity of a 200 nm particle over time.

**Fig. 2.6:** Exemplary comparison of two signals over time belonging to different particle sizes.

for a position $(x, y)$ and times $t_i$, $t_j$ than a smaller particle. With smaller particle sizes, the particle signal $P$ becomes weaker, while inaccurate adjustments of the optical instruments, less clean samples, and increasing external influence imposed on the device increase the intensities of the artifact signal $A$ and the residuum $R$. This results in a lower signal-to-noise ratio (SNR) [127]

$$SNR(P, Q) = \frac{|\mu(P) - \mu(Q)|}{\sigma(P)} \tag{2.2}$$

that is determined based on the particle signals $P$ and the non-particle signals $Q := A \cup R$, where $\mu$ is the average value and $\sigma$ is the standard deviation of a set of intensity values. While not suitable for analyzing structured artifacts, the measurement can illustrate the visibility of particles in an image that predominantly contains random noise besides the particle signals. A low SNR value indicates a bad detectability of particles caused by a low intensity of particle signals $P$ or a high intensity of random noise $R$. Specifying a metric indicating the strength of the influence of structured artifacts would require

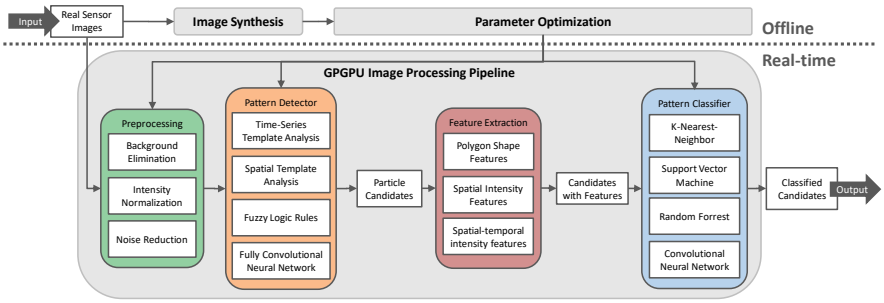



**Fig. 2.7:** An abstract processing pipeline for PAMONO images showing different methods that can be used for preprocessing, pattern detection, candidate feature extraction, and classification [385].

a more complex definition that is less interpretable. For this reason, the SNR value is used when a directly understandable indication of particle visibility is desired.

#### 2.2.4.1 Stages of Detection

With respect to the characteristics described in Section 2.2.4, different methods exploiting spatial and temporal features for nanoparticle detection were evaluated. Figure 2.7 shows the stages of an abstract detection pipeline containing alternative approaches for the different stages. In general, the task of automated nanoparticle detection can be divided into different subtasks. The first step of any presented detection approach is preprocessing with the goal of image restoration based on a signal model. In addition, image enhancement techniques are used to improve the distinguishability between particles and other signals. With the preprocessed images at hand, a segmentation of the image areas and the detection of candidate regions takes place, which is summarized as the *pattern detector*. Then a *feature extraction* method determines features that are used in a *pattern classifier* to check the properties of candidates so that non-particle candidates can be filtered out.

The whole pipeline can be optimized for a specific criterion using the offline parameter optimization described in Section 2.2.5.1.

For preprocessing images, a sliding-window method with a fixed window size in different variations [399, 733], as well as a constant background removal method [613], have shown to be effective in different approaches. Depending on the downstream algorithm for detection, additional noise reduction techniques such as Gaussian or median filtering [613], wavelet denoising [401], brightness correction [399], or dynamic contrast enhancement [733] are used to provide a better separability between particles and other signals.

While a variety of approaches is applicable for subsequent tasks, different approaches offer diverse strengths and weaknesses for different data characteristics. The

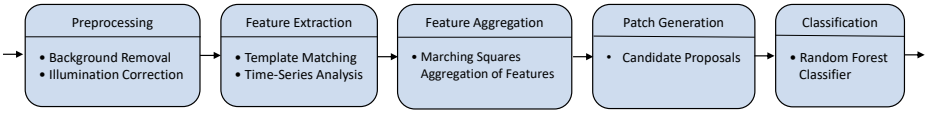



**Fig. 2.8:** Example of a classic pipeline for the detection of nanoparticles based on template matching and time-series analysis for the generation of particle candidates and the evaluation of polygon shape features in a random forest approach for classification (adapted from [399]).

following are available operations for the subtasks of particle detection. An appropriate selection of operations is presented in Section 2.2.5.1.

For the task of pattern detection, (fuzzy) template matching in different variants [399, 400, 401, 613] and convolutional network approaches [387, 733, 746] were developed. With particle candidate regions generated by the pattern detector, feature extractors obtain spatial or spatiotemporal features, which are then evaluated by a pattern classifier. Extractors generate, for example, polygon shape features like the covered area or the circularity of a polygon [399] or measures describing representations in other spaces like the Fourier or wavelet space [746].

Evaluated classifiers are, for example, k-nearest-neighbor [615], support vector machines (SVM) [615], random forests [387], and convolutional neural networks [387, 733].

In the end, a connection between regions to single particle traces distributed over consecutive frames is established. This is particularly important when counting the detected particles is desired instead of just detecting if particles are present at all.

A concrete example of a classic detection pipeline is shown in Figure 2.8. It utilizes template detection and fuzzy time series analysis [399] for pattern recognition and classifies particle candidate patches with random forests based on polygon shape features.

Template matching, which has proven effective in various examples, uses a previously recorded particle region as the predefined template patch $T$ to detect similar regions of the same size in the current image $I$. This is done by calculating a normalized cross-correlation

$$R(x,y) = \frac{\sum_{x',y'} T(x',y') I(x+x',y+y')}{\sqrt{\sum_{x',y'} T(x',y')^2 + \sum_{x'} \sum_{y'} I(x+x',y+y')^2}}.$$

for each pixel position (x,y) in the image with the template patch [100, 399]. In simplified terms, the template patch is moved over the image while the correlation of the template patch with the underlying image area is determined at each position.

Although using classic methods like template matching with optimally adjusted parameters can lead to high detection accuracies, slightly outperforming some neural network approaches [385], it has to be noted that this is only possible if the parameters are individually optimized for each change in the setup and, in the worst case, for each dataset [385, 613]. This is a disadvantage for use as a rapid test on-site since

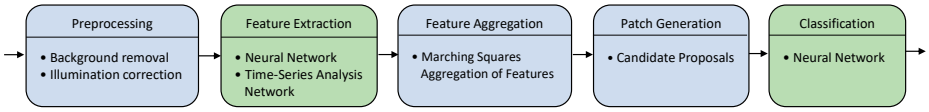



**Fig. 2.9:** Modification of the classic detection pipeline shown in Figure 2.8 for the use of neural networks. Template matching, time-series analysis, and the random forest classifier were replaced by a neural network. These replaced modules are colored green.

changes in the characteristics of artifacts have to be covered without the need for manual adjustments for each dataset. This demand leads to the employment of neural network approaches, which are presented in the following section.

#### 2.2.4.2 Spatiotemporal Deep Learning

The detection of signals of interest from a real-world measurement always comes with irregularities. Changes in the environmental influence have to be taken into account, as well as the varying cleanliness of the samples. Dealing with these influences requires a technique that can adapt to the changes while concentrating on the typical characteristics of the particles of interest. At this point, the advantages of deep learning-based methods can be exploited. Instead of adjusting to a restricted scenario, a deep learning-based approach can take advantage of previous recordings by using them to approximate patterns that are typical for a particle of interest. Rather than calculating the features based on a static method, neural networks choose from a large set of possible feature extraction operations limited only by the number of freely learnable parameters and their architecture. Several deep learning approaches have been applied to the PAMONO recordings to evaluate their detection performance [385, 387, 733, 746].

**Deep Learning Integration Into the Detection Pipeline** There are different architectures of neural networks that can be integrated into the pipeline presented in Section 2.2.4.1. One pipeline modification using neural networks created to improve the flexibility of particle detection is presented in Figure 2.9. Several modules of the former pipeline shown in Figure 2.8 are replaced by neural networks for spatial or temporal classification of the respective inputs.

The key difference between the deep learning approach and the use of more direct methods like template matching is the way the parameters are used. By the layer-wise connection of learnable operators, this adaptation takes place on lower levels, such as edge detection, as well as on higher levels where possible particle shapes learned from training data can be correlated with a given image. For this purpose, backpropagation [264] is used in combination with a loss function, which in this case is the cross-entropy loss that is predicted in the current training step [387]. The whole process aims at the minimization of the evaluated loss functions, that is, the creation of a minimal divergence between predicted and expected classes.

The presented version with parts of the operations replaced by neural networks was able to achieve results on data with a median SNR value of 0.7 (see Equation 2.2) for images that were mainly affected by random noise. With the classic pipeline, this level was only possible on datasets with a median SNR value of 1.25. Although achieving slightly worse results compared with a template-matching approach which was optimized for each dataset separately, the neural networks can cover different situations without having to be adjusted anew [385, 733]. This property is highly desirable when handling shifts in the data characteristics, which are described in more detail in Section 2.2.6.2.

**Deep Learning-Focused Pipeline** The modified pipeline described in the previous paragraph increases the adaptability to changed characteristics of nanoparticle signals by replacing some classic operators with neural networks. At the same time, the modified structure still relies on inflexible methods for the extraction of candidate regions and patch extraction. The next step towards a completely adaptable structure is the employment of a neural network that is capable of proposing candidate regions by itself. In this way, different sizes of particle regions can be taken into account while learning the characteristic features of particles. When evaluating the architecture on datasets with different characteristics due to changing optical instruments and different image resolutions, accuracies of over 80 % could be achieved without adjusting parameters between analyses [733]. The pipeline which was proposed to achieve this goal is illustrated in Figure 2.10. It puts a stronger focus on the use of learned functionalities. The first step of detection with this approach is the spatial prediction of candidate regions in sliding window-preprocessed images. The downstream filtering of the candidates takes the temporal changes in pixel intensities into account and decides whether it corresponds to a characteristic of interest. Each of the two steps is learned using a neural network. The precise implementation of the spatial predictor is based on a Mask R-CNN [263], which has already been shown to be capable of handling the task of nuclei detection [302, 752, 761]. The Mask R-CNN itself uses a ResNet-50 Feature Pyramid Network [414] to generate abstract features for downstream detections. In terms of the classic pipeline shown in Figure 2.8, this functionality can be called a pattern detector. To speed up the training process and improve the generalization capability, we employ the concept of transfer learning and use the initial weights of a ResNet model pretrained on the Microsoft Common Objects in Context [415] dataset. Since the more universal, low-level patterns used for the detection process are already set, the training process can focus on adjusting the weights in the last layers containing high-level features. An additional advantage of the used network is that it is not bound to one specific tile size as in previous approaches. Rather, it can determine the sizes of particle regions from a given set of possible dimensions. This results in higher flexibility while retaining the possibility of restricting accepted sizes to a specific range based on external knowledge.

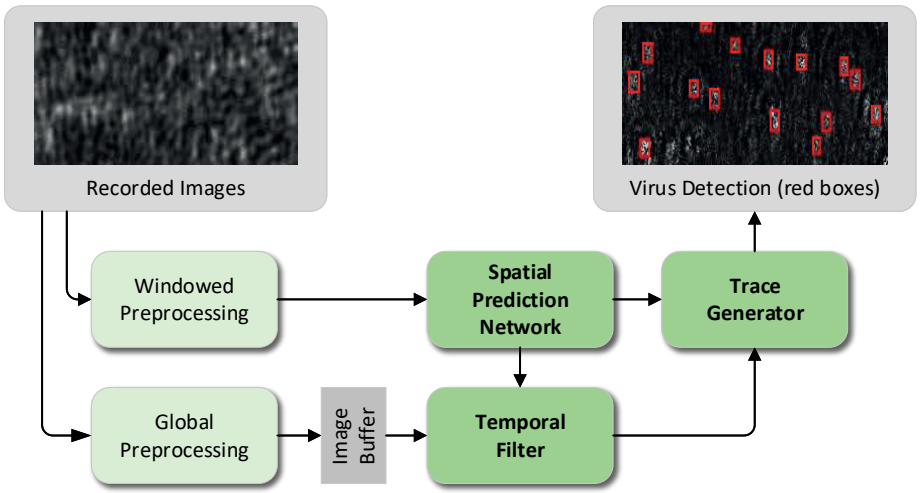



**Fig. 2.10:** Architecture of the spatiotemporal pipeline for particle detection. It preprocesses the recorded image stream in two ways, each fitting to its downstream task, predicts particle regions for each image, and connects these regions to countable particle traces stretching over multiple, consecutive frames [733].

The temporal filter network is kept simple, consisting of three fully connected layers with intermediate activation functions. It checks if the temporal view of the candidate region fits a particle signal and sorts out artifact signals that resemble the signals of interest spatially only for a short time. The still hand-crafted, fixed parts of the detection process can be found in two places: the preprocessing and the connection of confirmed particle regions to countable particle traces. By learning the core functionalities of the detection task, this approach was able to reach accuracies of over 80 % with respect to the expected particle counts of the test sets, which contained signals of 80 nm to 200 nm particles with different recording qualities, image sizes, and particle region sizes. The sets originate from different development stages of the PAMONO sensor, so setups with different optical instruments and camera configurations were included. The possibility of taking these differences into account indicates the high flexibility of the proposed solution.

In summary, classical methods such as template matching can work well for particle detection methods if they are specifically adapted to the specific imaging conditions. On the other hand, deep learning approaches provide better adaptability to distinct conditions with the drawback that they usually require a large amount of training data. However, specialized training approaches or the incorporation of domain knowledge can reduce the necessary training data. An example method dealing with this challenge explicitly is presented in Section 2.2.6.3.

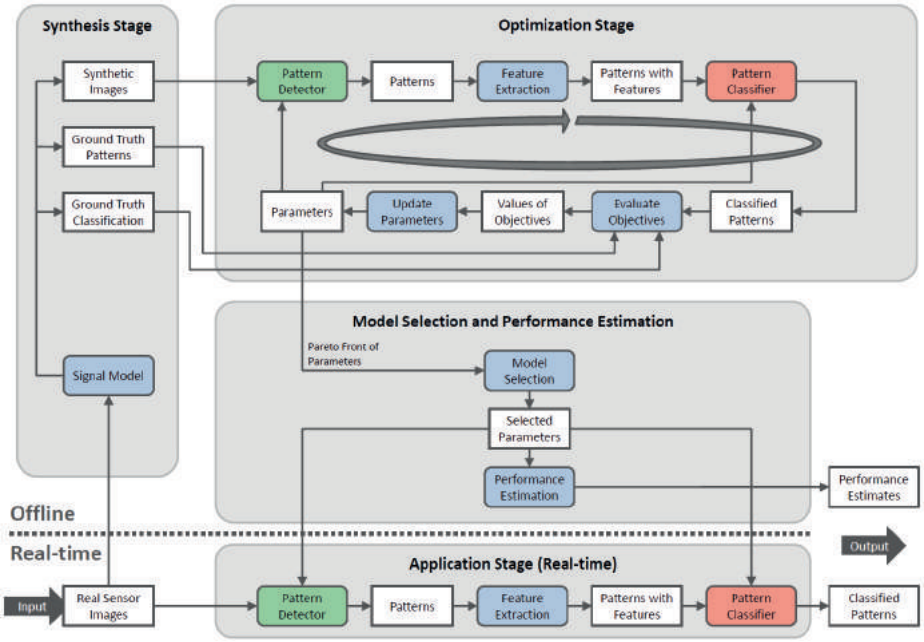



**Fig. 2.11:** The SynOpSis approach for automatic parameter adjustment. The optimization process tries to approximate the optimal parameters for the detection methods. Training is done on previously analyzed data. The amount of test data is increased artificially by a synthetic combination of background and particle signals of different training datasets. In the end, only the parameters of the detection method are changed without increasing the calculation complexity at test time [613].

### 2.2.5 Optimization

Most of the methods that can be used for object detection and other image-processing algorithms require suitable parameter values to be identified and set for improved detection. When trying to determine the best parameter settings for the detection of particles for a given algorithm or even for a complete pipeline as described in Section 2.2.4, the limit of what can be accomplished manually is reached quickly. A method utilizing a genetic optimization approach to handle this search automatically is presented in Section 2.2.5.1. Section 2.2.5.2 focuses on the possibilities of specialized optimization targeting an energy-efficient execution.

#### 2.2.5.1 Algorithmic Optimization

A method that was developed for the purpose of automatic parameter selection is the SynOpSis (synthesis/ optimization/ analysis) approach [613], which is illustrated schematically in Figure 2.11. It makes use of previous examples and synthetic augmentations based on new combinations of particle signals and background signals from

different datasets to artificially increase the number of available images for training and testing. With a given interval of plausible values for the underlying parameters, it is run repeatedly with different parameter choices and with the goal of approximating the ideal settings for the given data. In particular, the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [163], a multi-objective optimization approach, is exploited to find Pareto-optimal settings for the parameters used in the detection algorithms [613]. After that, the parameters that lead to the best results based on a predefined criterion are chosen for the single stages of the pipeline. While the optimization itself has to execute the detection pipeline after each optimization step, the actual detection is not slowed down. The reason for that is that after the optimization is done, the parameters are just transferred to the corresponding algorithms without conducting additional calculations while the actual detection takes place. Despite the benefits of tuning the algorithms to some training data, the optimization is limited to the given feature extractors and the parameter sets that are presented to it. A problem with this can appear when the selected operators fit a specific situation instead of generalizing, as can happen with template-matching approaches [385, 613].

#### 2.2.5.2 Resource Optimization

The focus of resource optimization for the task at hand can vary depending on the concrete application scenario, while several goals can hinder each other in their fulfillment. Depending on the place of use, the energy efficiency of the used calculation platform can be highly important as a reliable external power supply can not be assumed everywhere. An energy-efficient device can thus cover a wider range of operating locations and provide better portability in general, as with lower energy consumption, smaller computing platforms like embedded systems can be operated for some time over by a battery. For this purpose, work on this subject could demonstrate a concept for balancing between a reduction of the overall energy consumption resulting in a higher battery lifetime and a shorter execution time [403]. The computations were either executed locally on a mobile graphics processing unit (GPU) or offloaded over a wireless network to transfer the sensor images as well as computation requests to a server. A complete offloading of computations was compared with an alternative in which the method can select the calculations that are offloaded based on a given objective such as energy-saving or a low execution time. For this purpose, a calculation of the consumed energy is required. This is achieved using the energy model

$$e_{\text{total}} = p_{\text{GPU}} \cdot t_{\text{GPU}} + p_{\text{GPU}} \cdot t_{\text{CPU}} + p_{\text{LTE}} \cdot t_{\text{LTE}}$$

which is based on the average powers $p_{\text{GPU}}$, $p_{\text{GPU}}$, and $p_{\text{LTE}}$ of GPU, CPU, and the communication and the corresponding times $t_{\text{GPU}}$, $t_{\text{GPU}}$ and $t_{\text{LTE}}$ [403]. The result can be used to approximate the time that an algorithm can run with one charge. Although LTE was chosen as the communication technique, the energy model can be transferred

to other communication standards. To compute the overall runtime

$$t_{\text{total}} = t_{\text{GPU}} + t_{\text{CPU}} + t_{\text{LTE}} + t_{\text{Server}} - t_{\text{Parallel}}$$

the time $t_{\text{Server}}$ needed by the server and the time $t_{\text{Parallel}}$, which can be saved by parallel operations on the client and the server, are also taken into account.

Using simulation software to simulate different hardware setups such as more powerful and more energy-efficient GPU alternatives, a share of 90 % of the energy could be saved compared with the configuration with the fastest calculation while optimizing, with the goal of a low overall execution time, a speedup of 55, that is to say, a division of the execution time by 55, could be achieved in comparison to the most energy-efficient configuration [403]. It was also observed that using the most power-efficient GPUs in the example setups did not lead to the most power-efficient total configuration. Instead, a GPU with more computing capacity could calculate results faster, leading to a lower total energy consumption [403]. Additionally, it was recognized that the usage of a GPU, in general, can reduce energy consumption significantly compared with an execution purely on a CPU [402].

### 2.2.6 Application in Real-World Scenarios

Besides the direct optimization of the detection methods to a specific data basis, there are further factors to consider in a real-world scenario.

#### 2.2.6.1 Identification and Influence of Particle Size Distributions

While the most important task of the PAMONO sensor is to decide whether particles of interest are present in a sample, the determination of physical particle sizes brings additional advantages. A size distribution gives more information on the detected particles and enables plausibility checks helping to uncover outliers caused by impurities in the sample. Domain knowledge can be used for this purpose: since the device is adjusted beforehand to a specific particle type corresponding to the applied antibody coating, the expected range of particle sizes is known.

In tests with samples containing different, well-defined particle sizes, it could be demonstrated that the PAMONO sensor can be used to distinguish the different sizes from each other. The quality of the predicted size distributions was also compared with a commercial device showing that the predictions based on the PAMONO sensor rank at the same level of size prediction quality. For this purpose, the difference between the median signal intensity before and after a particle attaches is calculated. When comparing two particle sizes, this difference is proportional to the difference in signal intensities. An example of this is shown in Figure 2.6. A visualization of the analyses with the PAMONO sensor and, for comparison, a commercial nanoparticle tracking device for samples containing 100 nm, 200 nm, and 300 nm particles, as well as one mixture containing all three sizes, can be seen in Figure 2.12.

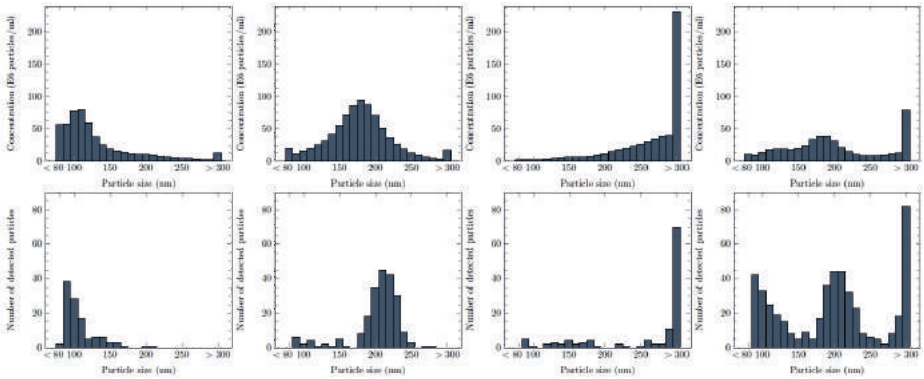



**Fig. 2.12:** Determined physical particle sizes for four different suspensions analyzed with (from left to right): 100 nm particles, 200 nm, 300 nm, and a composition of 100 nm and 200 nm and 300 nm particles. The top row shows reference distributions obtained with the commercial Malvern LM10 device while the bottom row shows the results of the PAMONO sensor [609].

The soft accuracy value of detected particle size distributions can be evaluated with

$$a^r = \frac{|\{e \in \mathcal{E} \big| |c^{\text{pred}}(e) - c^{\text{corr}}(e)| \leq r\}|}{|\mathcal{E}|}$$

calculating the share of predicted sizes $c^{\text{pred}}$ that approximately match the correct size class $c^{\text{corr}}$ for each patch $e \in \mathcal{E}$ of an analyzed image.

With the chosen division into classes of 10 nm, the classification of a 100 nm particle into say, the size class 80 nm, is tolerated with $r = 2$. At the same time, a prediction of 130 nm would be considered a false value for a 100 nm particle as a class interval of 10 nm together with $r = 2$ results in a tolerance of 20 nm. In an evaluation, particles of the sizes 80 nm, 100 nm, and 200 nm were analyzed. With classes of 10 nm intervals, an accuracy of over 70 % was reached for $r = 2$ [386].

In general, the PAMONO sensor can be calibrated by measuring the signal intensities of standardized particle sizes to map the measured values of temporal intensity steps to physical size. When changes are applied to the optical components of the sensor setup, the procedure has to be executed again as they can lead to a general shift in the measured signal intensities.

#### 2.2.6.2 Detection of Viral Infections

The PAMONO sensor is designed to be usable as a rapid test for detecting viral infections on-site. When considered for use at an airport, a city center, or the entrance of a stadium, however, additional requirements emerge. For use as a rapid test on-site, the device has to be transportable to different places, which is fulfilled by the sensor case being suitcase-sized. Energy efficiency can be important for enabling the execution on an embedded device and use at otherwise poorly accessible places. A possible solution

for miniaturization was evaluated using the Hardkernel Odroid-XU3 and Odroid-XU4 single-board computers as a basis for calculations [399]. Although execution times are increasing, the ability to calculate and manage calculation offloading on embedded devices was demonstrated to be feasible [399]. The related method of computation offloading is described in Section 2.2.5.2.

#### 2.2.6.3 Dealing With Imaging Artifacts

The amplification of imaging artifacts has to be expected when targeting a real-world application of the PAMONO sensor. Therefore, an essential aspect of reliable detection is the robustness of algorithms against imaging artifacts.

These artifacts originate from different sources. On the hardware side, there is the imperfection of the optical instruments and their adjustments. The smoothness of the gold film and its coating also influence the quality of the signals of interest. For example, scratches and other irregularities on the surface cause visible artifacts in the recordings. Another source that has to be considered is the limited and varying cleanliness of samples in real use cases. For example, air bubbles and dust particles in saliva or sputum samples cannot be avoided completely. In general, changing external influences have to be expected when targeting an application at places where the constancy of laboratory conditions cannot be achieved.

With variations of artifact types, such as line-like or wave-like structures, regions of constant intensities, pulsating regional patterns, or random noises, a detection has to tolerate both unstructured and structured artifacts.

This combination causes classic noise reduction methods to yield insufficient results. At the same time, changing patterns and intensities of artifacts also pose a problem for learning methods since many related approaches require a high amount of training data, especially when artifact characteristics differ in the recorded images.

Recent work tackles both problems by increasing the robustness of an existing learning approach relying only on a small amount of labeled training data.

This is achieved through a generative adversarial network (GAN) [237] that is not directly involved in the detection process but learns to simulate real artifact patterns to improve the detector before the actual detection takes place. Synthetically generated artifacts are used to overlay training images for the detection network. In this way, the detector learns how to tolerate the induced artifacts [547]. In addition, the architecture of the detection network itself does not need to be changed, so there is no decrease in the speed of the detection process.

The training of the GAN is not free from the demand of training data, but it uses reference images. These images are characterized by the fact that they do not contain particles of interest but only show imaging artifacts. The advantage of using this type of recordings instead of those carrying particle signals is that no physical test particles are required. This saves time and material costs and eliminates the risk of reproduc-

ing particle signals in the generated artifact images, as particles do not occur in the reference images. It is therefore ensured that only artifacts are learned.

Having demonstrated that it can produce realistic images with sufficient variation even with a small amount of training data, the StyleGAN2-ADA [312] was employed as the specific GAN architecture. While this GAN requires fixed-size training images and produces fixed-size images, different sensor configurations lead to different sizes of recorded images. Therefore, the GAN-based approach composes multiple patches of a fixed side length to cover the area of an annotated image for the training of the detector. As an illustration of the whole process, Figure 2.13 shows the generation of synthetic artifact signals and their usage in the training of the detection network.

Training a simple segmentation network with a downstream object detection shows the improvement in the robustness against artifacts when using the presented, GAN-based approach.

Two configurations were evaluated to compare the results with and without overlaying synthetic artifacts. The first configuration contained one dataset with particles of interest and only weak and unstructured artifacts. The second one uses the same dataset but adds synthetic artifacts generated by a GAN trained with reference images. Both employ the same test datasets, which contain different types and intensities of artifacts. After training two identical U-Net [544] architectures, each with one of the configurations, a clear difference becomes visible. While the training results without synthetic artifacts yield poor results, overlaying the images improved the mean F1 score by 22 % [547]. While improvements can be seen for all types of artifacts, they are most significant for images with structured artifacts of high intensities.

The GAN-based approach shows similarities to the approach presented in Section 2.2.5.1, as both create mixtures of particle signals and other signals from different datasets. The difference becomes clear when considering that the GAN-based generation does not directly use the limited set of natural images but artificially creates an arbitrary number of additional images in a learning process. This procedure can significantly increase the variability of the training images.

### 2.2.7 Current Research and Outlook

The techniques around the PAMONO sensor are expected to benefit from further research and development in the area of robust object detection. At the same time, it is worth evaluating an extension of the applications beyond virus detection, for which the sensor also forms a suitable basis. For each of the two perspectives, current and planned investigations are described below.

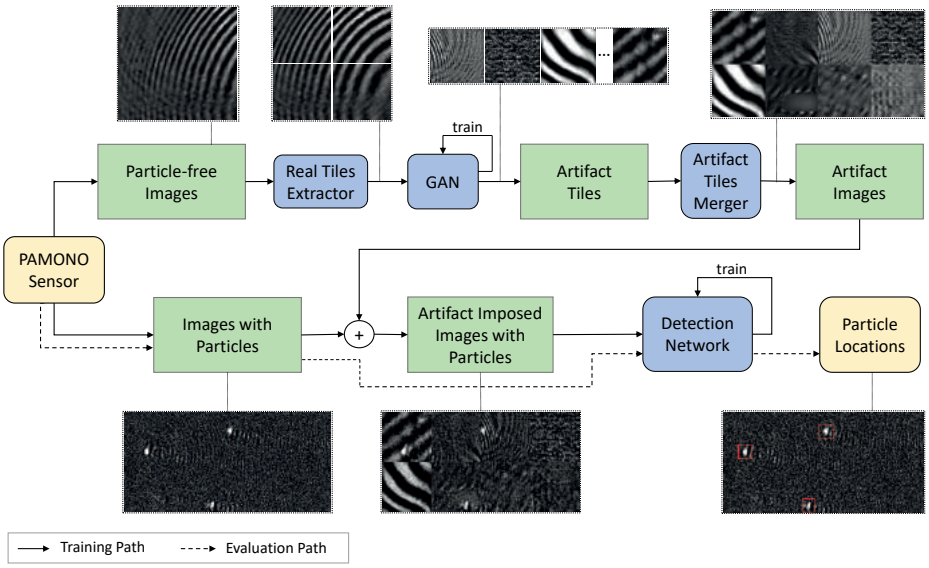



**Fig. 2.13:** Schematic representation of overlaying training images with generative adversarial network (GAN)-generated artifacts from composite tiles. The PAMONO sensor is used to record samples without particles of interest (upper part) and samples including such particles (lower part) for the training process. The trained detection model is then used to search for particles in images where their presence is unknown. Dashed arrows show the path of images in the evaluation process, while solid arrows represent the path of images in the training process. The images in dotted boxes visualize the single steps by examples. The yellow boxes illustrate the start and end of the pipeline, green boxes represent data, and blue boxes mark algorithms [547].

#### 2.2.7.1 Improvements of Detection Capabilities

A method to improve the overall results concerns the physical setup of the sensor. With the rotation of the prism and the position of the camera objective influencing the signal quality, the appropriate adjustments must be realized before a recording. Although good results can be achieved by manual adjustments, there are limits to human accuracy when tuning the optical parts. For this reason, a sensor-actuator control system is targeted to determine an optimal setting automatically and to adjust the components in feedback with the resulting image signals.

In any case, further development in the direction of an on-site application should consider the need for high robustness of the methods against external disturbances to ensure a reliable and widely available rapid test. While the synthetic generation of artifacts presented in Section 2.2.6.3 showed improvements in the detection robustness, temporal dynamics are not specifically taken into account. By including temporal information, further improvements in the robustness can be assumed.

A different way of improving results is the incorporation of domain knowledge. The knowledge regarding a specific pathogen suggests the inclusion of particle counts per

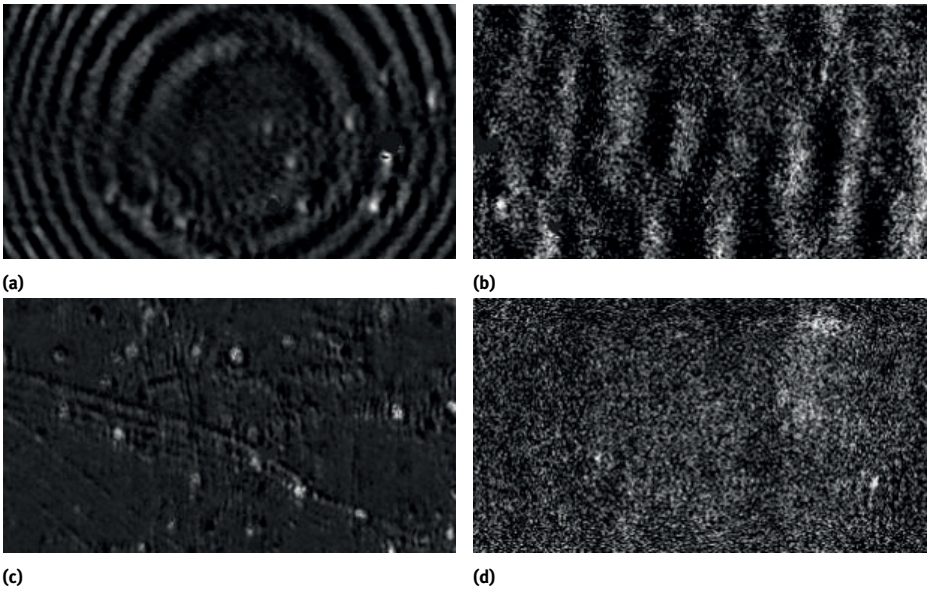



**Fig. 2.14:** Examples of different artifact characteristics in preprocessed images after the application of dynamic contrast enhancement.

image area as additional information. When a reference value for a usual concentration of those particles is known, the separation between samples with and without particles of interest can be improved based on an infection-specific concentration threshold. Although this option has not yet been quantitatively evaluated, it promises to improve reliability in practical applications.

A promising area of future exploration of mobile concepts for the execution of detection methods is the evaluation of embedded and other mobile devices. For example, the use of Field-Programmable Gate Arrays (FPGA), which are presented in Section 6.1 in Volume 1, promises to reduce power consumption while keeping the system size small. Another interesting work, which addresses hardware improvements, is the evaluation of learning approaches on modern memories in Section 7.2 in Volume 1. It focuses on the aspect of energy-saving in different approaches by targeting an optimal use of memory technologies.

#### 2.2.7.2 Perspective Applications of the PAMONO Sensor

The PAMONO sensor proved its power in the characterization of viruses and Virus-Like Particles (VLPs). The latter is especially important since VLPs are often used as medical products and serve as a basis for different vaccines. Under these circumstances, the PAMONO sensor can be applied as an analytical instrument for quality controls during routine production of VLP-based vaccines, estimating the size and concentration of

produced NPs in real time. Further, the PAMONO sensor represents a valuable analytical instrument for the characterization of extracellular vesicles (EVs). In theory, the PAMONO sensor does not simply enable the sizing and quantification of EVs; it also helps to gain information about their surface markers and the molecules transported inside EVs. Nowadays, EVs earn growing interest as a means of intercellular communication. Acting this way, EVs can potentially serve as drug-delivering vesicles and as biomarkers of the cellular status. In this case, the ability of the PAMONO sensor to characterize EVs without labeling is a promising opportunity. Moreover, the PAMONO sensor may serve as a platform for the development of cell-based assays. In this case, the status of cells cultured on the sensor surface can be estimated via the cellular production of EVs and soluble mediators measured by the PAMONO sensor.

## 2.3 Cancer Diagnostics and Therapy from Molecular Data

*Sven Rahmann*
*Alexander Schramm*
*Johannes Köster*

**Abstract:** The past decade has seen unprecedented progress in the survival chances of cancer patients as a consequence of new treatments targeting tumor-specific cellular processes, which have been uncovered by molecular genetic analyses. From a data analysis perspective, the main challenge is the high dimensionality and multimodality of the genetic data relative to the small sample sizes (numbers of patients). From a computational perspective, the analysis of high volumes of data (about 100 GB of sequencing data for an individual tumor genome) currently requires high-powered computational resources and still remains challenging in the very short time frames that are desired to start treatment immediately.

We discuss two avenues of progress. First, we present methods that are able to extract most of the genetic variants from a sequenced tumor genome, but require only 2 % to 5 % of the computational resources compared with the current state-of-the-art procedures.

Second, we discuss a versatile unified statistical model for distinguishing true variants from technical artifacts of the DNA sequencing process.

Using analyses of paired samples from primary and relapse neuroblastoma tumors, we are able to extract patterns of tumor evolution that are correlated with cancer progression and the escape of tumors from therapeutic intervention.

As a result, a novel risk classification of neuroblastoma has been established based on genomic and mutational data.

### 2.3.1 Introduction

Cancer patients nowadays receive precise diagnosis and personalized therapy based on their individual molecular genetic data.

Here, we report on the analysis of DNA data from patients with neuroblastoma, a solid tumor typically occurring in children.

Diagnostics and prognoses are based on DNA sequencing, currently ranging from a few hundred targeted genes to entire genomes requiring 100 GB per patient in the near future.

Identifying relevant variants in the DNA that serve as biomarkers to distinguish between different risk classes, or primary tumors from relapses, or treatable versus non-treatable tumors, is and remains challenging, but every step of progress in this

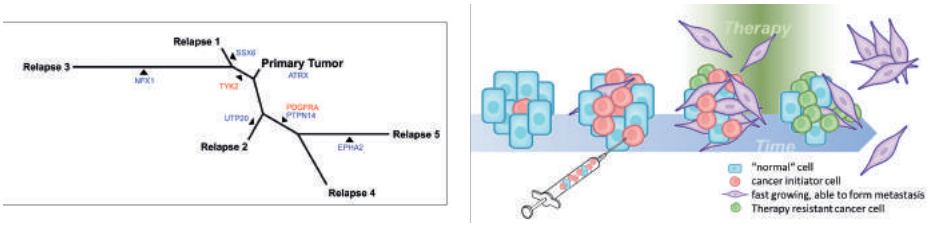



**Fig. 2.15:** Left: Evolutionary tree derived from binary features (presence/absence of informative single nucleotide variants) of a primary neuroblastoma tumor and several relapse samples taken from different tissues and at different times from the same patient [583]. Right: Illustration of evolution of tumor heterogeneity under therapy over time [588].

field helps to make better treatment decisions in the long run. The following molecular features derived from genomic data are of primary interest: (1) single nucleotide mutations (or variants, SNVs), (2) short insertions or deletions of DNA, (3) large structural variants (e.g., chromosomal translocations), (4) copy number changes (gain or loss of genetic material in tumor cells), (5) epigenetic changes, such as DNA methylation changes, and (6) differences in gene expression.

Over the past years, we have developed feature extraction workflows and data analysis processes for each type of feature mentioned above. For data analysis workflows in general, but especially in medicine, reproducibility of derived data from raw data is of utmost importance. The basis of each of these processes is our workflow management system called Snakemake [345, 450], which is now widely used worldwide, as it guarantees reproducibility in particular for large-scale DNA sequence analysis workflows. Furthermore, the Bioconda package repository [242] was founded by one of us (JK) and now, with widespread community support, acts as a central repository for semantically versioned bioinformatics software, which is made available in a reproducible way.

In the following, for simplicity, we focus on the first type of features (SNVs), but these findings also translate to the other variant types, if appropriately adjusted. In particular, we discuss whether we can determine genomic variants that distinguish primary neuroblastomas from those that re-occur after therapy (referred to as relapses or relapse samples). The latter are responsible for adverse disease courses and are currently considered to be incurable. It was therefore highly encouraging that we were able to identify several genes with recurrent mutations present only in relapse samples [583]. Figure 2.15 summarizes some of our key findings on tumor heterogeneity after relapse (left side) and illustrates the tumor evolution process (right side). It is mainly this developing molecular heterogeneity of tumor cells under treatment that currently prohibits effective long-term therapies.

The main resource constraints for this setting and similar situations are a limited number $n$ of samples (patients) versus an extremely high number $p$ of potential features (e.g., each potential variant in the genome observed in at least one sample).
So we face two challenges in particular:

1. resource-efficient detection of candidates of variants (Section 2.3.2)
2. accurate classification of candidates in each sample (true variant vs. noise, technical artefact, etc.; Section 2.3.3)

### 2.3.2 Resource-Efficient Detection of Variant Candidates

Standard genetic mutation or variant analysis starts with an extremely compute-intensive step: the localization of every single sequenced DNA fragment (or "read"; there are literally millions of DNA reads in a single dataset) in the genome, and a pairwise comparison between the fragment and the genomic sequence. Such pairwise alignments are the basis of variant calling: many reads showing a certain difference at the same position compared with the reference genome, this provides convincing evidence that the sequenced genome contains a specific genetic variant at that position, either in both inherited chromosome copies (homozygous variant) or in just one (heterogzygous variant). To be precise, complex statistical models and tests are necessary to distinguish true variants from possible technical artifacts (see Section 2.3.3).

This first localization and comparison step is performed by so-called *read mappers*, such as BWA-mem [391], bowtie2 [371], minimap2 [390], or PEANUT [344]. Extensive parallelism on both multi-core systems and GPUs keep the (wall clock) time of this step within a few hours. However, the overall CPU work consists of many CPU days or months for a single dataset, consuming considerable energy.

It is therefore of high interest to develop more resource-frugal methods to achieve the same task, or at least a large fraction of it. We explored *alignment-free* methods as an alternative to the above mapping and alignment-based method. In particular, we propose to use short DNA strings of length $k$ (so-called $k$-mers) to directly detect potential single-nucleotide variants, as we now describe.

#### 2.3.2.1 Genome Preprocessing

We first preprocess the reference genome.

1. Select an appropriate value for $k$, such that most $k$-mers are unique in the reference genome. Our studies indicate that $21 \leq k \leq 31$ works well for the human genome [517].
2. Build a (very large) hash table of $k$-mers in the human reference genome and the number of times that they occur. We need to take into account that double-stranded DNA is equivalent to its reverse complement.
3. Mark the unique $k$-mers; they point to a unique position in the genome.
4. Among the unique $k$-mers, mark those that are *robustly unique* against single substitutions, i.e., those for which no Hamming-distance-1 neighbor also occurs in the genome. The resulting robustly unique $k$-mers do not only point to a unique

location in the genome; they also cannot easily be changed into *k*-mers that occur at a different genomic location.

The alignment-free methods work either exclusively on the robustly unique *k*-mers or on all unique *k*-mers, giving the information from the robustly unique *k*-mers a higher weight. This pre-processing step has to be performed only once for any genome version.

We found that multi-way bucketed Cuckoo hash tables are ideally suited for the task, as they allow relatively quick construction times and yield very fast lookup times later. There are smooth trade-off options between lower memory usage and even faster lookup times.

In a preliminary study [756], we designed and implemented these hash tables for a simpler application than variant calling: xenograft sorting. Here, a human tumor is engrafted into another organism (typically a mouse) to be able to study its evolution and response to different therapies. When such a tumor is sequenced, one obtains a mixture between human and mouse DNA reads, so that all reads have to be assigned to the organism of origin before proceeding further. This assignment is called xenograft sorting. Even though human and mouse are quite similar on a genetic level, they can be sufficiently well distinguished on the *k*-mer level. We presented a classification method based on *k*-mer hash tables, as outlined above, with extremely high accuracy, but using much less CPU work than previous methods: less than 25 % of comparable hash-based methods and less than 5 % of classical alignment-based methods [756]. We additionally showed that the placement of keys in the hash table can be optimized to yield optimal average look-up times (based on the number of random memory accesses, i.e., likely cache misses), saving 10 % to 15 % of CPU work for each sample (after a 48 CPU hour optimization procedure that has to be run only once) [757].

#### 2.3.2.2 Basic Alignment-Free Variant Calling

The underlying idea of this method is as follows: We count all the *k*-mers in a sequenced sample and produce a histogram of the count values. A typical (unique) *k*-mer should have a copy number of two (in a diploid genome) when no variant is present. We therefore analyze the histogram of observed *k*-mer counts (Figure 2.16) from the sample. The leftmost peak (counts near zero) can be explained by rare *k*-mers due to sequencing errors or contamination; we can attempt to correct these, or ignore them entirely. We fit a negative-binomial mixture model to the remaining peaks occurring at equidistant counts. The main peak corresponds to a copy number of 2 in a diploid genome (from *k*-mers present in both the maternal and paternal chromosome set).

The initial analysis is restricted to the robustly unique *k*-mers from the reference genome. We expect that each such *k*-mer has a copy number of either 0 (homozygous variant), 1 (heterozygous variant) or 2 (no variant) in the sample. Higher copy numbers could be explained by segmental duplications, which we do not consider at this point. If we suspect a variant, we look for *isolated single nucleotide variants*, i.e., *k*-mers with

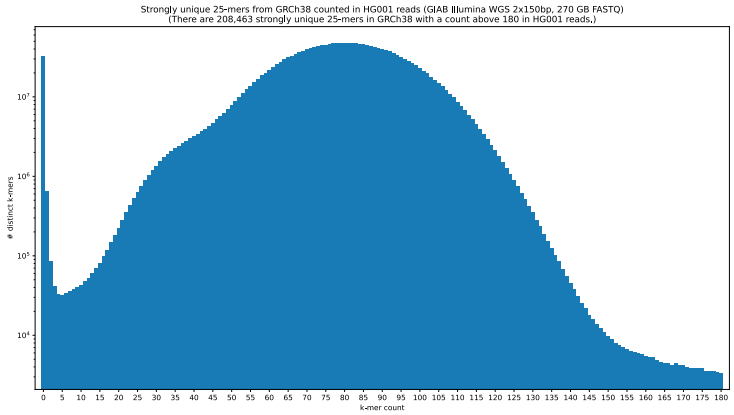



**Fig. 2.16:** Illustration of a *k*-mer count histogram, relating observed *k*-mer counts (x-axis) to their frequency (y-axis; logarithmic). The leftmost peak (close to zero) represents noise and erroneous *k*-mers, mostly due to sequencing errors. The main peak (near 80, approximately the sequencing coverage of this example) corresponds to the standard copy number of 2. The shoulder (near 40) then corresponds to a copy number of 1 and consists of *k*-mers that are part of heterozygous isolated mutations. This histogram was created from a control sample; in a tumor sample, more irregularities, especially additional peaks at higher copy numbers, can be expected.

a Hamming distance of 1 to the reference *k*-mer, among the *k*-mers in the sample. If we find a unique one (with the expected copy number), we store the pair of reference *k*-mer and modified *k*-mer as a candidate for a variant.

This process can be implemented very efficiently, and in addition, it can be trivially parallelized. It yields candidates for Single Nucleotide Variants (SNVs) that then can be checked by statistical methods (see next section). It can also reliably detect copy number variants on long segments. However, it cannot easily detect more complex variants, such as two SNVs in close proximity, short indels, or structural variants: Here translating *k*-mer information into an exact variant is more difficult, but can resort to alignment-based methods for the local regions around areas with suspicious *k*-mer frequency structure.

**Perspectives** Alignment-free variant calling is still an active research area, and while we made contributions to the underlying data structures (engineered Cuckoo hash tables) and were successful in calling selected SNVs, further ideas are necessary to call larger classes of variants reliably. Possible approaches include using locality sensitive hashing, in particular min-hashing, instead of exact *k*-mer hashing, combined with *hybrid methods* between alignment-free and alignment-based approaches. To assess the potential of min-hashing-based methods, we conducted a detailed statistical feasibility study, examining when it is useful to include known variants into a *k*-mer-based read mapper (and when not; see [515]), paving the way for novel approaches.

### 2.3.3 A Unified Statistical Model for Genomic Variant vs. Artifact Classification

We present an extension and generalization of a latent variable model originally published by Köster, Dijkstra, Marschall, and Schönhuth [342].

For this, we consider a set $S$ of samples. Samples can be related with each other in three ways.

1. There can be clonal inheritance between two samples $s_1, s_2 \in S$: sample $s_1$ inherits all constitutive genetic variants of sample $s_2$. In addition, the tissues of origin of both samples may have developed their own *somatic* mutations during their lifetime until sequencing.
2. There can be Mendelian inheritance [443] between samples $s_1, s_2, s_3 \in S$: the individual of origin of sample $s_1$ inherits constitutive genetic variants of two parental individuals ($s_2$ and $s_3$).
3. A sample $s \in S$ can be contaminated with a fraction of another sample $s' \in S$.

We represent the three relationships in a directed graph $G = (S, I_c, I_m, C)$ (the *sample graph*) with edge types $I_x \subseteq S \times S$ for clonal ($x = c$) and Mendelian ($x = m$) inheritance as well as $C \subseteq S \times S$ for contamination. The corresponding contamination fraction can be obtained with $c : C \to [0, 1]$.

The above representation can be used to model the three classical cases of genomic variant calling: single-sample or population calling (the graph has no edges) [171], pedigree based family variant calling (Mendelian inheritance edges) [171], and calling of tumor/normal sample combinations (clonal inheritance and contamination edges) [342]. Importantly though, instead of being limited to these, it can reach beyond them by combining the mechanisms into more complex scenarios.

#### 2.3.3.1 Variables and Notation

**Observed Variables** For each potential genomic variant of interest, we observe sequencing read data $\boldsymbol{Z}_s = (Z_1^s, \ldots, Z_k^s)$. If the read data consists of so-called paired-end reads (each investigated DNA fragment is sequenced from both ends), each observation in $Z_i^s \in \boldsymbol{Z}_s$ is a tuple $Z_i^s \in (\{\text{A,C,G,T}\}^+, \{\text{A,C,G,T}\}^+, \mathbb{N})$, with the first and the second element denoting the nucleotide sequence of the read and the last element denoting the so-called observed insert size, that is, the number of bases from the leftmost to the rightmost covered base when aligning the read pair to the most likely position of origin on the reference genome of the investigated species. If the read data consists of so-called single-end reads (each investigated DNA fragment is sequenced just from one end), each observation $Z_i^s \in \boldsymbol{Z}$ is simply the nucleotide sequence of the read, in other words $Z_i^s \in \{\text{A,C,G,T}\}^+$.

**Latent Variables** The central readout of our model is the allele frequency in each sample $s$, denoted as latent variable $\theta_s \in [0, 1]$. For each read observation $i$, there is

a binary latent variable $\xi_i^s$ with $\xi_i^s = 1$ denoting that the observation originates from the variant allele (i.e., from a genome copy hosting the variant under consideration) and $\xi_i^s = 0$ denoting that the read originates from the reference allele (i.e., from a genome copy hosting exactly the same sequence as in the reference genome of the corresponding species). In addition, a binary latent variable $\omega_i^s$, denoting whether the observation has been aligned to the correct ($\omega_i^s = 1$) location of origin in the reference genome, is used.

**Extensions for Bias Estimation** The model can be further extended in order to estimate biases in additionally observed properties of the read data, that is, the strand, the read position supporting the variant, the read orientation, and whether the alignment against the reference genome covers the entire read. Biases from an equal distribution in the observed values of variant supporting reads for any of these properties typically indicate an artifact. For clarity and brevity, we omit the integration of these biases in our model here. An integration of strand bias can be already found in [342].

#### 2.3.3.2 Latent Variable Model

In the following, we briefly introduce the latent variable model used for calculating allele frequency likelihoods that has been published recently [342], and then provide a generalization of the method. When evaluating if a read deviates from the reference genome, two types of uncertainty are to be considered. First, there is *alignment uncertainty*: often, a read can be aligned at multiple loci in the reference genome (also see Section 2.3.2).

Depending on their similarity, there is more or less certainty about the optimal positioning of the read. Read mappers and alignment tools, such as BWA [391], report this uncertainty as mapping quality (MAPQ), which can be translated into a probability $\pi_i^s$ associated with each read observation $Z_i^s$ to be aligned to the correct locus. Second, there is *typing uncertainty*: the observed read sequence is not a perfect representation of the true DNA fragment that has been sequenced, but instead a measurement entailing potential errors and artifacts. The DNA sequencing machine provides an estimate of the certainty of each reported base as the so-called base quality, which can again be translated into a probability of the reported base to be correct. In addition, depending on the sequencing technology, there are known rates of false insertions or deletions of bases in the reported read sequences, as remarked on for example by Schirmer, D'Amore, Ijaz, Hall, and Quince [578].

We now model the relationships between our observed and latent variables, while taking above mentioned uncertainties into account. For each observation $Z_i^s$ in sample $s$, we handle alignment uncertainty by defining the distribution of the latent variable $\omega_i$ as

$$\omega_i^s \sim \text{Bernoulli}(\pi_i^s). \tag{2.3}$$

The distribution of the latent variable $\xi_i^s$ depends on the expected fraction of observations from the variant allele. If $s$ is not contaminated by another sample, we define

$$\xi_i^s \sim \text{Bernoulli}(\theta_s \tau). \tag{2.4}$$

Thereby, $\tau \in [0, 1]$ denotes a sampling bias that occurs because it is usually harder to obtain observations from the variant allele: it is harder to align, and depending on the size of the variant, harder to obtain reads that sufficiently cover it [342]. If, in contrast, $s$ is contaminated by a $s'$ (i.e., $e = (s, s') \in C$) with fraction $\alpha = c(e)$ we define

$$\xi_i^s \sim \text{Bernoulli}(\alpha\theta_s \tau + (1 - \alpha)\theta_{s'} \tau). \tag{2.5}$$

In other words, the expected fraction of observations from the variant allele becomes a mixture of the allele frequencies in $s$ and $s'$.

Then, typing uncertainty can be modeled as

$$Z_i^s \mid \xi_i^s, \omega_i^s \sim \begin{cases} p_i \text{ if } \xi_i^s = 1, \omega_i^s = 1 \\ a_i \text{ if } \xi_i^s = 0, \omega_i^s = 1 \\ o_i \text{ if } \xi_i^s = 0, \omega_i^s = 0. \end{cases} \tag{2.6}$$

Here, $a_i$, $p_i$, and $o_i$ are probability distributions modeling the case that the observation comes from a genome copy where the variant is present ($p_i$), absent ($a_i$), or from a different locus ($o_i$). These can be computed using Pair Hidden Markov models, which essentially realign the read sequence against the sequence of reference and alternative allele while statistically considering sequencing error rates, as shown in Köster, Dijkstra, Marschall, and Schönhuth [342] for deletions and insertions. Since then, via analogous approaches, our model has been extended to also support all other common variant types ranging from small (SNV, MNV) to structural variants such as inversions, duplications, and arbitrary chains of breakpoints.

By combining the above relations, the model can be used to calculate the likelihood of a given combination of allele frequencies of samples $S = \{s_1, \ldots, s_n\}$ as

$$\Pr(\boldsymbol{Z}_{s_1}, \ldots, \boldsymbol{Z}_{s_n} \mid \theta_{s_1}, \ldots, \theta_{s_n}) = \prod_{j=1}^{n} \prod_{i=1}^{|\boldsymbol{Z}_{s_j}|} \Pr(Z_i^{s_j} \mid \theta_{s_1}, \ldots, \theta_{s_n}) \tag{2.7}$$

while assuming independence between the read observations. Note that the computation of the likelihood function is linear in the total number of read observations, as we have shown previously [342].

#### 2.3.3.3 Prior Distribution

The prior probability of a given allele frequency combination $\theta_{s_1}, \ldots, \theta_{s_n}$ in our generalized model can be computed by considering the dependencies between the samples modeled by the sample graph $G$ (see beginning of Section 2.3.3). In addition, we assume

that for each sample $s \in S$, a ploidy $\rho_s \in \mathbb{N}$ (which may differ by chromosome, e.g., it may be sex-specific), a somatic effective mutation rate $\mu_s \in [0, 1]$, and a germline mutation rate $\nu_s \in [0, 1]$ are known. For calculating a prior probability, the key is to explain the total allele frequency $\theta_s$ by a germline allele frequency $\iota_s$ and a somatic allele frequency $|\theta_s - \iota_s|$. Usually, one of the two will be zero, such that variants are explained either by germline or by somatic mutation, but combinations thereof can also happen in rare cases. From the known ploidy $\rho_s$ of a sample $s_j \in S$, we can calculate the set of possible germline allele frequencies $\zeta_s \subseteq [0, 1]^{\rho_s+1}$. For example, for $\rho_s$, we obtain $\zeta_s = \{0, 0.5, 1\}$; in other words, any germline variants may occur either in no, one allele (0.5 or 50 %), or two alleles (1.0 or 100 %). The prior probability can then be calculated by recursively exploring all possible explanations of a given total allele frequency combination.

For a combination of germline and somatic allele frequencies we can then distinguish between the following cases:

1. All samples that are not direct descendants of other samples (have no incoming edges in $I_c$ and $I_m$ in the graph $G$) are considered a population and the prior probability of their combination of germline allele frequencies is calculated, as defined by DePristo et al. [171], based on a so-called heterozygosity (i.e., the expected proportion of heterozygous sites in the genome), which is usually known for the investigated species.
2. For any sample $s \in S$ that inherits clonally from another sample $s' \in S$, we calculate the prior probability for the somatic allele frequency $f = |\theta_s - \iota_{s'}|$ according to the method of Williams, Werner, Barnes, Graham, and Sottoriva [730], who report a formula for the expected cumulative number of somatic mutations per frequency. The latter can be translated into the corresponding density by normalizing with the genome size $g$ and taking the first derivative, resulting in $h(f) = \frac{\mu}{f^2 \cdot g}$ for $f > 0$. In order to also be able to calculate the probability for $f = 0$, we define a reasonably small $\epsilon$ and define $h(0) = 1 - \int_{\epsilon}^{1} h(f)\mathrm{d}f$.
3. For any sample $s \in S$ that inherits in a Mendelian [443] way from two parents $s' \in S$ and $s'' \in S$, we first calculate the number of expected constitutive alternative alleles in the child and the parents by multiplying the ploidy with the respective germline allele frequency, i.e., $\rho_s \cdot \iota_s$. We then sum over the probabilities of all cases of inheriting chromosomes with or without the variant allele from the parent samples that could explain the expected constitutive alternative alleles. The individual probabilities can be calculated by modeling an urn drawing process without replacement, yielding a hypergeometric distribution. Finally, additional somatic variation, i.e., cases where $\theta_{s_j} - \iota_{s_j} \neq 0$, are handled by multiplying the corresponding prior probability for the somatic allele frequency.
4. Finally, sometimes it might not be possible to formulate prior assumptions about allele frequencies of a sample $s \in S$. In such cases we specify an allele frequency universe $U_s \subseteq [0, 1]$ for a sample and assume a uniform distribution.

By taking the product over the priors for individual or groups of samples derived from distinguishing the above three cases, the prior probability for any combination of germline and somatic allele frequencies can be obtained.

#### 2.3.3.4 Variant Calling Grammar

The above model is implemented in the software Varlociraptor (https://varlociraptor.github.io). Varlociraptor offers a *variant calling grammar* that allows to define a scenario that configures all aspects of the model (prior parameters, sample graph) via a textual representation in YAML format (YAML Ain't Markup Language; https://yaml.org/). A scenario consists of the following sections.

**Species** In this section, general prior knowledge about the investigated species is defined, such as the heterozygosity (see Section 2.3.3.3) and the ploidy (number of chromosome copies in a cell). The latter may be defined with sex-specific exceptions (such as the X and Y chromosome distribution in humans).

**Samples** In this section, the samples and their dependencies (i.e., the sample graph) are defined. For each sample, it is necessary to either define an allele frequency universe (leading to a uniform prior across the defined frequencies) or the sex. In the latter case, ploidy and heterozygosity are taken from the species definition and used to configure the prior accordingly. Each sample may be annotated with a contamination by another sample in a given fraction (this can be used to define the common case of having a tumor sample that also contains healthy normal tissue). Finally, each sample may define a type of inheritance (Mendelian or clonal), while referring to the corresponding parental samples.

**Events** The heart of a scenario is formed by the definition of mutational *events* of interest. These can be used to define any kind of Boolean logic expressions over allele frequencies (discrete or intervals) in the given samples.

An example for a scenario modeling the calling of variants in a patient for which a normal healthy blood sample, a tumor sample, and a relapse sample is used can be seen in Figure 2.17. Here, for simplicity, we have initially not defined any prior knowledge regarding mutation rates etc., thereby modeling a uniform distribution over the defined allele frequency universes. An equivalent scenario including this kind of prior knowledge is shown in Figure 2.18. Here, it can be seen that we are able to define inheritance between the normal and the tumor sample. For the relapse sample, although in principle it should inherit mutations from the tumor sample, it is unknown to what extent this happens, because usually only one or a few subclones survive the therapy. Hence, we refrain from specifying an inheritance between the tumor and the relapse, and instead impose a uniform prior on the possible allele frequencies in the relapse sample.

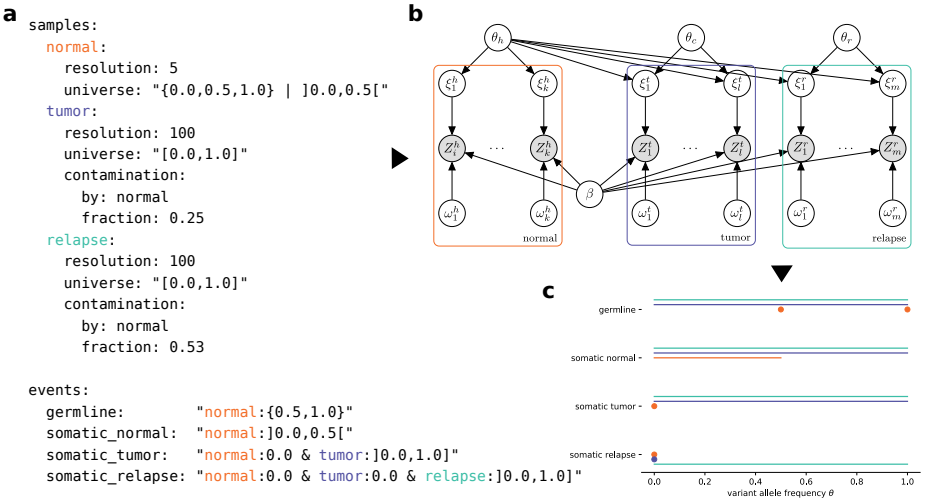



**Fig. 2.17:** Example of a Varlociraptor scenario specification to distinguish between germline variants and those occurring as somatic events in the primary or relapse sample. (a) Scenario definition via Varlociraptors variant calling grammar. The first section defines the three involved samples normal healthy blood, primary tumor, and relapse after therapy, along with their contaminations and expected allele frequency universe. The second section defines the events of interest via Boolean logic formulas. (b) The resulting structure of the latent variable model, automatically derived from the scenario definition. (c) Visualization of the expected allele frequencies in the three samples for each defined event.

### 2.3.4 Application and Results

It was previously shown that Varlociraptor is able to significantly improve the recall, while precisely controlling the false discovery rate without the need to tune any technical filter parameters in the absence of a biological interpretation [342]. Here, we illustrate the application of the model by re-analyzing the aforementioned previously published neuroblastoma dataset [583]. In this manuscript, we analyzed genomic data from 17 neuroblastomas, for which DNA was available from the primary tumor and the tumor at relapse. Obtaining the sequence of the entire coding region of the human genome (usually referred to as the "exome") was especially useful for modeling intra-tumor heterogeneity and clonal tumor evolution.

We use the normal-tumor-relapse model formulation from Figure 2.18 and parametrize it as follows. The effective somatic mutation rate in the tumor sample is set to $2.93 \cdot 10^{-6}$. This roughly models the expectation of at most 100 de-novo somatic mutations in typical neuroblastoma tumors found in our original study [583].

Since somatic mutation can also appear in the normal tissue, we set the corresponding effective somatic mutation rate to $2.8 \cdot 10^{-7}$, as reported by Oota [485]. Finally, the

```
species:
  heterozygosity: 0.001
  genome-size: 3.1e9
  ploidy:
    female:
      all: 2
      X: 2
      Y: 0
    male:
      all: 2
      X: 1
      Y: 1

samples:
  normal:
    sex: female
    somatic-effective-mutation-rate: 2.8e-7
  tumor:
    sex: female
    somatic-effective-mutation-rate: 2.93e-6
    inheritance:
      clonal:
        from: normal
    contamination:
      by: normal
      fraction: 0.1
  relapse:
    resolution: 100
    universe: "[0.0,1.0]"
    contamination:
      by: normal
      fraction: 0.53

events:
  germline:        "normal:{0.5,1.0}"
  somatic_normal:  "normal:]0.0,0.5["
  somatic_tumor:   "normal:0.0 & tumor:]0.0,1.0]"
  somatic_relapse: "normal:0.0 & tumor:0.0 & relapse:]0.0,1.0]"
```

**Fig. 2.18:** Extension of the Varlociraptor scenario specification in Figure 2.17 to include prior knowledge. We define the species (here *Homo sapiens*) in terms of genome size, heterozygosity (expected fraction of heterozygous loci), and sex-specific ploidy (number of chromosome copies). In addition, we model known somatic mutation rates, and define that the tumor inherits germline mutations from the normal sample.

tumor and the relapse sample tissue is usually contaminated by healthy cells. We use the amounts of contamination reported in the original study [583].

**Workflow** Analyzing sequencing data for genomic variants entails a variety of steps, which we outline in Figure 2.19. The entire analysis is implemented as a Snakemake workflow [343].

First, raw reads are processed by (a) trimming so-called sequencing adapters, (b) mapping them to the reference genome of the corresponding species, (c) removing putative duplicates from the Polymerase Chain Reaction (PCR), and (d) recalibrating base qualities. Sequencing adapters (a) are non-biological artifacts of the sequencing process. Since they are known beforehand, they can be removed from the reads by

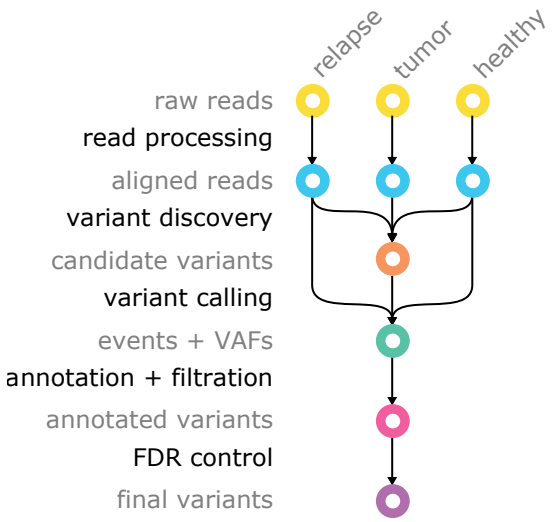



**Fig. 2.19:** Schematic representation of applied genomic variant calling workflow. Nodes represent original or derived data (gray labels in the left column), arrows represent processing steps (black labels in the left column).

performing an error-tolerant alignment between each read and the known sequence. We use Cutadapt to perform this step [431]. By mapping reads to the reference genome (b), we obtain the correct order and individual differences of each read compared with the representative genome of the underlying species. The resulting read alignments already contain all necessary observations for applying the Varlociraptor model. In order to obtain a signal of sufficient strength, sequencing protocols often entail the amplification of the DNA material via polymerase chain reaction [24]. The result is that there can be multiple reads from the same DNA fragment. Since Varlociraptor assumes each read to be an independent observation, it is important to remove such putative PCR duplicates, which we achieved using Picard tools [500]. Finally, the sequencing process sometimes causes artifacts to appear next to certain motifs [19]. In (d), we therefore use the base recalibration process from the Genome Analysis ToolKit (GATK [171]), which systematically investigates base alteration causing motifs and recalibrates the per base confidence scores in each sequencing read to reflect the uncertainty about whether an altered base is a true signal or a motif-induced artifact.

Second, the aligned reads are used to generate candidate variants. We use the tools Freebayes [222] and Delly [523] for this purpose. While the former covers small variants that can be covered by a single read (SNVs, MNVs, small insertions and deletions), the latter covers large, structural variants (large insertions and deletion, inversion, and duplications). Importantly, while both Freebayes and Delly provide their own statistical models for calling variants, we utilize them to generate candidate hypotheses. Both

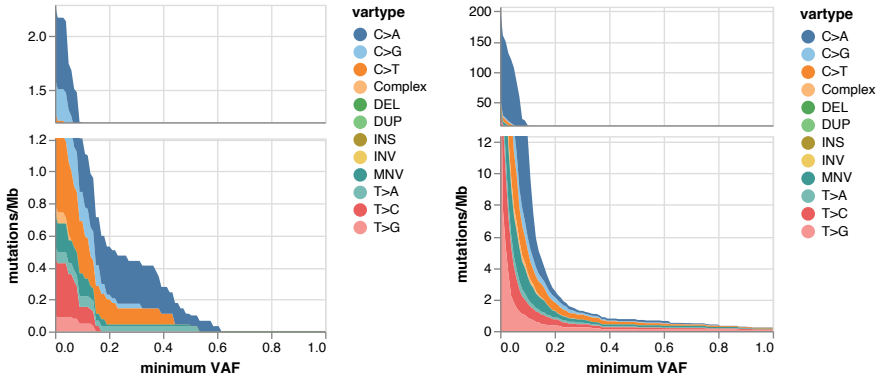



**Fig. 2.20:** Mutational burden of patient 1 from Schramm et al. [583] in the primary tumor (left) and relapse sample (right). The horizontal axis shows the minimum allele frequency, vertical axis shows the mutational burden as number of coding somatic mutations (calculated as expected value over the posterior probability for having a somatic mutation) per megabase of coding genome. The colors represent different types of mutations (see legend).

models are designed only for specific cases and are not generic enough to handle the composition of samples available in this dataset.

Third, we use Varlociraptor to (a) extract observations for each sample and each candidate variant and (b) apply the model as defined in the corresponding scenario for each patient in the study data.

Fourth, we (a) annotate the variant calls from Varlociraptor with their impact on proteins via the VEP tool [440] and (b) filter them for those that are of interest. In this case, we strive for three disjoint sets of variants

1. Variants that have been previously described as pathogenic or likely pathogenic in other studies.
2. Variants with high impact on the protein but which have not been previously described by other studies.
3. Variants with moderate impact on the protein but which have not been previously described by other studies.

Finally, we separately control the local false discovery rate for somatic variants in either the tumor or the relapse sample on each of the three sets.

**Insights** In the following, we summarize the most important insights from reanalyzing the study data with this workflow.

Figure 2.20 shows the mutational burden as a curve over the minimum allele frequency on an example patient. It can be seen that the burden for higher frequencies in general increases in the relapse sample compared with the tumor sample. This supports the hypothesis that the relapse sample originates from a subclone of the tumor

sample, which has survived therapy. Thus, one can expect that resistance-inducing mutations in the relapse sample become more abundant. Our findings contribute to the emerging view of resistance to cancer therapies as an evolutionary process. Selection of surviving clones results in mutational fingerprints that are specific for resistant or recurrent tumors. A better understanding of these genetic fingerprints is a prerequisite for identifying markers allowing early detection of resistance or tumor recurrence and enabling timely adjustment of therapies to further improve the survival and cure of cancer patients.

Future work entails the interpretation of individual recurrent deleterious gene and pathway alterations across the analyzed samples. Moreover, we aim to further improve the prior model of Varlociraptor such that assumptions about subclonal inheritance patterns can be incorporated as well.

Finally, we will combine the statistical approach of Varlociraptor with alignment free methods, as outlined in Section 2.3.2. Since Varlociraptor has to perform a realignment of read sequences anyway (see Section 2.3.3.2), we may replace the initial read alignment with an alignment free approach that yields a rough positioning of reads on the reference genome so that they can be selected for validating a given candidate variant with Varlociraptor. For this, it is necessary to accurately estimate the alignment uncertainty from the $k$-mer hits via, say, the strategy proposed in our previous work on PEANUT [344]. Finally, the detection of candidate variants with alignment free methods has to be extended beyond single nucleotide variants. Here, a possible strategy might be a hybrid approach where aberrations in $k$-mer counts are translated into an exact variant call by (a) collecting the causing reads, (b) assembling them into one or more consensus sequences [114], and (c) aligning these against the reference genome to determine the nature of the variant.

## 2.4 Bayesian Analysis for Dimensionality and Complexity Reduction

*Zeyu Ding*
*Katja Ickstadt*
*Alexander Munteanu*

**Abstract:** In this contribution, we will present Bayesian approaches for dimensionality and complexity reduction in the context of health-related problems. Following an introduction to Bayesian analysis in general, we will first show two examples of Bayesian variable selection methods for reducing the number of variables, one for binary data with an application to Single Nucleotide Polymorphisms (SNPs) for the HapMap dataset, and one for time-to-event endpoints with an application to glioblastoma data from the Cancer Genome Atlas. Second, we will present an approach for reducing statistical models, where we transfer the Merge & Reduce principle to maintain statistical summaries in streaming models. The variable selection approaches as well as the Merge & Reduce approach are important steps towards resource-aware data analyses.

### 2.4.1 Introduction

Machine learning obtains predictions by constructing a predictive model. Many machine learning algorithms are built on black-box models, as opposed to statistical learning, which is more concerned with the statistical properties of the model, such as the distribution of the variables and its parameters. In this section, we focus on the problem of data- and dimensionality-reduction in Bayesian statistics. Bayesian regression does not assume a fixed optimal solution for a dataset as in the frequentist case, but introduces a distribution over the parameter space. The likelihood function models the information that comes from the data, and the prior distribution models problem-specific prior knowledge. Our goal is to explore and characterize the posterior distribution, which, as a consequence of Bayes' theorem, is a compromise between the observed data situation and the prior knowledge that we assume for the parameters. For very large and high-dimensional datasets and settings where computational resources are scarce, the posterior distribution is hard to obtain. In general, our work focuses on algorithmic approaches that can be implemented in streaming and distributed environments to reduce the underlying problem in order to enhance the scalability of modern Bayesian regression approaches. In this contribution, we particularly concentrate on biomedical applications. Depending on the large-scale high-dimensional problems at hand, our interest centers around a) reducing the number of observations, b) reducing the number of variables, or c) reducing the underlying statistical model. Task a)

comprises approaches such as sketching or coreset methods. We have made various contributions on this topic; see, e.g., Geppert et al. [226] or Munteanu et al. [463]. For more details on this topic, see Section 3.2 in Volume 1 ("Coresets and Sketches for Regression Problems on Data Streams and Distributed Data"). Task b) is particularly relevant for high-dimensional settings in which the number of variables exceeds the number of observations. This is the main problem in many genetic applications. We will present two variable selection approaches for reducing the number of variables, one in the context of a single data source for Bayesian (logic) regression, one in the context of integrating multiple genomic data sources in a Bayesian Cox model. For task c) of reducing statistical models, we transfer the Merge & Reduce principle to maintain statistical summaries in streaming models (Geppert et al. [227]). We can compute the necessary results for a regression model by analyzing them blockwise and combining the summaries of each block in a structured way; more details are given in Section 2.4.3.

### 2.4.2 Variable Selection

The variable selection topic is one of the central problems in modern statistics. In some fields, researchers are often faced with the problem that the number of variables is larger than the sample size, which is commonly known as the $p$ larger than $n$ problem. In this case, training statistical models directly on the original dataset may lead to many problems, such as overfitting or the inability to use the Ordinary Least Squares (OLS) regression method. Therefore, selecting only those variables that are truly informative becomes a very important step in the process of constructing a model.

A popular approach for variable selection is to make the model sparse by forcing the coefficients of some variables to be zero or converge to zero during the regression process by $L_1$ or $L_2$ regularization. Another often-used approach is usually seen in the ensemble algorithm: all variables are first included in the model, and then the variables are ranked according to their importance by a variable importance measure, and the variables that have an impact on the dependent variable are selected after modeling. The following two approaches are well suited for medical applications: a variable importance measure approach employed prior to model building, and a regularized modeling approach using suitable priors and a stochastic search for variable selection.

#### 2.4.2.1 Variable Selection for High-dimensional Binary Data

For high-dimensional binary data, e.g., genetic marker data, interactions between variables are often more important than main effects, which increases the number of variables even further. In this section, we describe how to use so-called *leverage scores* and *cross-leverage scores* as measures of variable importance to select subsets of

explanatory variables while retaining valuable interactions. The leverage and cross-leverage scores are also compared with the more popular variable selection criteria *correlation coefficients* and $p$-values for univariate linear regressions. In contrast to common variable selection criteria, our approach focuses on variable selection prior to building the model. For more details on our method, see Parry et al. [492].

Obtaining the leverage and the cross-leverage scores requires the calculation of the *hat matrix* of the data matrix, which is a projection matrix that carries the information about the impact of variables instead of observations or responses.
For a data design matrix $\boldsymbol{X}$ and response vector $\boldsymbol{y}$, we set

$$\tilde{\boldsymbol{X}} = [\boldsymbol{X}, \boldsymbol{y}]^T \in \mathbb{R}^{(p+1)\times n} \tag{2.8}$$

and obtain the hat matrix

$$\boldsymbol{H} = \tilde{\boldsymbol{X}}(\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})^{-1}\tilde{\boldsymbol{X}}^T \in \mathbb{R}^{(p+1)\times(p+1)}. \tag{2.9}$$

Calculating the hat matrix requires the calculation of the inverse matrix $(\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})^{-1}$, which is not a stable calculation (see, e.g., Geppert et al. [226]). An alternative method is to obtain the hat matrix by *QR-decomposition*.

The $i^{th}$ element on the main diagonal of the hat matrix for the first $p$ elements $i = 1, \ldots, p$ is an importance measure of the $i^{th}$ variable, and is also known as its *leverage score* (see, e.g., Drineas et al. [181]). The *cross-leverage scores* are obtained by the off-diagonal elements of a hat matrix (Chatterjee and Hadi [122]) and describe the mutual influence of the $i^{th}$ variable on the $j^{th}$ one for $j = 1, \ldots, p$ and on the response variable for $j = p + 1$.

When selecting the most important variables prior to building the model, one important consideration is how many variables should be included in the pre-selection. Parry et al. [492] point out, based on the available literature, that using $\mathcal{O}(n \ln n)$ is a valid indicator for the optimal number of the pre-selected variables and that choosing that many variables allows the subsampled data matrix to remain at full rank.

For studying the relationship between binary variables as well as their interactions with respect to a response of interest, Ruczinski et al. [550] developed the so-called *logic regression*, an adaptation of a generalized linear regression model. In logic regression, the binary variables are not employed directly as independent variables, but are combined using the logical operators $\wedge$ (and), $\vee$ (or), and the negation !. The resulting Boolean expressions $L_i$ are called logic trees, and the corresponding logic regression model can be written as:

$$g(\mathbb{E}(\boldsymbol{y})) = \beta_0 + \sum_{i=1}^{T} \beta_i L_i, \tag{2.10}$$

where $g(\cdot)$ is the link function of the generalized linear model and $L_i$ for $i \in 1, \ldots, T$ are the Boolean combinations of the binary variables. In our genetic application, we employed the logit link function. The models are fitted with an iterative search algorithm based on Simulated Annealing (see, e.g., [550]).

While logic regression models are well suited for analyzing the association of binary variables and their interactions with a response, a drawback is that they can handle only a limited number of variables. Here, our approach using leverage or cross-leverage scores is helpful, since it reduces the number of independent variables prior to the analysis, while keeping the most influential ones.

We evaluate our approach using simulated data from different scenarios as well as the real HapMap dataset. We compare the leverage and cross-leverage scores for variable selection with correlation coefficients and p-values as selection criteria. In our simulation study, we have created different data scenarios in order to explore the ability of these variable selection measures to select the correct variable when the number of independent variables and the sample sizes vary. We also study these measures for main effects and for the presence of different higher-order interactions.

The genetic HapMap data was collected as part of an international collaboration to develop a haplotype map of the human genome. The subset considered in our work is available in the R-package `SNPassoc` [234]. For our analysis, we considered 7648 human single-nucleotide polymorphisms (SNPs) employed as binary variables for $n = 120$ individuals from two separate ethnic groups. The dependent variable was set to 1 when an individual is from central Europe (CEU), and 0 when it belongs to the Yoruba group (YRI) that inhabits western Africa.

For the simulated data, we show the results by plotting the distributions of leverage scores and cross-leverage scores for the different scenarios (see Parry et al. [492]). The leverage scores perform better in selecting important variables when only main effects are present, whereas cross-leverage scores distinguish variables of higher-order interactions better. A change in the number of variables $p$ did not have much effect on the overall performance of the two measures when the number of irrelevant variables increased. By contrast, when the sample size increases, both scores for the informative variables increase while both measures for the irrelevant variables decrease, meaning that more samples make it easier to find the truly informative variables for both measures. Moreover, we compare the leverage and cross-leverage scores to the correlation coefficients and to p-values.

We also propose and investigate a way of combining cross-leverage scores and leverage scores by selecting the variables using these two metrics separately and taking the union of all selected variables. We show that this combined approach is superior to using just one metric alone (see Parry et al. [492]).

For the HapMap data we present here a raster plot (see Figure 2.21). It is a graphical representation of a matrix containing SNP data, where the rows correspond to individuals and the columns correspond to SNPs. We take subsets of size $\lceil 120\ln(120) \rceil = 575$ of the most important SNPs using cross-leverage scores (CLS), leverage scores (LS), correlations (COR), and p-values, respectively to be comparable. We can easily distinguish the two groups using cross-leverage scores; thus, the selected SNPs can be used to classify individuals of the two groups. Using leverage scores, it is almost impossible to distinguish the two groups. The variables selected using correlation coefficients or

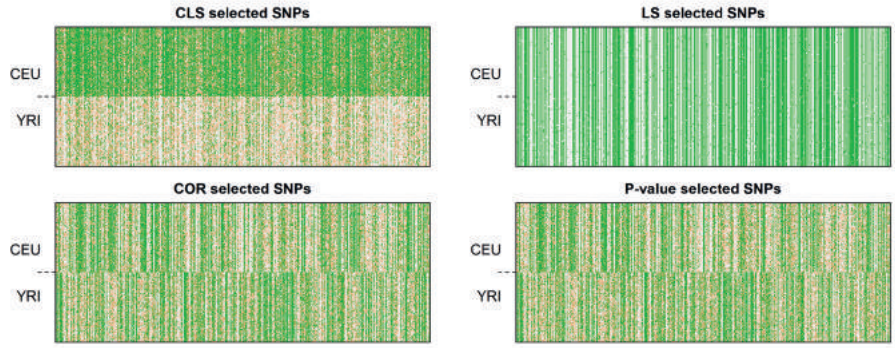



**Fig. 2.21:** Raster plots of all 120 subjects from two continental regions with respect to a subset of 575 out of 7648 SNPs (green/light pink denotes homozygotic individuals and white denotes heterozygotic individuals), selected using CLS, LS, COR, and p-values, respectively.

p-values show hints of blocks corresponding to the two groups, but much less than those selected by the cross-leverage scores.

Based on our simulation study and the real data example, cross-leverage scores turn out to be a promising tool for variable selection prior to model building, especially in the presence of higher-order interactions, and leverage scores prove useful for selecting main effects. A combination of leverage and cross-leverage scores usually further improves variable selection.

#### 2.4.2.2 Variable Selection for High-Dimensional Survival Data

In the following, we briefly summarize a Bayesian approach that combines the recent progress in the following areas in one model:

1. analyzing time-to-event data in high dimensions
2. variable selection in a high-dimensional setting, and
3. integration of several data sources.

Here, we give a brief overview. For more details on this research, see Treppmann et al. [691].

We usually encounter time-to-event endpoints or survival data in cancer studies. To analyze them, Cox [149] created the semi-parametric proportional hazards regression model that considers the relation between covariates and the hazard function. Therefore, the Cox model has often been applied to low-dimensional data. However, in biological applications with genomic data, we often deal with high-dimensional settings with more variables than subjects. This shows the need for a high-dimensional survival time model. With this in mind, Lee et al. [382] developed a Bayesian version of the Cox model for right-censored survival data, where high dimensions are treated by a

regularization of the regression coefficient vector via Laplace priors. Another contribution to survival prediction for high-dimensional data based on the Cox model in this volume is presented by Rahnenführer et al. in Section 2.5.

A second aspect resulting from a huge number of variables is the need for variable selection. In a high-dimensional setting, common methods like best subset selection as well as backward and forward selection prove to be unsuitable for various reasons. Bayesian techniques provide a good alternative to search stochastically over the entire parameter space, especially as they implicitly address model uncertainty. One example is the stochastic search variable selection (SSVS) by George and McCulloch [224], an approach commonly used in regression analyses. It is a flexible and intuitive procedure that uses data augmentation for the selection task and includes shrinkage.

Moreover, the Bayesian setting provides a way for incorporating additional data sources. The interest in such integrative statistical analyses is growing steadily, as technological progress makes it possible to collect different genome-wide data systematically. The integration of more than one information source can lead to an improvement in the performance of risk prediction models and, therefore, to a more detailed understanding of the biology of diseases. For a recent overview of integrative Bayesian analyses in molecular biology, see Ickstadt et al. [295].

In conclusion, our approach combines the variable selection procedure of George and McCulloch [224] with the Cox proportional hazards model of Lee et al. [382] in one Bayesian model and integrates a further data source by means of an informed prior.

As mentioned, for right-censored survival data in high dimensions with the number of variables $p$ being (much) larger than the number of subjects $n$, Lee et al. [382] developed a Bayesian variant of the semiparametric proportional hazards model $\lambda(t|x) = h_0(t) \cdot \exp(x^T\beta)$ by Cox [149]. Here, $h_0(t)$ denotes the underlying baseline hazard function, $t$ the survival time of a person with covariable vector $x = (x_1, \ldots, x_p)^T$, and $\beta = (\beta_1, \ldots, \beta_p)^T$ the vector of regression coefficients. By a finite partitioning of the time axis, $0 < s_0 < s_1 < s_2 < \ldots < s_J$ with $s_J > t_r$, $\forall r = 1, \ldots, n$, such that the breaks are points where at least one event occurs, and the last event lies inside the last interval, Lee et al. [382] obtain the following grouped likelihood introduced by Burridge [110]:

$$
\begin{aligned}
L(\mathcal{D}|\beta, h) \propto \prod_{j=1}^{J} \Bigg( & \exp\Bigg( - h_j \cdot \sum_{l \in (\mathcal{R}_j - \mathcal{D}_j)} \exp(x_l^T \beta) \Bigg) \\
& \cdot \prod_{\xi \in \mathcal{D}_j} \left( 1 - \exp\left( -h_j \cdot \exp(x_\xi^T \beta) \right) \right) \Bigg) \qquad (2.11) \\
& h_j \sim \Gamma(a_{0j} - a_{0j-1}, c_0) \\
\text{with } a_{0j} = c_0 \cdot H^*(s_j) & \text{ and } c_0 > 0, \quad j = 1, \ldots, J.
\end{aligned}
$$

In this context, $\mathbf{D} = \{(x, \mathcal{R}_j, \mathcal{D}_j) : j = 1, \ldots, J\}$ denotes the observed data, with $\mathcal{R}_j$ being the risk set and $\mathcal{D}_j$ being the event set regarding the $j^{th}$ interval. In case of choosing a

Weibull distribution for the monotonously increasing function $H^*(t)$ (with $H^*(0) = 0$), we obtain $H^*(t) = \eta_0 \cdot t^{\kappa_0}$ with hyperparameters $(\eta_0, \kappa_0)$.

For the variable selection, we apply the SSVS procedure of George and McCulloch [224]. Under the assumption that the variances of the regression coefficients of variables included in the model are equal, the prior distribution for $\beta_i$ conditioned on $\gamma_i$ has the following form:

$$\beta_i|\gamma_i \sim (1-\gamma_i) \cdot N(0, \tau^2) + \gamma_i \cdot N(0, c_b^2 \cdot \tau^2), \quad i = 1, \ldots, p, \tag{2.12}$$

with small $\tau^2 > 0$ and $c_b^2 > 1$. Following the concept of data augmentation (Tanner and Wong, 1987 [673]), the indicator vector $\gamma$ states whether the associated variables are included in the model or not.

Inference is based on Markov Chain Monte Carlo (MCMC) algorithms. For updating the full conditional distribution $P(\beta_i|\beta_{-i}, \gamma, h, \mathbf{D})$ with

$$\beta_{-i} = (\beta_1, \ldots, \beta_{i-1}, \beta_{i+1}, \ldots, \beta_p)^T \tag{2.13}$$

we use the special random walk Metropolis-Hastings method with adaptive jumping rules proposed by Lee et al. [382]. Moreover, the conditional distributions $P(\gamma_i^{it} = 1|\beta^{it}, \sigma^{it}, \gamma_{-i}^{it})$ with $\gamma_{-i}^{it} = (\gamma_1^{it}, \ldots, \gamma_{i-1}^{it}, \gamma_{i+1}^{it}, \ldots, \gamma_p^{it})^T$ are derived by means of the Bernoulli distribution. The full conditional distribution $P(h_j|h_{-j}, \beta, \gamma, \mathbf{D})$ with $h_{-j} = (h_1, \ldots h_{i-1}, h_{i+1}, \ldots, h_J)^T$ is approximable by a Gamma distribution. To update $\beta$, $\gamma$ and $h$ iteratively according to the full conditional distributions described above, a Gibbs sampler is appropriate.

In addition to a simulation study, which will not be discussed here, we applied our method to a dataset of glioblastoma multiforme (GBM) patients, retrieved from the Cancer Genome Atlas [471] database. In adults, glioblastoma is the most frequent and the most rapidly growing brain tumor. The used dataset comprises 210 patients and includes survival and gene expression data as well as associated copy number variation (CNV) data, which are used to construct an informative prior. We restrict the analysis to the 1000 genes that show the greatest variance in their values. The underlying assumption is that genes with low variability are probably not well suited to distinguish between patients with a good and patients with a poor survival prognosis. We divide the dataset into a training dataset for the model fitting of 140 patients and a test dataset for the evaluation of 70 patients.

For the analysis, we assume a prior expected number of selected variables of $k = 20$. We construct an informative prior such that the prior inclusion probability $\pi_i^{\text{CNV}}$ of the $i^{th}$ variable is proportional to its standard deviation $\sigma_i^{\text{CNV}}$ of the copy number variation data for the associated genomic region across patients. Thus $\pi_i^{\text{CNV}}$ is defined as

$$\pi_i^{\text{CNV}} = k \cdot \frac{\sigma_i^{\text{CNV}}}{\sum_{j=1}^{p} \sigma_j^{\text{CNV}}}, \quad i = 1, \ldots, p. \tag{2.14}$$

For comparison purposes, we use the non-informative prior $\pi = (k/p, \ldots, k/p)^T$.

In the course of the evaluation, we consider the posterior means and standard deviations of the parameters $\beta$ and $\gamma$. To decide which variables to select, we first determine the mean model size $p_m$ by rounding the average of selected variables per iteration. Then we select the $p_m$ variables with the highest selection probability.

We conduct a combined analysis of five Markov chains, each with a length of 100 000 of which the first 10 % are removed as burn-in.

The simulation result from Table 1 in Section 2 in Treppmann et al. [691] shows that the posterior selection probabilities differ greatly depending on whether the informative or uninformative prior was used. Only three genes are among the $p_m = 10$ (uninformative) or $p_m = 9$ variables (informative prior) with the highest posterior selection probability in both cases.

To evaluate the goodness of the prediction, we consider prediction error curves and determine the integrated Brier score [241, 589] in comparison with the Kaplan-Meier estimator without any covariates (reference approach). This shows that in the case of the informative prior, our model improves the prediction performance relative to the reference approach, while this is not observed for the uninformative prior. The examination of trace plots to the simulated MCMC chains indicates that the chains move quickly into desired regions of the model space and exhibit good mixing performance.

Both in our application to glioblastoma data and in our simulation study, we have shown that the inclusion of a second data source has distinct potential for improvement in terms of prediction quality. However, this is only the case if the second data source provides an informative prior. This requires that variables with an increased prior selection probability tend to be truly associated with the response. However, since this is usually not known in practice, a comparison with the model using an uninformative prior is always appropriate.

Due to the Bayesian modeling, we obtain full inference, especially concerning the posterior selection probabilities. The joint analysis of all variables offers the great advantage that posterior selection probabilities of whole sets of variables can be considered. One example would be a group of genes that has been shown to be particularly influential in previous studies.

Since in MCMC approaches there is usually a trade-off between the computational cost and the accuracy of the results, efficient programming is of particular importance. We recently re-implemented our approach in Python, eliminating some inefficiencies of our previous implementation of the algorithm in R.

### 2.4.3 Merge & Reduce for Statistical Models

Datasets with a massive number of observations have become more and more common, making scalability a major challenge for modern data analysis. For many statistical methods, these amounts of data lead to an enormous consumption of resources. A prominent example is linear regression, an important statistical tool in both Bayesian

and frequentist settings. On very large datasets, regression analysis becomes increasingly demanding with regard to running time and memory consumption, making the analysis tedious or even impossible. We propose a method called Merge & Reduce to address these scalability limitations in regression analyses. Merge & Reduce is well known in computer science and has mainly been used for transforming static data structures to dynamic ones with little overhead (Bentley and Saxe [56]). Instead of reducing the data to approximate the full dataset with respect to some model, we propose using the statistical models derived from small batches as concise summaries. Combining these statistical models via the Merge & Reduce framework, we can turn an offline algorithm into a data stream algorithm.

Here, we focus on streaming to deal with massive datasets, where a data stream algorithm is given an input stream of items, like numerical values, vectors, or edges of a graph at a high rate. The algorithm is allowed to make only one single pass over the data. As the items arrive one by one, it maintains a summary of the data that was observed so far in the form of, say, subsample or a summary statistic. Despite our focus on a streaming-setting, we stress that the Merge & Reduce scheme can also be implemented in distributed environments.

Our contribution is to develop the first Merge & Reduce scheme that works directly on statistical models. We show how to design and implement this general scheme for the special cases of (Bayesian) linear models, Gaussian mixture models, and generalized linear regression models in Geppert et al. [227]. Here, we will restrict ourselves to Bayesian linear models and evaluate the resulting streaming algorithm on simulated datasets. We demonstrate that we obtain stable regression models from large data streams and that the Merge & Reduce schemes produce little overhead.

#### 2.4.3.1 Method

In our Merge & Reduce method for statistical data analysis, we iteratively load as many observations into the memory as we can afford. On each of these blocks, we apply a classical algorithm to obtain, say, the parameters of a statistical model, some (sufficient) statistics or a summary of the presented data; in short, a model. Models are merged according to certain rules, eventually resulting in a final model that combines the information from all subsets. Merge & Reduce leads to stable results, where every observation enters the final model with equal weight, thus ensuring that the order of the data blocks does not bias the outcome toward single observations.

In order to design a streaming algorithm for a specific statistical analysis task, we need to choose an appropriate model as a summary statistic for each block of data. The two main ingredients that we need to implement for this particular choice of a model are called *merge* and *reduce*.

1. Let $M_1, M_2$ be the models obtained from the analysis of data blocks $B_1, B_2$, then the output of **merge**$(M_1, M_2)$ is a model $M$ for the union $B_1 \cup B_2$ of the input data blocks.

2. Let $M$ be a model for data block $B$ that has become too large, i.e., $|M| \geq 2T$ for some threshold $T$ (e.g., by repeated merge operations), then **reduce**($M$) computes a model $M^{'}$ of size $|M^{'}| \leq T$ for the same block $B$.

Summarizing the statistical analysis and implementing the Merge & Reduce functions on statistical models are not trivial undertakings. The approach heavily depends on the statistical method employed and on the representation chosen to store the model. Here, we present the novel, general concept and discuss how to design the Merge & Reduce functions for the example of linear regressionindexRegression!linear in the Bayesian setting.

We first describe how the Merge & Reduce functions interact in a structured way to perform the statistical analysis task on the data block-by-block while maintaining a model for the whole subset of data presented so far. The data structure consists of $L = O(\log(n/n_b)) = O(\log n)$ buckets for a sufficiently small block size $n_b$ to fit into the main memory of the machine. The buckets store one statistical model each. Initially, they are all empty. One bucket, the working bucket $B_0$, is dedicated to store the model for the current batch of data, while each of the other buckets $B_i$ stores one model on its corresponding level $i \in \{1, \ldots, L\}$ of a binary tree structure formed by the merge operations, see Figure 2.22. The data structure works in the following way. First, we read one block of data of size $n_b$. We perform the statistical data analysis on this block only. The model that summarizes the analysis is stored into $B_0$. We begin to propagate the model in the tree structure from bottom to top by repeatedly executing Merge & Reduce operations on each level. If $B_1$ is empty, then we just copy the model from $B_0$ to $B_1$ and empty $B_0$. Otherwise, we have two models that are siblings in the tree, so we merge the two into $B_0$, empty $B_1$ and proceed with $B_2$. Again, if it is empty, the model from $B_0$ is stored in $B_2$ and the propagation terminates. Otherwise, we have two siblings that can be merged and propagated to the next higher level in the tree. In general, the propagation stops as soon as the bucket on the current level is empty. When this happens, the update of the data structure has completed, and we can move on to reading and analyzing the next block of input data. This is repeated until the end of the stream. Notice that except for the additional working bucket, we need to store at most one bucket on each level at a time, since two siblings are merged immediately.

A linear regression model is given by

$$Y = X\beta + \epsilon, \tag{2.15}$$

where $Y \in \mathbb{R}^n$ is the dependent variable and $X \in \mathbb{R}^{n \times d}$ is the design matrix containing the observations $x_1, \ldots, x_d$ of the independent variables. The error term $\epsilon$ is assumed to be unobservable and is usually modeled by a normal $N(0, \sigma_\epsilon)$ distribution, $\beta \in \mathbb{R}^d$ is the unknown parameter vector of regression coefficients that we wish to estimate. In Bayesian regression, $\beta$ is assumed to be random and follows a distribution. Interest

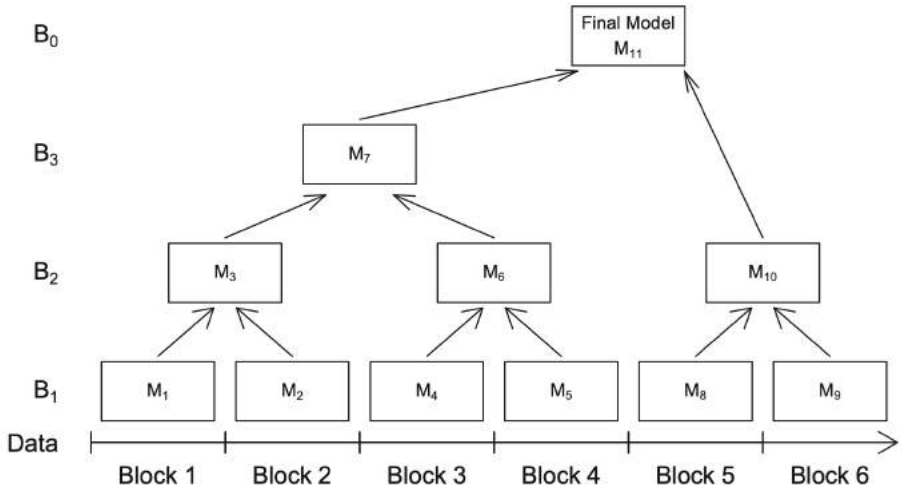



**Fig. 2.22:** Illustration of the Merge & Reduce principle taken from Geppert et al. [227]. The data is presented in the form of a stream and subdivided into Blocks 1 through 6 of equal size. Models $M_1$ to $M_{11}$ are numbered in order of their creation throughout the execution. Arrows between models indicate the Merge & Reduce operations. Sibling models are deleted right after their parents' creation. Thus, only one model is stored on each level, i.e., in buckets $B_1$ to $B_3$, at a time. The working bucket $B_0$ acts on all levels, eventually holding the final model after postprocessing at the end of the stream.

centers around the posterior distribution of the parameters that can be written as

$$p_{post}(\beta|X, Y) \propto \mathcal{L}(Y|X, \beta) \cdot p_{pre}(\beta). \tag{2.16}$$

Thus the posterior distribution $p_{post}$ can be seen as the product of the prior distribution $p_{pre}$ and the likelihood function $\mathcal{L}$ of the parameters. In many cases, the posterior distribution cannot be obtained directly by analytical means, but must be approximated by some sampling approaches. The most popular method is to apply Markov Chain Monte Carlo (MCMC) random sampling, but this requires a very large number of simulations. When the sample size is also very large, such sampling simulations are demanding in terms of computer memory and computing speed.

We will apply the Merge & Reduce method to Bayesian linear regression. Since we use the MCMC method to approximate the posterior distributions of the parameters $\beta$, for each estimated parameter $\beta_j$, we collect the mean $\bar{x}_j$, the median $\tilde{x}_{.5}$, the lower and upper quartiles $\tilde{x}_{.25}$ and $\tilde{x}_{.75}$, 2.5 % and 97.5 % quantiles $\tilde{x}_{.025}$ and $\tilde{x}_{.975}$, and the standard deviation $\sigma_j$ of the posterior distributions. Then, the collected statistics can be summarized as

$$S = (\bar{x}_1, \ldots, \bar{x}_d, \tilde{x}_{p,1}, \ldots, \tilde{x}_{p,d}, \sigma_1, \ldots, \sigma_d), \tag{2.17}$$

where $p \in \{0.025, 0.25, 0.5, 0.75, 0.975\}$. We use a weighted average of the statistics to merge these statistics vectors, and the weights are dependent on the size of the sample.

#### 2.4.3.2 Simulation and Results

The main parameters used to generate datasets for the simulation study are the number of observations $n$, the number of variables $d$ and the standard deviation $\sigma_\epsilon$ of the error term $\epsilon$. Different numbers of observations per block $n_b$ are also chosen. The setup of the simulation study and the parameter values are similar to those in the simulation study in Geppert et al. [226].

In this study, all assumptions of a linear regression model are met. A varying fraction of the variables has an influence (large or small) on the dependent variable, while the remainder is not important for the explanation of $Y$.

We evaluate the Merge & Reduce approach by calculating the squared Euclidean distances $e_m^2$ of the statistics in $S$, specified in Equation 2.17, between the original model and the Merge & Reduce model for all simulation models $m = 1, \ldots, M$. If the Euclidean distance is close to 0, then the Merge & Reduce approach approximates the results of the original model accurately. We see in Figure 2.23 that, as the ratio $\frac{n_d}{d}$ increases, the difference between the medians, obtained from the Merge & Reduce approach and from the Bayesian original model, evaluated by their squared Euclidean distance, is quite small. The majority of the squared distances is close to 0. For more details on the results of other summary statistics of the posterior distributions, see Geppert et al. [227].

#### 2.4.3.3 Conclusion and Outlook

Merge & Reduce is suitable for Bayesian regression models. The goodness of the approximation depends on the ratio of observations per block and variables $\frac{n_d}{d}$ and the goodness of fit of the original model. The first condition can easily be controlled by the data analyst, especially in a setting with large $n$. The second condition may require care when building the model.

For the implementation of the Merge & Reduce approach in principle, it is only necessary to choose an appropriate statistical model and to implement Merge & Reduce operations for this specific type of model. However, the design of such operations is not trivial in general. In particular, we showed in Geppert et al. [227] how to design such operations for the case of Bayesian linear regression, Gaussian mixture models, and generalized linear models. Implementing the Merge & Reduce approach for the Bayesian Cox model is a next step in future work.

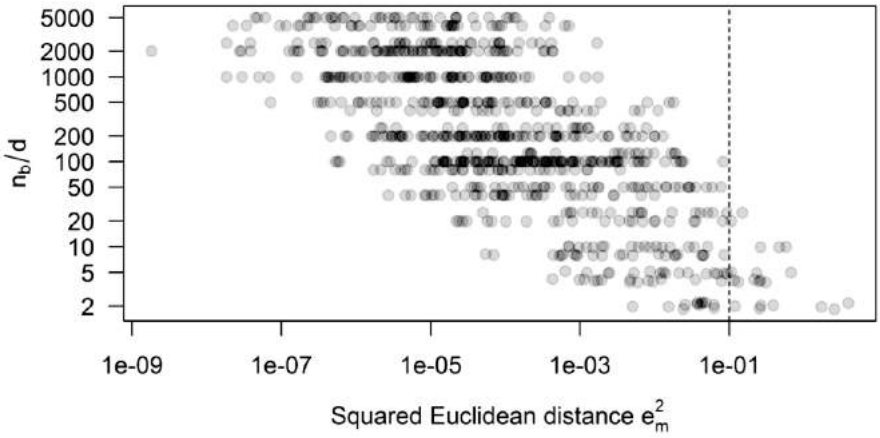



**Fig. 2.23:** The figure, taken from Geppert et al. [227], shows a scatterplot of the effect of observations per block per variable $\frac{n_b}{d}$ on squared Euclidean distances $e_m^2$, $(m = 1, \ldots, M)$ between posterior medians for the Bayesian Merge & Reduce approach. The *x*- and *y*-axes are drawn on a logarithmic scale, and observations are drawn as partially transparent points: gray points mean single observations; black points represent multiple observations at roughly the same location. The vertical dashed line is at 0.1.

### 2.4.4 Overall Conclusion

We are concerned with several issues that arise when dealing with insufficient computational resources. In this contribution, we show ways of reducing high-dimensional data and simplifying complex models in the context of biomedical applications.

In the first two sections, we introduce variable selection strategies in two different high-dimensional datasettings. In the first scenario, we describe a variable importance measure approach for genetic SNP data, employed prior to model building. In the second scenario, we formulate a variable selection strategy for high-dimensional time-to-event data integrating several data sources in the context of a Bayesian Cox model. In the third section, we use the Merge & Reduce approach for massive data to analyze and build statistical models on small batches separately and recombine them in order to cope with the problems of limited running time and insufficient memory.

All approaches in this contribution improve the efficiency of statistical learning when facing the limited computational resources of host devices. They reduce the computation time, energyconsumption, and memoryusage of learning individual models without losing the high accuracy of the model. This, in turn, allows us to analyze more complex models on massive data with an efficient use of computational resources.

## 2.5 Survival Prediction and Model Selection

*Jörg Rahnenführer*
*Michel Lang*
*Jakob Richter*

**Abstract:** Survival analysis comprises statistical methods for time-to-event data. The main prediction tasks include the estimation of the influence of prognostic factors for, say, medical treatments, and the modelling and prediction of survival times using regression models. In recent years, in molecular medicine, many omics technologies have been developed, generating complex high-dimensional genetic data that can be used as predictors.

For such complex tasks, the selection of the best prediction method out of a large set of candidates, along with potential feature selection and hyperparameter optimization, represents an optimization task under resource constraints. In this section, approaches for tackling the model selection problem in survival analysis are presented, specifically using Bayesian optimization and addressing feature selection for high-dimensional data.

### 2.5.1 Introduction

In medicine, times to events are compared between groups to estimate the effect of prognostic factors and medical treatments, and regression models are used to model and predict survival times of cells, animals, or patients. For two decades, high-dimensional genetic and genomic variables have been generated and analyzed as potential predictive and prognostic factors in biological and medical scenarios. The very large number of variables requires developing and using tailored methods to describe the complex relationships. Popular modeling approaches are based on penalized regression methods, gradient boosting methods, survival trees, and survival forests, often combined with suitable feature selection methods.

In recent years, machine learning approaches were used to find the best survival method from a large set of candidates. Efficient approaches are required, since it is crucial that runtimes especially in resampling scenarios with many repeated estimation tasks be kept short, especially for complex high-dimensional predictor settings. In CRC 876, we applied modern Bayesian optimization (BO) [303] techniques to efficiently identify the best survival prediction method, by modeling the relationship between the choice of the survival prediction method (as well as its hyperparameters) and its performance or quality, using so-called surrogate functions. On several lung cancer datasets the new approach was superior to established benchmark approaches [367,

369]. After a short introduction into the analysis of time-to-event data in Section 2.5.2, model selection for survival analysis is discussed in Section 2.5.3. To solve this task, various R packages were implemented both for the general candidate selection and for parallelization, as presented in Section 2.5.6.

The same principle idea was used in a scenario, where survival predictions for a specific cancer dataset are to be improved, by adding data from similar datasets. This is a frequent situation in, say, cancer survival analysis, where patient numbers in clinical trials are limited due to ethical, financial, and administrative reasons, but similar treatments are applied, e.g., in other clinical centers. However, simply adding similar datasets to the one of interest potentially deteriorates the predictions, due to structural differences between the datasets. Instead, one can estimate dataset-specific weights that determine how strong these datasets should be considered. In CRC 876, we applied BO to determine such optimal weights for inclusion of the respective observations in appropriate weighted likelihood-based modelling approaches [531], as shown in Section 2.5.4 below. In two other projects related to feature selection, we developed improved methods based on two-fold subsampling schemes [383] and benchmarked filter methods against each other for high-dimensional data [95]. These analyses are described in Section 2.5.5.

### 2.5.2 Analysis of Time-to-Event Data

Survival analysis, also called event-time analysis, deals with the analysis of times to certain events and is used in many application fields. In medicine, the overall survival (OS) of patients is often of direct interest. Alternatively, Progression-Free Survival (PFS) is frequently analyzed, which includes Event-Free Survival (EFS) and recurrence-free survival. An important property of survival data is that they are often not fully observable, such as when patients in a clinical trial have not yet experienced the event of interest at the time the trial ends and the data is analyzed. This situation is called right-censoring, since for patients without an observed event, the survival time must be greater than or equal to the time until the end of the study. Depending on the type of missing information, many other censoring mechanisms are defined and considered in the analysis techniques.

Specialized statistical methods for analyzing survival data have been developed and are widely used in literature and in practice. Most prominent are the Kaplan-Meier estimator for estimating survival curves under right-censoring, the log-rank test for comparing survival between patient groups, and the Cox proportional hazards model [149] for estimating survival dependent on a number of explanatory variables, such as tumor size or age in oncological studies.

In regard to the evaluation of performance, seemingly obvious approaches lead to wrong interpretations of the results. For example, simply predicting the event indicator that indicates if a patient has survived until the end of the study, neglects the different

time intervals that have passed since the patients entered the study. By contrast, methods based on hazard rates that model the instantaneous failure rates at different time points can cope with the censoring mechanisms. Alternatively, parametric likelihood methods that consider the missing information can also be used.

For evaluating the prediction accuracy of survival models, several suitable measures have been developed. Concordance statistics, in particular Harrell's C-index and the area under the (time-dependent) ROC curve, are the most popular measures. However, they consider only the discrimination ability of a survival model and not the calibration. This means that monotone transformations of predicted values of survival outcomes do not change the concordance score, which limits the interpretability of the score for clinicians. Alternatively, the Brier score is also widely used. It considers both calibration and discrimination, but interpretation is also difficult. An advantage is that it can be related to a time-specified horizon. A discussion of these important properties and an adaptation of the Brier score can be found in Kattan and Gerds [313].

In preclinical and clinical studies, genetic factors are of interest, and modern high-throughput technologies provide many thousand potential explanatory variables. Even the popular but controversial rule of thumb that the number of events per variable should be at least 10 cannot be used as a basis for sample size planning. Instead, tailored statistical and machine learning approaches are required. Aspects to consider for model selection in this scenario are discussed in the next subsection.

### 2.5.3 Model Selection for Survival Analysis

Model selection in survival analysis, compared with model selection in classical machine learning setups such as regression or classification, presents numerous additional challenges.

First, instead of having to solve a learning task with many observations (e.g. patients) and comparatively few variables, in survival analysis we often face a low sample size problem. Even worse, with the rise of omics technologies, thousands to hundreds of thousands of genetic features need to be included in the analysis to be able to identify the most important genes. However, most machine learning or statistical learning algorithms have been designed and heavily optimized for a large sample size $n$, and usually have worse than quadratic runtime in the number of features $p$. For this reason alone, we often face runtime issues in the $n \ll p$ scenario.

Second, a dual objective is often pursued, and the predictive performance of the models is not the only target criterion. Instead, it is desired to identify the important features (clinical covariates, genetic dispositions, or genes) in the given medical context. A good predictive performance often ensures that the model describes the data in a meaningful way, which is the prerequisite for extracting a set of important features. This restricts the analysis to models that either come with an embedded feature selection or models that still work reasonably well after a feature filter has been applied.

Third, all performance measures in survival analysis require a large enough test set to yield performance values that lead to reliable statements. For example, the popular C-index mentioned above assesses the ranking of the predictions for survival in the test set by comparing it with the true, observed ranking of survival times (while correcting for censoring). Obviously, having too few observations in the set results in a high variance of the performance estimator. Since usually only a few hundred observations are available in total per dataset, the number of repetitions must be increased during resampling in order to account for the larger variance. Of course, this exacerbates the runtime problems one is already facing.

These points taken together form a hard tuning problem with the following characteristics:

1. The models form a black box from the perspective of the tuner, as there are no known derivations. Therefore, the optimization problem itself is also called "black box".
2. To assess the predictive performance of a hyperparameter configuration $\theta$ and its resulting model, the data needs to be split into a training set and an independent test set. This introduces stochastic components into our tuning problem at the latest (some learners are non-deterministic either way).
3. The search space spanned by the hyperparameters to be tuned usually includes both numerical and categorical variables. This precludes the use of many tuners derived from discrete and steady optimization.
4. Each model fit is potentially resource demanding, in terms of computational time, memory requirements, or communication costs. The key word here is "potentially". Some models, e.g., a simple Cox model augmented with an aggressive feature filter, can easily be fitted in less than a minute even with $n = 200$ observations and $p = 10^6$ variables (features). Other learners, such as support vector machines, require a complete day for the same task on the same hardware while simultaneously consuming several orders of magnitude more memory. Obviously, the resource requirements are very heterogeneous, which should be taken into account during the mandatory parallelization.
5. Last but not least, during hyperparameter optimization, we generally have to deal with an additional type of censoring (besides the censoring of the survival times): It is not unusual that the learner implementations crash from time to time due, say, to numerical problems. And since the tuning is usually distributed on larger computation sites with shared and contested resources, computational jobs can hit a wall time and be killed by a scheduler. In such a case, the missing performance score must be imputed with a number to continue with the tuning, and it is unclear which value to choose.

Over the last decade, special strategies addressing the difficulties of hyperparameter optimization have emerged. An overview is given by Bischl et al. [68]. Roughly speaking, hyperparameter optimization is about finding the configuration $\theta$ of a model, which

leads to the best predictive performance (evaluated on an independent test set). If the evaluation of a single configuration is sufficiently expensive with regard to computational resources like runtime, every evaluation counts, which also means that rather wasteful optimization methods are not applicable. This applies, among others, to Evolutionary Algorithms (EAs). EAs usually require many hundreds of configurations before being able to make the first targeted decisions. Instead, a tuner that optimizes more aggressively from the start is needed.

One tuner that addresses all the problems of the outlined expensive black-box optimization problem is *iterated F-racing* [421]. The basic idea of F-racing is to race a population of configurations against each other, and to eliminate in each iteration candidates that are underperforming based on a Friedman test. Iterated F-racing extends this approach by assuming a probability distribution over the search space. This distribution gets updated iteratively so as to be centered around some elite configurations. We applied this tuning approach to a broad range of survival pipelines (consisting of the feature filter and the survival model) [369]. The benchmark considers 12 different datasets of four breast cancer cohorts where each dataset consists of clinical and/or genetic variables (features). The architecture of the pipeline, i.e. the choice of filter and the choice of model, is encoded as virtual hyperparameters passed to the tuner. This way, dominated combinations of filters and models are getting fewer evaluations, giving the tuner more opportunity to exploit hyperparameters of more promising combinations. As a baseline, four reasonable approaches that are popular in practice but are arguably less computationally intensive have been evaluated. To the best of our knowledge, this was the largest benchmark of survival models up to that point. In comparison with the baseline approach, the tuning yields significantly better results in terms of the C-index. The caveat is the effort to archive the results: with more than 10 000 hyperparameter evaluations, the tuning cannot be applied easily on new data or cohorts.

Another tuner which perfectly fits the requirements of hyperparameter optimization in survival analysis is Model-Based Optimization (MBO). Its performance has been verified by Lang [367] where the benchmark study from Lang, Kotthaus, Marwedel, Weihs, Rahnenführer, and Bischl [369] has been extended, with more datasets, more filters, more models, and more time budget.

Figure 2.24 visualizes the survival probability in the included cohorts. Although the studies are all on lung cancer, and share the same set of clinical and genetic features, they differ considerably with respect to survival times. This is a frequently observed characteristic, and makes the careless merging of the datasets into a larger dataset with more observations inappropriate. As a result, in this domain, it is usually not possible to configure a single model to perform sufficiently well on all cohorts. Instead, for each cohort, tuning starts from zero. One goal of the analysis was to thin out the portfolio of methods to consider for a new tuning run. If, e.g., only two pipelines consisting of a filter and a model have to be tried, the computational effort required for tuning is significantly reduced, rendering the tuning for new cohorts on a single computer

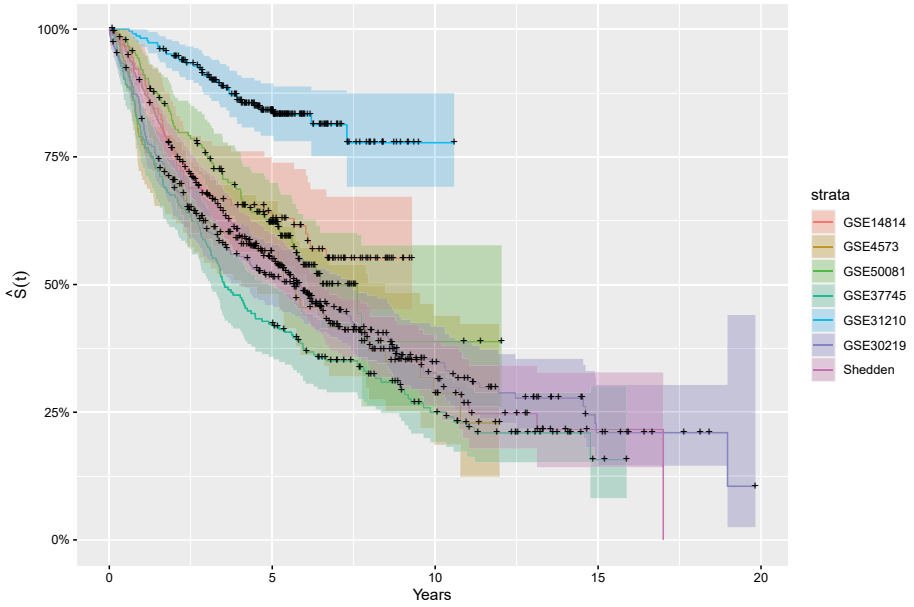



**Fig. 2.24:** Plot of Kaplan-Meier estimators including confidence bounds for the survival time $S(t)$, stratified by the cohorts that are included in [367]. In the Kaplan-Meier plots, the time $t$ is plotted against the estimated proportion of patients still alive at $t$. Lines represent survival curves of the seven cohorts. A vertical drop in a curve indicates an event, and a plus on a curve means that an observation was censored at this time.

possible and therefore applicable for practice. This has been systematically analyzed by Lang [367]. Parts of the results are summarized in Figure 2.25. Additionally, the mean ranks of filters and learners have been analyzed and revealed the following important take-home messages in the context of the datasets analyzed:

- If one base learner has to be chosen, random survival forests perform best on average.
- One of the most popular approaches due to its embedded feature selection–fitting a Cox proportional hazards model with a LASSO penalty ($L_1$)–performs the worst on average.
- Tuning over multiple base algorithms jointly with MBO results in the best performance on average.
- Tuning each pipeline individually and picking the best performing pipeline (approach BenchOpt in Figure 2.25) in a second step is not only a waste of computational resources; it also leads to overoptimistic performance estimates. As each tuning run is stochastic, and the pipelines often perform comparably well, picking the best configuration is determined by the stochastic noise to some degree. This

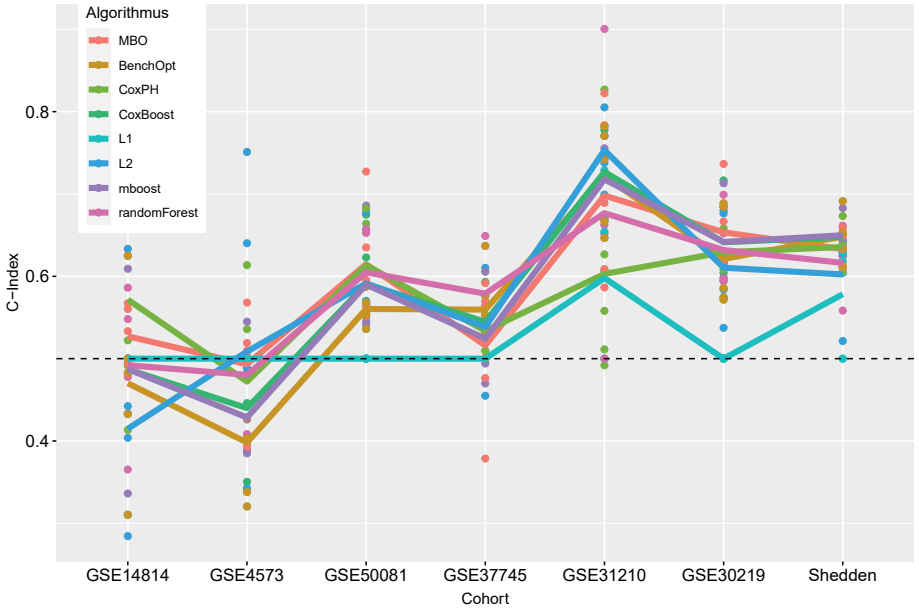



**Fig. 2.25:** Resulting average C-index on independent test sets of multiple algorithms, stratified by cohort. All base algorithms are individually tuned together, jointly with the choice of filter and the filter's hyperparameters. MBO tunes over all filters and algorithms simultaneously. BenchOpt expresses the resulting performance on an independent test set after picking the best performing base algorithm on the training data [367].

is in particular a very alarming result, as the described manual benchmarking is common practice.

The heterogeneous runtimes (or more general, the heterogeneous resource demands) have been addressed by Richter, Kotthaus, Bischl, Marwedel, Rahnenführer, and Lang [530] and Kotthaus et al. [346]. Instead of fitting only a single surrogate model, guiding the optimization to areas with the best predictive performance, multiple surrogate models are fitted in each iteration. On the one hand, the usual surrogate based on the observed predicted performance is calculated. On the other, one or more surrogate models for computational resources are fitted, e.g., one surrogate for the runtime and one surrogate describing the memory consumption. As a result, we can query the models for the estimated predictive performance and the estimated resource demands for all hyperparameter configurations. All the information is fed into a scheduler that selects a subset of the configurations and maps them to multiple CPUs or workers based on their priority (as derived from the estimated predicted performance) while minimizing the idle times (based on the estimated runtime).

### 2.5.4 Weighted Subgroup Selection for Survival Analysis

Obtaining a reliable prediction model for a specific cancer subgroup or cohort $s^\star$ is often difficult due to a limited sample size and, in survival analysis, due to potentially high censoring rates. Sometimes similar data from other patient subgroups is available, e.g., from other clinical centers. Simple pooling of all subgroups can decrease the variance of the predicted parameters of the prediction models, but also increase the bias due to heterogeneity between the cohorts.

Different approaches exist to improve the predictive quality by including data from other patient subgroups in a weighted fashion. One possible way is to include one further weighted subgroup, as proposed by Weyer and Binder [726]. Alternatively, individual weights for each patient can be estimated from the training data, as described by Bickel et al. [63]. The idea is that weights match the joint distribution of the combined data to the distribution in each subgroup, such that a patient who is likely to belong to the target subgroup receives a higher weight in the subgroup-specific model. Weights correspond to the conditional probability of belonging to the target subgroup $s^\star$ given the observed covariates and outcome divided by the prior probability for $s^\star$. The former is estimated from the training data by multi-class classification, and the latter by the relative frequency of $s^\star$.

The goal is to optimize the predictive performance of our model for the target subgroup $s^\star$. Including data from additional subgroups in the training data should increase the predictive performance of the target subgroup. Accordingly, this forms a combinatorial optimization problem where additional subgroups must be chosen to maximize the predictive performance.

However, completely abstaining from using certain subgroup data seems overly drastic since there might be relevant information contained in each additional subgroup data. Luckily, most machine learning methods and also those that can be used for time-to-event data allow observational weights. This allows us to give a low weight to observations that do not represent our problem. However, finding an optimal weight for each observation is exceedingly complex. Instead, we introduce subgroup weights as presented by Richter et al. [531]. The observation weight is then determined by the subgroup membership of each observation. This enables the inclusion of certain subgroups with a specific weight. Hence, including additional data in a weighted way might lead to a better solution than the binary choice of including a subgroup with full weight or not at all.

By introducing subgroup weights, we relaxed the combinatorial problem into a numerical optimization problem. However, setting those subgroup weights in an optimal way remains a difficult optimization problem. First, each additional subgroup leads to a further weight parameter that has to be chosen optimally. Second, the evaluation of a weight parameter combination can take fairly long, since the datasets themselves tend to be rather large, especially when they include high-dimensional genetic mea-

surements in the scenario of survival analysis. In this case, it becomes infeasible to try out many weight parameter combinations in order to find an optimal one.

Therefore, we can apply state-of-the art optimization methods for expensive black-box problems such as MBO (model-based optimization) in order to find the optimal subgroup weights without the cost of having to evaluate hundreds of different weight parameter combinations. For our evaluation, we optimize the subgroup-specific weights $w^{(g)}$ in the weighted Cox model. Note that in Section 2.5.3 a study was reported where random survival forests performed on average better than fitting a Cox proportional hazards model with a LASSO penalty. However, here we use the much more frequently used penalized regression approach to evaluate the potential improvement due to the weighted analysis.

**Weighted Cox Model** Assume the observed data of the patient $i$ consists of the tuples $(t_i, \delta_i)$, the covariate vectors $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^{'} \in \mathbb{R}^p$, and the subgroup membership $s_i \in \{1, \ldots, S\}$ with $S$ being the total number of available subgroups, and $i = 1, \ldots, n$. $t_i$ denoting the observed time of patient $i$, the minimum of the event time, and the censoring time. $\delta_i$ indicates whether a patient experienced an event ($\delta_i = 1$) or was (right-)censored ($\delta_i = 0$). As mentioned above, one of the most popular regression models in survival analysis is the Cox proportional hazards model [149]. It models the hazard rate $h(t|\boldsymbol{x}_i)$ of a patient at time $t$ as

$$h(t|\boldsymbol{x}_i) = h_0(t) \cdot \exp(\boldsymbol{\beta}^{'} \boldsymbol{x}_i) = h_0(t) \cdot \exp\left(\sum_{j=1}^{p} \beta_j x_{ij}\right),$$

where $h_0(t)$ is the baseline hazard rate, and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^{'}$ is the unknown parameter vector. The parameters are estimated by maximizing the partial log-likelihood [326, Chapter 8.3]. A version of the partial log-likelihood that uses observation weights is introduced in [726]:

$$l(\boldsymbol{\beta}) = \sum_{i=1}^{n} \delta_i w_i \left( \boldsymbol{\beta}^{'} \boldsymbol{x}_i - \ln \left[ \sum_{k=1}^{n} \mathbf{1}_{(t_i \leq t_k)} w_k \exp\left(\boldsymbol{\beta}^{'} \boldsymbol{x}_k\right) \right] \right). \tag{2.18}$$

Instead of an individual weight for each patient, we introduce an individual weight for each subgroup. Therefore, we assign the same weight to each patient of the same subgroup:

$$w_i = \begin{cases} 1, & \text{if } s_i = s^{\star} \\ w^{(g)}, & \text{if } s_i = g,\ g \in \{1, \ldots, S\} \setminus s^{\star} \end{cases} \tag{2.19}$$

where $w^{(g)} \in [0, 1]$ is the specific weight for the subgroup $g$, and $s^{\star}$ is the subgroup for which we want to obtain predictions. Patients for subgroup $s^{\star}$ enter with full weight 1 in the prediction model.

Standard subgroup analysis is based only on the patients in the subgroup of interest (target subgroup $s^\star$), which corresponds to $w_i = 0$ for all patients not belonging to $s^\star$. A combined model that pools patients from all subgroups corresponds to $w_i = 1$ for all patients.

In high-dimensional settings where the number of covariates $p$ is typically much larger than the sample size $n$, standard maximum likelihood cannot be used for parameter estimation, since it does not result in a unique solution. Therefore, we add a LASSO penalty [677] to the partial log-likelihood. Lasso regression performs feature selection and yields a sparse model solution. The resulting maximization problem of the penalized partial log-likelihood is given by

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}} \left\{ l(\boldsymbol{\beta}) - \lambda \cdot \sum_{j=1}^{p} |\beta_j| \right\}.$$

The LASSO penalization parameter $\lambda$ is optimized through an internal 10-fold cross-validation.

**Evaluation** We are interested in maximizing the predictive performance for a target subgroup $s^\star$. The predictive performance of the weighted Cox model is evaluated using the C-index. For the evaluation of the model for a given target subgroup $s^\star$, a dataset that contains $S$ subgroups, and a subgroup weight vector $w = (w^{(1)}, \ldots, w^{(S-1)})$, we conduct a modified 10-fold cross-validation. The validation data should only contain the target subgroup, because we are only interested in the predictive performance on the target subgroup. In order to obtain the 10 necessary splits for the cross-validation, we only divide the data of the target subgroup into 10 chunks. To obtain the prediction for one chunk, all remaining 9 chunks plus all observations from the additional subgroups are combined to the training dataset. By performing the modified cross-validation, we obtain an estimation on the C-index for the given combination of dataset, target subgroup and subgroup weight vector.

Now, the goal is to find the subgroup weight vector that maximizes the C-index. This optimization problem can be solved with MBO, with a search space $[0, 1]^{S-1}$ that directly maps to the weight vector. The acquisition function that selects the next weight vector to be evaluated should take into consideration that results are not deterministic. Therefore, we proposed the augmented expected improvement [288], which is well suited for such scenarios. For the Gaussian process regression within the Bayesian optimization, we proposed the Matern 3/2 kernel with an estimated *nugget effect* to account for the noisy response of our objective.

The benefit of optimizing the subgroup weights is twofold: First, the resulting optimal subgroup weight vector does not only maximize the C-index for the target subgroup; it also allows drawing conclusions about the similarity of certain subgroups. If a certain subgroup weight is small, it can be assumed that this subgroup does not have a similar relationship between the explanatory variables and the event times as

the target subgroup. Second, as shown by Richter et al. [531], the predictive performance of the method does not deteriorate if additional subgroups are included that contain inconsistent data.

Benefits could arise from using the penalization of the weights, which would allow researchers to completely exclude data with weights close to zero. Then the model becomes computationally cheaper and possibly more stable. Finally, the presented approach can be used for any situation where data is pooled from different cohorts and a machine learning method is used that supports observational weights.

### 2.5.5 Feature Selection for High Dimensional Data

The problem of feature selection is particularly important in the domain of high dimensional data, as already described in detail in Section 2.5.3. One challenging problem in this context is the stability of feature selection. Some learners can be restricted to using only a small subset of the thousands of available features, and learners can be combined with a feature filter to achieve the same in a generic fashion. However, the set of selected features is often highly variable. For example, if a Cox proportional hazard model is extended with an $L_1$ penalty $\lambda$ tuned to include only up to 20 features in a 3-fold cross-validation, the resulting three sets of selected features can be pairwise disjoint. This has a simple reason: if two features $x_1$ and $x_2$ are highly correlated, they are also comparably good predictors. If the model has to choose between $x_1$ and $x_2$, a few observations can tip the scales in one direction or the other. If the dataset is now resampled and these observations are removed from the training set, the scale can easily swing in the other direction. This is particularly annoying because in this way no features can be reliably selected for a later analysis, such as a biological analysis.

Lee et al. [383] tackle this problem in two ways: First, a special extension to the LASSO regression is used. The preconditioned LASSO [495] is a two-step procedure designed to address the problems of high bias in LASSO estimates. Second, the preconditioned LASSO is embedded in a two-fold subsampling procedure to improve the stability of the feature selection via model averaging and extra shrinking of covariates based on the selection probability in the inner subsampling.

The approach has been applied to datasets on neuroblastoma, lung adenocarcinoma, and breast cancer. Both predictive performance (measured by the C-index) and stability (measured by the Jaccard index and the Kuncheva index) are improved. However, the comparison with popular univariate selection methods does not provide a clear picture.

Another take on this topic was presented by Bommert et al. [95], where more than 20 filter methods are benchmarked against each other for high-dimensional data. Although this work is based on classification data, there is no reason why the core results should not be transferable to survival problems, and confirming this is currently work in progress. One key result is shown in Figure 2.26. There are clear groups of

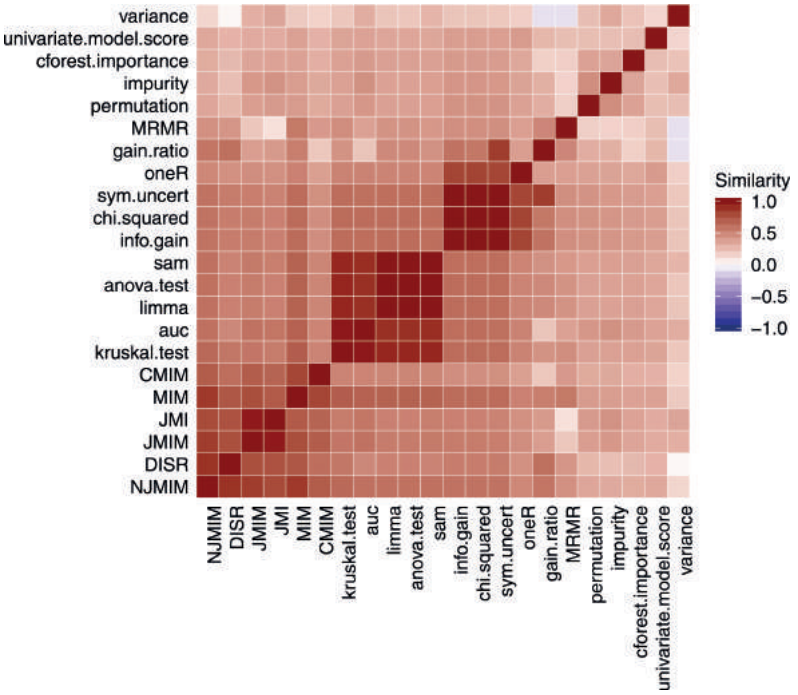



**Fig. 2.26:** Rank correlations between the feature selection order, for all pairs of a large set of filter methods, averaged across several datasets by the arithmetic mean. The filter methods are ordered by average linkage hierarchical clustering using the mean rank correlation as a similarity measure [95].

feature filters available. Filters from the same group are expected to give very similar results across different datasets. Therefore, instead of including more than 20 filters into the machine learning pipeline, it is sufficient to thin out this portfolio to a smaller set. Additionally, the filters have been analyzed regarding performance and stability to provide general recommendations for feature filtering in high-dimensional settings.

### 2.5.6 Software

Many machine learning frameworks exist that can be conveniently employed for model selection or feature selection. However, most of these frameworks have a strong focus on classification and regression. Support for survival analysis is often not existent or insufficient. For proper evaluation, two frameworks have been extended to support time-to-event data.

First, the R package `mlr` [69] has been extended with an object for survival tasks, including the most common survival learners and survival measures. By building upon

the existing infrastructure for resampling and tuning, survival learners can be tuned with state-of-the-art tuners such as model-based optimization. For larger tasks, i.e. tasks with thousands of features of genetic data, parallelization of the benchmark experiments is mandatory. The package `BatchJobs` [67] and its successor `batchtools` [368] provide the bridge between `mlr` and managed high-performance computing clusters, allowing to compute comprehensive benchmarks on hundreds of CPUs simultaneously. In this way they can define and execute exhaustive benchmark studies, such as those from Lang, Kotthaus, Marwedel, Weihs, Rahnenführer, and Bischl [369], Lang [367] and Richter et al. [531].

The second framework extended for survival analysis is `mlr3` [370], the successor of `mlr`. The extension package `mlr3proba` [640] provides a general framework for probabilistic regression. Compared with the survival capabilities of `mlr`, `mlr3proba` connects much more learners and, even more importantly, connects and implements much more survival measures. Additionally, `mlr3proba` can be embedded in the infrastructure of the `mlr3pipelines` [65] package, which provides a language to build complex machine learning workflows as directed acyclic graphs. `mlr3pipelines` is also used to convert and unify the many predict types of survival models: while some models return a linear prediction vector, others return a continuous ranking, relative risks, or a complete time-dependent distribution such as individual survival function estimates. `mlr3proba` provides several pipeline operators for converting between predict types or even for composing multiple types.

Thanks to the unified interface of `mlr3` and `mlr3proba`, it is directly possible to use state-of-the-art methods to optimize the hyperparameters of the survival methods via `mlr3tuning`. Especially in the survival context, data preprocessing is often a crucial step. Decisions on how to configure the preprocessing should be included in this optimization process to obtain an unbiased estimate of the predictive performance. Modeling preprocessing through `mlr3pipelines` allows the building of a whole pipeline that can be resampled and optimized. To obtain an unbiased estimate of the performance of a pipeline identified through optimization, the whole optimization can be resampled, resulting in a nested resampling setting. Multi-criteria optimization is also supported, e.g. to tune for predictive performance, sparsity, and feature selection stability simultaneously by connecting the `stabm` [94] package.

### 2.5.7 Conclusion

The analysis of survival data requires the use of adequate statistical methodology, especially when it comes to accounting for missing information due to censoring. Corresponding methods are available and established. However, for modern high-dimensional data increasingly being generated today, omics data in particular, additional challenges emerge. Estimating prediction models often requires elaborate feature selection and hyperparameter optimization. For this task, Bayesian optimization meth-

ods provide a beneficial solution. They can efficiently identify models with competitive prediction accuracy out of a large set of candidate models. Of great importance is the availability and use of software frameworks for reproducible analysis pipelines. One valuable example is the widely used R package `mlr3`, which provides efficient, object-oriented programming on the building blocks of machine learning, together with its extension package `mlr3proba`, which provides a general framework for probabilistic regression, including many popular survival models and survival measures.

## 2.6 Protein Complex Similarity

*Bianca K Stöcker*
*Till Schäfer*
*Petra Mutzel*
*Johannes Köster*
*Nils Kriege*
*Sven Rahmann*

**Abstract:** Proteins have manifold functions in living cells, including structural integrity, transport, defense against pathogens, or message transmission, to name but a few. Recent advances in Machine Learning appear to have solved the protein folding problem, i.e., how to obtain the three-dimensional functional protein structure from the amino acid sequence of the protein. However, proteins rarely act alone, but instead perform their functions together with other proteins in so-called *protein complexes*. Quantifying the similarity between two protein complexes is essential for numerous applications, e.g., for database searches of complexes that are similar to a given input complex. While similarity measures have been extensively studied on single proteins and on protein families, there is little work on modeling and computing the similarity between protein complexes yet. Because protein complexes can be naturally modeled as graphs, graph similarity measures may be used, but these are often computationally hard to obtain and do not take typical properties of protein complexes into account. We introduce a parametric family of similarity measures based on Weisfeiler-Leman labeling see "The Weisfeiler-Leman Algorithm for Machine Learning with Graphs" in Section 4.2 in Volume 1. Based on simulated complexes, we show that the defined family of similarity measures is in good agreement with edit similarity, a similarity measure derived from graph edit distance, though it can be computed more efficiently. Moreover, in contrast to graph edit similarity, the proposed measures allow for an efficient similarity search in large volumes of protein complex data. It can therefore be used as a basis for large-scale machine learning applications.

### 2.6.1 Introduction

Proteins fulfill manifold tasks in living cells, but they rarely act alone. Indeed, most cellular functions are enabled only when proteins physically interact with other proteins, forming protein complexes. DNA transcription is a typical example, where RNA polymerase II, general transcription factors, cell type specific transcription regulators, and mediator proteins interact.

Understanding protein complex formation and function is one of the big challenges of cell biology, and is approached by both experimental techniques and computational modeling. While the constituent protein sequences can be obtained from the genome, the computational prediction of real protein complexes from protein interaction networks appears to be much more difficult [61, 643], and we presently face a lack of experimental datasets on verified real protein complexes. Fortunately, new experimental technologies such as high-resolution protein-protein docking are about to enhance our understanding of complexes significantly in the near future [348, 490].

When studying biological entities such as protein sequences or protein complexes, a fundamental task is to define a measure of similarity between two such entities. For protein sequences, there is a well-established theory based on scoring matrices and alignment scores [496]. For protein complexes, it appears that no systematic effort to quantify similarity has been made yet. The purpose of the present article is therefore to discuss the different options for defining a similarity measure on protein complexes and for proposing a reasonable and computationally tractable definition of protein complex similarity. Establishing a similarity measure is not only important fundamentally, but there are many immediate applications.

**Database search** In the *database search problem* we are given a query complex and a large collection (database) of complexes, and the task is to find the complexes in the database whose similarity to the query exceeds a given threshold.

**Comparing predictions** Several complex prediction methods predict putative complexes by locating dense regions in a protein interaction network [180, 272, 424, 497], and for comparing complexes predicted by different algorithms, it is of interest to compute a maximum-weight matching between the output of two algorithms, where the weighting is given by a similarity function.

**Summarizing and clustering** When simulating complex formation based on available knowledge such as possible interactions and interaction constraints, it is helpful to aggregate the simulation output to focus on frequently seen or typical complexes, ignoring small differences. Aggregation or clustering by similarity thereby reduces data size and complexity. Such a task requires quantifying the similarity between two protein complexes.

When there are tens of thousands of different complexes subject to pairwise comparison, a similarity measure must be efficiently computable.

This contribution is based on a conference paper by the authors [644], adapted by permission from Springer Nature; 

#### 2.6.1.1 Models for Protein Complexes

We first discuss models for protein complexes at different levels of detail, namely the *set*, *multiset*, and *graph* models.

While intuition suggests that protein complexes can be naturally described as graphs with proteins as vertices and physical interactions as edges, there are in fact different ways to formally describe a protein complex. We start with a given set $P$ of all proteins of an organism, the building blocks of the complexes.

**Set** In its most simple form, a protein complex can be defined as a set (in the mathematical sense, i.e., without multiplicities) of proteins, i.e., as a subset $\{p_1, p_2, \ldots, p_n\}$ of $P$. Sets neither capture the multiplicities nor the nature of the physical interactions between the constituent proteins of a complex. Some experimental techniques only give set-type information, and several existing databases only provide this type of information, e.g., the CORUM database [551].

**Multiset** Formally, a multiset is a function $C : P \to \mathbb{N}_0$ that assigns a multiplicity to each protein $p \in P$ with $C(p) = 0$ for proteins $p$ that are not part of the complex. We also use the multiset notation $C = \{\!\{p_1, p_1, p_2\}\!\}$ to express that $C(p_1) = 2$, $C(p_2) = 1$ and $C(p) = 0$ for all other $p \in P$. Defining a protein complex as a multiset of proteins gives a more accurate representation of the complex, but still does not consider the interaction topology.

**Graph** To add more information, we can define a protein complex as an undirected graph $C = (V, E, \ell)$ with labeled vertices $V$, such that each vertex $v \in V$ represents a protein and hence has a label $\ell(v) \in P$, each edge $e \in E \subseteq V \times V$ represents a physical interaction between the corresponding proteins, such that $E$ is symmetric and $C$ is connected. The graph description provides the interaction topology. We call this representation a *protein complex graph* and define its *size* as $|C| := |V| + |E|$. In the following, we use the terms protein complex and protein complex graph interchangeably.

A remark is in order to avoid confusion with *protein interaction networks*. Our definition of the graph model of protein complexes is formally identical to the definition of protein interaction network. However, there are important differences. The protein complex graph represents an assembly composed of multiple proteins that physically bind and co-exist in one temporal and physical space, while a protein interaction network represents proteins that may interact at some point of time, where individual interaction time points may be different. Hence, the protein complex graphs considered in this work typically consist of only a few vertices and edges, while interaction networks are much larger.

For the set and multiset models, a similarity measure is readily given by the *Jaccard similarity* (see Section 2.6.2.1). For graphs, various techniques exist such as graph kernels [353] or graph matching [741]. A particularly intuitive approach is the *graph edit distance*, which has been proposed for pattern recognition tasks more than 30 years ago [574]. A graph edit distance between graphs $C$ and $C'$ measures the total costs of the edit operations required to transform $C$ into $C'$. Defining similarity via graph edit operations appears natural, but has computational disadvantages, as the graph edit distance generalizes the classical maximum common subgraph problem [109], which

is NP-complete [220] and hard to approximate with given guarantees [311]. Therefore, a large number of heuristics for computing the graph edit distance without provable guarantees have been proposed. A particular successful class of heuristics derives the edit costs from the solution of a linear sum assignment problem [90, 352, 535]. Recently several elaborate exact algorithms for computing the graph edit distance have been proposed, but are still limited to small graphs [132, 239, 388]. The binary linear programming formulation of [388] allows researchers to compare graphs of moderate size using highly-optimized general purpose solvers. However, when we want to compare many complexes, evaluating the edit distance between all pairs becomes infeasible in practice.

We therefore propose an efficient alternative. We define a family of similarity measures on graphs using the Jaccard similarity, which can be efficiently computed and even more efficiently estimated with locality-sensitive hashing techniques. Taking the graph structure into account is achieved by the Weisfeiler-Leman labeling of the vertices [724], propagating vertex labels between neighbors. This approach is different from recent work that approximates and bounds the graph edit distance [534] and has the advantage of scaling better to large-scale studies.

We find that the Weisfeiler-Leman (WL) similarity approximates edit similarity well, but can be computed much more efficiently. In addition, in large-scale database searches for complexes similar to a given query complex, we obtain an additional speed-up by an order of magnitude when filtering for high WL similarity using min-hashing. Finally, we discuss limitations and possible extensions of this work.

### 2.6.2 Methods

Our goal is to define a similarity measure between protein complexes that captures not only the (multisets of the) constituent proteins, but also the interaction topology (graph structure). We introduce a parameterized family of similarity measures on protein complexes, which are based on multiset comparisons of vertex labels in the graph representation and take the local neighborhood of each protein into account by using Weisfeiler-Leman labels.

#### 2.6.2.1 Jaccard Similarity of Sets and Multisets

To compare sets or multisets, Jaccard similarity coefficients are an established quantity.

Let $M \subseteq U$ and $M' \subseteq U$ be two subsets of a common universe $U$. Then the *Jaccard similarity* between $M$ and $M'$ is defined as

$$J_{\text{set}}(M, M') := \frac{|M \cap M'|}{|M \cup M'|} \in [0, 1]\,. \tag{2.20}$$

This definition is extended to multisets as follows. Recall that multisets $M$ and $M'$ are functions $U \to \mathbb{N}_0$, assigning multiplicities $M(o)$ and $M'(o)$ to each object $o \in U$. (The

set definition can be seen as the special case where the value set is only $\{0, 1\}$ instead of $\mathbb{N}_0$.) Then the *Jaccard similarity* between $M$ and $M'$ is defined as

$$J_{\text{multiset}}(M, M') := \frac{\sum_{o \in U} \min\{M(o), M'(o)\}}{\sum_{o \in U} \max\{M(o), M'(o)\}} \in [0, 1]\,. \tag{2.21}$$

The definition of $J_{\text{multiset}}$ can be reduced to that of $J_{\text{set}}$ by augmenting the element names with a running index in each multiset. For example,

$$J_{\text{multiset}}(\{\!\{A, A, B, C, C\}\!\}, \{\!\{A, C, C, C\}\!\}) = J_{\text{set}}(\{A_1, A_2, B_1, C_1, C_2\}, \{A_1, C_1, C_2, C_3\})\,.$$

Using this transformation, sketching techniques like min-hashing [105] that primarily work on sets can be extended to multisets.

#### 2.6.2.2 A Parametric Family of Protein Complex Similarity Measures

Instead of comparing (protein complex) graphs directly by their labels and graph topology, we extract and compare multisets of derived features that represent local neighborhood information. Encoding the local structure surrounding a vertex is a general method widely used in graph matching and machine learning with graphs. Various concepts and techniques for this have been proposed, e.g., the $k$-hop neighborhood [319, 459], or the $k$-sphere neighborhood [37]. Weisfeiler and Leman developed an iterative label refinement procedure to derive a canonical graph representation for graph isomorphism testing [724]. The same procedure is often used to define graph similarities or graph kernels [602]. This approach recently became popular in machine learning for its expressivity and favorable algorithmic properties. Several highly efficient graph kernels based on Weisfeiler-Leman refinement have been proposed [351, 602], and several graph neural networks were shown to be at most as expressive as the Weisfeiler-Leman method [737].

The technique works as follows. Initially, the feature multiset of a graph consists of the union of all vertex labels, i.e., the protein names. After the initialization, the vertex labels are iteratively augmented by the labels of the neighboring vertices from the previous iteration, thereby encoding the (local) graph structure in the vertex labels. As previously mentioned, we label the vertices of graphs with the protein names; so two vertices that refer to the same protein type are labeled identically. Thus, the initial feature multiset is identical to the multiset representation as described above. Let us now formally define the process.

**Definition 1** (Weisfeiler-Leman labeling)**.** Let $C = (V, E, \ell_0)$ be a graph with label function $\ell_0 : V \to L_0 := P$, where $P$ is the initial set of vertex labels (e.g., protein names for protein complex graphs). Furthermore, let $N(v) := \{u \mid \{v, u\} \in E\}$ denote the neighbors of vertex $v \in V$. Then, the Weisfeiler-Leman labeling of iteration $i$ is defined as a re-labeling of the graph. It replaces the current labeling function $\ell_{i-1} : V \to L_{i-1}$ with a new labeling function $\ell_i : V \to L_i$. The value of $\ell_i$ for a vertex $v \in V$ is recursively

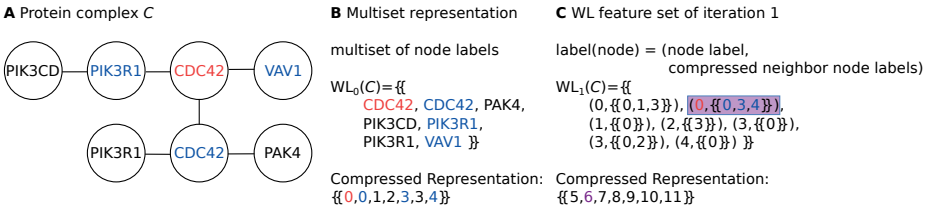



**Fig. 2.27: Example of a protein complex and its representations.** The colors highlight the labels of an example node in $\mathrm{WL}_0(C)$ and $\mathrm{WL}_1(C)$. **A:** Graph representation of protein complex $C$. **B:** Multiset representation of $C$ which is equal to $\mathrm{WL}_0(C)$. **C:** Result of the first WL iteration.

defined as

$$\ell_i(v) := (\,\ell_{i-1}(v),\ \{\!\{\ell_{i-1}(u) \mid u \in N(v)\}\!\}\,). \tag{2.22}$$

Note that the second component of the new label is a multiset.

To avoid increasingly complex labels consisting of nested multisets, label compression is performed after each step. This is achieved by applying an injective function, which maps a pair consisting of a label and a multiset of labels of the form given in Equation 2.22 to an integer label. The label compression step must be consistent across multiple graphs in order to construct comparable feature sets. If all graphs in the dataset are known from the beginning, we can sort all the multisets of one iteration, identify the identical pairs, and assign them to the same new integer label. This step can be realized in time linear in the total number of edges of all graphs by applying variants of bucket sort [602]. A less efficient, but more flexible approach, which is suitable even when the graphs are only revealed successively, is to manage the injective map used for label compression in a hash table.

Given the Weisfeiler-Leman labeling function for any iteration $i$, we can now define the multiset of Weisfeiler-Leman features for iteration $i$.

**Definition 2** (Weisfeiler-Leman feature set)**.** Let $C = (V, E, \ell_0)$ be a graph with label function $\ell_0 : V \to L_0 = P$, where $P$ is the initial set of vertex labels. Then, the Weisfeiler-Leman features of iteration $i$ are defined as multiset $\mathrm{WL}_i(C) = \{\!\{\ell_i(v) \mid v \in V\}\!\}$.

Note that $\mathrm{WL}_0(C)$ always corresponds to the initial multiset of labels (protein names). Accordingly, $\mathrm{WL}_1(C)$ integrates the neighborhood labels of each node. Figure 2.27 shows an example protein complex graph, together with the associated feature sets $\mathrm{WL}_0(C)$ and $\mathrm{WL}_1(C)$. A node and its neighborhood are highlighted in red and blue to demonstrate the relation between $\mathrm{WL}_0(C)$ and $\mathrm{WL}_1(C)$.

We use the Jaccard coefficient to obtain a normalized similarity based on multiset intersection. We apply the Jaccard coefficient to the feature sets of each iteration individually and compute a convex combination of the results. Let $w = (w_i)_{i\geq 0}$ be a sequence of non-negative weights with $\sum_{i\geq 0} w_i = 1$. We quantify the weighted similarity between

two labeled graphs $C$ and $C'$ by

$$S_w(C, C') := \sum_{i \geq 0} w_i \cdot J_{\text{multiset}}(\text{WL}_i(C), \text{WL}_i(C')), \tag{2.23}$$

where $J_{\text{multiset}}$ is given by Equation 2.21. This defines a family of similarity measures between labeled graphs with values in $[0, 1]$, parameterized by the weight vector $w = (w_0, w_1, \dots)$.

It is easy to see that, as long as $w_0 > 0$, we have $S_w(C, C') = 0$ if and only if the label sets of $C$ and $C'$ are disjoint. If $S_w(C, C') < 1$, the graphs are not isomorphic. However, $S_w(C, C') = 1$ does not necessarily imply that $C$ and $C'$ are isomorphic even if $w_i > 0$ for all $i$. There exist examples of non-isomorphic graphs $G$, $G'$ with $\text{WL}_i(G) = \text{WL}_i(G')$ for all $i \geq 0$. (As a simple example, take $G$ to be a cycle of six vertices, and $G'$ to be two cycles of three vertices, all with the same label.) On the other hand, there exist classes of graphs, such as the CR-graphs, for which the implication "$S_w(C, C') = 1 \Rightarrow C, C'$ are isomorphic" is true if $w_i > 0$ for all $i$ [31]. Moreover, the implication holds with high probability for random graphs (without vertex labels) even when $w_i = 0$ for all $i \geq 3$ [34].

In our application scenario, we may assume that most protein complexes are non-adversarial graphs with sufficiently simple structure and expressive initial labels such that their Weisfeiler-Leman features are appropriate to characterize their similarity. In fact, we put forward the hypothesis that using a single iteration is frequently sufficient for practical purposes, and we set $w_i := 0$ for $i \geq 2$ in our computational experiments (see Results) and only have a single free parameter $w_1 \in [0, 1]$ that defines $w_0 := 1 - w_1$. In this case, $S_{w_1}$ is efficiently computable: a proof of the following lemma can be found in the work of [602].

**Lemma 3.** *For $w_1 \in [0, 1]$, each of the one-parameter similarity measures*

$$\begin{aligned} S_{w_1}(C, C') := (1 - w_1) \cdot J_{multiset}(WL_0(C), WL_0(C')) \\ + w_1 \cdot J_{multiset}(WL_1(C), WL_1(C')) \end{aligned}$$

*can be computed in $O(|C| + |C'|)$ time, where $|C| = |V| + |E|$.*

#### 2.6.2.3 A Similarity Measure Based on Graph Edit Distance

To compare the family of Weisfeiler-Leman multiset-based similarity measures defined above with graph edit distance, we state a formal definition of the edit-based similarity. We allow the following elementary operations to edit a graph: vertex deletion, vertex insertion, vertex relabeling, edge deletion, and edge insertion. A sequence $(o_1, \dots, o_k)$ of such edit operations that transforms a graph $G$ into another graph $H$ is called an *edit path* from $G$ to $H$. Each operation $o$ is assigned a cost $c(o)$, which is zero for substituting vertices and edges with the same label. We use a cost of 1 for all operations except vertex relabeling, which has a cost of 2, corresponding to one deletion and one insertion (leaving the edges in place). Note that deleting or inserting a vertex of degree $k$ otherwise

has cost $k+1$ for deleting $k$ edges and the vertex itself. We denote the set of all possible edit paths from $G$ to $H$ by $Y(G,H)$.

**Definition 4.** Let $G$ and $H$ be labeled graphs. The *graph edit distance* from $G$ to $H$ is defined by

$$d(G,H) = \min\left\{\sum_{i=1}^{k} c(o_i) \mid (o_1,\ldots,o_k) \in Y(G,H)\right\}. \tag{2.24}$$

Intuitively, the graph edit distance preserves a subgraph $G'$ of $G$ that is also contained in $H$ using zero-cost substitutions, deletes the vertices and edges in $G$ that are not in $G'$, and then inserts vertices and edges to obtain an isomorphic copy of $H$. Therefore all non-zero costs can be attributed to the elements that are in one of the graphs, but not in their common subgraph. In this sense the graph edit distance is similar to the symmetric difference of two sets. This observation motivates the following normalized similarity measure derived from the graph edit distance. We define the *graph edit similarity* as

$$S_g(G,H) := \frac{|G|+|H|-d(G,H)}{|G|+|H|+d(G,H)} \in [0,1], \tag{2.25}$$

where $|G| := |V(G)| + |E(G)|$. Note that the graph edit distance between $G$ and $H$ is at most $|G|+|H|$, which is achieved by deleting all vertices and edges of $G$ and inserting all vertices and edges of $H$. In this case the graph edit similarity is zero. Similarly, $S_g(G,H) = 1$ if and only if $d(G,H) = 0$. In this respect the similarity measure resembles the Jaccard similarity. In fact, the following lemma shows that, if the edges are not taken into account, the graph edit similarity equals the multiset Jaccard similarity. Therefore, the graph edit similarity can indeed be seen as a natural extension of the multiset Jaccard similarity to graph structured data.

**Lemma 5.** *For two vertex-labeled graphs $G$, $H$, let $C$, $D$ denote their respective label multiset. For the edge-free graphs $G' = (V(G),\emptyset)$ and $H' = (V(H),\emptyset)$ it holds that $S_g(G',H') = J_{multiset}(C,D)$.*

*Proof.* An optimal graph edit path is obtained as follows: We substitute the vertices with common labels free of cost, which are $Z = \sum_{p\in P} \min\{C(p), D(p)\}$ in total. We delete the remaining $|G'| - Z$ vertices in $G'$ and insert $|H'| - Z$ vertices to obtain an isomorphic copy of $H'$ at a total cost of $|G'| + |H'| - 2Z = d(G,H)$. Instead we may also substitute up to $||G'| - |H'||$ vertices, each at cost two, which results in the same total cost. Using the fact that $|G'| = \sum_{p\in P} C(p)$ and $|H'| = \sum_{p\in P} D(p)$, we obtain the result

by calculating

$$S_{\mathrm{g}}(G', H') = \frac{|G'| + |H'| - d(G', H')}{|G'| + |H'| + d(G', H')} = \frac{Z}{|G'| + |H'| - Z}$$

$$= \frac{Z}{\sum_{p \in P} C(p) + \sum_{p \in P} D(p) - Z}$$

$$= \frac{\sum_{p \in P} \min\{C(p), D(p)\}}{\sum_{p \in P} C(p) + D(p) - \min\{C(p), D(p)\}}$$

$$= \frac{\sum_{p \in P} \min\{C(p), D(p)\}}{\sum_{p \in P} \max\{C(p), D(p)\}} = J_{\mathrm{multiset}}(C, D)\,.$$

□

#### 2.6.2.4 Database Searches

An important application of a similarity measure $S(\cdot, \cdot)$ is for similarity searches in large databases. When searching for similar protein complexes to a given input complex ("query" $Q$) in a large database, one can perform a linear scan and compute $S(Q, C)$ for each complex $C$ in the database and report those with $S(Q, C) \geq T$ for a given threshold $T$. However, computing $S(Q, C)$ exactly may be computationally expensive, and in many cases, a quickly computable upper or lower bound can be used as an initial filter.

In Section 2.6.3, we evaluate the proposed similarity measure $S_{w_1}$ against graph edit similarity $S_{\mathrm{g}}$, and for database searches we use individual filtering techniques as described in this section.

**Weisfeiler-Leman Similarity** For $S_w$, a speed-up is possible using min-hashing [105], which is a locality-sensitive hashing scheme for the Jaccard similarity between sets that can be extended to multisets, as described in Section 2.6.2.1. We use a simple approach that maps the $\mathrm{WL}_i(C)$ multisets to integer hash values $h_{i,j}(C)$ using a large number ($j = 1, \ldots, K$) of different random hash functions. The exact $S_w(Q, C)$ value is only computed if any of the hash values agrees with that of the query, i.e., if $h_{i,j}(C) = h_{i,j}(Q)$ for any $i = 0, 1$ and $j = 1, \ldots, K$. The number $K$ of hash functions is chosen such that the false negative error rate is lower than a given probability threshold (0.01).

**Graph Edit Similarity** The graph edit *distance* is widely used for searching graph databases and several approaches tailored to this task have been proposed [132, 323, 397, 712, 755, 768, 769, 770]. These methods typically follow a *filter-verification* approach [755]. In the filter phase, efficiently computable lower bounds on the graph edit distance are used to eliminate dissimilar graphs. The remaining graphs are then verified using upper bounds on the graph edit distance and, if necessary, by an exact algorithm to obtain the final result. The methods can be categorized according to different criteria.

Several methods compute a lower bound for every graph in the database [132, 323, 755], while the others use an index data structure to find candidates without scanning the whole database. The lower bounds are often derived from overlapping substructures, while some methods partition the graphs into disjoint parts [323, 397, 769]. Recently, the large number of proposed lower and upper bounds were systematically studied in an extensive experimental evaluation according to their running time and tightness [90].

Most of the above mentioned algorithms assume a uniform edit cost function and cannot directly be applied to solve database search problems with respect to the graph edit similarity as defined in Equation 2.25: To make graph similarity compatible with the Jaccard index, we use a cost of 2 instead of 1 for vertex relabeling. Nevertheless, several of the efficiently computable bounds for uniform graph edit distance can be generalized for this case, and we have implemented such generalizations (see below). Then, by substituting $d(G, H)$ in Equation 2.25 by known lower (upper) bounds on the graph edit distance, we obtain upper (lower) bounds for the graph edit similarity.

Since the vertex labels denoting proteins are highly specific when comparing protein complexes, we derive a first lower bound on the graph edit distance from the cardinality of the symmetric difference of the two vertex label multisets [323, 768]. This is equivalent to using $J_{\text{multiset}}$ as an upper bound for the graph edit similarity.

As a second lower bound we use a more expensive approach based on the linear sum assignment problem, which was shown to provide a good trade-off between tightness and running time [90]. The assignment instance is defined on the vertices of the two graphs, where additional dummy vertices are introduced that represent vertex insertion and deletion. The costs for assigning individual vertices is made up of the costs for substituting the vertex label and the cost of an optimal assignment between the incident edges [535]. Since we do not consider edge labels, the assignment cost matrix can be computed in quadratic time (cf. heuristic `BRANCH-CONST` in [90]) and the instance can be solved in cubic time. The cost of the assignment instance serves as a lower bound on the graph edit distance. Following [535], we obtain an upper bound from the cost of the edit path derived from the assignment. If this is not sufficient for verification, we compute the exact graph edit distance using the binary linear programming formulation of [388].

### 2.6.3 Results

#### 2.6.3.1 Hypothesis

We hypothesize that finite truncations of the Weisfeiler-Leman-based family of similarity measures $S_w$ (defined in Equation 2.23), in particular $S_{w_1}$ (Lemma 3), are in good agreement with the edit similarity (defined in Equation 2.25) for typical protein complex graphs. Especially $S_{w_1}$ has the advantage that it can be efficiently computed.

#### 2.6.3.2 Experimental Setup

We have implemented the similarity measures based on Weisfeiler-Leman labeling and the graph edit similarity in Java 8. To compute the exact graph edit distance, we used a recent binary linear programming formulation [388] and solved the instances using Gurobi 7.5.2. All experiments were run on 18-core Intel Xeon E5-2699 CPUs with 2.3 GHz and 512 GB RAM, using 64-bit Ubuntu Linux 18.04. Our analysis is available as a Snakemake workflow [345] on github (https://github.com/BiancaStoecker/complex-similarity-evaluation).

#### 2.6.3.3 Data Generation

As mentioned in the introduction (Section 2.6.1), obtaining graphs from real protein complexes is difficult at the moment, because experimental techniques that resolve the (graph) topology of the complexes are still in the developmental stage. Therefore we resort to the simulation of complexes, based on two types of knowledge: possible physical protein-protein interactions, formalized by a *protein interaction network*, and *constraints between protein interactions*. Especially the second type of information allows us to simulate more realistic complexes than what we would get from interaction networks alone.

Formally, a protein interaction network is an undirected graph $N = (P, I)$, where $P$ is the set of protein types of a cell (or an organism), and $I \subset P \times P$ indicates the pairs of protein types that may potentially physically interact. Since $N$ describes the entirety of possible interactions, any protein complex can be seen as a connected subgraph of $N$.

Protein interactions are not independent of each other, but interdependent. Those interaction dependencies are generated by two major mechanisms: allosteric regulation, in which the capability of a protein to bind other proteins is affected by a conformational change upon one interaction [374], and steric hindrance, which prevents proteins from binding to identical or nearly identical protein domains, leading to mutual exclusiveness of interactions [573]. The dependencies between interactions constrain the set of possible protein complexes and their assembly paths. One possible model for this are *constrained protein interaction networks*, where the protein interaction network is enhanced by the interaction dependencies (*constraints*) modeled as propositional logic formulas [649]. With constrained protein interaction networks, we can stochastically simulate complex formation based on the available knowledge and obtain a detailed interaction topology (which proteins physically interact) for each complex.

For the simulation, a *constrained protein interaction network* was generated from the human adhesome protein network and a set of interaction dependencies obtained from protein domain interaction databases and manual curation (for details, see [649]). Then, protein complex assembly was simulated in a step-wise process, with association and dissociation rates calibrated to fit the complex size distribution of the CORUM database [551], until reaching convergence. The obtained complexes mimic the size distribution of known complexes, while also providing information about the actual

**Tab. 2.5:** Properties of the 2 242 972 simulated protein complex graphs. There are 717 distinct vertex labels (protein names), the most frequent ones being 'GRB2' occurring 100 009 times, EGFR occurring 84 706 times, and 'CRK' occurring 83 117 times. The least frequent labels are 'PHF20' (55×), 'CDYL' (63×), and 'PHF20L1' (75×). The average label frequency is 14561.6. The right table shows how the number of distinct labels increases with WL iterations; here $|\mathrm{WL}_{si}|$ refers to the cardinality of the set of all labels up to the $i$-th WL iteration.

| Quantity | Average | Min | Max |
|---|---|---|---|
| $\lvert V\rvert$ | 4.655 | 3 | 126 |
| $\lvert E\rvert$ | 3.655 | 2 | 125 |
| Degree | 1.570 | 1 | 28 |
| Diameter | 2.709 | 2 | 28 |
| Density | 0.530 | 0.016 | 0.667 |

| Label set sizes | |
|---|---|
| $\lvert \mathrm{WL}_0\rvert$ | 717 |
| $\lvert \mathrm{WL}_{s1}\rvert$ | 461 657 |
| $\lvert \mathrm{WL}_{s2}\rvert$ | 4 114 569 |
| $\lvert \mathrm{WL}_{s3}\rvert$ | 9 237 071 |
| $\lvert \mathrm{WL}_{s4}\rvert$ | 14 703 231 |

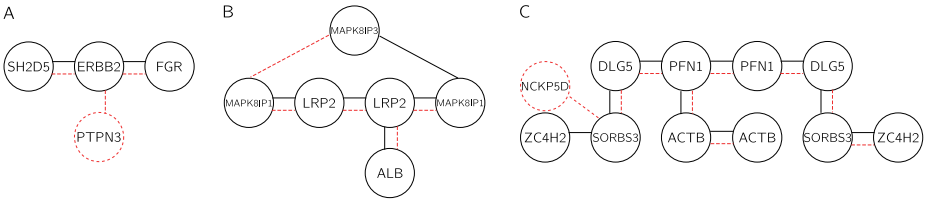



**Fig. 2.28: Three exemplary pairs of protein complexes.** Each labeled node is a protein instance, each edge a protein interaction, and solid black vs. dashed red edges distinguish between the two complexes. **A:** Edit similarity 0.714; WL similarity in [0.4, 0.75] depending on weight $w_1$. **B:** Edit similarity 0.838; WL similarity 1.0 (independent of $w_1$). **C:** Edit similarity 0.9; WL similarity in [0.667, 0.818] depending on $w_1$.

physical interactions happening inside the complex, an information that is currently not yet available for real data. Over 2.2 million protein complex graphs were simulated in this way. Some statistics are given in Table 2.5: Most simulated graphs are small and tree-like ($|V| = |E| + 1$) and consist of low-degree nodes. We were able to verify that all distinct simulated graphs can be distinguished by WL labels after at most two iterations.

To evaluate the Weisfeiler-Leman based similarity ("WL similarity") against the edit distance based similarity ("edit similarity"), we computed both measures on selected pairs of simulated complexes.

### 2.6.3.4 Illustrating examples

We first consider three exemplary pairs (Figure 2.28 A–C) with edit similarities of approximately 0.7, 0.8 and 0.9, respectively, the latter being the most similar observed pair. Our simulation has been calibrated to yield complexes of a realistic size distribution that additionally reflect all currently known interaction dependencies. However, since this data is likely incomplete and we also did not consider the law of mass action, we

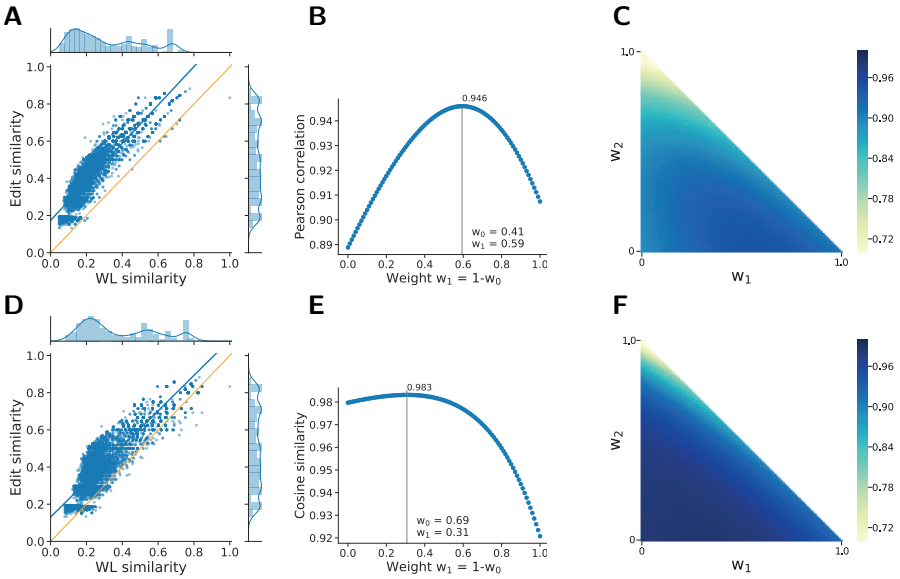



**Fig. 2.29: Comparison of edit similarity and WL similarity. A** and **D:** Scatterplot between edit similarity and WL similarity for weight $w_1 = 0.59$ with maximum Pearson correlation (A) and $w_1 = 0.31$ with maximum cosine similarity (D), including marginal distributions and least-squares regression line. Each point represents a pair of complexes. **B:** Pearson correlation coefficient between edit and WL similarity as function of $w_1$. The maximum 0.946 occurs for $w_1 = 0.59$ (scatterplot A). **E:** Cosine similarity as a function of $w_1$. The maximum 0.983 occurs for $w_1 = 0.31$ (scatterplot D). **C** and **F:** Heatmap showing the Pearson correlation (C) and cosine similarity (F) over weights $w_1$ and $w_2$ when using 2 WL iterations and $w_0 = 1 - (w_1 + w_2) \geq 0$.

do not claim that the particular combination of proteins in these examples is likely to occur in reality. The examples are therefore only meant to illustrate the behavior of the two measures and give an intuition of cases where WL similarity fails to properly approximate edit similarity.

In example A, an additional protein (PTPN3) is added to an existing complex, a linear chain of 3 proteins. The edit similarity is $10/14 = 0.714$, the WL similarity is between 0.75 for $w_1 = 0$ and 0.4 for $w_1 = 1$. Because the edit similarity is between the extreme WL similarities, there exists a unique weight $w_1^* \approx 0.102$, for which WL and edit similarities agree for this particular complex pair. Example B is a noteworthy case, because the WL similarity is 1.0, independently of $w_1$, because the vertex labels are identical even after the first Weisfeiler-Leman iteration. (Further iterations would show a difference.) The edit similarity is $20/24 = 0.83$, which is obtained by attaching ALB to the other LRP2 protein. In example *C*, one protein is replaced by another one in a fairly large complex. The edit similarity (0.905) is relatively high and outside the WL similarity range between 0.667 for $w_1 = 1$ and 0.818 for $w_1 = 0$.

#### 2.6.3.5 Large-Scale Comparison

For the following comparison, we considered only a stratified subset of all possible pairs for the analysis, because calculating the edit similarity is computationally costly. To obtain candidate pairs, we considered all pairs of complexes that have at most 20 proteins (larger complexes are so rare that high similarities are unlikely), that have a size difference of protein multisets of at most 10, and that share at least one protein. These were sorted in descending order according to the number of shared proteins. Then the edit similarity was computed on the first 500 000 candidate pairs, and the similarity values were grouped into bins of width 0.1. Because most pairs of complexes share a small number of proteins, we find many pairs with small edit similarity (but none in the range [0.0, 0.1[ because we required one common protein) and fewer pairs with edit similarity above 0.5. To achieve a uniform distribution among bins for the comparison, we randomly selected 1000 pairs from each bin, excluding the bin [0.9, 1.0[ which contained a single pair. This yielded 8000 pairs of complexes from 8 bins.

Because most protein complexes are small and do not exhibit properties of examples B or C of Figure 2.29, the overall agreement between WL similarity and edit similarity is high. For each of the selected complex pairs, we computed the exact edit similarity and the WL similarity for each weight $w_1 \in W := \{0.0, 0.01, \ldots, 1.0\}$ and $w_0 := 1 - w_1$. Let $e$ be the vector of edit similarity values and $s(w_1)$ the corresponding vector of WL similarity values using weight $w_1$ for WL in the first WL iteration. To compare the similarity measures, we calculated both the Pearson correlation coefficient and the cosine similarity of $e$ and $s(w_1)$ for all $w_1 \in W$. As can be seen from Figure 2.29B, the highest Pearson correlation values occur for $w_1$ between 0.56 and 0.62. The maximum Pearson correlation coefficient of 0.946 is obtained for $w_1 = 0.59$. Figure 2.29A shows the scatter plot between the similarities for this weight. For the cosine similarity, the maximum value is reached for weight $w_1 = 0.31$, but the function is less peaked, and all values of $w_1 < 0.6$ lead to high agreement (Figure 2.29E).

To quantify the possible benefit of additional WL iterations (and hence a larger space of possible weight vectors), we first repeated the calculations with an additional second WL iteration. We calculated the similarity measure for all weight combinations $(w_1, w_2) \in W \times W$ with $w_0 := 1 - (w_1 + w_2) \geq 0$ and $w_0 + w_1 + w_2 = 1$. The Pearson correlation and cosine similarity over the weights $w_1$ and $w_2$ are shown in Figure 2.29 C and F. The weight combinations that lead to maximum correlations all have $w_2 = 0$ and therefore are the same as in the similarity measure without second iteration (Figure 2.29 A and D). Therefore, the additional WL iteration provided no benefit for approximating edit similarity.

Overall, we find good agreement between edit similarity and WL similarity for appropriate values of $w_1$; values around $w_1 = 0.5$ yield both high Pearson correlation and cosine similarity.

#### 2.6.3.6 Runtime Comparison

To study the practical running time of both similarity measures, two different settings were evaluated. In the first setting, all pairwise similarities in a subset of the dataset were computed. This setting is a typical sub-task of clustering or machine learning tasks on metric distances. In the second setting, the database search application was evaluated. Given a set of query complexes and a similarity threshold, similar complexes in the dataset were searched.

**Pairwise Similarities** We measured the time required to compute all 10 000 pairwise similarities for a subset of 100 graphs, drawn at random from the dataset described in Section 2.6.3.3. For the Weisfeiler Leman-based similarities, the computation can be divided into two steps. In the first step, the Weisfeiler-Leman feature vectors are computed for each of the 100 graphs (WL-FV). In the second step, they are used to compute the similarity values between all 10 000 pairs (WL-SIM). Thus, the computational costly part is computed only once for each graph, while the quadratic number of comparisons is lightweight. This kind of preprocessing is not possible for the graph edit similarity, where each pairwise similarity computation is costly (GES).

Figure 2.30 shows the violin plot of the measured times for each single feature vector and pairwise similarity calculation. Running times are shown separately for one and two WL iterations (WL-[FV|SIM]-[1|2]; GES). As expected, the calculation with two WL iterations is slower than using only one iteration, but the difference is small. Most importantly, we observe that a single graph edit similarity computation is two to four orders of magnitude slower than a single Weisfeiler Leman-based similarity computation (median of over $10^6$ vs. approx. $10^3$ ns, but extreme outliers are visible from the violin plots). While Figure 2.30 shows the times for single-instance calculations, Table 2.6 shows the total times for all 100 (FV) and 10 000 (SIM) calculations. For Weisfeiler-Leman similarity, we observe that computing the feature sets dominates the running time. (This eventually changes if more graphs are considered since the WL-FV time grows linearly but the WL-SIM time grows quadratically with the size of the dataset.) Overall, the GES computation is slower by more than four orders of magnitude, which is influenced by a few extremely slow GES computations.

**Database Search** We used the entire set of graphs described in Section 2.6.3.3 as a database and 500 randomly selected graphs as queries. For the Weisfeiler Leman-based similarities, we evaluated a linear scan over the database (`wl_linear`) and the min-hashing speed-up described in Section 2.6.2.4 (`wl_minhash`) with false negative rate 0.01 and weights $w_0 = w_1 = 1/2$. Times were measured separately for the Weisfeiler-Leman feature set calculation, the hash table creation (for `wl_minhash` only), and the queries itself. For the graph edit similarity (`ges`), a linear scan over the database is prohibitive and hashing schemes are not readily available. Therefore, we employed the filter-verification approach described in Section 2.6.2.4 and measured the time for filtering

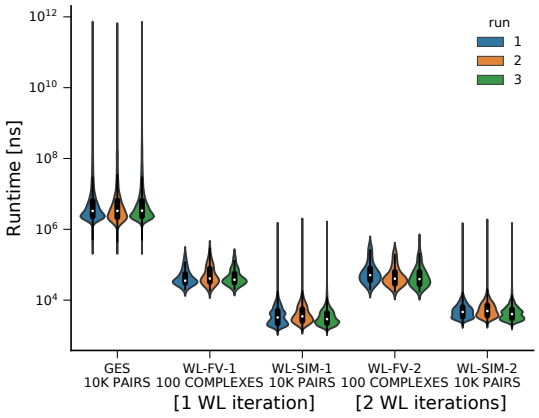



**Fig. 2.30: Running times.** Violin plots of wall-clock times for computing each of the 10 000 pairwise graph edit similarities (GES), the 100 Weisfeiler-Leman sets (WL-FV) and their 10 000 pairwise Jaccard coefficients (WL-SIM) with either one or two Weisfeiler-Leman iteration(s) in three independent runs. The time axis is logarithmic.

**Tab. 2.6:** Total running times for each of three runs in seconds. WL-Total = sum(WL-FV) + sum(WL-SIM) for either 1 or 2 Weisfeiler-Leman iteration(s). Here sum() refers to the sum over the 100 feature vector calculations (WL-FV) and 10 000 similarity calculations (WL-SIM) whose time distribution is depicted in Figure 2.30.

| Time [s] | Run 1 | Run 2 | Run 3 |
|---|---|---|---|
| GES | 851.222 | 809.087 | 845.883 |
| WL-Total-1 | 0.050 | 0.056 | 0.045 |
| WL-Total-2 | 0.066 | 0.071 | 0.055 |
| sum(WL-FV-1) | 0.005 | 0.006 | 0.005 |
| sum(WL-SIM-1) | 0.045 | 0.050 | 0.040 |
| sum(WL-FV-2) | 0.008 | 0.006 | 0.006 |
| sum(WL-SIM-2) | 0.058 | 0.065 | 0.049 |

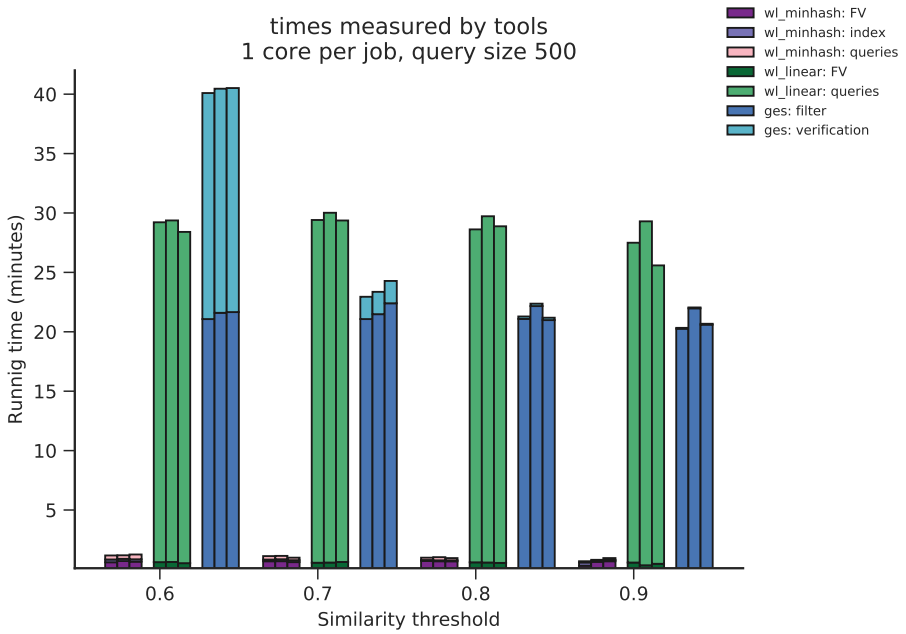



**Fig. 2.31:** Running time comparison of the three approaches wl_minhash (purple), wl_linear (green), and ges (blue) for a database search with 500 queries and a database of ≈ 2 million complexes.

and verification separately. All three algorithms were run with the similarity thresholds 0.6, 0.7, 0.8 and 0.9 and each run was repeated three times. Further, each run was executed using only a single CPU core for better comparability of CPU time usage; in practice, many database complexes can be evaluated in parallel, independently of each other. The results are shown in Figure 2.31.

It can be seen that `wl_minhash` and `wl_linear` need a similar amount of time for the feature vector calculation, but the query time is much smaller for `wl_minhash` because only a small fraction of the database complexes needs to be evaluated. These savings outweigh by far the additional indexing time required to build the hash tables, even for only 500 queries. Therefore, `wl_minhash` will scale even better to larger numbers of queries. The filter-verification approach of the graph edit similarity (`ges`) is slightly faster than `wl_linear` for thresholds 0.7, 0.8, and 0.9, because using the similarity bounds often suffices for a decision and the actual graph edit distance does not need to be computed often. For threshold 0.6, however, many verifications and hence exact computations are necessary; so the running time for ges increases above that of `wl_linear`.

### 2.6.4 Discussion

Our original motivation to consider protein complex similarity was to reduce the size of the simulation output of our constrained protein interaction network simulator [649] via clustering, and we were surprised to see that apparently, no similarity measures have been proposed explicitly for protein complexes in the literature. Depending on the underlying representation (set, multiset, or graph), different alternatives suggest themselves. However, most graph-based measures are both theoretically and practically hard to compute for larger complexes or for large amounts of complexes. While different tractable graph similarity measures [705] and an approximate graph edit distance [534] have been developed, none of these appear to be specifically tailored to the properties of protein complexes (often less than ten vertices; sparse).

Our proposal to define the similarity as a convex combination of two Jaccard coefficients (protein label multiset and Weisfeiler-Leman label multiset after one iteration) has several beneficial properties. We have shown that it can approximate edit-based similarity with high Pearson correlation and cosine similarity, and at the same time, that it can be computed much more efficiently. Further, for weight $w_0 = 1$ of the 0-th WL iteration, WL similarity reduces to the natural similarity measure of the multiset representation. Our framework hence allows for a smooth transition between multiset and graph representation. The comparison with an edit-based similarity seems to indicate that the protein label multiset plays an important role if one wants to approximate the edit similarity, the first WL iteration provides additional graph information, and the second WL iteration does not provide further benefits, probably because most complexes consist of few proteins. In addition, in large-scale similarity searches, using Jaccard coefficients allows us to efficiently pre-filter for high similarity using locality-sensitive hashing. In combination with the preprocessing abilities discussed in the experimental running time comparison, this allows for very fast search queries, clusterings, and other applications that rely on intensive distance computations.

In the present work, we have not considered individual similarity between vertex labels (i.e., protein types): We treat two labels as either equal or distinct. While it is relatively straightforward to allow arbitrary label similarities in the graph edit distance framework, and, via Equation 2.25, in the graph similarity framework, this generalization appears less straightforward for WL similarity and will be investigated in future work.

From a biological point of view, a high similarity value between two protein complex graphs should indicate a high probability that the complexes share biological functions and can (partially) substitute each other in a cellular process. If comprehensive functional information were available, we could use it for evaluating different similarity measures and decide which ones best capture the biological reality. At present, when not even the interaction topology of most complexes has been determined, such an evaluation is unfeasible, but this may change in the future.

# 3 Industry 4.0

## 3.1 Keynote on Industry 4.0

*Michael ten Hompel*
*Moritz Roidl*

**Abstract:** Industry 4.0 is the connecting element of all contributions in this chapter. The notion of a physical thing moving within an industrial process unites the research presented. Logistics is a driving force behind Industry 4.0 developments and that the concept of a cyber-physical twin is one of its fundamental building blocks. This keynote introduces and develops the mindset for fully appreciating the following contributions. They range from the creation of a steelbar to autonomous swarms of drones, providing a wide-ranging selection of current developments.

### 3.1.1 Introduction

The term "Industry 4.0" heralds a fourth industrial revolution. The factory of the future is built upon a hyper-connected, smart, and autonomous infrastructure. It promises to deliver high adaptability with optimal use of resources. Industry 4.0 is expected to bring considerable benefits for production sites, factory equipment suppliers, and business software providers, on both the productivity and the revenue side. Under Industry 4.0, digitization penetrates all areas of industrial process chains–from production to distribution to recycling and waste management–and is thus spurring expectations and ideas regarding the future design and implementation of processes, facilities, and systems.

### 3.1.2 Industry 4.0

Amid the wide variety of views on the term Industry 4.0, this keynote adheres to the German concept of a Plattform Industrie 4.0. According to this, the term Industry 4.0 "stands for the fourth industrial revolution, a new stage in the organization and control of the entire value chain" and is intended to establish a link to the three previous industrial revolutions. Many authors refer to Industry 4.0 collectively as the digital transformation of the manufacturing sector. A comprehensive overview of the field is given in [279], [548], and [697].

As the term revolution suggests, the transformation is likely to present a variety of opportunities for an economy that can deal with the disruption that comes with


https://doi.org/10.1515/9783110785982-003

profound change. In addition, the successful implementation of Industry 4.0 concepts and technologies requires the ability to easily change business processes with regard to new requirements. This sense of a high adaptability calls for the creation of a kind of "deep transparency" to efficiently evaluate decision-making situations. The basic vision of Industry 4.0 thus stems from the age-old desire for the constant availability of all relevant information about all physical objects at all levels of industrial process chains. Technologically, this "deep transparency" is to be achieved through the massive use of low-cost sensor technology. To ensure constant availability of information, state-of-the-art communication technologies provide the means for both networking the production technology and networking between the hierarchical levels of the IT architecture. From a business perspective, the hope is that this comprehensive networking will give rise to highly adaptable supply-chain networks that can organize and optimize themselves and thereby also form the basis for a wide range of business model innovations (cf. [372], [189]).

**Concepts**

Industry 4.0 encompasses various concepts that not only cover different technologies, but also affect various structural aspects of the organization within a company and between companies.

The constant availability of information requires a smooth exchange of data within the entire value creation network through the integration of physical objects and IT systems. This gives rise to the concepts of *vertical and horizontal integration*. Vertical integration describes the creation of a coherent network for objects and systems within a company. All internal IT systems are interconnected via harmonized interfaces that serve to exchange data between a single sensor, a production machine, or the production planning system. Horizontal integration takes place across the entire value creation network. The vertically integrated IT systems of customers, suppliers, or the company's own distributed sites are integrated into a horizontal system landscape. This enables the exchange of information in real-time across company boundaries (cf. [310], [548]).

While the goal of constant availability of information can theoretically be achieved through a centralized approach, Industry 4.0 explicitly recognizes the geographically distributed nature of the value creation network. This is covered by the concepts of *decentralized control and autonomous behavior* of physical objects. These concepts encompass two important aspects. On the one hand, the disruptive dissolution of centralized, rigidly planned control systems is necessary simply from the point of view of transmission and computing power due to the large amounts data that are to be delivered in real-time. On the other hand, decentralized control enables the use of autonomous, automated decision support systems. Based on the available data, these systems should support human decisions or make them partially autonomous (cf. [617]).

The digitization of all physical objects is reflected in the concept of Cyber-Physical Systems (CPS). These are created by combining a purely physical system with computing

power, memory and a communication interface. In the context of Industry 4.0, this combination enables the creation of smart products, production tools and machines as part of the production process. CPS are expected to generate very large amounts of data, which in turn requires the on-demand availability of decentralized, high computing capacity and methods for processing and analyzing the information.

Several CPSs are the building blocks of the concept of Cyber-Physical Production Systems (CPPS). These are production systems that use CPSs and newly defined interfaces between machines and between humans and machines to accomplish the production task. CPPS are designed to enable decentralized, semi-autonomous, cross-enterprise production control (cf. [437], [617]).

Another concept related to Industry 4.0 is *end-to-end digital engineering*. It encompasses the digital mapping of the entire physical production process in a company, from product development to product completion. All planning, control and monitoring processes are constructed and simulated in a virtual environment. The basis is a digital image of the factory with all its physical objects. This includes production facilities, personnel, products and other working and operating resources that are present in the real factory (cf. [617]).

**Maturity Levels**

The idea of Industry 4.0 is often reduced to a visionary description that represents an already fully realized implementation of all concepts. Although almost all the technologies that would be necessary for successful implementation are already available, it is often only the right combination of approaches and solutions that brings positive results for a company.

The "Acatech Industrie 4.0 Maturity Index" describes a maturity-based development path towards a fully developed Industry 4.0 capability (cf. [584]):

- The first two development levels are not seen as part of Industry 4.0 per se, but form the basis for all further levels. *Computerization* refers to the isolated use of information technologies, while *connectivity* refers to the networking of these isolated systems. Both terms cover the first and second levels of the maturity model.
- The third level describes the ability to *perceive* through extensive equipment of physical objects with sensor technology. This enables the comprehensive collection of data points about interrelated processes. The aim of this stage is to generate the ability to create an up-to-date digital model of reality at any time.
- The fourth development stage describes a state of *transparency* in which, building on the third development stage, interrelationships and causes become comprehensible through the analysis and interpretation of these interrelationships.
- The fifth level describes the presence of *forecasting capability*. For this purpose, the system is equipped with the ability to simulate dynamic processes. On this basis, automated simulation experiments can be carried out that depict different

future scenarios. The evaluation of the experimental results leads to the forecasting capability of the system.
– At the sixth level, *adaptability*, the forecasts are used to derive decisions independently and implement suitable measures automatically.

These six developmental levels represent a development path for the Industry 4.0 capability of a company (see Figure 3.1). In addition, the development levels provide a reference point for classifying existing Industry 4.0 technologies (cf. [584]).

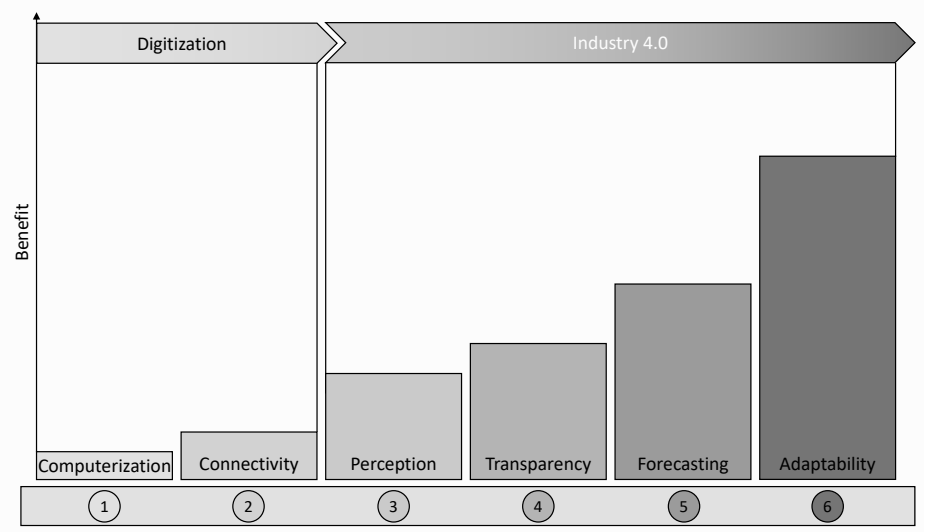



**Fig. 3.1:** Maturity Index for Industry 4.0.

### 3.1.3 The Role of AI in Industry 4.0

From the perspective of Industry 4.0, AI technologies can be used to enable a company for reaching the higher levels of the maturity index. AI technology is intended to enable technical systems to autonomously perceive their environment based relying on the data available to them. In particular, it can be used to locate and identify physical objects and determine their state. The subsequent interpretation of the perceived objects and the meaning of their state as well as the relationships between the objects shall help to achieve the necessary transparency for reasoning about industrial process chains. The automatic creation and updating of simulation models for an automated continuous prediction system, as envisioned for the fifth maturity level, could be supported using certain AI methods such as imitation learning.

While AI is sometimes understood in its function as a technological form of human decision-making capability, in the context of Industry 4.0 it is not necessarily intended to directly copy human behavior. On the one hand, AI is intended to achieve the classic goal of automating industrial processes: cost reduction, time savings, quality assurance, etc. On the other hand, AI technologies are expected to manage the emerging complexity of the vast amounts of data coming in from the newly deployed Industry 4.0 sensor systems. To some extent, AI systems may solve problems, discover unexpected issues, or reduce complex situations by uncovering associations in the data hidden to the human eye.

**The Data Crisis**

However, AI technologies are not "magic fairy dust", but are limited by the underlying data and mathematical models used to try to solve a specific problem [377].

One of the most important engineering issues for the successful use of AI technology in Industry 4.0, especially machine learning methods, is therefore the management of the underlying data collection and its representation in a database. Generating an error-free and uninterrupted data stream is a prerequisite for successful operation. If the data is inconsistent and of poor quality, the digital images of physical objects cannot represent the true physical state. Predictions and decisions made on the basis of such a virtual model would then also be of poor quality.

The design of any Industry 4.0 data stream begins with developing a fundamental, ontological notion about the true nature of the physical object in question. Critical factors are the physical attributes of the object and the selection of sensors as well as the programming of the transformation of sensor data into semantically correct observations. The overarching challenge is that in many places the tools to adequately assess these aspects are still lacking. In particular, in many cases 80 % of the time building a system is spent on data collection and cleaning, a situation that has been called the *data crisis* [377]. "The main reason for the data crisis is the increasing interconnectedness of computers. Access to data is therefore easy and cheap, but its quality is often poor. What we need are cheap data streams of high quality. This means that efficient methods for improving and verifying data quality must be developed."[377] This crisis becomes even more apparent when considering continuous data acquisition by a multitude of sensors, as envisaged for Industry 4.0 systems. Challenges include monitoring the quality of the data stream throughout the life cycle of an Industry 4.0 system and the ability to respond appropriately to changes in the underlying assumptions made at the beginning of the design process.

### 3.1.4 From Digital to Cyber-Physical Twin

In response to the data crisis, the development of a system should more closely link the design of data acquisition with the design of the physical objects concerned. This results in the concept of the digital twin , which combines the connectivity of a CPS with the virtual representation of objects in end-to-end digital engineering. The concept of the digital twin focuses in particular on the quality of data transmission between the physical and the virtual counterpart.

Like the Industry 4.0 maturity index for companies, the literature on digital twins knows different implementation levels for the digitization of physical objects [354]. The development stages are differentiated according to the degree of automation:

- A digital model has a purely manual data transmission. It can be, for example, a simulation model of a planned factory, a mathematical model of a new product or another model of a physical object that is stored digitally.
- A digital shadow automates the data flow between the physical and the virtual object. Any change in the state of the physical object is transferred and applied to the state of the digital object.
- The digital twin eliminates all manual data transfers. It extends the concept of digital shadow by automating the flow of data directed to the physical object.

In order to be able to map data automatically onto the virtual object, correct identification of the physical object is a prerequisite for both the digital shadow and the digital twin. In addition, the complete elimination of manual data transmission makes the design of data reception at the physical object particularly relevant. Necessary conditions for the digital twin can be described as follows:

- The data of the virtual object must be able to be received and processed automatically at the physical object.
- A physical object can only be part of a digital twin if it can be perceived separately from other objects.
- A perceived physical object must be uniquely identifiable so that the captured sensor data can be assigned to the correct virtual object.

The concrete technical implementation of the identification can vary. It can be done by observation, e.g. by a scanner reading 2D barcodes. It can also be done by communication, e.g. via radio transmission, where recognition and identification result from the defined standards of radio transmission, i.e. how receiver and transmitter can identify each other is defined by communication protocols.

Many of today's Industry 4.0 applications can be classified under the concept of the digital shadow. One example is a low-cost tracker that is built into the bottom of a pallet and records data about the pallet's current location, movement, impacts and temperature profile. This data is sent over the cellular network to a server where the corresponding digital shadow is located (cf. [354]).

**Cyber-Physical Twin**

Due to the very general metaphor, digital twins can vary greatly in size and complexity. Therefore, the application areas and the possible types of digital twins can be very broad. In particular, there is no distinction between individual physical objects and large, complex systems. The term digital twin can describe a system that includes only a small object or an entire factory. In the case of a large system such as a factory, a central, monolithic approach in the form of a central database would reach its obvious technical limits.

For use in an Industry 4.0 environment, a special class of digital twins is therefore required that provides for a decentralized, modular architecture based on individual physical objects. Their networking should enable scalable, adaptable systems that support autonomous behavior. This new extended concept can be called a cyber-physical twin . It has the following additional characteristics:

- The physical object is considered to be a self-contained and relatively small entity. Its physical extent can be confined to a finite space, and it has a definite location. It is fundamentally mobile, even if it remains in a particular place for a long time.
- The transmission path to the physical object is primarily for changing its behavior rather than issuing direct commands. All behavioral changes are first made to the digital object, with the physical object adjusting its internal behavior accordingly. The behavioral changes can be made by transmitting parameters, compiled source code, or other behavioral data (e.g., a trained neural network).
- The physical object acts largely autonomously and makes local decisions where possible.
- The physical object monitors its environment and can trigger an update process in the virtual image when situations arise in the physical world that require it to adjust its behavior.
- Cyber-physical twins are designed as multi-agent systems that communicate in both the physical and virtual worlds.

Figure 3.2 shows the concept of two cyber-physical twins negotiating with each other in the physical and virtual environments. Note that communication between the environments takes place via an exclusive link between the physical and virtual objects to ensure synchronization of the twins.

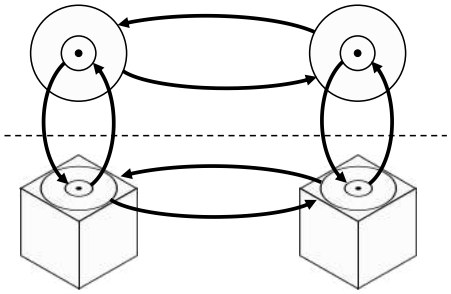


**Fig. 3.2:** Cyber-physical twin concept

Although not all cyber-physical twins are controlled by AI, the challenges for software and hardware engineering are similar, as high-quality data streams are required in both cases.

The mobility of the physical object is required for the adaptability of an Industry 4.0 system, as it can then change the arrangement of its components. It follows that the embedded computer system of the physical object is in principle resource-constrained, while the computer system in which the virtual object runs is considered unconstrained.

### 3.1.5 An Excursion into Logistics

Since the introduction of Industry 4.0, the area of *logistics* has been considered its outstanding application domain. In no other area of industry is such a fundamental change expected in the near future. On the one hand, many of the central technical and social challenges are directly or indirectly linked to logistics and efficient supply chain management. On the other hand, this is due to the rapid development of Industry 4.0 technologies. In addition to global data processing capabilities, resource-efficient sensor hardware, communication technologies, and embedded systems are increasingly usable on a large industrial scale. This enables widespread rollout at the level of individual logistics objects (cf. [164]).

In the context of an increasingly volatile production and trade environment, the topology of logistics networks and thus the location of an individual logistics node, such as a transshipment point or a distribution center, can no longer be permanently fixed. In fact, the idea of a fixed, ideal location has not been viable for many years. A logistics network and its nodes must constantly adapt to new circumstances. Therefore, logistics centers should be able to relocate on their own in the future. This rules out many forms of traditional, technical infrastructures and underscores the need to introduce new concepts such as the cyber-physical twin (cf. [279]).

To an increasing extent, swarms of autonomous vehicles will take over intra-company transport. In production systems, the arrangement of workstations can thus

be changed at any time. The vehicles' virtual software agents negotiate orders and business processes, while the physical software agents negotiate movement paths and constantly exchange their locations and those of new stations or storage locations. Autonomous vehicles are able to approach the racks and move bins or pallets in and out. There is virtually no stationary conveyor system in this vision.

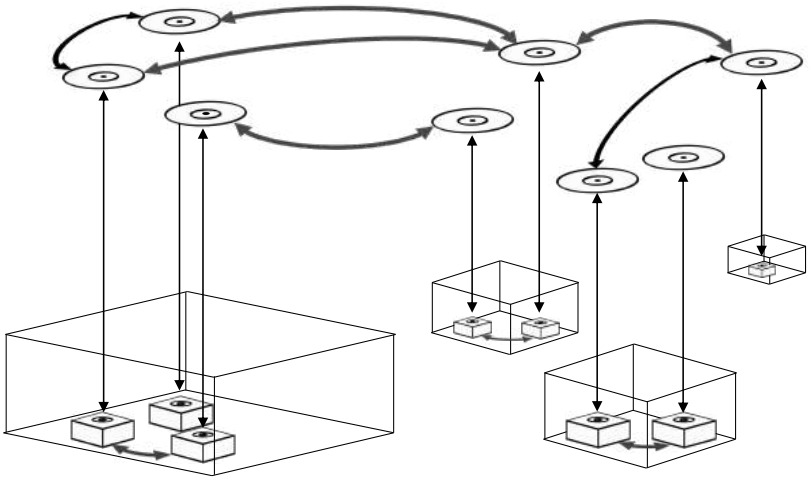


**Fig. 3.3:** Cyber-physical twins managing warehousing tasks in a supply-chain

Even the shelf and each bin within it can become part of a cyber-physical twin. The bins in the warehouse handle inventory management, check minimum stock levels, and order replenishment (see Figure 3.3). They communicate with shelf displays and transport vehicles. The classic, RFID-based "Internet of Things," as it was devised at the turn of the millennium, is literally getting eyes, ears, arms, and legs.

Direct challenges quickly arise from such all-encompassing networking of physical and virtual objects. Although the absolute amount of data has been increasing significantly in all areas of the economy for a long time, the massive increase in the collection and processing of logistics process data is becoming one of the central challenges to be mastered. The potential for AI applications in cyber-physical twins handling complex logistics processes is significant. From predicting arrival times in transportation and production logistics to dispatching logistics networks and the coming high-frequency logistics, distributed AI solutions can automate and autonomize complex processes that were previously closed to classic control methods. It is expected that AIs will be able to learn how to cope with the complexity of a large number of cyber-physical twins representing logistics objects.

### 3.1.6 Resource-Constrained Smart Objects That Can Move

The contributions in this chapter address four key areas associated with physical objects in Industry 4.0 systems: the creation of things, their localization, the resource constraints of objects once they become mobile and smart, and their behavior with respect to each other.

The first three contributions deal with the genesis of a thing long before it becomes intelligent or autonomous. Using the example of steel production, machine learning methods are used to investigate how the act of physical separation creates a self-contained entity from a primordial mass and how this separated thing can be assigned to a concrete class by determining or predicting certain properties. An early prediction of undesirable developments enables a controlling intervention in the production process (Section 3.2). The second contribution is dedicated to challenges for machine learning techniques that arise in the context of changing, physical aggregate states (Section 3.3). Combining expert knowledge and collected data in simulation is the topic of the third contribution (Section 3.4).

The fourth contribution deals with the automated localization of things–a basic prerequisite for a continuously (self-)perceiving infrastructure that supports flexible (self-)control of processes. Industry 4.0 thus requires a comprehensive ability to observe physical object movements. Wireless ultra-wideband localization is an example of the technical solution to this task in indoor environments (Section 3.5). In this case, the radio medium is the limiting resource that affects the accuracy and scalability of the application. The contribution is related both to the *smart city* concept (see Chapter 4), which deals with a cross-location infrastructure based on future wireless technologies, and to communication networks (see Chapter 5).

The fifth contribution is motivated by smart things that can move. Free mobility is at its core a deeply logistic property ("the ideal logistic space is empty"). It is also the main cause of resource constraints in terms of energy supply, computational capability and communication for embedded systems. The topic of "Indoor Photovoltaic Energy Harvesting" (Section 3.6) deals in depth with the requirements for ultra-low power devices and their modeling. An important application of this technology is the standardized logistics container, which becomes smart by an embedded ultra-low power computer system. Such smart containers are destined to be one of the key building blocks of Industry 4.0: they are the external interface for the non-smart things that they contain.

The last contribution deals with the behavior of smart and in principle autonomously moving things. It is based on the development of a micro-UAV drone swarm (Section 3.7). The resource constraints of the small drones place high demands on the system architecture. The contribution describes the setup of a testbed environment in which simulation and the physical world are tightly coupled. The drones negotiate their individual movements through simple rules that mimic natural swarm behavior. It is shown how individual rules can be replaced by knowledge learned in

a simulation. The drone swarm testbed illustrates the challenges of developing autonomous swarm systems for Industry 4.0. Since these systems will increasingly be used across companies in business-critical areas in the future, the issue of data protection in multi-agent systems will also become important in this context (cf. Section 6.1).

## 3.2 Quality Assurance in Interlinked Manufacturing Processes

*Jochen Deuse*
*Katharina Morik*
*Amal Saadallah*
*Jan Büscher*
*Thorben Panusch*

**Abstract:** Interlinked manufacturing processes are characterized by the dependence of downstream process steps on the previous ones. If it can be predicted that a particular workpiece will not reach the desired quality, anticipatory measures can be taken early in the process. By its prediction, machine learning saves resources, both in processing and in the material. One example is the model-based quality prediction in electronics manufacturing on a Surface Mount Technology (SMT) production line. Here, the application of a learned classifier predicting the quality must be fast so that, say, the routing of a piece may be changed. Hence, machine learning itself needs to save its resources, in this case runtime. Another example is the hot rolling mill process for steel bars production. There, several sensors and process parameters deliver process data that need to be aligned and useful features are to be extracted from the resulting stream automatically, in real time. Here, machine learning saves testing time in the factory's quality inspection process.

### 3.2.1 Introduction

Undetected quality deviations passing through the entire manufacturing chain have a severe impact on internal failure costs due to the increasing rejection and reworking of defective products without being labeled as defective. Therefore, early quality prediction of a specific workpiece indicates whether it will reach the required quality requirements or if some anticipatory measures should be executed in a timely manner to save resources (e.g. time, material) resulting in rejection or further processing. However, due to technological and temporal restrictions, physical product quality inspections are limited to the final process step. In this context, data mining and machine learning techniques can be used to predict the intermediate product's quality, thus gaining transparency on quality properties of intermediate process steps and enabling real-time process adaptation to sustainably increase its efficiency [339, 654].

In general, process modeling can be done on three levels [654]: process understanding; designing better processes and equipment; and online process control in real time. Online control requires integrating data from different sensors at different steps in the process, taking into account the communication between sensors and even

integrating different models in real time. Over the last decades, the majority of works were focused around the first two levels, where models based on domain knowledge have been developed. More recently, with the emergence of the "industrial Internet of Things" or "Industry 4.0" (see Section 3.1), the adoption of different monitoring technologies using sensor technologies [155, 156], has offered new opportunities for applying data-driven approaches for process modeling, in general, and for intermediate quality prediction, in particular.

This section is about making Intermediate Quality Prediction (IQP) along a process chain by embedding data analysis directly into the manufacturing chain. We start by showing how IQP using data mining and machine learning can be integrated into a comprehensive Intelligent Manufacturing Process Control (IMPC) framework for industrial applications. Section 3.2.3 dives deeply into the steps of data analysis. These steps include a detailed description of the data acquisition process, cleansing, the choice of data representation, extracting the right features, , modeling, and evaluation. A framework for processing data streams with the right level of abstraction in real-time, in addition to the real-time management of many machine learning models, is also presented. It glues diverse contributions of data mining and modeling together to form an application.

We present two real-world case studies in Section 3.2.4, starting with the description of the production process and data acquisition followed by modeling and deployment. The first case study addresses the use of data mining in an electronics production environment for the purpose of reducing the quality inspection volume. The case study is conducted on a Surface Mount Technology (SMT) manufacturing line in the Siemens plant in Amberg. The motivation is to relieve the optical end-of-line test, consisting of an X-ray inspection system. The second case study consists of a hot rolling mill process. It showcases the importance of embedded data analytics. The system is developed in close collaboration with experts in machining, production, and the steel mill. Data analysis results are validated by domain experts. The conclusion summarizes our findings and gives an outlook for future research work.

### 3.2.2 Intermediate Quality Prediction in Intelligent Manufacturing Process Control

Different quality-related tasks for the application of data mining and machine learning in manufacturing can be distinguished [335]: description of product/process quality; modeling of the product quality; quality prediction; and parameters/process optimization.

In literature, there are several standardized procedures for the implementation of data mining or machine learning in general and one approach for optimizing quality-related tasks in particular. Four widely used process models include the Knowledge Discovery in Databases (KDD), the Cross-Industry Standard Process for Data Mining (CRISP-DM), the 5-step SEMMA model by SAS, and the 5A model by SPSS [121, 201, 484,

668]. The general framework for continuously monitoring and optimizing a process under quality considerations is known as IMPC [339]. A characteristic feature of the models mentioned is that they are very similar in terms of their basic objectives and structure. The procedures are subdivided into distinct iterative phases that differ in terms of the number and the content of the respective phases. While the IMPC solely focuses on the technical perspective of a data mining project, some other models include a phase for elaborating the business cases, which indicates the necessary involvement of expert knowledge in every data science project.

Product or process quality description is usually the first step to be performed, especially in the context of highly complex manufacturing systems with non-linear interactions between the different process steps [579]. This model can be based on a physical model designed by domain experts, or it can be a data-driven model using machine learning techniques. The model is subsequently used to predict the quality of new unseen products. Following these predictions, a variety of measures for process optimization can be applied [405, 579, 654]. These measures include early control interventions, optimization of process parameters setting, stabilization of processes, dynamization of inspection plans, and the design of model-based inspection processes.

The IMPC introduces data mining techniques to perform IQP and adjust further production processing steps according to the results of the IQP. The IMPC consists of different functional modules that can be summarized as follows [339]:

1. Data acquisition and storage
2. Intermediate Quality Prediction-IQP
3. Process optimization

While the first module seems to be a prerequisite for building the IMPC, modules 2–3 represent the different process control stages.

The IMPC itself can be realized following two different paradigms. The first one is based on an online optimization of the process based on the current observed state of the process in order to compensate for previous process deviations. Such a decision is made following the domain knowledge provided by production experts and engineers. The second one is a data-driven approach where the first and second modules are integrated into the decision support system. The second type of process control relies on estimating the quality of the intermediate product in real time. Anticipatory measures are taken to prevent possible predicted process deviations. The modular design of the entire process control approach is detailed in Figure 3.4.

The IMPC concept can be viewed as a separate building block in a company's process control landscape [339]. The IMPC does not introduce any change in the process or production structure. However, it generates recommendations in process planning and optimization. Therefore, the focus is brought on analyzing processing states in real time and forecasting the intermediate product's quality properties. This knowledge is used for either deriving recommendations on whether the product should be processed any further or for stopping the processing of the product early, since the final quality

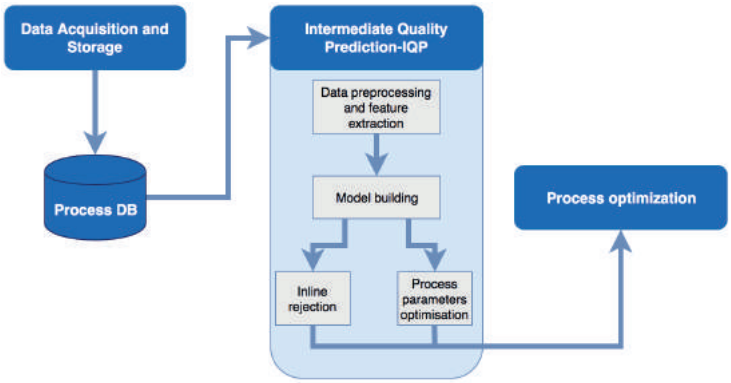



**Fig. 3.4:** Intelligent manufacturing process control model.

requirements will not be met, or for optimizing the next processing step's parameters to align the product with the required quality standards.

The IMPC requires information on the current processing state and parameters that are gained from the first module, in addition to information on historical processing data. Data is most often collected from sensors implanted to monitor some process variables. Collected data has to be preprocessed and meaningful features should be extracted to be fed afterwards into a quality prediction model in order to assess the intermediate product's quality correctly. Hence, the Intermediate Quality Prediction IQP module translates all available information irrespective of its actual meaning into a quality assessment. This is done by means of data mining, i.e., supervised learning models [339, 654]. The result of the IQP module can then be applied to decision rules or recommendations on process optimization. The process optimization module is the final step in implementing the IMPC model. As described above, it aims at adjusting the parameters of upcoming process steps in such a way that quality deviations caused by previous processing steps are compensated in the remainder of the process chain. It is worth mentioning that when it comes to decision rules following the results of the prediction of intermediate products' quality, even more knowledge is required, including domain knowledge and artificial test-beds (e.g. process simulations) to validate the *correctness* and *feasibility* of the process optimization recommendations.

In the following sections, we will discuss in detail available methods for data analysis and quality modeling using machine learning.

### 3.2.3 Methods for Data Mining from Sensor Data

The dynamic evolving nature of manufacturing processes poses multiple challenges for data mining from sensor data [654]. For instance, different scenarios may occur, sensors and machines may fail, some deviations in processing steps could be observed

and products may take different routes through the process chain. With such events, difficult challenges on how data should be collected, stored, and preprocessed arise. Additionally, adaptive extraction and selection of which features may be relevant for the prediction task, are necessary. Moreover, to generate intermediate products' quality predictions in real time, stream data mining methods and online management of many machine learning models should be established. The following sections will discuss such problems and available methods in more detail. The structure of the section is depicted in Figure 3.5. It is based on the IQP process steps and comprises standardized procedures that are common steps in every quality-related data-mining task.

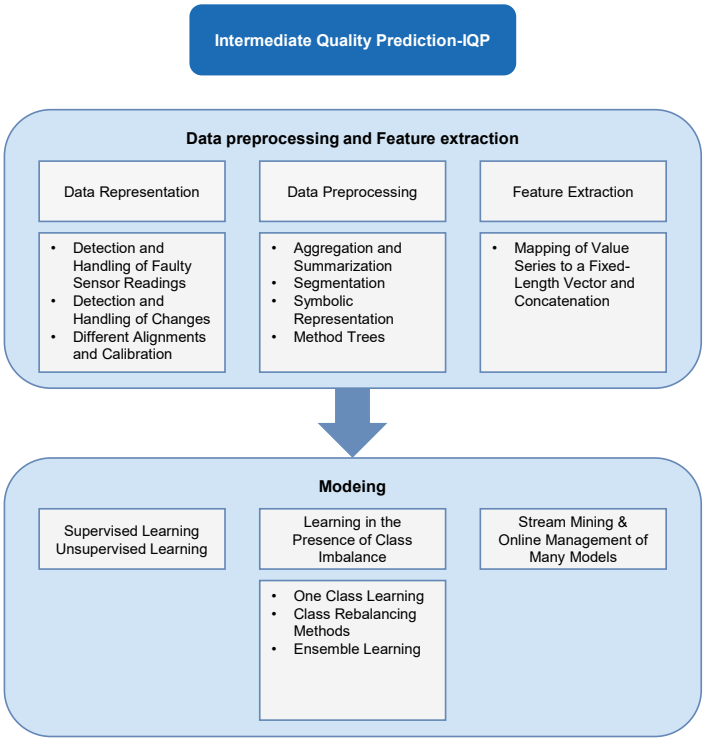



**Fig. 3.5:** Specification of the IQP process with regard to common steps in quality-related data mining tasks.

### 3.2.3.1 Data Acquisition and Storage

In intelligent manufacturing systems, data is most often collected using various sensing technologies [339, 405, 654]. The continuous measurements of sensors form an indexed stream of countably infinite data items $x_i$. Every data item contains an index, e.g., a timestamp, and can contain an arbitrary number of values and value types such as

images, strings, or numbers. A segment of this stream with length $n$ and only numerical values forms a value series that can be defined [445] as a mapping $x : \mathbb{N} \to \mathbb{R}\times\mathbb{C}^m; m \in \mathbb{N}$, where each element $x_i$ of a value series with length $n$ is an ordered pair $(d_i, w_i).d_i \in \mathbb{N}$ is called the index component and $w_i \in \mathbb{R} \times \mathbb{C}^m$ the value component. It is called a time series if the index dimension represents a temporal order. The value component is written using a complex number value space instead of a real number value space in order to formalize sensor readings and their transformations in the same form. In classification tasks (e.g. quality prediction) the goal of a machine learning model is to learn a functional mapping $f(x) : \mathbb{R} \to y : \mathbb{N}^n$, where $y$ formally denotes the class label of the process measurements. In general, this task is named a multi-class classification task. The special case of binary classification occurs when the class label consists of only two complementary classes, say, the Not Ok (NOk) class and the Ok class.

#### 3.2.3.2 Data Preprocessing

In manufacturing environments, sensors may deliver wrong readings or might experience failure periods [474]. Their readings are most of the time noisy. Henceforth, collected data may contain irrelevant readings, be wrongly aligned, or have different resolutions. In an offline setting, such cases can simply be excluded from the analysis or corrected. By contrast, the embedded real-time analysis of data must somehow detect such cases in an automated fashion and react accordingly. The first analysis step therefore usually consists of cleaning the sensor data.

**Detection and Handling of Faulty Sensor Readings** Faulty sensor readings such as sensor readings lying outside physically meaningful ranges need to be detected. Nevertheless, faulty readings may overlap with the normal data, requiring the automatic detection of faulty patterns using supervised learning techniques (see Section 3.2.3.5) [654]. However, if the faults are not highly frequent, it is difficult to detect them based on available training examples. Models for anomaly detection can be of great help in this context by describing only the normal data. They mark patterns deviating largely from the distribution as anomalies. Nevertheless, the correct definition of parameters, like threshold values, remains difficult with only a few noisy examples. Furthermore, it can be difficult even for domain experts to identify such examples correctly. There are several possible ways on how to handle missing values after being detected including replacement by their predecessor values or auto-regressive moving average approaches [98] or imputation based on predictive models that are trained on other existing values. Additionally, the production process itself might introduce some level of noise to sensor data. If the underlying noise model is known, it should be used. Otherwise, measurements should be filtered and smoothed [405, 654].

**Detection and Handling of Changes** Deviations and changes in sensor readings may also be explained by intended changes in the underlying hardware, such as when

new production equipment and new or different calibrations of sensors are introduced. Based on the type of changes, it must be decided which of the trained models need to be updated and how. Without any models describing such changes, methods for concept drift detection can be applied and it has to be decided if already trained prediction models must be updated and how these models should be managed [564]. More details are provided in Section 3.2.3.10. Incremental training methods, like streaming methods (see Section 3.2.3.9), can be used to integrate such changes into the trained models automatically.

**Different Alignments and Calibration** Different data alignments can be observed due to different data sampling strategies and sampling frequencies from sensor recordings. The measurement frequency of a data source is defined by the number of times the source delivers data per time unit. The higher the desired sampling frequency, the more memory it uses. The association and integration of data from many sensors require their synchronization with the environment description. The environment description is characterized by the adequate time of measurement. To obtain a precise synchronization, a sufficiently accurate global time of measurement for the different sensors is required to be defined or derived using synchronization techniques [309]. In this context, a multi-sensors data fusion system has to cope with different and varying measurement frequencies, measurement latencies, and asynchronous measurement times.

Synchronization techniques can be divided into two main families: deterministic and non-deterministic. In a deterministic setting, the measurement times of each sensor have to be known in advance and synchronization is performed on the slowest sensor using aggregation techniques [309]. In a non-deterministic setting, sensors are assumed to be asynchronous and there is no knowledge about measurement times or latencies. In the process recordings, different frequencies might be used. In such situations, recursive filtering approaches such as the Kalman filter or recursive autoregressive filters can be used for synchronization [291].

In addition, different hypotheses for drawing data samples from sensors may also lead to a mismatch in the sense of learning from different underlying distributions. Therefore, sensor data has to be calibrated (e.g. adjustment of sensor parameters, features, raw data cuts for out-ranging measurements using classification or clustering techniques) [654].

#### 3.2.3.3 Data Representation

As it will be shown in the case of steel production (Section 3.2.4.2), a single run through the process chain can be represented by a set of time series with different lengths and offsets, which may overlap in time, contain different numbers of segments at different levels of granularity, stem from different machines, and sensors and may also be entirely missing for optional processing steps.

Many common data analysis methods cannot work directly on such sets of time series; rather, they require that all observations to be represented by fixed-length feature vectors. Here, we discuss how the raw series values can be transformed to fixed-Length vectors and concatenated into one single representation.

**Mapping of Value Series to a Fixed-Length Vector and Concatenation** The goal is to transform raw sensor data into fixed-length vectors in order to make standard learning techniques applicable. We start by reserving enough space for the records of each sensor in a single fixed-length numerical vector. Original series values are then indexed by predefined positions in this vector. For the mapping, the maximum length of the time series needs to be known beforehand. For the projection, the original time series values might need to be rescaled, e.g., by interpolation. As a result, the application of the most popular distance-based data analysis algorithm becomes possible.

A difficult question is which values to assign to portions where no processing took place. We may simply fill missing portions with zeros or the last recorded value. However, filling with zero values can lead to inaccurate evaluation with several popular distance measures, including Euclidean distance. For example, both series would be marked as highly dissimilar by Euclidean distance, although both blocks (A and B) can have a similar quality. In such a case, the correct mapping between similar feature vectors and similar labels would be altered. Reserving the same portion for both finishing rolls (sensors 5 and 6) in the fixed-length vector seems to solve the problem, but it does not take into account that both finishing rolls might have different properties, e.g., value scales, which usually require a careful data calibration [654]. Instead of transforming all series values to a fixed-length vector, another option would be to use distance measures that can handle value series with different lengths, such as the Dynamic Time Warping (DTW) [461] or the Longest Common Subsequence (LCSS) distance [160]. In principle, two main approaches, for transforming the original time series appropriately, exist. The first approach simply concatenates all time series belonging to the processing of single manufactured pieces (e.g. a single steel block). The second approach consists of calculating distance values for each time series of each sensor independently and then summing them up to a total distance.

#### 3.2.3.4 Feature Extraction and Selection

Instead of using a raw data stream, we can characterize production processes by a devised set of features extracted from raw data. The transformation of the raw data into a feature vector should maximize the prediction accuracy and increases the resource awareness of the machine learning algorithms by reducing the dimensionality and the amount of stored data. (See Chapter 1 in Volume 1.) The challenge in this task consists of the huge search space (i.e. exponential size) over all the possible transformations. Traditional approaches are based on manual feature extraction to build features one at a time using data analysis and domain knowledge. While these approaches are

accessible and interpretable to users, they are tedious, time-consuming, error-prone, and usually not adaptive to data changes. In this context, automatic feature engineering appears to be a promising way to go. It consists of automatically extracting and generating a large number of features and selecting an effective subset of these features to ensure better performance of the machine learning algorithm. Many works have been conducted for automated feature engineering in literature either by automating the whole process [594],[445], or by focusing on particular steps such as feature extraction, feature generation [314] or feature selection [582].

The following sections present a non-exhaustive list of transformation and feature extraction methods that look especially promising in the context of production processes.

**Aggregation and Summarization** Aggregation and summarization methods for streaming time series data reduce the amount of collected data as much as possible and try to retain its most important patterns. The simplest type of aggregation is based on the calculation of summary statistics such as minimum and maximum values, the mean, median, standard deviation, percentiles, and histograms. Such simple features can already encode sufficient information for the learner and sometimes outperform sophisticated methods [654]. More sophisticated methods search other representations of the time series based on time series transformations. These include the Discrete Fourier Transform (DFT) [445] and the Discrete Wavelet Transformation (DWT) [445].

**Segmentation** The salient features approach by Candan, Rossini, Sapino, and Wang [115] transfers ideas from the segmentation of two-dimensional images and the extraction of Scale Invariant Feature Transformation features (SIFT) from images to the space of one-dimensional value series. Salient points in the series, which are points that deviate much from their surrounding values, are used for segmenting the series. Then, from each segment, characterizing features are extracted. The method determines salient points at different resolutions, allowing for a description of value series at different levels of granularity. One segmentation approach was developed in the context of the steel production use case (Section 3.2.4.2), which determines segments based on domain knowledge and signals from machines in the process chain [654]. After segmentation, different statistics are computed on the segments such as the mean, the standard deviation, and the minimum and maximum values. Other features are differences between values and histograms. The biggest advantage is that the approach allows for the combination of many features in a highly interpretable manner [654], since the feature transformation is handled in a multivariate manner in the sense that features from different value series and their parts are combined together. For example, a classification rule that is formed based on such features may be: “Predict a steel block as defective if it was heated for less than one hour at 900 degrees and if the maximum rolling force in the first rolling step exceeds the value of 10 000”. The approach has

already been used successfully for the identification of coarse-grained patterns such as processing modes (see Section 3.2.4.2).

**Symbolic Representation** SAX (Symbolic Aggregate Approximation) [413] first determines the elements of a sequence $C = (c_1; \cdots ; c_n)$ by piece-wise aggregate approximation and maps them to a new sequence $C^{'}$ with $w < n$:

$$c_i^{'} = \frac{n}{w} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j \tag{3.1}$$

The elements $c_i^{'}$ are then discretized by mapping them to a fixed number of symbols, keeping the upper bounded Euclidean distance between all series. A gradient-based approach for the symbolization of streaming sensor data was introduced by Morik and Wessel [458]. This approach was originally applied in the context of text mining and has been successfully used in areas such as text classification or intrusion detection.

**Method Trees** All of the aforementioned methods, with different parameter settings and combinations, are useful for extracting functional features from value series. Instead of trying and combining all the existing methods and different parameter values manually, Mierswa and Morik [445] developed an automatic representation learning method that optimizes the composition of a representation for best classification learning. Basis transformations, filters, mark-ups, and a generalized windowing present elementary methods that are combined in the form of a method tree. The tree applies the operators (nodes) in a breadth-first manner to transform the original value series. The root of each tree consists of a windowing function, while the children of each parent node are operator chains representing basis transformations, filters, and a finishing function. Learning the feature extraction tree is performed by a genetic programming algorithm. The method can be used for the analysis of time series from production processes. However, its demand for stratified datasets with respect to the labels is not always met by the production data. The method had been implemented within RapidMiner.

#### 3.2.3.5 Modeling

The following sections give a short overview of widely used methods that are assumed to be of special relevance for the embedded data analysis in production processes.

#### 3.2.3.6 Supervised Learning

The most popular task in the context of Supervised Learning is inferring functions representing the relationship between a set of explanatory variables (i.e. features) and a response target variable which can be discrete (i.e. classification) or continuous (i.e. regression), also known as *function learning*. Let $X$ be a set of possible explanatory

variables, $Y$ be a set of possible target values and $D$ an unknown probability distribution on $X, Y$. Let further $H$ be a set of possible functions. Given examples$(x, y) \in X \times Y$, drawn from $D$, where $y = f(x)$ for an unknown function $f$, the goal is to find a function $h \in H : X \to Y$, such that the error $er_D(h, f)$ is minimized.

In the case of a production process, explanatory variables are feature extracted from sensor measurements (i.e. records for a given steel block) and $y \in Y$ is the corresponding quality label to be predicted. Examples of learning functions $h \in H$ include Decision Trees, K-Nearest Neighbour (kNN), Naive Bayes and, Support Vector Machines (SVM). For a detailed in-depth overview of such methods, see Hastie et al. [260].

The aforementioned methods assume that all training data is available in batches. Hence, they cannot be trained during the manufacturing process, and their models require to be retrained if concepts change. In case of the absence of concept drifts, models can be trained offline but deployed online for the prediction.

#### 3.2.3.7 Unsupervised Learning

If no labels are provided, unsupervised learning methods may be employed to reveal the most prominent patterns in the data. Cluster analysis [300] tries to group observations, following a similarity measure. The number of clusters $k$ is usually a user-defined hyper-parameter. A well-known clustering algorithm is k-means [426]. Dimensionality reduction techniques such as principal component analysis (PCA), aim at simplifying high-dimensional datasets. Some of them may be used for better data visualization, like SOMs [654], which map high-dimensional input vectors to a low-dimensional grid. Vectors that are similar to each other in the input space lie close to each other on the grid. SOMs have also been used for analyzing the data in the steel production case study (see Section 3.2.4.2). The biggest disadvantage of unsupervised methods is that, without any labeled data, their results can only be validated by domain experts.

#### 3.2.3.8 Learning in the Presence of Class Imbalance

Class imbalance occurs when data classes are not equally frequent. Generally, it occurs when some classes represent rare events, while the other classes represent the counterpart of these events. Rare events, especially those that may have a negative impact, often require informed and prompt decision-making. However, the class imbalance is known to induce a learning bias towards majority classes, which implies a poor detection of minority classes. For example, production processes with high-quality standards usually output more high-quality goods. Similarly, certain events, like machine or sensor failures, may only occur rarely. In such cases, many positive examples but only a few or even no examples of the negative class are available. Measuring class imbalance performance in classification tasks based on the accuracy leads to the problem that the metric is biased towards the majority class. Class imbalance can be mitigated using different methods including one-class learning, class rebalancing methods, and ensemble learning.

**One Class Learning** One way of handling class imbalance is to treat minority class instances as outliers or anomalies and majority class instances as the normal class using the task of one class learning [460]. Tax and Duin [675] propose a Support Vector Data Description (SVDD) that computes a spherical boundary around the given data points. The diameter of the enclosing ball and thereby the volume of the training data falling within the ball can be chosen by the user. Observations inside the ball are then classified as normal whereas those outside the ball are treated as outliers or anomalies. Schölkopf et al. [581] have proposed the 1-Class SVM, which separates all training examples with a maximum margin from the origin. An active approach to such data domain descriptions generalizes this approach [238].

**Class Rebalancing Methods** Several methods have been proposed to handle the class imbalance problem using resampling of the data [563]. Resampling strategies rebalance the data in order to mitigate the effect of the bias of machine learning models towards the majority class [125]. Resampling methods are considered to be flexible and are widely used since they are independent of the selected classifier. The class imbalance problem can also be solved using algorithmic modifications of existing machine learning classifiers, e.g., support vector machines, k-nearest neighbors, or neural networks. Modifications can be introduced by enhancing the discriminatory power of the classifiers towards the minority class using kernel transformation to increase the separability of the original training space [219].

Cost-sensitive learning can also be applied in the context of class imbalance by modifying the loss functions to increase miss-classification costs of the minority samples [322].

**Learning Ensembles in the Presence of Class Imbalance** Combining classifiers in ensemble frameworks is another common approach to handle the class imbalance problem [563]. Within ensemble-based classifiers, we can distinguish four main families. The first family includes resampling based ensembles. An ensemble of classifiers is created after training base classifiers on balanced datasets obtained with a resampling technique. In the second family, the ensemble is built based on boosting [576] after applying a data resampling strategy (e.g. SMOTEBoost [126]). Within the third family, we find Bagging-based ensembles [101] (e.g. UnderBagging, OverBagging, SMOTEBagging [715]).

Recently, we proposed a probabilistic ensemble method to handle the class imbalance explicitly at training time [563]. Unlike existing ensemble methods for class imbalance, which use either data-driven or randomized approaches for their construction, our method leverages both directions. On the one hand, ensemble members are constructed from randomized subsets of training data. On the other hand, we design different scenarios of class imbalance for the unknown test data. For each of the resulting scenarios, an ensemble is obtained by combining random sampling with an

estimate of the relative importance of specific loss functions. The final predictions are generated by computing a weighted average over the individual ensemble predictions. In contrast to existing methods, this approach does not attempt to correct imbalanced datasets. Instead, it has been shown how imbalanced data sets can facilitate classification, given a limited range of true class frequencies. This method promotes diversity among ensemble members and is insensitive to certain parameter settings.

#### 3.2.3.9 Stream Mining

To cope with real-time evolving production systems, every step of the analysis should be applied and handled in an online fashion. Whereas learning an effective representation needs to be run offline as shown above with model trees, using the learned extraction must be applied online. Many applications demand the transformation of the raw sensor data, the deployment of the model, and sometimes even the model update and the detection of novel events being performed online. Change detection is usually done by updating the model with every new arriving data item or learning a model on a batch of the most recent data items [218].

The biggest challenge in applying classical stream mining settings to production processes is, that the labels are usually measured at the end of the process chain. Hence, at each point in time we predict the final quality. For the time series starting with $t_{i0}$ and ending with $tie$, we have the observations $x_{ti0}, ...x_{tia}, ...x_{tie}$, but only the final quality measurement $y_{tie}$. Hence, at each observation point we predict the final quality. The model $f$ uses $p$ extracted features from $x \in X$ (i.e. $f(\phi_1(x), \cdots, \phi_p(x)) = y$). Assume $t_{i0}$ to be the starting time of the process $p_i$, $t_{ie}$ its end, and $t_{ia}$ the actual time instant, one possible approach is to segment the time series and learn an individual model on every single segment to predict the final quality. This approach is useful when important events can not be identified in time and memory constraints are imposed, i.e., training models on time series subsequences instead of considering all the historical data. If it is possible to identify important events within the corresponding step of the process occurring at a given time instant $t_{ic}$, then every feature extracted from the segment $[t_{i0}, t_{ic}]$ can be used to learn a model. It is therefore possible to generate the first prediction of the label if $t_{ia} > t_{ic}$.

A second approach consists of using a combination of static and statistical features, like the minimum, maximum, or average of the time series. The static data won't change over the complete process and the probability of change of the statistical features will decrease to the end of the process. That means, that there exists an index $t_{is}$, where the prediction error is bounded by $\left(\hat{f}(\phi_1([t_{i0}, t_{is}]), \cdots, \phi_p([t_{i0}, t_{is}])) - y\right)^2 \leq \xi$. Another approach would be to use one or a set of algorithms, that predict the set of features for the unseen part of the time series $[t_{ia}, t_{ie}]$ and train a model on the full feature set to predict the quality label. The overall prediction will therefore be strongly dependent on the accuracy level of the feature prediction.

#### 3.2.3.10 Online Management of Many Models

As it is mentioned in stream mining, machine learning models can be trained offline and deployed in real time to generate quality predictions. It may also happen that the stream-mining setting requires the training of many models on different parts of the data, e.g., different segments of the time series (see Section 3.2.3.9). In addition, the increasing individualization of products and processes may result in small heterogeneous groups of observations. Learning distinct models is then necessary to represent each group of observations and capture its underlying properties. Therefore, efficient online management of many models should be established through the dynamic combination of many models built to comply with detected changes in the data by adaptively changing their combination and integration rules in real time. The dynamic combination can formally be established with adaptive ensemble methods that are built as a weighted combination of distributions characterizing the target concepts and enabling flexible management of the models from the individual model selection to the weighted aggregation of many models [561, 562, 564, 565]. The application of adaptive ensemble methods is made in connection with concept changes detection in the data that enable the ensemble update on different levels (i.e. base models selection [561], informed base models/ ensemble parameters adaption (i.e. after a detection of a concept drift) [562]).

In this context, we have proposed an adaptive ensemble selection framework that manages online two main ensemble construction stages: pruning and integration [564]. Since the performance of ensemble-based models changes over time, it is also considered to be subject to concept drifts. A drift detection mechanism is employed to exclude models whose performance becomes significantly worse compared with the remaining models and to identify the top base models in terms of performance. Performance is evaluated in this context using a custom measure based on the Pearson Correlation (i.e. commonly used to deal with time series data between base models forecast and target time series on a sliding window validation set. After each drift detection, top base models are identified. Since diversity is a fundamental component in ensemble methods, we perform a second stage selection through clustering model outputs. Clusters and top base models are updated after each drift detection (i.e. whenever an alarm is triggered by our drift detection). At each cluster computation, the models that belong to the cluster representatives are selected. In a final step, the selected models' outputs are combined together using a sliding-window weighted average.

We have also developed a framework for online ensemble aggregation using deep reinforcement learning for time series forecasting [562, 565]. There, we leverage a deep reinforcement learning framework for learning linearly weighted ensembles as a meta-learning method. In this framework, the combination policy in ensembles is modeled as a sequential decision-making process and an actor-critic model, that aims at learning the optimal weights in a continuous action space, is used. The policy is updated following a drift detection mechanism for tracking performance shifts of the ensemble model over time [562].

These frameworks were developed initially to manage forecasting models and can be applied to predicting continuous quality-related measures over time, but can also be transferred to classification models operating in streaming environments.

### 3.2.4 Case Studies

The following case studies highlight different facets of the introduced theoretical framework and give a decisive impression of how the described methods can be applied in practice, in order to solve quality-related problems. However, not all concepts have been validated in the particular industrial environment. In the SMT manufacturing use case, the steps of data acquisition and storage, learning in the presence of class imbalance as well as feature extraction and modeling are explained. Lastly, the process optimization in the SMT line is described. In the hot rolling mill process, the data acquisition and storage are highlighted, since the processing of the sensory data was relatively challenging due to incomplete sensory measurements and other factors. Also, the feature extraction and modeling phase are explained.

#### 3.2.4.1 Model-Based Quality Prediction in Electronics Manufacturing

The case study of electronics production covers the production of programmable logic controllers of the Simatic type. At the end of the soldering process, the correct position of soldered components is checked. This is accomplished by an Automatic Optical Inspection (AOI) for variants with visible connection points (pins) and by an X-ray inspection for variants with the pins underneath the components from two different perspectives ($X_1$ and $X_2$). The Printed Circuit Boards (PCB) are placed on a panel and are tested in a pool test by placing 48 PCBs in eight Fields Of View (FOV). The number of pins on each PCB are 79 and 52 for $X_1$ and $X_2$ directions, respectively. A typical PCB board is illustrated in Figure 3.6 to showcase one typical product variant that is manufactured using the SMT technology. Due to the long inspection time as well as the large number of units produced, X-ray inspection is a bottleneck in the production of PCBs, especially when considering the 100 % inspection, i.e. the testing of every PCB, that is conducted and the constantly growing demand for programmable logic controllers [579].

For this case study, the focus has been narrowed to one product variant, its respective manufacturing line, and the data sources of SPI and X-ray inspection.

**Data Acquisition and Storage** Historic datasets from Serial Peripheral Interface (SPI) and X-ray are matched by different manufacturing databases via a unique identifier. Resolving the use case at the pin level, where each pin is assigned a quality label, is a low-dimensional machine learning task as was shown by [580] since only seven different features are mapped to one categorical class label. However, from a process perspective,

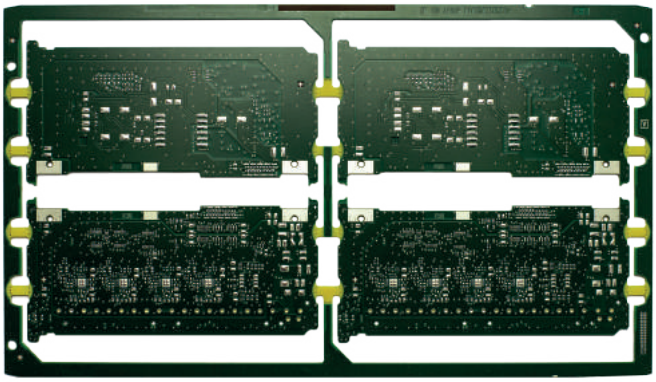


**Fig. 3.6:** The image illustrates a PCB board from the $X_1$ direction that is manufactured using SMT.

the quality test is not feasible on a pin level and the process data is aggregated to the FOV level. The dimensionality of the process data thereby increases dramatically. The considered dataset consists of numeric SPI features (see Table 3.1) and a binary X-ray label on the aggregation level of FOVs, which is formalized as a binary classification task. The features characterizing each pin are summarized in Table 3.1.

**Tab. 3.1:** Descriptive PCB features on the pin level.

| SPI feature | Unit |
|---|---|
| Height | % |
| Shape 2D | % |
| Shape 3D | % |
| Surface | % |
| Volume | % |
| Offset X | $\mu m$ |
| Offset Y | $\mu m$ |

Historical datasets for a period of five production months are used for the case study. In total, 1 461 037 321 data points are parsed, of which 800 parts per million (ppm) are NOk.

Non-representative datasets, which, for example, are recorded under obsolete process configurations or during manufacturing trials, are eliminated using expert knowledge. The result is a prepared and cleaned training dataset for subsequent modeling, including a unique identifier, all relevant features and the quality label, which can be continuous or discrete depending on the applied measurement method.

**Learning in the Presence of Class Imbalance** The considered use case constitutes an extraordinary case of class imbalance. When using no specific countermeasures, the classifier degenerates due to frequent misclassifications of the minority class. Classifying the defect class as OK is often unacceptable in an industrial environment. Therefore, specific measurements are chosen in order to increase the performance of the algorithm on the NOk class. To rebalance the dataset, a combination of different resampling techniques are examined, such as oversampling (SMOTE, random oversampling) and undersampling (random undersampling). Furthermore, the output class-membership probability threshold is systematically tuned in order to achieve a better trade-off between both the precision and recall of the classifier.

**Feature Extraction and Model Learning** Initially, the PCB process data was described on the pin level. Process expertise led to the conclusion that aggregation to the FOV level is necessary so that each FOV consisted of either 79 pins in the $X_1$ direction, or 52 pins in the $X_2$ direction with seven features for each pin. While [580] results on the FOV level were not promising, further experiments are conducted. One direction is to reduce the dimensionality using automatic feature extraction methods in combination with machine learning models. For this we use the Rapidminer extension Value Series Plugin. This reduces the dimensionality of the process data for $X_1$ from 553 features to 240 features and from 364 features to 50 features for $X_2$.

**Model Evaluation** The training of the models takes place in a nested structure of inner and outer cross validation and hyper-parameter optimization. As the a priori selection of adequate algorithms is not achievable in a generalized way, different learning methods and algorithms are tested and evaluated for each individual application [579], including Gradient Boosting Trees (GBT), Random Forest (RF), and Multilayer Perceptron (MLP), which is based on fully connected neurons that compute a complete weighted sum in the affine transformation stage of the connections. These methods are used as a baseline for the experiments. Opposed to a fully connected neuron, a 1-Dimensional Convolutional Neural Network (1D-CNN) is based on the principle of weight sharing on all connection units. For example, in the discrete one-dimensional (1D) case: $(d \star K)(x) = \sum_a I(x - a)K(a)$ is a convolution of a data sample $d$ with the kernel (or filter) $K$ [767]. A 1D-CNN with a kernel size of $k = 3$ is used in order to integrate knowledge from the direct neighborhood of pins in question into the learning process. Two convolutions are stacked and, additionally, dropout with a rate of $d = 0.5$ is used for regularization. The most discriminative features are obtained by combining max pooling and two consecutive dense layers. The best performing rebalancing technique on this use case is random oversampling which is applied before feeding the data to the classifiers. The summarized results can be found in Table 3.2.

The model results indicate that the overall performance of the classifiers is modest when it comes to overall correctness. The GBT model shows the overall best performance,

**Tab. 3.2:** Cross-validated classification results.

| Metric | Recall(NOk) | Recall(Ok) | Accuracy |
|---|---|---|---|
| RF | 69.83 % | 53.76 % | 73.46 % |
| GBT | 73.98 % | 39.70 % | 43.26 % |
| MLP | 16.08 % | **95.13 %** | 80.15 % |
| 1D-CNN | **75.4 %** | 43.85 % | **74.71 %** |

but the difficulty in this use case is the high-class imbalance rate. It appears that the distribution of the NOk class is not accurately represented by the classifiers or only at the cost of a lower recall on the Ok class. However, from a practical perspective, the 1D CNN can be used in order to reduce the testing effort of the X-ray machine, which can save up to 75.4 % of testing volume, which is shown in Table 3.2 as the recall of the NOk class. Surprisingly, the experiments using feature selection methods do not improve the performance as we expected even though the dimensionality is reduced. It appears that in this use case, every individual pin has to be considered, summarizing the data through statistic features leads to an information loss. Lastly, since deep learning models usually lack interpretability, it is not clear which discriminative features led the classifier to draw its conclusion. By contrast, tree-based methods offer more interpretability. Therefore, the decision for a specific model is also a question of its accuracy and to some extent its interpretability by domain experts.

**Process Optimization** The model deployment is achieved through organizational integration into the inspection planning process. As described by [580], the models were trained on a cloud infrastructure and stored on an edge device close to the process to reduce latency and bandwidth issues. Here, the inspection strategy determines the role of the model with respect to inspection planning and design. While an inspection, exclusively based on the prediction model, requires high confidence in the model and extremely high model accuracy to reach or exceed the level of conventional inspection principles, hybrid approaches seem promising for the current state of development. The inspection reliability is given by the combination of quality prediction and conventional inspection. The introduction of quality prediction in quality assurance can facilitate the generation of additional added value by reducing physical inspection volume without sacrificing inspection reliability. Two different strategies can be deduced depending on the trustworthiness of the model. Either only those parts are subjected to a physical inspection whose prediction result was Ok or whose result was NOk. As the class imbalance of datasets in the quality context is usually quite high, selecting only NOk predicted parts to undergo the physical inspection offers vastly superior potential savings.

### 3.2.4.2 Real-Time Quality Prediction in a Hot Rolling Mill Process

In the hot rolling mill case study, steel blocks move through a process chain to become bars. The process chain is shown in Figure 3.7. First, the blocks are heated for 15 hours in five different heating zones of a furnace. They are then rolled at the block roll and the first finishing roll. The rolling in the second roll is optional.

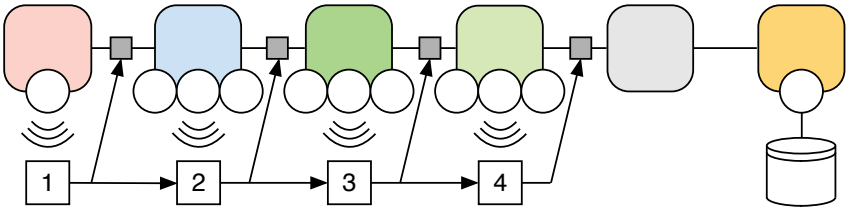



**Fig. 3.7:** Sensor measurements for two different blocks. ©[2016] Springer. Reprinted, with permission, from [654].

Each block usually moves over a single roll several times, where the number of rolling steps, each taking a few seconds, is determined. The blocks are finally cut into smaller bars whose quality is assessed using an ultrasonic test, several days later. Online measurements on how the blocks are processed are provided by sensors installed along the production chain. The sensors measure various physical qualities including the air temperature, rolling force, rolling speed, and the height of the roll, with 10 Hz. The ultrasonic test results indicate the amount of material containing defects for each bar. It is impossible to assess the physical quality of hot steel blocks or smaller bars at intermediate steps of the process chain. The blocks must cool down before their final quality can be tested in an ultrasonic test. Quality deviations are assessed on specific internal quality parameters such as location, manifestation, and frequency of core and border displays [404]. In cases where some of the blocks are, for example, wrongly heated, energy, material, and human workforce are wasted if they nevertheless continue the process chain. Therefore, the goal of the case study is to identify quality-related patterns in the sensor data, and to predict the final quality of steel blocks in real-time, hence detecting NOk blocks as early as possible. Energy savings can be achieved if, following the quality prediction, blocks with estimated defects are sorted out from the process early enough or reinserted again in previous steps. In addition, it may happen that the required final quality can be reached by adjusting the parameters of subsequent processing stations [339].

In the following, we describe in more detail which sensor measurements are recorded and how they are stored, preprocessed, and analyzed in the context of the given case study. See also the Section 3.3, which presents a novel learning method inspired by exactly this use case.

**Data Acquisition and Storage** During the period of one year, over one billion measurements from 30 different sensor types are collected during the processing of about 10 000 steel blocks, together with the corresponding quality information. Among the readings are the air temperature for each furnace zone, rolling speed, force, position, and temperature. Domain experts consider these to be the most relevant quality-related parameters. For validation purposes and guaranteeing the reproducibility of results, all data has been stored in a single SQL database. A Java tool was developed for reading and importing the raw data delivered in different files and formats. Once imported, sensor measurements can be exported based on filters written in SQL to CSV files, that contain all measurements recorded by a particular sensor during the processing of a single steel block. For the preprocessing of time series data in production environments, a highly modular process has been developed and implemented with Rapidminer [405, 654]. The following sections provide a summary of the procedure and results already presented in [404, 654]. At first, all value series are cleansed by cutting away irrelevant parts where no processing happened, as discussed in Section 3.2.3.2. In addition, data measurements that lie outside meaningful ranges are marked as outliers and replaced by their predecessor value.

Afterwards, the time series are segmented based on background knowledge as described in Section 3.2.3.3. In the case of the heating furnace, for instance, the five different heating zones make up natural borders for the segments. Similarly, individual rolling steps are natural divisions for all series stemming from the three different rolls.

**Feature Extraction and Model Learning** Each segment in the time series is described by several statistics, and mapped to portions of a fixed-length vector. The 60 000 raw series values recorded for each steel block are aggregated to about 2000 features. The resulting dataset can then be fed to common feature selection and learning algorithms. For 470 processes, the mapping of the resulting cut bars to the steel blocks could be established. The feature vectors of these processes have then been used for comparing diverse machine learning methods: Naive Bayes, Decision Trees, k-NN, and the SVM [404, 654]. It has been shown that including features about the individual segments decreases accuracy in comparison to including global information about the value series and segments. Features of individual segments were therefore excluded for the following analysis, resulting in 218 remaining features.

**Model Evaluation** Even with extracted and selected features, none of the classifiers mentioned is able to reach a better prediction accuracy than the baseline, which predicts the majority label. For getting a better impression of the data, the feature vectors were mapped to a two-dimensional Self-Organizing Maps (SOM) (Figure 3.8) where points close to each other have similar features (see Section 3.2.3.4). The shading is used to indicate a weighted distance between the points, where lighter shades represent larger distances. In the SOM on the left-hand side, the points represent the feature vectors of

production processes and their colors indicate the final quality of the resulting steel bars as Ok (red) and NOk (blue). In many cases, NOk bars are very close to Ok ones (see also the zoomed area in Figure 3.8), which means they have highly similar features. As a result, the features extracted so far do not suffice to correctly classify low and high-quality processes.

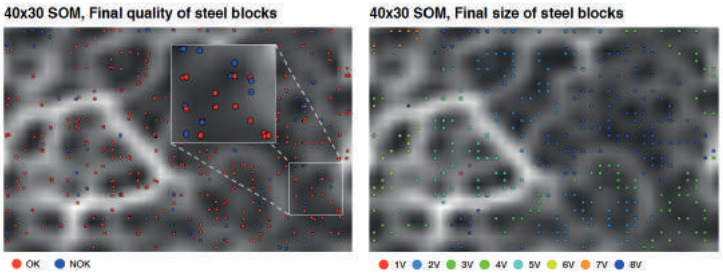



**Fig. 3.8:** Similarity relationships between feature vectors. ©[2016] Springer. Reprinted, with permission, from [654].

In comparison, the SOM on the right-hand side of Figure 3.8 shows the final size of the produced bars. As it seems, the extracted features are highly correlated with distinct operational production modes for the different bar sizes. The hypothesis could be verified by training a decision tree on features of the first finishing roll. The accuracy as estimated by a 10-fold cross-validation is 90 %, while k-NN ($k = 11$) even achieves 97 %. Most important for the decision is the position of the roll (sensor 501). This indicates the height of the roll and naturally implies the pressure on the block. This correlates with the size of blocks after milling, expressed by $1V$, $2V$, .... Domain experts have verified that the results reflect the real operational model in the rolling mill. That the features are correlated with distinct operational modes and not the quality could mean that large absolute quantitative differences between the modes, i.e. the global patterns, overshadow local patterns. The clustering with SOMs thus gives important hints for improving the quality prediction. For instance, in the future, separate models for the modes could be trained, more scale-invariant features could be extracted, or the value series could be better normalized.

As the results demonstrate, data analysis methods are capable of detecting meaningful patterns in production processes. Even though identifying the exact features that are relevant for the prediction task is sometimes not straightforward, as already discussed in Section 3.2.3.4, each insight into the data can be useful for providing new ideas for better feature extraction and process understanding that can be validated afterward by interaction with domain experts.

### 3.2.5 Conclusion and Future Outlook

This contribution gave an overview of data mining for Industry 4.0. The first sections offer a general guideline for applications similar to ours. In Section 3.2.2, we present the need for intermediate quality prediction in order to reduce material and energy consumption and how it can be integrated into an IMPC module in order to prepare factories for the adequate use of data mining techniques in a quality-related context. Carefully going through all the steps of the data-mining process in Section 3.2.3, we present a large variety of methods for each step. The contribution presents two real-world case studies in Section 3.2.4: quality prediction in electronics manufacturing (Section 3.2.4.1) and in a hot rolling mill process (Section 3.2.4.2). Both use cases demonstrate the benefits of real-time embedded data analysis in the production chain for quality prediction.

A new application of multiclass time series classification predicts the quality of a bolt in a real-world automotive industry use case. Shorter subsequences of time series were determined that already allowed to train a model achieving high recall and F-measure (both 97 %) for almost all of the 8 classes except for the two that were only present in 1 % or 5 % of the data. Detecting the quality as early as possible enables to do corrective actions, thus avoiding costly rework and waste of resources through further processing of defective components. Moreover, anticipating the type of defect helps to estimate the reworking time to correct it, which varies from 5 seconds to 5 hours [557].

Another more recent study is about saving quality testing efforts in surface mount technology. Since it would take too much time to assess the quality at pin level, the quality information of the panels is aggregated at a FOV level, which corresponds to the aggregation level of the X-ray inspection. One FOV consists of 6 PCB and is denoted as "NOk" if one PCB is detected as defective, whereas it is declared as "Ok" when all PCBs are defect-free. In addition to excellent recall and accuracy, the trained model was explained using a heat map [558]. This is an important step into the interpretability of learned models, but further work is needed.

Despite several success stories, there are still limitations that slow down the application of the developed machine learning-based solutions in the industry. The first of the two most-important obstacles that are still in the way of machine learning adoption in real environments is the lack of reliable labeled data in many manufacturing scenarios. Data gathering, data fusion from heterogeneous sources, and data cleaning require ongoing efforts. The second is that the internal organization of companies needs to integrate computer scientists with a qualification in machine learning into the higher levels of engineering departments. The social integration of employees with diverse backgrounds not only at the board of directors is a challenging task, but it is necessary to benefit from the full potential of machine learning.

## 3.3 Label Proportion Learning

*Marco Stolpe*
*Katharina Morik*

**Abstract:** For the interlinked process of producing steel bars, quality information is given in statistical form for whole charges of blocks, i.e. we know the fraction of blocks which had quality related problems. This poses a new kind of problem for machine learning, since almost all supervised learning methods assume to be given labels for individual observations instead of groups. The problem of learning from fractions of labels has become known as the problem of learning from label proportions. In this contribution, the learning problem and existing methods for solving it are introduced, as well as a clustering-based method called *Learning from Label Proportions by Clustering* (LLPC) developed in the context of project B3. It is demonstrated that LLPC outperforms methods that were considered the state of the art at that time in terms of prediction and runtime performance. Moreover, the relation to resource-constrained learning settings such as distributed learning is shown.

### 3.3.1 Introduction

In smart manufacturing, it can be difficult to track products through the whole process chain. For instance, in the interlinked production process of hot rolling, the temperature of steel blocks is so high that they cannot be stamped or equipped with RFID chips. Once cut to smaller rods, tracking object identity, i.e. which rods belonged to which block in which customer order (or batch), can therefore become a big technical challenge. In the steel-rolling scenario, quality labels are usually given as percentages for whole batches, but not for individual blocks. This is similar to other industries, in which due to cost reasons only the quality of a small sample is checked. Depending on the estimated fraction of faulty products, either the whole batch needs to be thrown away, checked again, or it is accepted that a very small fraction of faulty products is delivered to the customer. An interesting question is if we can check only a sample, but nevertheless derive information about the properties of individual products. Moreover, can we sort out some rods already during the production process itself, before the quality check? To put it in more general terms: can we derive a model that assigns correct labels to individual products based on their properties, if we are only given the proportions of each label for different batches of products?

The aforementioned problem of *learning from label proportions* (LLP) not only has applications in industry, but in application areas as diverse as privacy-preserving data mining, election forecasting, bank customer classification, bankruptcy predic-

tion, or marine litter beaching prediction. Especially relevant for us is its relationship to resource-constrained distributed learning settings. Distributed machine learning receives subsets of the overall data in two ways, horizontally and vertically. Horizontally partitioned data are separating sets of observations with all their features, for instance, the sales of different shops each offering the same items. By contrast, vertical partitioning means that each location senses a different subset of features. This is a common scenario in the Internet of Things [653]. Our steel rolling scenario is of this kind. It also appears, for instance, in traffic prediction problems (see also Section 4.1). Here, we might like to reduce communication costs by transmitting only aggregated label information between nodes. Again, the question is, if we can learn a model that is sufficiently accurate in assigning class labels to individual instances, based only on aggregated label information.

The problem of LLP not only deviates from that of supervised learning, where we learn from individually labeled training examples, but also from many other learning settings known in machine learning and data mining. It is different from *semi-supervised learning* [120], where we are given at least some examples that are labeled. It is not strictly *unsupervised learning*, since we are given at least *some* additional information about labels. It is different from *anomaly and outlier detection*, where we might know about observations that belong to a normal class. It comes close to *multiple instance learning* [731], where whole bags of observations are either labeled as positive or negative. However, LLP is not exactly the same problem, since we are not given binary information on each bag, but real-valued statistical information about the labels in each bag.

In Section 3.3.2, the task of LLP is defined more formally. Then, Section 3.3.3 gives an overview of related work. In Section 3.3.4, we discuss the difficulty of the problem from a Bayesian perspective. Afterwards, Section 3.3.5 defines loss functions for the scenario. Section 3.3.6 introduces a clustering approach and variants that minimize aforementioned loss functions. In Section 3.3.7, we compare the algorithm's prediction performance and runtime to other existing methods. Finally, Section 3.3.8 concludes and gives a short summary.

### 3.3.2 The Problem of Learning from Label Proportions

To the best of our knowledge, Musicant et al. were the first who formally formulated both the classification and regression tasks of the problem [466]. We extend their problem definition to multi-class problems and relate it to the unknown joint distribution $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ from which all observations and labels are drawn.

**Definition 1** (Learning from label proportions (LLP))**.** Let $\mathcal{X}$ be an instance space and $\mathcal{Y}$ be a space of categorical class labels $Y_1, \ldots, Y_l$. Let $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ be an unknown joint distribution on the instances and class labels. In the setting of *learning from label*

**Labeled examples (unknown)**

$B_1 = \{(x_1, 1), (x_3, 1), (x_7, 0)\}$
$B_2 = \{(x_2, 0), (x_4, 0), (x_5, 1), (x_6, 1)\}$
$B_3 = \{(x_8, 0), (x_9, 0)\}$

**Unlabeled examples (known)**

$B_1 = \{x_1, x_3, x_7\}$ $n = 9$
$B_2 = \{x_2, x_4, x_5, x_6\}$ $h = 3$
$B_3 = \{x_8, x_9\}$ $l = 2$

**Label proportions (known)**

$Y = \{0, 1\}$

$$\begin{array}{cc} y=0 & y=1 \end{array}$$

$$\Pi = \begin{pmatrix} 0.33 & 0.67 \\ 0.50 & 0.50 \\ 1.00 & 0.00 \end{pmatrix} \begin{array}{l} |B_1| = 3 \\ |B_2| = 4 \\ |B_3| = 2 \end{array}$$

$\eta$ 0.56 0.44

**Fig. 3.9:** Example for given bags of observations, a label proportion matrix, and related notations

*proportions*, we are given a sample of $N$ unlabeled observations $X = \langle x_1, \ldots, x_N \rangle$ with $X \subseteq \mathcal{X}$, drawn i.i.d. from $\mathbb{P}(\mathcal{X})$. Let $y_i \in Y$ be the individual class label for observation $x_i$, where $Y \subseteq \mathcal{Y}$. The individual labels are unknown. Instead, we are given a partitioning of $X$ into $h$ disjunct bags $B_1, \ldots, B_h$ and for each bag $B_u$ and label $Y_v$, we are given the proportion $\pi_{uv} \in [0, 1]$ of observations with that label in bag $B_u$. Only based on this information, we seek a function (model) $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that predicts the label $y \in \mathcal{Y}$ for an observation $x \in \mathcal{X}$ drawn i.i.d. from $\mathbb{P}$, such that the expected risk

$$R_{\text{exp}} = \int \ell(\mathcal{Y}, \hat{f}(\mathcal{X})) d\mathbb{P}(\mathcal{X}, \mathcal{Y})$$

is minimized. Here, $\ell$ is a convex *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+$ which measures the cost of assigning the wrong label to individual observations.

The given label proportions $\pi_{uv}$ can more conveniently be written as a $h \times l$ matrix $\Pi = (\pi_{uv})$, where the values in a row $\Pi_{u,\cdot} = (\pi_{u1}, \ldots, \pi_{ul})$ sum up to one. The frequency count $\mu_{uv}$ of observations with label $Y_v \in \mathcal{Y}$ in bag $B_u$ can easily be reconstructed by multiplying the label proportion $\pi_{uv}$ with bag size $|B_u|$.

The proportion $\eta(\Pi, Y_v)$ of label $Y_v$ over the whole sample can then be calculated from $\Pi$ as the sum of the frequency counts for bag $u$, divided by the total number of observations $N$:

$$\eta(\Pi, Y_v) = \frac{1}{N} \sum_{u=1}^{h} \mu_{uv} \,. \tag{3.2}$$

Figure 3.9 gives an example of the notations previously introduced, the division of observations into disjunct bags, and the label proportion matrix as derived from the original (now unknown) labels.

### 3.3.3 Related Work

When starting the work on LLP in 2010, only a few publications on the topic were available. In the following, we'll discuss related work insofar as it has been relevant for comparison with the clustering approach we published at that time [651] [652] and which is going to be presented in this contribution. Since its publication, our approach has been cited more than 70 times according to Google Scholar and even more works on the problem setting have appeared. For instance, different clustering methods have been applied [154]. Fish and Reyzin cast light on the theoretical properties of the problem in the context of probably approximately correct (PAC) learning [209] (for an easy introduction into PAC learning see Mitchell [447]). Saket et al. present an approach that does not rely on the underlying distributions of the bags, and give some guarantees for any learner [571]. However, they do not cover multiclass learning as we do here. Kobayashi and colleagues show estimation bounds for multiclass LLP [333]. A probabilistic method for LLP is developed to estimate the individual votes during presidential US elections [660]. Also new applications are dealt with such as bank customer classification [513], marine litter beaching prediction [273], and bankruptcy prediction [133].

**Related Semi-Supervised Methods** There are some approaches that seem similar to the scenario of LLP, but are actually semi-supervised learning tasks. For instance, Dara et al. first cluster the given data with SOMs and then label the resulting clusters [159]. However, labeled observations are given, which are usually not available when learning from label proportions. Demiriz et al. adapt the k-means optimization problem to respect labeled data [167]. Again, this is a semi-supervised setting, with labeled observations.

**Basic Methods** To the best of our knowledge, Kueck and Freitas were the first who introduced the problem of LLP by proposing a probabilistic model based on group statistics that is trained by an efficient Markov-Chain-Monte-Carlo (MCMC) sampling algorithm [357]. Musicant et al. were the first who defined the problem of learning from aggregate values for regression and classification tasks in a more formal way [466]. They modify well-known methods such as k-NN [13], backpropagation neural networks [447] and the linear SVM [700] to respect the given label proportions. Their experimental results focus on regression tasks, while we are mainly interested in classification.

**Mean Map and Laplacian Mean Map** The Mean Map method we use for comparisons in Section 3.3.7 has been proposed by Quadrianto et al. [514]. It estimates the conditional class probability $\mathbb{P}(\mathcal{Y}|\mathcal{X}, \vec{\theta})$ by conditional exponential models, using a joint feature map $\phi$ that maps observations and labels into a new feature space. The parameters $\vec{\theta}$ can be estimated by solving a convex maximization problem for the conditional

log-likelihood. The conditional log-likelihood in turn can be expressed in terms of the so-called *mean operator*, which can be expanded into bag-wise and label-wise components. The unknowns in this formulation can then be found by solving a system of linear equations, without knowing the individual labels. This is possible by making a *homogeneity assumption*, which states the conditional independence of feature vectors from bags, given the label. Once the mean operator is estimated, the parameter vector $\vec{\theta}$ can be derived by standard methods for maximum likelihood estimation. It is shown that Mean Map outperforms kernel density estimation, discriminative sorting, and MCMC [357].

Patrini et al. relax Mean Map's restrictive homogeneity assumption such that whenever bags are similar to each other, it is assumed that also their feature vectors are similarly distributed, given the label [494]. The relaxed assumption is encoded into a regularized least-squares minimization problem, which can be rewritten in matrix form by the Laplacian of a symmetric matrix whose entries consist of the similarities between bags. The solution to the stated optimization problem can then be obtained in closed form. On ten datasets from the UCI standard repository [32], LMM and AMM outperform Mean Map, Invcal, and the $\propto$SVM in terms of prediction performance and runtime. However, since LLM is not kernelized and can only find linear decision boundaries, the results and LLM cannot be directly compared with the non-linear clustering algorithm introduced in Section 3.3.6.

**Inverse Calibration** Rüping proposes the Inverse Calibration (Invcal) method [552]. The regression SVM (SVR) is converted into a probabilistic classifier by applying a scaling function $\sigma$ to the outputs. According to the author, it is sufficient that the predictions of the classifier approximate the given label proportions well on average, for each bag. These constraints are integrated as auxiliary conditions into the standard SVR optimization problem. As a large margin method, the formulation allows for the reduction of model complexity, while the class probability estimates for each bag are kept close to the given label proportions for each bag, up to some maximum tolerable error. The primal problem can be transformed into its dual, and then solved with a standard solver for quadratic optimization. It is shown empirically over twelve standard datasets from the UCI repository that Invcal significantly outperforms Mean Map in terms of prediction accuracy.

**$\propto$SVM** Invcal treats the mean of each bag as some kind of super-instance, and gives each bag a regression label that corresponds to the label proportions. Instead, the $\propto$SVM proposed by Yu et al. explicitly models the labels of individual observations [751]. The label proportions, as calculated from labels assigned to individual observations, should match the given label proportions as closely as possible. This criterion is encoded as an additional term into the primal problem of the standard SVM. The task is to find a vector of labels such that the loss over label proportions and the standard loss

over individual observations are minimized. This ensures that observations lying on the same side of the hyperplane will be assigned the same label, depending on the particular values of the trade-off parameters $C$ and $C_\pi$. Although the formulation seems intuitive, the optimization problem is a NP-hard non-convex integer programming problem. The authors propose two different efficient algorithms for solving it, one based on an alternating optimization strategy, and another based on convex relaxation. In experiments, the $\propto$SVM outperformed Mean Map and Invcal in terms of accuracy on several datasets from the UCI standard repository. However, the authors do not report which significance test they used. It should be noted that in the work by Patrini et al. results are not always in favor of the $\propto$SVM in comparison to Invcal, even on the same datasets [494]. Moreover, Mean Map outperformed the $\propto$SVM in many cases, while Invcal outperformed Mean Map in the work by Rüping [552].

**AOC Kernel K-Means** AOC Kernel k-means (AOC for Aggregate Output Classification) introduced by Chen et al., called AOC-KK in the following, clusters the observations such that clusters correspond to classes, and the assignment of observations to clusters (classes) matches the given label proportions [131]. The authors present variants of k-Means and kernel k-means [174], which is a kernelized version of the original k-means algorithm. Here, the cluster centers can no longer be written in explicit form, but have to be expressed in terms of a kernel function induced by some feature map $\phi$.

In the objective function formulated, the first term is the same as in the original optimization problem of kernel k-means, while the second term measures the deviance between the given label proportions and those that would result from the current assignment of observations to clusters (classes). In that way, the authors try to find a good clustering, i.e. an assignment of observations to clusters (classes), such that the within-cluster scatter is minimized, but at the same time that also the given label proportions are matched as well as possible. The trade-off between the two criteria can be controlled by parameter $\lambda$. As standard tools for convex optimization cannot be used, the authors propose an alternating updating algorithm based on expectation maximization (EM) [168]. On two datasets from the UCI standard repository, AOC-KK outperforms k-NN and neural networks in terms of accuracy [466].

Although AOC-KK shares similarities with the clustering approach LLPC developed in Section 3.3.6, there are some fundamental differences. The first is that AOC-KK restricts the number of clusters to the number of classes, while LLPC allows for classes being represented by more than one cluster. This allows for a better control of bias vs. variance, by changing $k$. Another difference is that AOC-KK combines the loss over label proportions with the original kernel k-means objective in the same objective function, while LLPC first clusters observations as usual, and then tries to find a good assignment of labels to the resulting clusters. LLPC thus has the advantage that it can be used with arbitrary partitional clustering algorithms, while AOC-KK works only

with k-means and kernel K-means. LLPC is compared to AOC-KK with a quadratic loss function in Sect. 3.3.7.

**Theoretical Results** In an unpublished work, Yu et al. cast LLP into the PAC learnability framework [750]. The authors prove that under certain conditions, the labels of individual observations can be predicted well when the label proportions per bag can be predicted well. The generalization error over bag proportions in turn can be bounded by the empirical proportion error if the number of bags is large in relation to the Vapnik Chervonenkis (VC) dimension of the underlying hypothesis class $\mathcal{H}$ [1]. The authors further show that the probability for classifying instances correctly increases with the purity of bags, i.e. if many instances per bag belong to the same class. In extreme cases, where all label proportions are equal (i.e. they are the least pure), it can happen instead that a hypothesis achieves zero bag proportion error, but nevertheless classifies all instances incorrectly. The true error can be even further bounded by making additional assumptions on the distribution of bags [750].

The aforementioned findings imply that special care must be taken when comparing the performance of label proportion learning methods. For instance, it must be ensured that algorithms are trained and validated on the exact same data splits. Moreover, since the individual bag distributions can play a big role for the performance of algorithms, performance should be assessed over different diverse datasets and results need to be tested for their significance, even more so than with traditional supervised methods. That a method outperforms another does not mean then that it shows better performance in an absolute sense, under all circumstances, but on average.

**Other Works** Hernández-González et al. apply a structural EM strategy to learn Bayesian network classifiers from label proportions [274]. They compare their method to Mean Map and report lower error rate of their method for four of seven domains. However, the significance of results is not reported. Fan et al. introduced a generative classifier called DNLP, which learns from label proportions by following a deep belief network approach [198]. The authors compare their method to Mean Map and Invcal on several standard datasets from the UCI repository. In terms of prediction performance, they report no significant differences. However, the runtime of DNLP is much lower than that of Mean Map and Invcal. Fan and Taylor combine convolutional neural networks (CNN) with probabilistic graphical models trained by an EM approach to learn from label proportions in the context of ice and open water classification from image data [199]. Their algorithm shows good performance in the context of the mentioned application, but isn't evaluated on other domains.

1 For an easy introduction we recommend Mitchell [447] explaining PAC learning and the VC dimension.

### 3.3.4 Difficulty of the Problem

For getting a better idea about the difficulty of the problem, we discuss the problem of LLP from a more Bayesian perspective and relate it to different kinds of better-known learning tasks such as supervised, semi-supervised, and unsupervised learning.

For the supervised learning scenario, we can construct an optimal classifier, called the *optimal Bayes classifier*, if the distribution $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ is known. From a Bayesian perspective, a prediction model can be obtained from estimating the conditional class density $\mathbb{P}(\mathcal{Y} \mid \mathcal{X})$. Applying Bayes theorem, one recognizes that $\mathbb{P}(\mathcal{Y} \mid \mathcal{X})$ may also be estimated from other unknown densities—the class-conditional density $\mathbb{P}(\mathcal{X} \mid \mathcal{Y})$ and the class prior density $\mathbb{P}(\mathcal{Y})$:

$$\mathbb{P}(\mathcal{Y} \mid \mathcal{X}) = \frac{\mathbb{P}(\mathcal{X} \mid \mathcal{Y}) \cdot \mathbb{P}(\mathcal{Y})}{\mathbb{P}(\mathcal{X})} \tag{3.3}$$

Here, $\mathbb{P}(\mathcal{X})$ doesn't necessarily need to be known or estimated, since it can be calculated from $\mathbb{P}(\mathcal{X} \mid \mathcal{Y})$ and $\mathbb{P}(\mathcal{Y})$. $\mathbb{P}(\mathcal{Y})$ may be estimated directly from the data, if the number of data points is high enough. Moreover, if the joint distribution $\mathbb{P}(\mathcal{X}, \mathcal{Y})$ is known, as is assumed by the optimal Bayes classifier, all other quantities can be derived from it. For a given observation $x \in \mathcal{X}$ to classify, the optimal Bayes classifier would predict the most probable class, which is also known as the MAP criterion. Here, optimal means that the Bayes classifier is the best classifier over all possible classifiers for the given data.

**Best Case** With respect to LLP, the class prior $\mathbb{P}(Y_v)$ for label $Y_v$ can be estimated as $\eta(\Pi, Y_v)$, the proportion of $Y_v$. This is done, for instance, by the Mean Map method. Finding a good estimate for $\mathbb{P}(\mathcal{X} \mid \mathcal{Y})$, however, is at least as difficult as in the supervised scenario and equates to it if each bag $B_u$ only contains observations from a single class and at least $l$ bags contain observations from different classes. This scenario may be called the *best case*, since in LLP, usually less information about individual labels is given. Our intuition matches the findings of Yu et al. where it has been proven that the probability of classifying instances correctly increases with the purity of the bags [750]. When each bag only contains examples from the same class, each bag is as pure as possible.

**Worst Case** In the *worst case*, all label proportions $\pi_{uv}$ in matrix $\Pi$ are equal, i.e. least pure, and labels can only be guessed correctly with probability $1/l$. If sample size is large enough, the worst case can only occur if also all class priors $\mathbb{P}(Y_v)$ are equal, and the label does not depend on the bag, i.e. $\mathbb{P}(\mathcal{Y}|u) = \mathbb{P}(\mathcal{Y})$. Otherwise, we can estimate $\mathbb{P}(\mathcal{Y})$ from the data and at least predict the class that has highest probability to occur (i.e. the majority class). In this case, the probability for predicting the correct label would be higher than $1/l$.

The worst case can only occur with large amounts of data, where the label proportions, given as relative frequencies, will approach the true probabilities of classes in each bag, or in the context of privacy-preserving data mining, where we have full control over the formation of bags and want to make the problem as difficult as possible. In general, a small number of large sized bags can make the problem more difficult, as Yu et al. have shown [750]. For smaller random samples, we may usually expect slight deviations of the proportions in matrix $\Pi$, which may help with making a correct decision about class labels. For instance, when randomly uniformly sampling the 50 observations per class from the well-known Iris dataset into bags, in many cases the clustering approach developed in Section 3.3.6 classifies at least 96 % of the observations correctly on average.

**Average Case** In cases where observations have been sampled more or less randomly into bags, a first intuition might be that bags that are more "pure", i.e. that contain more instances of the same class, provide more information. Yu et al. also show that the probability of classifying individual instances correctly increases with the purity of bags [750]. However, only an upper bound is shown for the *probability* of misclassifying a fraction of individual observations incorrectly. In practice, cases may occur where we perform well, despite label proportion matrix $\Pi$ having high entropy, or badly, despite $\Pi$ having low entropy.

For instance, bags with low information content in terms of labels may nevertheless provide information about the underlying distribution of observations, $\mathbb{P}(\mathcal{X})$. Getting more information about $\mathbb{P}(\mathcal{X})$ by taking unlabeled observations into account as well can increase prediction performance when only a few labeled examples are given [120]. Conversely, even if a bag has high information content in terms of labels, learning might not profit from it if the sample doesn't represent the underlying data distribution.

For practical cases, it is therefore hard to find a measure of problem difficulty. While it is easy to measure the entropy of $\Pi$, it is difficult to measure how well bags reflect the overall data distribution given a concrete sample, without knowing the underlying data distribution—which is the crux of learning.

### 3.3.5 Loss and Risk

There are different possible ways to define loss functions over label proportions. First we define measures of the quadratic deviation between the label proportions as being derived from a previously trained prediction model $\hat{f}$ and the given label proportions. Applying the trained model to a set of observations $x_i \in X$, the resulting label proportions can be calculated by counting the number of observations $x_i$ with $\hat{f}(x_i) = Y_v$, in each bag for each label $Y_v \in \mathcal{Y}$ and dividing such counts by the size of their respective

bag. This leads to a new matrix $\Gamma_{\hat{f}}$, containing the model-based label proportions:

$$\Gamma_{\hat{f}} = (\gamma_{uv}^{\hat{f}}), \quad \gamma_{uv}^{\hat{f}} = \frac{1}{|B_u|} \sum_{x \in B_u} I(\hat{f}(x), Y_v), \quad I = \begin{cases} 1 : \hat{f}(x) = Y_v \\ 0 : \hat{f}(x) \neq Y_v \end{cases} . \tag{3.4}$$

Similar to when defining a loss function for individual observations, it is now possible to define a loss function over individual matrix entries by say, taking as loss the squared error $(\pi_{uv} - \gamma_{uv}^{\hat{f}})^2$. The total deviance between $\Pi$ and $\Gamma_{\hat{f}}$ can then be defined as the average squared error over all matrix entries:

$$\ell_{MSE}(\Pi, \Gamma_{\hat{f}}) = \frac{1}{hl} \sum_{u=1}^{h} \sum_{v=1}^{l} (\pi_{uv} - \gamma_{uv}^{\hat{f}})^2 \tag{3.5}$$

The average squared error $\ell_{MSE}$ doesn't take into account the relative group and class sizes; nor can it catch the situation where two hypotheses $\hat{f}_1$ and $\hat{f}_2$ appear indistinguishable from each other, because the total error sum over all matrix entries is the same. In practice, it can make sense to measure the error between $\Pi$ and $\Gamma_{\hat{f}}$ by $\ell_{\Pi}$, which we define as the geometric mean of two different error measures $\ell_{weight}$ and $\ell_{Prior}$ which deal with the previously mentioned disadvantages:

$$\ell_{\Pi}(\Gamma_{\hat{f}}) = \sqrt{\ell_{weight}(\Pi, \Gamma_{\hat{f}}) \cdot \ell_{Prior}(\Pi, \Gamma_{\hat{f}})} \quad \text{with} \tag{3.6}$$

$$\ell_{weight}(\Pi, \Gamma_{\hat{f}}) = \frac{1}{hl} \sum_{u=1}^{h} \sum_{v=1}^{l} \eta(\Pi, Y_v) \frac{|B_u|}{N} (\pi_{uv} - \gamma_{uv}^{\hat{f}})^2 \quad \text{and} \tag{3.7}$$

$$\ell_{Prior}(\Pi, \Gamma_{\hat{f}}) = \frac{1}{l} \sum_{v=1}^{l} \Big( \eta(\Pi, Y_v) - \eta(\Gamma_{\hat{f}}, Y_v) \Big)^2 \tag{3.8}$$

$\ell_{weight}$ weights the squared error of individual matrix entries by their relative group and class size. $\ell_{Prior}$ measures how well a chosen hypothesis matches the class priors, as estimated by $\eta(\Pi, Y_v)$. The choice to include the prior in the loss function has been made based on empirical evaluations and a close examination of the label proportion matrices which have lead to misclassifications. What we have observed in our experiments now has a theoretical justification. As shown by Yu et al., whenever a hypothesis matches the class priors and observations in bags are distributed i.i.d., the probability of misclassfiying a fraction of individual observations is bounded [750].

Moreover, if in addition to the label proportions, the true labels $y(x)$ of a subset $T \subseteq X$ of observations $x \in T$ are given, error criterion (Equation 3.6) can be easily extended to include the average loss $\ell_T$ over these labeled training examples:

$$\ell_{\Pi} = \sqrt[3]{\ell_{weight} \cdot \ell_{Prior} \cdot \ell_T} \quad \text{with} \quad \ell_T = \frac{1}{|T|} \sum_{x \in T} \ell(y(x), \hat{f}(x)) \tag{3.9}$$

Algorithms that optimize over $\ell_{\Pi}$ can thereby easily consider also labeled observations in addition to the given label proportions.

### 3.3.6 Learning from Label Proportions by Clustering

The goal in LLP is to find a function $\hat{f}$ that predicts the proportions of previously unseen bags as well as possible, which in turn bounds the risk of misclassifying individual observations, as shown by Yu et al. [750]. The authors pose this problem in terms of empirical risk minimization. However, if we allow for arbitrarily complex hypotheses, we can always match the given label proportions. In particular if we tried all different possible labelings of observations exhaustively, we would always find a set of labelings that minimizes one of the previously introduced loss functions. We would expect only a few of such labelings to also minimize the empirical loss over individual observations, i.e. we somehow need to control the capacity of our hypothesis class.

The particular LLP approach proposed in the following is based on the assumption that observations lying close together in regions of the input space also share the same class label. It first forms clusters of similar observations using an arbitrary partitional clustering algorithm and respective distance measure. Instead of trying all possible labelings of observations, the algorithm heuristically tries different labelings of clusters, such that a loss function over label proportions is minimized. The capacity of the hypothesis space can thus be controlled by varying the number of clusters $k$. A small number of clusters leads to high bias, but low variance. A larger number of clusters allows for ever smaller divisions of sample $X$, and therefore leads to low bias, but high variance.

The assumption that clusters represent classes is not necessarily correct. Hastie et al. demonstrate that especially the weighting of features can have an enormous influence on clustering results [259]. In fact, one advantage of supervised methods over unsupervised ones is that they can determine the relevance of features in relation to the target variable. We therefore allow for a certain flexibility in distance measures. Such measures should respect weights $w_j \in [0, 1]$ for each feature $A_j$, as given by a vector $\vec{w} = (w_1, \ldots, w_d)$. Usually, such weights are specified by a domain expert. In the clustering approach introduced in the following, however, the relevance weights can be approximated automatically by an evolutionary strategy, based on one of the loss functions defined in Section 3.3.5 (or other loss functions for LLP).

In the Section 3.3.6.1, the accompanying optimization problem is stated. Then, in Section 3.3.6.2, an approach for solving it is described. The algorithm can be used with different labeling strategies which are presented in Section 3.3.6.3. The approach's runtime is analysed in Section 3.3.6.4, while Section 3.3.6.5 explains how to classify new examples, based on a set of labeled clusters.

#### 3.3.6.1 Optimization Problem

Let the vector $\vec{\lambda}_{\mathcal{C}} = (\lambda_1, \ldots, \lambda_k)$ with $\lambda_j \in \mathcal{Y}$ represent a labeling for a clustering $\mathcal{C} = \{C_1, \ldots, C_k\}$. Let $\hat{f}_{\vec{\lambda}_{\mathcal{C}}} : \mathcal{X} \to \mathcal{Y}$ be a mapping that returns the label $\lambda_i$ for a given observation $x \in C_i$. Given a clustering $\mathcal{C}$, we search for a labeling $\vec{\lambda}_{\mathcal{C}}$ of the clusters

that minimizes the error, according to some error measure $\ell_{\vec{\lambda}_\mathcal{C}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}_\mathcal{C}}})$, between the model-based label proportions $\Gamma_{\hat{f}_{\vec{\lambda}_\mathcal{C}}}$ and the known label proportions $\Pi$:

$$\min_{\vec{\lambda}_\mathcal{C}} \ell_{\vec{\lambda}_\mathcal{C}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}_\mathcal{C}}}) \tag{3.10}$$

The error measure could be, for instance, the average squared error $\ell_{MSE}$ or a combined error measure such as $\ell_\Pi$.

Let $q_{\vec{w}}$ be a function which is able to assess the quality of a clustering based on a similarity measure that respects feature weights. This usually means that observations $x_i \in \mathcal{X}$ are represented as $d$-dimensional feature vectors $\vec{x}_i = (x_{i1}, \ldots, x_{id})$ with $x_{ij} \in \mathbb{R}$. We are trying to solve the optimization problem

$$\min_{\vec{w}} \ell_{\vec{\lambda}_\mathcal{C}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}^\star_\mathcal{C}}}), \quad \vec{\lambda}^\star_\mathcal{C} = \operatorname{argmin}_{\vec{\lambda}_{\mathcal{C}^\star}} \ell_{\vec{\lambda}_\mathcal{C}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}_{\mathcal{C}^\star}}}), \quad \mathcal{C}^\star = \operatorname{argmax}_\mathcal{C} q_{\vec{w}}(\mathcal{C}), \tag{3.11}$$

i.e. we are searching for a clustering $\mathcal{C}^\star$ which maximizes $q_{\vec{w}}$ and whose labeling $\vec{\lambda}^\star_\mathcal{C}$ minimizes $\ell_{\vec{\lambda}_\mathcal{C}}$, for all possible weight vectors $\vec{w}$. As formulated, with arbitrary functions $q_{\vec{w}}$ and $\ell_{\vec{\lambda}_\mathcal{C}}$, the problem is non-convex. Since we want to allow for flexibility in the choice of such functions, in the following we approximate solutions by an evolutionary strategy.

#### 3.3.6.2 The LLPC Algorithm

The LLPC (Learning from Label Proportions by Clustering) algorithm solves problem (3.11) by an evolutionary strategy. For each weight vector $\vec{w}$, the sub-optimization problem of maximizing $q_{\vec{w}}$ is solved by an inner clustering algorithm, where the particular $q_{\vec{w}}$ depends on the algorithm. The only prerequisite for the clusterer is that it returns disjunct clusters and respects different feature weights. The sub-optimization problem (3.10) is independent from the clusterer and currently can be solved by different labeling strategies, of which two are introduced in Section 3.3.6.3.

In more detail, LLPC takes a clustering algorithm *clusterer*, a labeling algorithm *labeler* and an error measure $\ell_{\vec{\lambda}_\mathcal{C}}$ as parameters, in addition to $\Pi$, $X$, $\mathcal{B} = \{B_1, \ldots, B_h\}$ and $Y = \{Y_1, \ldots, Y_l\}$, which are related to the task of LLP, and a set of parameters *evo* related to the evolutionary learning strategy. LLPC then approximates the optimal weight vector and returns $\vec{w}^\star$, as well as the related clustering $\mathcal{C}^\star$ and labels $\vec{\lambda}^\star_\mathcal{C}$ for the clusters. The returned weights $\vec{w}^\star$ can be interpreted as the importance of individual features and thus give valuable additional information for the interpretation of cluster models.

We use the evolutionary strategy described in [444]. The evolutionary strategy starts with a random population $P$ of normalized weight vectors $\vec{w}$, i.e. $w_j \in [0, 1]$. For each individual in $P$, the clustering algorithm *clusterer* is called. The clusters are labeled according to the given labeling algorithm *labeler* and the fitness is evaluated by criterion $\ell_{\vec{\lambda}_\mathcal{C}}$. If the fitness is higher than the best fitness seen so far, the newly found

**Algorithm 1:** The LLPC algorithm

**Function** LLPC($\Pi$,$X$,$\mathcal{B}$,$\mathcal{Y}$,*clusterer*,$k$, *labeler*,$\ell_{\vec{\lambda}_\mathcal{C}}$,*evo*)
  *best_fit* := $-\infty$; *generation* := 0 ;
  Randomly initialize a population $P$ of *psize* normalized weight vectors ;
  **while** *generation* < *maxgen* **do**
    **for** $\vec{w} \in P$ **do**
      $\mathcal{C}$ := *clusterer*($X$, $k$, $\vec{w}$) ;
      ($\vec{\lambda}_\mathcal{C}$, *err*) := *labeler*( $\mathcal{C}$, $\mathcal{B}$, $\Pi$, $\mathcal{Y}$, $\ell_{\vec{\lambda}_\mathcal{C}}$);
      **if** *best_fit* < $-$*err* **then**
        *best_fit* := $-$*err*; $\mathcal{C}^\star$ := $\mathcal{C}$; $\vec{\lambda}^\star_\mathcal{C}$ := $\vec{\lambda}_\mathcal{C}$; $\vec{w}^\star$ := $\vec{w}$ ;
      **end**
    **end**
    *generation* := *generation* + 1 ;
    **if** *generation* < *maxgen* **then**
      *Pcopy* := $P$ ;
      Gaussian mutation of weights in *Pcopy* with variance *mutvar* ;
      *Pchildren* := Uniform crossover on $P \cup$ *Pcopy* with probability *crossprob* ;
      $P$ := Tournament selection with size *tournsize* on $P \cup$ *Pcopy* $\cup$ *Pchildren* ;
    **end**
  **end**
  **return** $\mathcal{C}^\star$, $\vec{\lambda}^\star_\mathcal{C}$, $\vec{w}^\star$ ;
**end function**

clustering, labeling, and weight vector are memorized as the new best ones. In each generation, the weight values in a copy of $P$ are mutated by a Gaussian distribution and, with a certain probability, exchanged with $P$ by a crossover operator. The individuals then take part in a tournament and only the best ones are kept in the next generation. This process is repeated until the maximum number of generations as specified by the user is reached.

Using an evolutionary strategy as a wrapper has the advantage that it is not necessary to integrate the error measure $\ell_{\vec{\lambda}_\mathcal{C}}$ into the optimization problem of the inner clustering algorithm, as was done in AOC-KK. The clustering algorithm can thus be treated as a black box and easily exchanged, without any further adaptation. It should also be noted again that in contrast to AOC-KK, LLPC allows for classes being represented by more than just one cluster ($k > l$). Thereby LLPC allows for ever smaller divisions of sample $X$, i.e. parameter $k$ may be seen as a control parameter that trades off bias against variance, as previously discussed.

The free choice of clustering algorithm allows for respecting different kinds of data distributions. For example, LLPC was run successfully with k-means [426], kernel k-means [174], EM clustering [168, 731], DBSCAN [191], PROCLUS [11] and Support Vector Clustering (SVC) [55], without modification. Moreover, LLPC can be used with different

error measures, such as criterion (Equation 3.9) that can respect individually labeled examples.

LLPC may therefore be looked at as a meta-algorithm for learning from label proportions, which allows for the use of different clustering algorithms, labeling strategies and loss functions. In a further step, one might also exchange the evolutionary algorithm. For instance, it might be adapted to not only minimize $\ell_{\vec{\lambda}_{\mathcal{C}}}$ over weight vector $\vec{w}$, but also over hyperparameters such as $k$ in the case of k-means clustering, or $C$ and the RBF kernel $\gamma$ in the case of SVC.

**Algorithm 2:** Labeling of clusters by local search with multistarts

**Function** LocalSearchMultiStart($\mathcal{C}, \mu, \Pi, k, \mathcal{Y}, \ell_{\vec{\lambda}_{\mathcal{C}}}, starts$)
  $best = -\infty$ ;
  **for** $iteration \leftarrow 1, starts$ **do**
    $\vec{\lambda}_{\mathcal{C}}, \vec{\lambda}_{\mathcal{C}}^{\text{bestIter}} \leftarrow (\lambda_1, \ldots, \lambda_k)$ with $\lambda_j \in \mathcal{Y}$ chosen uniformly at random
    $start, bestIter \leftarrow -\ell_{\vec{\lambda}_{\mathcal{C}}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}_{\mathcal{C}}}})$ ; *// calculate initial fitness*
    $improving \leftarrow$ true ;
    **while** *improving* **do**
      **for** $kpos \leftarrow 1, k$ **do**
        *// at each position . . .*
        **for** $lpos \leftarrow 1, |\mathcal{Y}|$ **do**
          *// . . . try all labels . . .*
          $\lambda_{kpos} \leftarrow Y_{lpos} \in \mathcal{Y}$ ;
          $fitness \leftarrow -\ell_{\vec{\lambda}_{\mathcal{C}}}(\Pi, \Gamma_{\hat{f}_{\vec{\lambda}_{\mathcal{C}}}})$ *// calculate fitness*
          **if** $fitness > bestIter$ **then**
            $\vec{\lambda}_{\mathcal{C}}^{\text{bestIter}} \leftarrow \vec{\lambda}_{\mathcal{C}}$ ; $start, bestIter \leftarrow fitness$ ;
            **break** *// leave both for loops*
          **else**
            $\lambda_{kpos} \leftarrow \lambda_{kpos}^{\text{bestIter}}$ *// reset to best label found at kpos so far*
          **end**
        **end**
      **end**
      **if** $bestIter = start$ **then**
        *// Nothing better found*
        $improving \leftarrow$ false ;
      **end**
    **end**
    **if** $bestIter > best$ **then**
      $best \leftarrow bestIter$; $\vec{\lambda}_{\mathcal{C}}^{\text{best}} \leftarrow \vec{\lambda}_{\mathcal{C}}^{\text{bestIter}}$ *// remember best solution*
    **end**
  **end**
  **return** $\vec{\lambda}_{\mathcal{C}}^{\text{best}}, -best$
**End Function**

#### 3.3.6.3 Labeling Strategies

The following two labeling algorithms solve the sub-optimization problem (3.10) and can be used as the *labeler* in LLPC.

**Exhaustive Labeling** As long as $k$ can be restricted to a small number and $l = 2$ for a binary classification problem, trying $l^k$ possible labelings for a clustering $\mathcal{C}$ is no problem. In experiments (see Section 3.3.7), good solutions often were found for $6 \leq k \leq 12$. For each labeling, we need to calculate $\text{err}_{\vec{\lambda}_\mathcal{C}}$. For error measures like $\ell_{MSE}$ or $\ell_{\Pi}$, this takes linear time in the number of observations $N$. In case of the aforementioned error measures, the calculations only involve basic operations such as count, addition, multiplication, and division.

**Local Search with Multistarts** For cases where the number of clusters $k > 12$ or the number of labels $l > 2$, a local search that is started multiple times with different random combinations of labels is proposed. The local search greedily improves on the current labeling of clusters by trying all possible labels at each component of the labeling vector $\vec{\lambda}_\mathcal{C}$. Fitness measures how well the model-based label proportion matrix $\Gamma_{\hat{f}}$, as calculated from the current labeling, matches the given label proportions $\Pi$. If the fitness improves, the search starts again from the first component of the labeling vector $\vec{\lambda}_\mathcal{C}$. Otherwise, it resets the label at the current position $kpos$ to the label of the best (local) solution found so far. The best labeling found over all starts of the different greedy searches is returned.

In each iteration, the greedy search runs until no further improvement is possible (which is a stopping criterion). Moreover, at each step of the algorithm, the fitness either improves or stays the same. Therefore, each search finds a local minimum. Since the number of searches is finite, the returned labeling vector is also locally minimal. Although, in contrast with the exhaustive labeling strategy, it cannot be guaranteed that a globally optimal solution will be found, it has been demonstrated that the heuristic labeling strategy performs well in real-world applications such as reducing communication costs in distributed machine learning applications [655].

#### 3.3.6.4 Runtime Analysis

The user-specified parameters *maxgen*, *psize* and *tournsize* in LLPC are constants. They do not depend on the number of observations $N$ and limit the number of iterations of the evolutionary strategy to be constant. As discussed in Section 3.3.6.3, the asymptotic runtime of the labeling strategies is linear in $N$, as $k$ and $l$ are constants and the evaluation of $\text{err}_{\vec{\lambda}_\mathcal{C}}$ usually takes linear time. The asymptotic runtime of LLPC will otherwise depend on the used cluster algorithm. For example, if we allow for approximate solutions and limit the number of iteration steps, k-means has a linear runtime. Hence, overall LLPC has linear runtime, which makes it especially well suited for use in resource-constrained settings such as applications in the Internet of Things settings.

However, when used with an algorithm like kernel k-means, the runtime of LLPC can also become quadratic, for instance.

#### 3.3.6.5 Generating a Prediction Model

The LLPC algorithm returns labeled clusters of sample $X$. It is then possible to assign labels to individual observations $x_i \in \mathcal{X}$ with $\hat{f}_{\vec{\lambda}_e}$. The question is how to predict the labels of new observations, i.e. how to transform a clustering into a prediction model.

In the case of clustering algorithms which return a model-based description of clusters such as k-means which returns cluster means, one can simply use the model to assign new observations to a cluster, and then predict the cluster's corresponding class label. For instance, in the case of k-means, one can assign new observations to their closest cluster mean and predict the corresponding class label, by applying function $\hat{f}_{\vec{\lambda}_e}$. Whenever a clustering algorithm is purely descriptive, i.e. in cases where it only returns a clustering of $X$, but no model to assign new observations to clusters, one may use a nearest neighbors approach such as k-NN for classification.

In general, one option for getting a prediction model after running LLPC is to train a standard classifier such as Naïve Bayes [301] or a SVM [700], based on the current labeled observations. Taking this approach, LLPC may be regarded as a preprocessing step before modeling, in which the missing labels of observations in sample $X$ are restored, based on the given label proportions.

### 3.3.7 Evaluation

In this section, we evaluate the general method. Motivated by the steel scenario, the method needs to be carefully evaluated in order to be trustfully applicable in diverse industrial applications. Since the method is general, the LLPC algorithm is compared with three state-of-the-art methods for LLP: the Mean Map [514] method, Inverse Calibration (Invcal) [552], and AOC Kernel k-Means (AOC-KK) [131]. The comparisons are performed using standard benchmark datasets or generated test data, as is usually done.

LLPC is written in Java and has been implemented in the form of several operators in RapidMiner (https://rapidminer.com). All results are based on using fast k-means [187] as an inner clustering operator, which is a variant of k-means utilizing the triangle inequality for faster distance calculations. Observations $x_i \in \mathcal{X}$ are represented as $d$-dimensional feature vectors $\vec{x}_i = (x_{i1}, \ldots, x_{id})$ with $x_{ij} \in \mathbb{R}$. As a distance measure we have used the weighted Euclidean distance with weight vector $\vec{w} = (w_1, \ldots, w_d)$

$$d_{\vec{w}}(\vec{x}_i, \vec{x}_i^{'}) = \sum_{j=1}^{d} (w_j x_{ij} - w_j x_{ij}^{'})^2 \tag{3.12}$$

**Tab. 3.3:** UCI datasets used for the experiments. ©[2011] Springer. Reprinted, with permission, from [651].

| Dataset | *N* | *d* | Dataset | *N* | *d* |
|---|---|---|---|---|---|
| CREDITA | 690 | 42 | SONAR | 208 | 60 |
| VOTE | 435 | 16 | DIABETES | 768 | 8 |
| COLIC | 368 | 60 | BREAST CANCER | 286 | 38 |
| IONOSPHERE | 351 | 34 | HEARTC | 303 | 22 |

which weights each feature $x_{ij}$ of example $\vec{x}_i$ differently. In all experiments, we used the exhaustive labeling strategy (see Section 3.3.6.3) with loss function $\ell_\Pi$ (see Section 3.3.5). AOC-KK has been implemented using a combination of Java, RapidMiner, and Matlab. For Mean Map and Invcal, R scripts were used, which were provided by the author of Invcal [552].

#### 3.3.7.1 Prediction Performance Experiments

The accuracy of LLPC, AOC-KK, Invcal, and Mean Map has been assessed on the eight UCI [32] datasets shown in Table 3.3. Each possible value of a nominal feature has been mapped to a binary numerical feature with values 0 or 1. Numerical features were normalized to the [0, 1] interval. Table 3.3 shows the number of features $d$ after this preprocessing step.

In each single experiment, the accuracy has been assessed by a 10-fold cross-validation. For LLP, we have partitioned the training set of a particular fold into bags of size $\sigma$, by uniform sampling of observations. While such uniform sampling might not reflect the way in which bags are formed in a real-world setting, it allows for a more homogeneous interpretation of results across different datasets than domain-specific sampling based on feature values. We tried several bag sizes $\sigma$: 2, 4, 8, 16, 32, 64, and 128 (with the last bag smaller than $\sigma$, if necessary). The label proportions were calculated and the individual labels removed. In each fold, the accuracy of the learned prediction model has then been calculated on a labeled test set.

The kernel methods Mean Map, Invcal, and AOC-KK have been tested with the linear kernel, polynomial kernels of degree 2 and 3, and radial basis kernels ($\gamma = 0.01$, 0.1 and 1.0). LLPC has been tested for cluster sizes $k \in [2, 12]$. As parameters for the evolutionary strategy, we used $maxgen = 10$, $psize = 25$, $mutvar = 1.0$, $crossprob = 0.3$ and $tournsize = 0.25$. Running LLPC with k-means provides a prediction model consisting of cluster means with associated class labels. The same is true for AOC-KK. However, the cluster methods also assign labels to each observation in sample $X$, allowing for a subsequent training of other classifiers, as described in Section 3.3.6.5. Based on such labeled examples, we have trained models for Naïve Bayes [301], k-NN [13], decision trees [516], random forests [102], and the SVM [700] with linear and

radial basis kernel. The model parameters of each method have been optimized by a grid or evolutionary search.

The combination of all datasets, bag sizes, classifiers, their variants and parameters results in a total of 13 216 experiments: 672 for Mean Map and Invcal, 2688 for AOC-KK, and 9856 for LLP. For bag sizes 16, 32, 64 and 128 on the datasets COLIC and SONAR, and for bag size 128 on CREDITA, we conducted additional experiments with LLP for $maxgen = 5$ and $psize = 100$. In some cases, we achieved better prediction accuracy. All experiments took about three weeks. They were run in parallel on up to six machines with an AMD Dual-Core or Quad-Core Opteron 2220 processor and a maximum of 4 GB main memory.

#### 3.3.7.2 Prediction Performance Results

Figure 3.10 contains plots of the highest achieved accuracies for all datasets and bag sizes, based on the best performing models of LLPC, AOC-KK, Invcal, and Mean Map, over all conducted experiments. LLPC shows a higher accuracy than Invcal for many bag sizes on the datasets CREDITA, VOTE, COLIC, SONAR, and BREAST CANCER. On CREDITA, VOTE, IONOSPHERE, SONAR, and DIABETES, the variance of accuracy between bag sizes is smaller for LLPC compared with the other methods. Mean Map performs worse than LLPC and Invcal in many cases. The performance of AOC-KK varies, depending on the dataset. It shows good performance on BREAST CANCER and HEARTC, but not on the others. Except for the BREAST CANCER and VOTE datasets and a few other accuracy values, the overall accuracy of *all* methods decreases with a larger bag size. The results thus confirm the theory: with larger sizes of bags, without increasing the size of sample $X$, learning becomes more difficult.

The statistical significance of results can be assessed with the adjusted version of the Friedman test, as proposed by [169]. The test is a non-parametric equivalent of ANOVA and ranks the classifiers for each dataset separately. Under the null-hypothesis, the average ranks of the classifiers should be equal. For comparing LLPC to all others, we proceeded with the two-tailed Bonferroni-Dunn test as a post-hoc test.

Table 3.4 can be understood as a summary of the detailed plots shown in Figure 3.10, giving a better understanding and overview of LLPC's overall performance. The table shows the average ranks of the compared classifiers and their difference to LLPC's rank. Each rank was calculated based on the best performing models (including the standard classifiers), over all conducted experiments. The table also shows the critical difference (CD) values for the Bonferroni-Dunn test. The CD for $\sigma = 128$ is different, because Mean Map was not included in the comparison, due to missing values. LLPC has the highest rank in six cases, for $\sigma > 2$. At the 10 %-level, LLPC is significantly better than AOC-KK for $\sigma = 8$, better than Invcal for $\sigma = 128$ and better than Mean Map for $\sigma = 4, 8, 32$, and 64. In all other cases, LLPC performs equivalently.

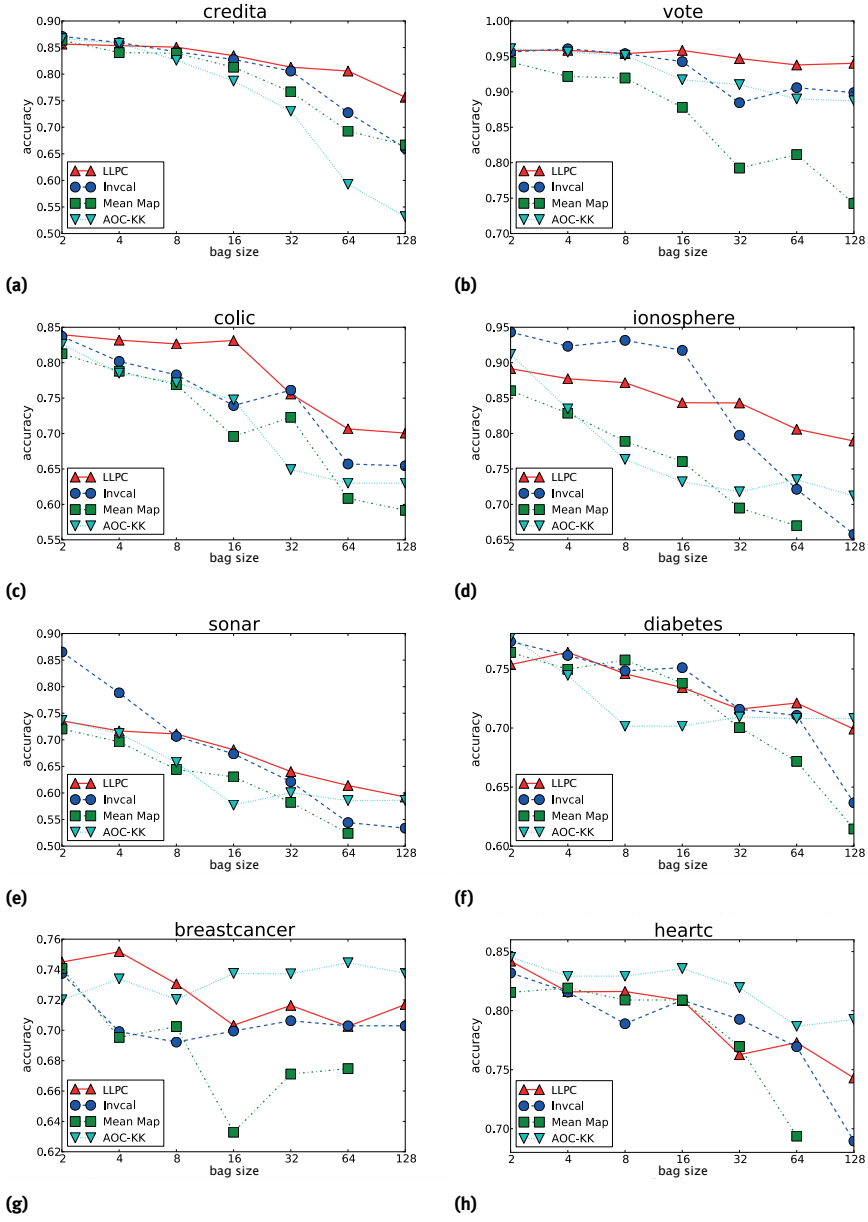
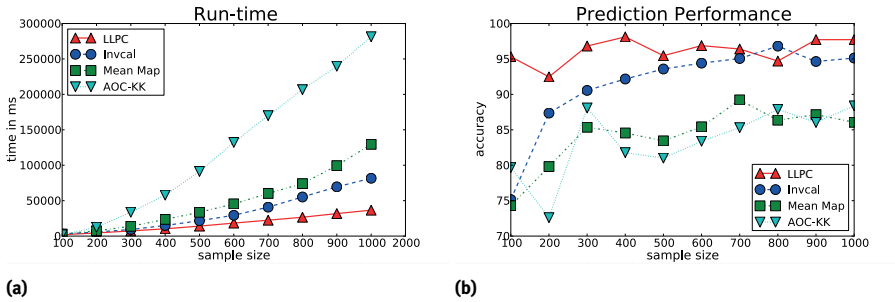



**Fig. 3.10:** Highest accuracies for all datasets and bag sizes, over all 13 216 runs of LLPC, AOC-KK, Invcal, and Mean Map (plus the additional runs of LLPC with $maxgen = 5$ and $psize = 100$). Some values for Mean Map and bag size 128 are missing in the plots, due to an error in the R script. ©[2011] Springer. Reprinted, with permission, from [651].

**Tab. 3.4:** Average ranks of classifiers by bag size, and their difference to LLPC's rank, based on the best models for each dataset and bag size. Positive difference values indicate a better performance of LLPC. Highest ranks and significant differences (higher than CD) at the 10 %-level are marked in bold. ©[2011] Springer. Reprinted, with permission, from [651].

| $\sigma$ | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
|---|---|---|---|---|---|---|---|
| | | | | **Average Ranks** | | | |
| LLPC | 2.500 | **1.875** | **1.500** | **1.875** | **1.625** | **1.375** | **1.375** |
| AOC-KK | **2.000** | 2.750 | 3.000 | 2.875 | 2.625 | 2.375 | 2.000 |
| Invcal | **2.000** | **1.875** | 2.375 | 2.125 | 2.125 | 2.275 | 2.625 |
| Mean Map | 3.500 | 3.500 | 3.125 | 3.125 | 3.625 | 3.875 | - |
| | | | | Differences, $CD_{<128}$=1.4317, $CD_{128}$=0.98 | | | |
| AOC-KK | -0.500 | 0.875 | **1.500** | 1.000 | 1.000 | 1.000 | 0.625 |
| Invcal | -0.500 | 0.000 | 0.875 | 0.250 | 0.500 | 1.000 | **1.250** |
| Mean Map | 1.000 | **1.625** | **1.625** | 1.250 | **2.000** | **2.500** | - |




(a) (b)

**Fig. 3.11:** Average runtime and accuracy of 10-fold cross-validations with LLPC, Invcal, Mean Map, and AOC-KK on several samples of random data. The data was generated for a two Gaussian mixture classification problem ($N = 10000$, $d = 10$, feature values normalized to [0, 1]). ©[2011] Springer. Reprinted, with permission, from [651].

### 3.3.7.3 Runtime Comparison

For an empirical runtime comparison of the algorithms, we have generated random data for a two Gaussian mixture classification problem (10 000 observations and 10 features, with values normalized to [0, 1]). Then, the average runtime for training and the accuracy of the classifiers for 10-fold cross validation has been assessed for different samples of the data, with varying sizes (see Figure 3.11). The bag size for LLP has been $\sigma = 16$ for all runs. A radial basis kernel with $\gamma = 0.1$ has been used for the kernel methods. LLPC has been run with the exhaustive labeling strategy and fast k-means ($k = 6$), with parameters $maxgen = 3$, $psize = 25$, $mutvar = 1.0$, $crossprob = 0.3$ and $tournsize = 0.25$ for the evolutionary optimization. Both LLPC and AOC-KK used the cluster mean model for prediction.

LLPC shows a high prediction performance for all sample sizes. Moreover, LLPC has the lowest runtime. However, since the methods are implemented in different programming languages (Java, Matlab, R), one should not compare the absolute times, but the slope of the curves. The curve of LLPC's runtime is very flat and almost a straight line, while the slopes of the other curves indicate runtimes that are faster growing. The results empirically demonstrate LLPC's small runtime, which makes the algorithm well suited for resource-constraint settings, especially since centroid cluster models also have a small memory footprint.

### 3.3.8 Summary and Conclusions

We have presented an approach for LLP known as the Learning from Label Proportions by Clustering (LLPC) algorithm. The approach is general enough to accommodate for the use of different clustering algorithms, labeling strategies, and loss functions. With k-means as the inner clustering algorithm and a constant number of iterations, LLPC has only linear worst-case training time and its cluster mean models are small and fast to apply. In comparison with state-of-the-art methods, which need more training time, the cluster mean models show a significantly higher or equivalent prediction accuracy in the conducted experiments. By training other classifiers on the labeled clusters, the highest achieved accuracy of LLPC is significantly higher for even more bag sizes. Here, LLPC has the highest average rank for all $\sigma > 2$. In addition, LLPC has other beneficial properties of which, to the best of our knowledge, other approaches don't possess all at once: LLPC can handle (1) non-linear decision boundaries, depending on the choice of clustering algorithm, (2) multiple classes, (3) additionally given labeled observations, and (4) it can weight the relevance of features.

LLP has relevance for real-world applications such as guaranteeing the privacy of democratic free elections, or the reconstruction of labels for objects that are hard to track, like those in smart manufacturing. Due to the small memory footprint and fast application of centroid cluster models, as well as a linear training time, LLPC is also well suited for running on resource-constrained devices in the Internet of Things, like edge devices in distributed computing. It has been developed for a series of processing steps in a steel rolling mill allowing for the early quality prediction during the processing (see Section 3.2.4.2). It also has been applied successfully in the field of traffic prediction (see Section 4.1), where it was used to reduce communication costs in a vertically-distributed machine learning setting [655].

Parts of this contribution were previously published in conference proceedings by Springer [651], [654] and in the first author's dissertation [652].

## 3.4 Simulation and Machine Learning

*Petra Wiederkehr*
*Katharina Morik*
*Amal Saadallah*
*Felix Finkeldey*

**Abstract:** The integration of sensor and simulation data for Machine Learning (ML) has become a hot topic. On the one hand, machine learning in the form of Active Learning (AL) allows running exactly those simulations that are needed for predicting future events based on the analysis of explanatory variables. Since simulation is a time-consuming process, we save computational resources by selecting the most informative simulation configurations. On the other hand, simulations represent expert knowledge, so that joining simulation and machine learning from observations leads to better predictions.

The combination of simulation and machine learning has been successfully used to optimize milling processes. Regarding undesirable vibrations of milling tools, a learning-based prediction of a stability criterion is realized. Furthermore, forces of milling operations are predicted using a developed data fusion of sensor and simulation data. Apart from forecasting process characteristics directly, machine learning also identifies parameter values for simulation models. In particular, the machine tool dynamics of a geometric physically based milling simulation system are successfully parametrized for different poses of the machine tool axes to reduce the number of calibration measurements required.

### 3.4.1 Introduction

In production engineering, many challenges arise regarding process design due to the high complexity and huge variety of engagement situations between the cutting tool and the workpiece to be machined [20]. For milling processes, different process parameter values can lead to different results for the machined component. Especially in the aerospace industry, where deep cavities have to be milled when machining structural components, the required long and slender milling tools can be susceptible to undesirable vibrations that can negatively affect the machined workpiece surface and can lead to increased tool wear. The tool wear can in turn influence the dynamic process behavior [20, 275].

Simulations can support the design of such processes [728]. There are different simulation approaches. Finite element (FE)-based methods represent complex interactions between the cutting edges of the tool and the machined material by numerical

approximations of differential equations [676]. Since the material is constantly being removed from the workpiece, the resulting high degrees of the deformation require relatively small simulation time steps. As a consequence, the simulation runtimes are high and only a few tooth engagements can be simulated in a reasonable time which makes the transfer to complex and long-running processes challenging [208]. By contrast, using geometric physically based simulation approaches, entire processes can be investigated [728]. This is due to the use of simple surrogate models to represent the tool and workpiece. However, this entails that complex interactions, such as tool wear, cannot be modeled reasonably. Moreover, while such methods are significantly faster than FE-based approaches, they are still not real-time capable.

In the context of Industry 4.0, the vision of realizing self-optimizing machining systems by incorporating machine learning methods has emerged recently and started to attract the attention of the manufacturing community [449]. By using these methods, predictions of process characteristics for unseen input data can be achieved in real-time [128, 491]. Several investigations can be found in literature, which focus on the prediction of different process characteristics for milling operations using machine learning methods, e.g. surface roughness [337, 384, 536], process forces [498, 699], or chatter vibrations [482, 690]. Furthermore, over the past decade, fusing multiple sensor signals has been a popular approach to increase the information gain for different tasks, especially for tool condition monitoring [230, 717, 765].

The predictive accuracy of machine learning models can be further improved by combining sensor data with simulation results. Denkena, Dittrich, and Uhlich [170] trained a model using support vector regression. Simulated and measured data are used as features to predict shape deviations of the workpiece. The simulated data is calculated using measured values of the axis positions of the tool. Plakhotnik et al. [503] combined sensor data, such as machine axis positions and spindle torque, with results from computer-aided design and geometry simulations to visualize specific process characteristics and support process design. Peng, Bergs, Schraknepper, Klocke, and Döbbeler [498] utilized FE-based simulations and measurements to train a tool wear-dependent cutting force model based on neural networks.

In this section, different aspects concerning the combination of simulation techniques and machine learning are discussed. As shown in Figure 3.12, it is analyzed how simulations can be replaced by machine learning models to enable real-time predictions [560]. The models learned on simulation data can then be refined by a selected number of experimentally acquired data to close the gap between simulation and experiment, which may be caused by the simplifications of simulation models. In this context, to improve resource efficiency, experiments should only be performed for scenarios whose inclusion is expected to maximize the improvement of the prediction accuracy of the models.

The research concerning AL involves two real-world applications. Reducing the computational resources of an expensive FE simulation is studied in the field of tunnel-

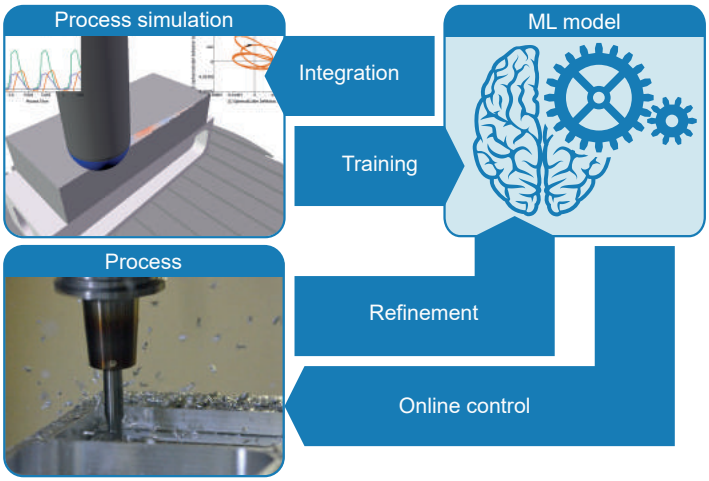



**Fig. 3.12:** Concept of combining simulation and machine learning [560].

ing (Section 3.4.3.1). The cheaper geometric physical simulation for sampling training data is studied when refining a model for milling (Section 3.4.3.2).

Another focus of the investigation is the fusion of simulation results with sensor data to predict future process characteristics for milling operations [206]. To this end, different fusion strategies are evaluated and compared for the milling application (Section 3.4.4). Furthermore, the first results are obtained for the integration of machine learning methods into a geometric physically based simulation system in order to learn pose-dependent dynamic models [207]. Using ML for initializing simulation models is illustrated by applications in milling (Section 3.4.5.1) and grinding (Section 3.4.5.2).

### 3.4.2 Simulation of NC-Milling Processes

In milling applications, we use a geometric physically based simulation system [728]. For the tool and workpiece model, the Constructive Solid Geometry (CSG) [210] technique is used. Thereby, the tool model can be realized by modeling the envelope of the rotating tool by combining simple geometric primitives through Boolean operators. For modeling the initial workpiece, the use of a cuboid is usually appropriate. The movement of the tool is defined by discrete positions along an NC path, which can also be used on a real machine. The step size between the tool positions is the feed per tooth defined by the process under consideration. The workpiece geometry of the $i$-th feed per tooth

$$W_i = W_0 \setminus \bigcup_{j=0}^{i-1} T_j \tag{3.13}$$

is the difference between the model of the initial workpiece $W_0$ and the union of the tool models $T_j, j = 0, \ldots, i-1$ for all previously processed discrete tool positions along the cutting path. The intersection between $W_i$ and $T_i$ represents the chip shape of the $i$-th cutting operation. For the calculation of process forces, this chip shape is sampled by rays that have their origin on the rotational axis of the tool. At each ray, the equation

$$F_i = b \cdot k_i \cdot d_0 \cdot \left(\frac{d}{d_0}\right)^{1-m_i},$$
$$d_0 = 1\,\text{mm},\ i \in \{c, n, t\}\ [321] \tag{3.14}$$

is evaluated to calculate process forces in cutting, normal, and tangential directions. Here, $d$ represents the chip thickness, $b$ the width of the cutting segments defined by the rays, and $k_c$, $m_c$, $k_n$, $m_n$, $k_t$, and $m_t$ the parameters to be calibrated. Using the directional information of the rays, the force vectors can be transformed into a global coordinate system in $x$-, $y$- and $z$-directions. To simulate tool vibrations, a set of decoupled damped harmonic oscillators represents the dynamic behavior of the machine-spindle-tool system. The parameter values of these oscillators have to be determined using measurements in advance so that deflections in $x$- and $y$-directions can be calculated as a vibration-induced displacement of the tool relative to the workpiece [662].

### 3.4.3 Learning from Simulation

As it was mentioned before, even though geometric physically based simulations allow for the investigation of long-running milling processes in a reasonable runtime [728], they are not yet real-time capable. As a result, predictions can be generated only for process configurations that have been simulated beforehand. By contrast, ML models can be evaluated in real time and, thus, offer the opportunity to predict future unknown events based on an analysis of past data [128] or to classify the current process state, using a set of features extracted from measured data [366]. Therefore, new trends in applied ML aim to replace simulations with ML surrogate models that are more appropriate for real-time applications [116, 212, 559, 560]. In such a manner, not only predictions are generated in real time, but computational resources required for running simulations are also saved.

#### 3.4.3.1 Active Learning for Simulation Data Acquisition

For the simulation of processes that exhibit a high degree of visco-elasticity or elasto-plasticity, numerical simulation methods that require high computational resources are often necessary [116, 212, 559]. Hence, we want to reduce the amount of data that is required to build an accurate surrogate ML model. The same idea was broached in the literature with AL. Starting from a small and non-optimal training set, AL procedures aim at selecting unlabeled data points whose inclusion in the training set improves

the performance of the ML model iteratively [593]. In this context, we develop an AL approach that selects the least amount of simulation configurations to learn an accurate surrogate ML model [559].

One simulation scenario is defined by a given combination of simulation input parameters and considered as a data instance characterized by a set of features. When the simulation runs for a given scenario, it is called a labeled data instance; otherwise, it is called an unlabeled data instance. The active selection can also be decided based on the label proportions of the instances that have so far been generated. In this case, the problem is formulated as Active Class Selection (for more details see Section 5.2.2 in Volume 2).We want to reduce the cost of running process simulations for collecting labeled training data instances. Hence, our Hybrid AL approach (HDAL) combines error-based with distance-based methods to select the minimal number of simulation scenarios that are necessary to build an accurate ML model. We start by randomly selecting a small set of simulation scenarios (i.e. configurations) to run. Then, our framework operates in three stages. First, it trains the ML model on the available labeled scenarios. Second, it computes the training error measures of the ML model for each individual scenario in the labeled set. This step allows us to identify input data regions where the ML model is weak (i.e. uncertain about the label) and probably would need to see more samples from these regions for a better generalization. We select the labeled scenarios with the highest estimated error rates. In the third stage, we determine the closest unlabeled scenarios to these labeled scenarios. The assumption is that two close data instances probably share similar characteristics and thus similar estimated surrogate ML models. However, to avoid clustering problems around these labeled scenarios (i.e. already investigated regions), we devise a "min-max" selection procedure that chooses the furthest ones from the closest unlabeled scenarios. The entire process is iterated until a stopping criterion is met. The stopping criteria for AL procedures is still an open research question. It can be set according to a maximum budget of iterations or when the model accuracy improvement on an independent calibration/validation set over the last iterations becomes insignificant.

We use a 3D FE simulation designed specifically for process-oriented computational simulations of shield tunnelling processes [116] to validate our framework. The simulation models predict soil displacements over time in different measuring points during tunnel excavation given two machine input parameters, namely the grouting and support pressures. Each scenario results in a time series with 64 item steps of displacement observations over 154 monitoring surface points. 20 scenarios are initially selected for training. 130 scenarios are considered unlabeled and 10 are used for testing. The goal is to replace the simulation with an ML model to forecast future soil displacements. To do so, we use Vector Autoregressive with Exogenous time series features (VARX) as a surrogate ML model [559]. For VARX setting, the input data consists of the set of time series of settlements of the surface monitoring points (i.e. measured in millimeters), plus the time series records of the grouting and the support pressures (i.e. measured in *Pa*) of the machines, considered as exogenous variables. A $L1$-regularization is applied

to estimate the model coefficients. The VARX model is retrained with each update of the training set at each AL iteration. Our study shows the ability of our framework in reducing the number of training scenarios required for training up to 60 % while improving the prediction accuracy up to 20 %.

#### 3.4.3.2 Active Learning for Model Building

Active learning can be applied not only to reduce data annotation costs (i.e. reducing the number of simulation configurations to run) but also as an informed sampling procedure to accurately learn the ML surrogate model. The surrogate model should be able to represent the capabilities of the simulation correctly and applied afterward in an online setting to predict process characteristics. Since the quality of ML model depends largely on the training data, we exploit the interaction between the surrogate ML model and the simulation using AL to actively and iteratively sample from the simulation data.

In this context, in order to control milling processes in real time by adapting the process parameter values, we develop a novel ML framework based on a geometric physically based simulation of NC-milling process (cf. Section 3.4.2) [560]. We choose to focus more specifically on building a ML model to predict the Poincaré diameter, which is considered as a process stability criterion for a given simulation scenario, characterized by a given input spindle speed. Parts of this section are already published by the authors [206, 560].

Our experimental use case is a face milling process using a fixed width of cut of 2 mm, an increasing depth of cut from 0 mm to 1 mm, a feed per tooth of 0.1 mm and varying values for the spindle speed between 3000 $\text{min}^{-1}$ and 15 000 $\text{min}^{-1}$. For the tool, a torus cutting tool with a diameter of 6 mm and a corner radius of 1 mm is used to machine aluminum alloy 7075.

The proposed framework consists of a weighted ensemble of Multilayer Perceptrons (MLPs) trained on different subsets of features. In addition, an AL procedure is simultaneously used to iteratively design the optimal training set from the generated simulation data. A forecast for the Poincaré diameter is delivered every $P_1$ milliseconds for the next future $P_2$ milliseconds. This is achieved by training a committee of MLPs, each on a distinct subset of process features [129]. Each model member of the committee is an online regression model, where the expected Poincaré diameter is assumed to be a function of the historical values of the time series features, namely the forces in the three-dimensional global coordinate system $F_x$, $F_y$, and $F_z$, the deflections in the bi-dimensional space $D_x$ and $D_y$ and the expected values of the chip volume. Since many configurations (i.e. various spindle speed values) are simulated, an AL approach based on the Query By Committee (QBC) paradigm [129] is used to iteratively add simulation configurations, whose inclusion in the training set should improve the prediction accuracy. The final prediction output is computed by a weighted average over the committee members' outputs. The main steps of the framework are illustrated in Figure 3.13.

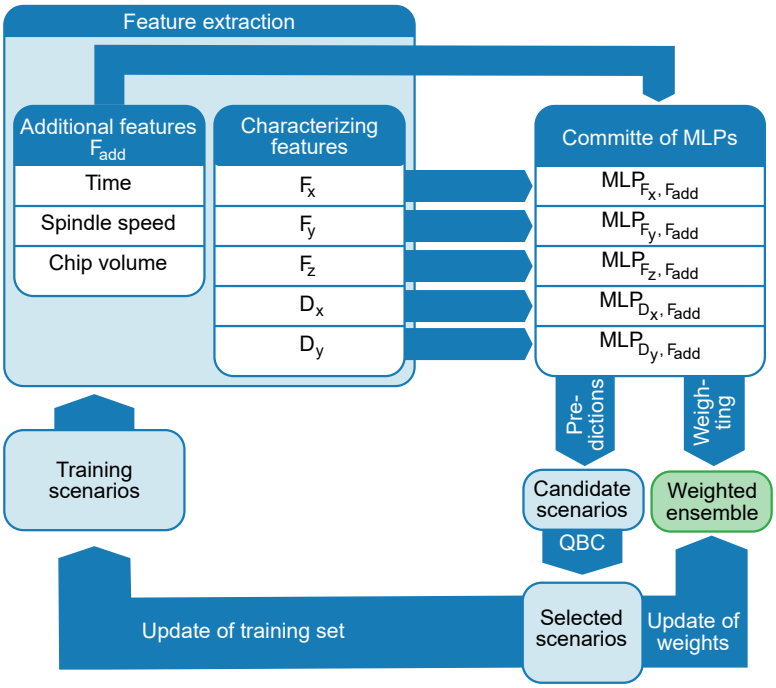



**Fig. 3.13:** Data acquisition and prediction framework [560].

**Pool-Based Active Learning with QBC** The output of the simulation depends to a large extent on the input value for the spindle speed. The goal is to identify simulation configurations whose inclusion in the training set improves the accuracy of the surrogate ML model. One way to do this is to carefully design the training set by controlling the selection of training simulation configurations using AL. This control is given by a problem-dependent heuristic, e.g., the decrease of the estimated prediction error on a test set if a given configuration or a pool of configurations are added to the training set. One of the most popular AL approaches is built based on the QBC paradigm [593].

A committee of learners is built following different assumptions on subspaces of instances, which may easily lead to a huge number of hypotheses to cover and quickly become computationally intractable for real applications [442]. It can also be built on disjoint subsets of features [129], usually generated by the Random Subspace Generation (RSG) approach [129], which may also lead to an intractable number of hypotheses. In our approach, each feature subspace represents one characteristic of the original process (cf. Figure 3.13)[560]. The QBC procedure selects the simulation configuration on which the committee members' predictions are maximally split. First, each committee member is trained on an initial small set of simulation configurations $N_i$. Then, at each iteration, the set of $N_c$ candidate configurations is sorted according

to a disagreement measure for each simulation configuration $c_s$:

$$f_{QBC}^{c_s} = log\left(\sum_{j=1}^{q}\frac{1}{n}\sum_{t=1}^{n}(\hat{y}_t^{c_s} - \hat{y}_{t,j}^{c_s})^2\right), \tag{3.15}$$

where $\hat{y}_{t,j}^{c_s}$ is the predicted value by the committee member $j$ at the instant $t$, and $\hat{y}_t^{c_s}$ is the estimated result of the committee composed of $q$ models, on the scenario $c_s$ and obtained as follows: $\hat{y}_t^{c_s} = \frac{1}{q}\sum_{j=1}^{q}(\hat{y}_{t,j}^{c_s})$. From the sorted configurations, the first $N_s$ are selected to be added to the training set. This entire process is iterated until a stopping criterion is met, e.g. a maximum number of configurations in the training set is reached, or the accuracy improvement on an independent calibration/validation set over the last iterations becomes insignificant. To output the final predictions, we use the already trained committee of MLPs and aggregate them in a weighted ensemble model. The weights are computed offline using a normalized version of the loss of the model on the training set. During the AL procedure, the weights are updated with each update of the training set. This mechanism may help to achieve a blind adaptation to drifting characteristics in time series observations by adjusting the contribution of each model in the final output [200]. Let $M = \{M_1, M_2, \ldots, M_q\}$ be the committee of $q$ MLPs and $\hat{y}_{t,j}^{c_s}$ is the output of model $M_j$ for a given simulation configuration $c_s$ at a time instant $t$. The final prediction output is obtained with:

$$\hat{y}_{t+1}^{c_s} = \sum_{j=1}^{q}\frac{\left[\left(1-\chi_j\right)\hat{y}_{t,j}^{c_s}\right]}{\sum_{j=1}^{q}\left(1-\chi_j\right)} : \chi_j \in [0,1], \forall j \tag{3.16}$$

where $\chi_j$ is the error of model $M_j$ on the recent obtained training set. To calculate such error, the Normalized Mean Squared Error (NRMSE) is used:

$$NRMSE_{c_s} = \frac{\sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t^{c_s} - y_t^{c_s})^2}{n}}}{\left|\max(Y^{c_s}) - \min(Y^{c_s})\right|}, \tag{3.17}$$

where $Y_s^c = \{y_1^{c_s}, \ldots, y_t^{c_s}, \ldots, y_n^{c_s}\}$ denote a time series of the Poincaré diameter for the simulation scenario $c_s$ and $\hat{y}_t^{c_s}$ is the predicted Poincaré diameter at the given instant $t$.

**Evaluation and Results** The simulation generates one observation for each feature with a frequency of 20 kHz, resulting in a step size of 0.05 ms [560]. The simulation configurations are generated for 240 different spindle speed values in the range of 3000 $\text{min}^{-1}$ to 15 000 $\text{min}^{-1}$ with a step size of 50 $\text{min}^{-1}$. The set of simulation configurations is randomly split into three independent sets. 180 are used for the training, while 20 others served to build a validation set. The remaining 40 are kept for testing. An aggregation period of $P_1 = 10$ ms is used for the preprocessing and a period of $P_2 = 50$ ms is set as a forecasting horizon. The MLP model parameters are tuned using a grid-search procedure on the validation configurations. For the AL procedure, 50 from

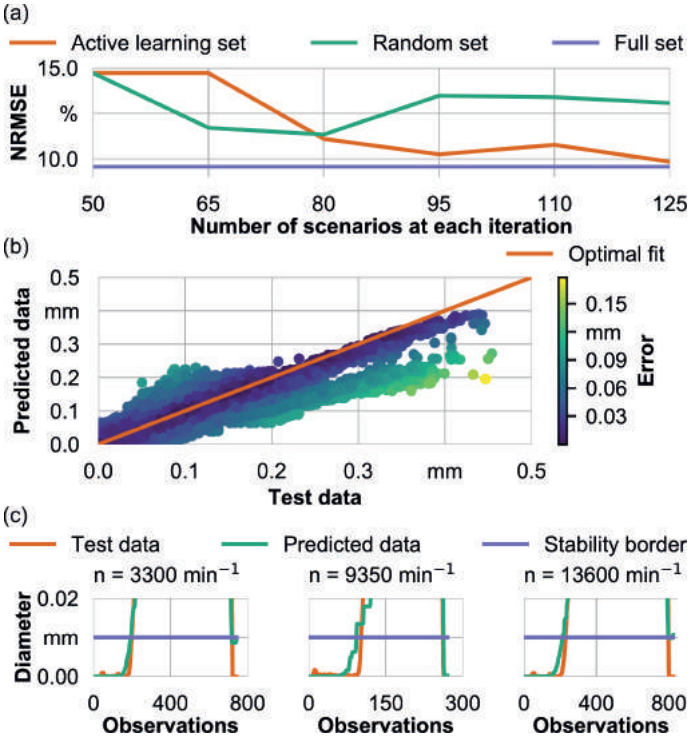



**Fig. 3.14:** (a) Active learning performance. (b) Comparison between test and predicted data. (c) Depiction of Poincaré predictions in critical areas for three different spindle speeds [560].

the 180 simulation configurations of the training set are randomly selected to construct the initial training set. From the remaining 130 configurations, 15 are added at each iteration. A maximum number of 5 iterations is used as a stopping criterion. The subset of features is constructed using two lagged values of each of the five characteristic features $F_x$, $F_y$, $F_z$, $D_x$, and $D_y$. The corresponding spindle speed value, the time index, and two future values of the chip volume are added to enrich each subset of features.

The results are presented from three perspectives [560]: 1) Figure 3.14(a) shows a comparison between the AL approach, a random sampling, and the performance of the ensemble, which is trained using the whole training set; 2) Figure 3.14(b) illustrates a comparison between the predicted Poincaré diameter values and the computed values on a subset of testing scenarios; 3) Figure 3.14(c) presents a comparison between the test and the predicted data for three different spindle speeds.

Figure 3.14(a) illustrates the performance of the AL method. The AL approach reaches almost the same performance as does training over the whole training set, while using only 70 % of the training set. The results show the advantage of designing an

optimal training set especially when a large pool of simulation scenarios with randomly chosen input parameters is available.

Figure 3.14(b) presents a comparison between the true and the forecasted Poincaré diameter values and illustrates that the ensemble is capable of accurately forecasting the Poincaré diameter in distinct scenarios and time periods.

Figure 3.14(c) shows that our framework predicts process instability in time or earlier for three different spindle speeds. This highlights its usefulness for real-time applications, where process parameter values should be monitored and adjusted to avoid process instabilities.

### 3.4.4 Fusion Between Simulation and Sensor Data

In general, the integration of diverse data sources provides useful and enriched new data. We refer to this combination as a data fusion process [502]. However, it presents many challenges [468]. Fusing different sources can generate conflicts. The conflicts are most often the result of incomplete, erroneous, and out-of-date records [468]. Another challenge comes along with complex sequences that are multi-dimensional, multi-modal and time-varying [331]. In addition, the process of data fusion may also lead to larger amounts of data, which poses difficulties for online application [331]. Solving these challenges requires not only substantial efforts and domain knowledge but also scalable and principled fusion approaches that cope with real-time constraints.

Process simulation can be viewed as background knowledge for domain experts. We want to integrate this knowledge into the machine learning process and, at the same time, use the simulation as an additional data source (i.e. generation of additional data points/data features or annotation of existing data). In this context, we aim at fusing both, simulation and sensor data, to predict active and passive forces in a real-time slot milling process [22, 206]. In [22], we present a framework that allows combining real-world observations collected from sensors and simulations at two levels: the data or the model level. At the data level, observations and synthetic data are integrated to form an enriched dataset for learning. At the model level, the models learned individually from observed and simulated data are integrated using an ensemble technique. Establishing a trade-off between model bias and variance, we perform an automatic selection of the appropriate fusion level. Figure 3.15 shows a schematic illustration of the conducted approach. Parts of this section are already published by the authors [22, 206].

To validate the developed framework, slot milling processes are conducted using a width of cut and depth of cut of 2 mm, a tilt angle of 30° [206]. A ball-end mill with a diameter of 10 mm and two cutting edges is used to machine AISI M3:2, hardened to approx. 62 HRC. During these milling experiments, process forces in $x$-, $y$- and $z$-directions are measured using a triaxial force dynamometer (Kistler 9257B) with a sampling frequency of 20 kHz. The goal of the considered use case is to predict upcoming active and passive forces, which are affected by tool wear. To this end, ten different process

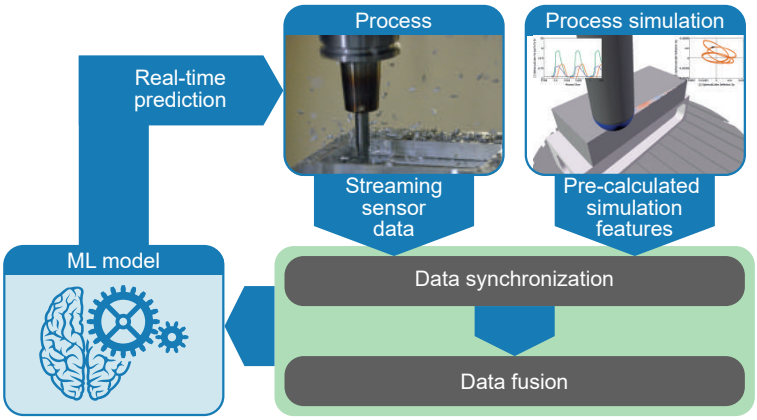



**Fig. 3.15:** Framework for synchronizing and fusing simulation and sensor data [206].

parameter configurations with varying values for the cutting speed and feed per tooth are used. For each parameter configuration, 180 slots are milled in order to provoke tool wear. After every twentieth machined slot, the width of flank wear is measured for both cutting edges and averaged to obtain a value $VB_V$ which indicates the wear state of the tool [206]. Intermediate values of $VB_B$ for all other slots $i$ are interpolated by

$$\widetilde{VB}_B^i = \begin{cases} \widetilde{VB}_B^{i-1} + \frac{F_\Sigma^i \cdot (VB_B^{j+1} - VB_B^j)}{\sum_k F_\Sigma^k}, & \text{if } F_\Sigma^i > F_\tau \\ \widetilde{VB}_B^{i-1}, & \text{otherwise} \end{cases}, \tag{3.18}$$

$$F_\Sigma^i = |F_x^i| + |F_y^i| + |F_z^i|, \tag{3.19}$$

where $i \neq 0$ and $F_\tau$ is a threshold that has to be defined in order to distinguish between signal and noise. The values $VB_B^j$ and $VB_B^{j+1}$ are estimated by measurements between which the interpolation is performed and $k$ is the number of interpolated values between these measured values. Using this approach, high force amplitudes are assumed to induce a high load on the cutting edges, resulting in increased values for $VB_B$ [206].

#### 3.4.4.1 Simulation-Sensor Data Mismatch Evaluation

The fusion of data collected from both sources into a single data representation is not straightforward and data mismatch between both sources needs to be checked. Data mismatch may mean that different data sources attribute different values to the same instance. This can be addressed by ensuring the *completeness* and the *correctness* of each data source [468].

The mismatch is often caused by different data alignments due to different data sampling frequencies from simulations and measurements. This results in a time delay between simulated and measured data. Therefore, data synchronization is required.

In addition, the mismatch can be caused by learning from different underlying distributions. In fact, due to the simplified models in the simulation system and the negligence of complex engagement behaviors, e.g. frictional effects, which are used to ensure a reasonable runtime, non-negligible deviations between simulation data and measurements of the corresponding process characteristics can occur, especially for process forces or tool vibrations. This deviation can differ for different process parameter values. As a result, a calibration of the simulation is required. Due to measurement noise and uncertainties, the quantification of simulation accuracy is a challenging task.

**Synchronization** If both, simulated and sensor data, are acquired using the same sampling frequency, only a constant time shift between each time series has to be determined. This can be achieved, e.g., manually, using change points, estimated by auto-regressive approaches [740] or by analyzing the continuous wavelet transform of the time series. In the context of the latter approach, the transformed signal is given by:

$$W(a,b) = \frac{1}{|a|^{1/2}} \int_{-\infty}^{\infty} \Xi(t) \Psi^{\star} \left( \frac{t-b}{a} \right) dt, \tag{3.20}$$

where $\Xi(t)$ is the original signal, $\Psi^{\star}(\frac{t-b}{a})$ represents the complex conjugate of a scaled and translated mother wavelet $\Psi(t)$, and $a$ and $b$ are the scale and translation parameters, respectively. Each scale corresponds to a frequency, resulting in information about the correlation between a given signal at a certain time instant and an investigated frequency without the need to make a trade-off between time and frequency resolution, which is a crucial issue of spectral analysis. In the milling application, the time-related delay between the two investigated time series can be identified by the points in time where the intensity of the wavelet transform at the tooth engagement frequency is greater than zero.

**Calibration** For the considered geometric physically based simulation system, simulation models can be calibrated using measurements as ground truth. This has to be performed for each combination of the tool geometry and the workpiece material of the regarded process. For simulated forces, for example, the parameters of the force model $\rho$ can be determined by applying an optimization procedure to minimize the squared Euclidean distance between simulated and measured forces

$$\ell(\rho) = \sum_{i=1}^{n} \left( \boldsymbol{F}_{\text{sen}}(t_i) - \boldsymbol{F}_{\text{sim}}(\rho, t_i) \right)^2, \tag{3.21}$$

acquired using the same process parameter values for both, the machining process and the corresponding simulation conduction. To this end, any optimization algorithm could be applied to solve the minimization task. However, in practice, quasi-Newton approaches often outperform other methods. Using the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [113] optimization algorithm, for example, an approximation of the

Hessian matrix $H_k$ is estimated, which is updated at each iteration $k$ of the procedure. According to Newton's method, the parameter values of the next iteration

$$\rho_{k+1} = \rho_k + \alpha_k s_k \tag{3.22}$$

are given by the line search along the descent direction

$$s_k = -H_k \nabla \mathcal{L}(\rho_k) \tag{3.23}$$

by estimating $\alpha_k$ through

$$\hat{\alpha}_k = \underset{\alpha}{\operatorname{argmin}}\, \mathcal{L}(\rho_k + \alpha s_k). \tag{3.24}$$

The update of $H_k$ is performed by adding a rank-two correction

$$H_{k+1} = H_k + auu^T + bvv^T, \tag{3.25}$$

$$u = \delta_k = \rho_{k+1} - \rho_k, v = H_k \gamma_k = H_k \nabla \mathcal{L}(\rho_{k+1}) - \nabla \mathcal{L}(\rho_k) \tag{3.26}$$

are typically chosen, so that the quasi-Newton condition

$$H_{k+1}\gamma_k = H_k \gamma_k + auu^T \gamma_k + bvv^T \gamma_k = \delta_k \tag{3.27}$$

is satisfied, resulting in

$$H_{k+1} = H_k + \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} - \frac{H_k \gamma_k \gamma_k^T H_k}{\gamma_k^T H_k \gamma_k}. \tag{3.28}$$

#### 3.4.4.2 Automatic Fusion-Level Selection

A sophisticated ML model should establish a trade-off between bias and variance. Such statement gives a guidance on how to automate the decision for the fusion-level selection. In fact, given a learned model $\hat{f}$ that approximates an unknown true model $f$, the expected mean-squared error between the target variable $y = f(x)$ and the model predictions on an unseen sample $x$, can be decomposed into bias, variance, and an irreducible error term [334]. One way to reduce the variance-type error is to use an ensemble model [101], that combines many models into one single model using an averaging technique [556]. Such a statement is derived from the ensemble error decomposition into the average bias of the ensemble single models, variance, and a covariance term [694]. Brown, Wyatt, and Tino [106] have proven that when using an average-based ensemble model $\bar{f}$ with equal weights (i.e. $\bar{f} = \sum_{i=1}^{N} w_i f_i$ , $f_i, i \in \{1, \cdots, N\}$ are the single models), the expected error decomposition is given by:

$$\mathbb{E}\left[\left(f(x) - \bar{f}(x)\right)^2\right] = \overline{Bias}^2 + \frac{1}{N}\overline{Var} + (1 - \frac{1}{N})\overline{Covar}, \tag{3.29}$$

where

$$\overline{Var} = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}\left[\left(\hat{f}_i(x) - \mathbb{E}\left[\hat{f}_i(x)\right]\right)^2\right], \tag{3.30}$$

$$\overline{Bias} = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{E}\left[\hat{f}_i(x)\right] - \mathbb{E}\left[f(x)\right]\right), \tag{3.31}$$

$$\overline{Covar} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1,\neq i}^{N}\mathbb{E}\left[\left(\hat{f}_i(x) - \mathbb{E}\left[\hat{f}_i(x)\right]\right)\left(\hat{f}_j(x) - \mathbb{E}\left[\hat{f}_j(x)\right]\right)\right]. \tag{3.32}$$

The variance in Equation 3.29 is the average variance divided by the number of base models $N$. When $N$ is big enough, the variance term in Equation 3.29 will diminish. However, Equation 3.29 states that also the averaged bias and covariance should be taken into account while adding more and more models. In our setting, we are concerned with a small number of base models (i.e. mainly 2, one built on sensor data and the other one on simulation data). In addition, the decision of the transition from a single model to an ensemble has to be made. Therefore, it is more straightforward to deal with the whole term $\frac{1}{N}\overline{Var} + (1 - \frac{1}{N})\overline{Covar}$, as a variance-type error for the ensemble model and the corresponding bias as the average bias of single models $\overline{Bias}$, since

$$Bias(\bar{f}) = \left(\mathbb{E}\left[\bar{f}\right] - f\right) = \left(\mathbb{E}\left[\sum_{i=1}^{N}\frac{1}{N}f_i\right] - f\right) = \overline{Bias}. \tag{3.33}$$

The decision for the transition from a *data-based* fusion to a *model-based* fusion should be based on the level of the expected variance-type error together with the expected bias of the *data-based* fusion model. Let $f_{simulation}$ and $f_{sensors}$, two models each trained on simulation and sensors data, respectively, and a model $f_{fus}$ be trained using a *data-based* fusion approach. From the decomposition in Equation 3.29, we can conclude that the *model-based* fusion using an averaged ensemble with equal weights contribute to reducing the variance-type error compared to the *data-based* fusion model if

$$Covar(f_{\text{simulation}}, f_{\text{sensors}}) \leq \tau, \tag{3.34}$$

where the threshold $\tau = 2\left(Var(f_{\text{fus}}) - \frac{Var(f_{\text{simulation}}) + Var(f_{\text{sensors}})}{2}\right)$.

Equation 3.34 complies with the decomposition in Equation 3.29, stating that a lower covariance is always desired to reduce the overall ensemble error. It also confirms that enforcing a degree of diversity between the ensemble members through low covariance is favorable as it reduces the ensemble ambiguity, presented in the more general ensemble error decomposition schema [355]. Furthermore, once the single models are trained and built, the average covariance term can be estimated entirely without any knowledge of the true data labels or the real function $f$ to be approximated. From a practical point of view, this result confirms the usefulness of using simulation for enriching data with samples that reflect different patterns than the ones observed with

sensor data. This enrichment includes either new features that cannot be measured by sensors or observations that cannot be detected with sensors.

The bias of the ensemble model will be equal to the average bias of the single models in case of equal weights. So, it will achieve a lower bias than the single model with the highest bias. If the covariance between single models is lower than the threshold $\tau$ in Equation 3.34, then the system compares their average bias with the bias of the *data-based* fusion model to check if the variance-type error reduction with the average bias will establish a better bias-variance trade-off or not. If it is not, the system sticks to the *data-level* fusion.

The systematic fusion-level selection is validated by the previously described slot milling process [206]. We measure the process forces ($F_a$ and $F_p$ ) using sensors, aggregate them using an aggregation period of 0.1 ms, and predict the future expected forces each 0.1 ms for the next 10 ms. Monitoring the milling process forces enables control of both, process stability and quality [560]. 10 different process scenarios are investigated by varying the input parameters, namely speed, and feed. The resulting length of the time series for each scenario depended on the values of the input parameters and varied from 13 250 to 54 500 observations. The simulation is used for feature enrichment by generating features that typically cannot be measured during the process. For this purpose, the chip volume, the sum of time of engagement, the feed, and the mean of the cutting speeds are generated for each point in time to potentially enrich the feature space of the force measurements [206].

After solving possible mismatches between the sources, a unified feature set is created by joining new features generated by the simulation together with the lagged sensor measurements of the forces as sensor features [206]. The Random Forest regressor (RF) [101] is chosen for the prediction of $F_a$ and $F_p$. The results are presented for 10 cross-validation folds, where 9 scenarios (i.e. time series) are kept for training and 1 scenario for testing for each fold [206]. The prediction error is evaluated using the NRMSE (Equation 3.17). We used 5 lagged values for the time series of the forces as sensor features. For each time step, lagged sensor values are joined together with the simulation features for the current time step (i.e. simulation features are pre-calculated and stored, and only sensor data is streaming). We have also devised a binary feature based on the simulation features called the *activity feature*, which indicates the engagement situation of the tool (0: no engagement, 1: engagement) and is added to the fused set of features.

The results in Table 3.5 show that the feature-based fusion model outperformed models trained separately on each data source. The feature-based fusion is automatically selected as the best way to perform data fusion using simulation and sensor data after empirically computing the threshold $\tau$ derived in Equation 3.34 for the covariance and the average bias of the single models trained separately on each data source. Our theoretical insights are validated by showing a comparison with the model-based fusion [206]. These results are presented in Table 3.5. Furthermore, examples of empirical evaluations of the covariance, the threshold $\tau$ derived in Equation 3.34, the average

**Tab. 3.5:** Comparison between the NRMSE of predicted active and passive forces using different methods.

| Method | $RF_{Simulation}$ | $RF_{Sensors}$ | $RF_{Feature\text{-}based\ fusion}$ | $RF_{Model\text{-}based\ fusion}$ |
|---|---|---|---|---|
| **NRMSE** $F_a$ | 15.29 % ± 3.70 % | 24.24 % ± 5.95 % | 9.95 % ± 2.90 % | 16.25 % ± 4.27 % |
| **NRMSE** $F_p$ | 21.50 % ± 11.15 % | 33.63 % ± 5.38 % | 13.30 % ± 6.30 % | 19.69 % ± 5.37 % |

**Tab. 3.6:** Comparison between different measures for the fusion-level selection.

| Measure | $F_a$ | $F_p$ |
|---|---|---|
| Covar ($RF_{Simulation}$, $RF_{Sensors}$) | 21.18 | 4750.80 |
| Threshold $\tau$ | 109.92 | -13092.95 |
| Average bias ($RF_{Simulation}$, $RF_{Sensors}$) | 782.50 | 13714.16 |
| Bias ($RF_{Feature\text{-}based\ fusion}$) | 121.21 | 3204.13 |
| Var ($RF_{Feature\text{-}based\ fusion}$) | 330.40 | 8614.31 |
| Var ($RF_{Model\text{-}based\ fusion}$) | 140.86 | 13367.52 |

bias of single models, and the empirical bias of the feature-based fusion model for the predictions of $F_a$ and $F_p$ are shown in Table 3.6 to describe how the fusion level selection is made. In addition, the model variances of the model-based (i.e. ensemble) fusion and the feature-based fusion are reported. For $F_a$, the value of the covariance between single models is lower than the threshold, which guarantees that computing the ensemble model will reduce the variance type error and this is also confirmed by the reported empirical variance values in Table 3.6. However, validating the covariance threshold is not sufficient. The average bias of single models should also be compared to the bias of the feature-based fusion model. Comparing these values clarifies whether the model-based fusion will contribute to reducing the variance type error, but also alters the bias by increasing it with approximately a factor of six. For $F_p$, the value of the covariance between single models is higher than the threshold, which indicates that computing the ensemble model will not contribute to reducing the variance type error. In addition, the bias of the feature-based fusion is lower than the reported average bias. This observation confirms that the model-based fusion will reduce neither the variance nor the bias.

### 3.4.5 Initialization of Simulation Models Using ML Methods

First investigations are performed with respect to the initialization of simulation models using machine learning methods. To this end, two different simulation systems are considered, which are developed to investigate the milling and grinding processes, respectively. Parts of this section have already been published by the authors [207, 727].

#### 3.4.5.1 Learning of Frequency Response Functions and Compliance Models

The underlying system of a geometric physically based milling simulation (cf. Section 3.4.2) models tool vibrations by a set of decoupled damped harmonic oscillators [662]. The parameters of these oscillators have to be calibrated by measurements of the Frequency Response Function (FRF) of the considered machine-spindle-tool system, acquired by impact hammer tests [325]. In addition, the dynamic behavior of this system changes with different poses of the tool. The modeling is performed separately for all considered spatial directions- Here this corresponded to the $x$- and $y$- directions. Each oscillator is parameterized by identifying values for the modal mass $m_m$, the natural frequency $f_m$, and the damping constant $\gamma_m$. The complex response function [207]

$$\begin{aligned} G(\omega j)_d &= \sum_{q=1}^{Q} \left( \cos(\phi_d^{(q)}(\omega)) \cdot A_d^{(q)}(\omega) + \sin(\phi_d^{(q)}(\omega)) \cdot A_d^{(q)}(\omega) \cdot j \right), \\ d &\in \{x, y\}, \\ A(\omega) &= \frac{\frac{1}{m_\mathrm{m}}}{\sqrt{\left(\omega_\mathrm{m}^2 - \omega^2\right)^2 + 4 \cdot \gamma_\mathrm{m}^2 \omega_\mathrm{m}^2}}, \\ \phi(\omega) &= \tan^{-1}\left( \frac{2\gamma_\mathrm{m}\omega}{\omega_\mathrm{m}^2 - \omega^2} \right), \\ \omega_\mathrm{m} &= 2\pi f_\mathrm{m} \end{aligned} \tag{3.35}$$

is represented by superposing the amplitude and phase of the parameterized oscillators $q$ for each angular frequency $\omega$. By using measured FRFs, the loss function

$$\ell = \sum_i \left( \tilde{A_t}^{(i)} - \tilde{A_p}^{(i)} \right)^2 + \sum_i \left( \tilde{\phi_t}^{(i)} - \tilde{\phi_p}^{(i)} \right)^2, \tag{3.36}$$

can be derived, which is the sum of the squared deviations between the normalized values of the calculated amplitudes $\tilde{A_p}$ and phase $\tilde{\phi_p}$ and the normalized measured data $\tilde{A_t}$ and $\tilde{\phi_t}$.

Two different learning tasks are pursued in the performed research. On the one hand, pose-dependent FRFs are learned in order to reduce the measurement effort. On the other hand, the resource- and time-consuming and only semi-automatic task of calibrating the oscillator parameter values for different tool poses is replaced by machine learning.

The amount of data needed to perform the investigations is obtained by frequency response measurements in our laboratory for two different machine tools, Heller FT 4000 ($M_1$) and DMG HSC 75 linear ($M_2$), using a centrally composed statistical experimental design with star points. A total of 46 and 49 poses are measured for $M_1$ and $M_2$, respectively. An impulse hammer (Kister 8206) are used to excite a ball end mill (Fraisa X7400) with a diameter of 10 mm. The impulse response is measured by an accelerometer (PCB Piezotronics 352C23) attached to the tool tip [207].

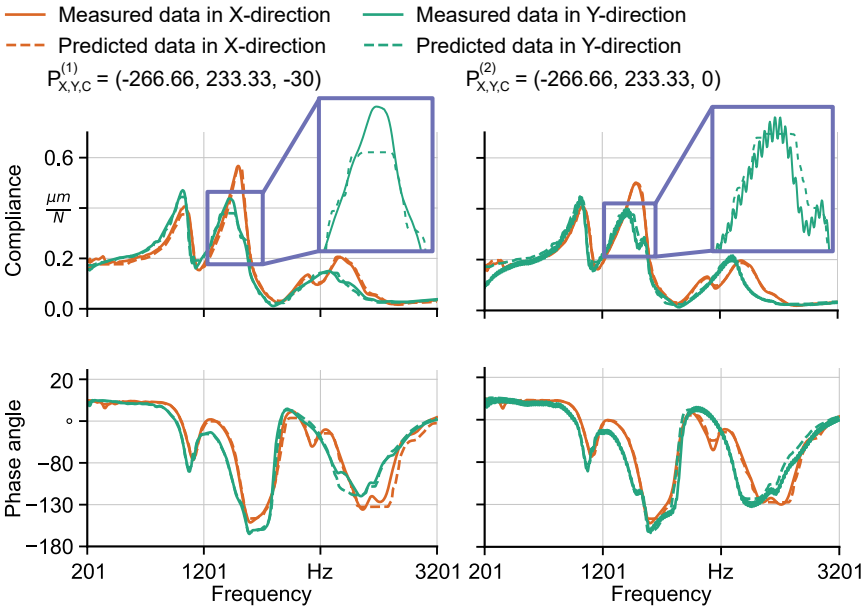



**Fig. 3.16:** Predicted and measured FRFs for machine tool $M_1$ [207].

As mentioned, two different objectives are investigated. For both objectives, let $\mathcal{X}$ be a set of $J \times N$ features sampled from an unknown distribution $\mathcal{D}$ and $\mathcal{Y}$ be a set of $K \times N$ targets labeled by a labeling function. The first objective involved the prediction of FRFs. For this, the measured FRFs of all $P$ considered measurement poses are discretized into data points by the frequency resolution $\Delta f$ such that $M$ is the number of frequencies examined for each pose. Each of the $N = P \cdot M$ data points contains a number of $K$ targets that included compliance amplitude and phase shift for the $x$- and $y$-directions of the machine coordinate system. Let $J$ be the number of features consisting of the frequency and the positions of the three axes defining the pose. For the second learning task, which is the prediction of modal parameter values for given poses, let $J$ consist of the three pose-dependent features and $N = P$. For the targets, let $K = 3 \cdot (Q_x + Q_y)$, where $Q_x$ and $Q_y$ are the number of oscillators in the $x$- and $y$-directions, respectively. Using this approach, the learning task attempts to represent the relationship between different interdependent oscillators of each compliance model across the two different vibration directions as well. For both learning tasks, the goal is to find a learner $h : \mathcal{X} \to \mathcal{Y}$ with respect to the distribution $\mathcal{D}$ [207].

Figure 3.16 shows an exemplary comparison between measured and predicted FRFs for two different poses using machine tool $M_1$. The phases are predicted with high accuracy in both $x$- and $y$-directions for both test poses. The amplitudes in the $x$-direction are predicted with a nearly non-visible deviation from the measured curves for both tests poses. There are two peaks visible in measured FRFs in $y$-the direction

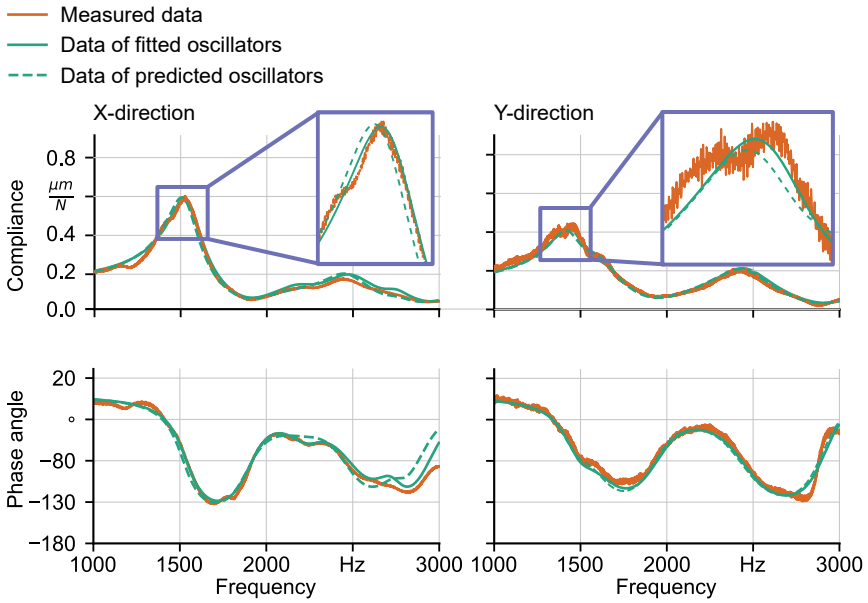



**Fig. 3.17:** FRFs resulting from predicted and manually calibrated compliance models for machine tool $M_2$ [207].

in a frequency range of 1200 Hz to 1700 Hz, which are very distinct for $P^{(2)}_{\mathrm{X,Y,Z}}$. For $P^{(1)}_{\mathrm{X,Y,Z}}$ the second peak can hardly be detected. This behavior could be represented by the model to a certain extent. In addition, the model also predicts a visible second peak for the remaining test poses, but the differences in the distinction between the two peaks across the poses can not be achieved. This effect is observed to be minimal in the data examined and needs to be analyzed in more detail in future research activities. In order to represent such behavior, more observations in which the fusion of peaks is present and that can be considered for the training procedure would be necessary.

Figure 3.17 shows a comparison between FRFs that resulted from predicted and manually calibrated compliance models in $x$- and $y$-direction for one specific pose using machine tool $M_2$. Generally, a high accordance is observed. Examining the zoomed-in areas of the FRFs, it can be seen that the shape of the measured FRF can only be represented coarsely by the manual fitting procedure. Since the fitted oscillator parameter values serve as target values for the learning task, the FRF, which is calculated based on predicted oscillator parameter values, can not reproduce the measured behavior in higher detail than the FRF that results from the fitting procedure. Nevertheless, there are only small deviations between the fitted and predicted data. Therefore, the learning of oscillator parameter values directly from given poses can be interpreted as successful.

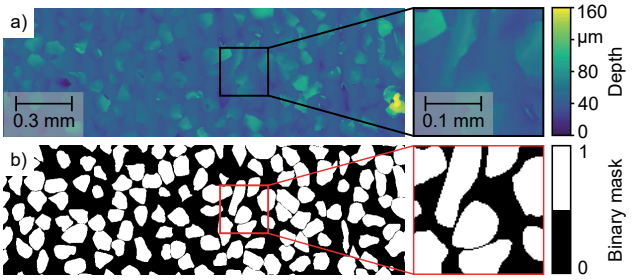



**Fig. 3.18:** Images of (a) measured depth information of grinding tool topographies and (b) the corresponding segmentation mask [727].

### 3.4.5.2 Augmented Semantic Segmentation for the Digitization of Grinding Tools

The following results are based on the investigations of [727]. The stochastic nature of grinding processes entails several challenges regarding process simulations. For modeling grinding tools, the choice of the methods for representing the individual grains and grain shapes have a significant influence on the accuracy of simulation results. Especially for single-grain scratch simulations and FE-based analyses, the identification of grains that adequately represent the overall characteristics of the tool used is crucial. In order to perform this identification successfully, an analysis of a huge amount of grains that have to be manually separated from the bond, is necessary. We developed a learning-based methodology, to automate this separation for digitized grinding tools by semantic segmentation. We have focused in particular on evaluating the prediction accuracy of the grain boundaries to be able to distinguish neighboring grains. This is crucial for a subsequent automated extraction of the grains. Figure 3.18a visualizes the measured depth information of grains in the bond. In addition, a manually generated segmentation mask is shown in Figure 3.18b. For the semantic segmentation, a novel neural network architecture [727] is developed, which is based on Fully Convolutional Networks (FCN) [420] (see Figure 3.19). In contrast to conventional FCN architectures found in literature, the channel information is gradually down-sampled by half in each transposed convolution layer instead of performing a single reduction operation. In addition, the up-sampling of the image dimensions is also spread across three twofold up-sampling steps.

Out of 4678 grains, 500 are used for testing purposes [727]. For hyper-parameter identification, random search [57] is used. In order to evaluate the prediction accuracy, the pixel accuracy [420] PACC $= \sum_i N_{ii} / \sum_i M_i$ is used, where $N_{ij}$ is the number of pixels of a class $i$ which are predicted to belong to class $j$ and $M_i = \sum_j N_{ij}$ is the number of pixels of class $i$. In addition, the boundary pixel accuracy (BPACC) is calculated as the pixel accuracy of the boundary pixels of each grain, which are estimated using a border following algorithm [665].

Figure 3.20 shows the results for grain segmentation for different numbers of grains incorporated for training using the developed approach versus applying an FCN-8. The

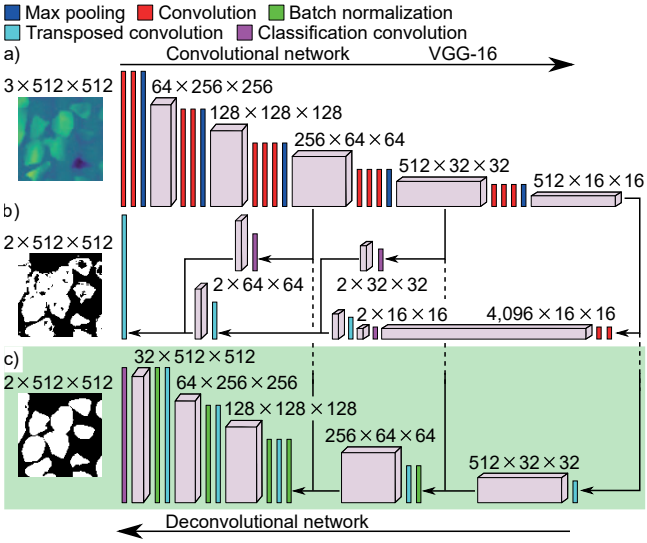



**Fig. 3.19:** FCN architecture including (a) the convolutional network, (b) a deconvolutional network based on FCN-8, and (c) the developed approach for the deconvolutional network [727].

conventional FCN-8 delivers insufficient results for all three investigated numbers of grains used for training. By contrast, using the developed FCN architecture, it is even possible to distinguish closely neighboring grains, if 2575 or 4178 grains are used for training [727].

To successfully train ML models, manual segmentation still has to be conducted in order to establish the required feature/target correspondences. To this end, data augmentation is used, to drastically reduce the necessary number of measurements. Different image manipulation techniques [605], e.g., rotation, flipping, or noise injection, are combined in a random sequence, to increase the amount of training data. In order to quantify the degree of augmentation, the augmentation factor per image (AFPI) is used as the number of generated images for each image based on measurements in a combined training set. Figure 3.21 shows the segmentation results using different numbers of grains used for training and different values for the AFPI, for the PACC, and the BPACC. The PACC value is higher the more grains are used and the higher the AFPI is. However, for the BPACC, a local optimum can be identified, indicating that high PACC values will not necessarily result in good segmentation results of the grain boundaries. Furthermore, choosing the AFPI as low as possible results in significantly lower training runtime [727]. Since grinding tools are constantly affected by tool wear during the process, the transferability of the model corresponding to the local optimum to different states of tool wear is also investigated. For further details, see the corresponding publication [727].

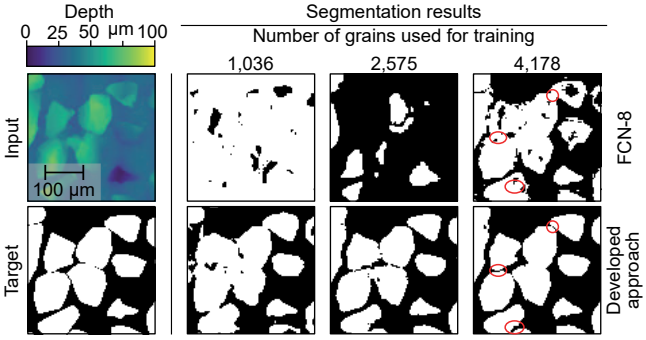



**Fig. 3.20:** Segmentation results using the developed approach in comparison with using an FCN-8 [727].

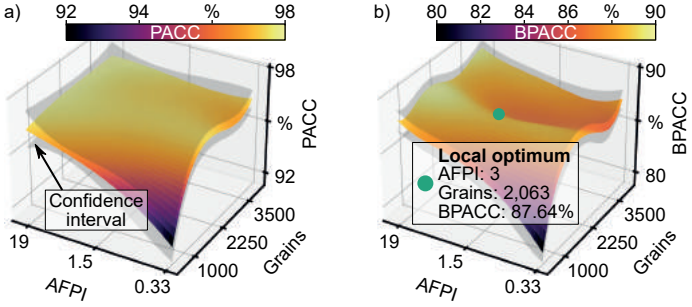



**Fig. 3.21:** Segmentation results with varying number of grains used for training and values of the AFPI [727].

### 3.4.6 Summary and Conclusions

This contribution has presented investigations that highlight different aspects of the combination of sensor and simulation data for scientific insights into machining. The methods are illustrated by milling, grinding, and tunneling processes as case studies. The main resource restriction in machining applications is processing time. Ensemble learning methods enable the real-time capability of simulated predictions. ML enhances the simulation. Another resource, which is often restricted in real-world applications, is (labeled) data. Various data fusion strategies were discussed to combine sensor and simulation data for real-time predictions of process characteristics of milling operations. Here, simulations and ML help each other.

In addition, ML methods were used to initialize simulation models, specifically the dynamic model of a geometric physics-based milling simulation system and the tool model of a grinding simulation. ML helps the simulation.

The presented results emphasize the significant potential of combining sensor data, simulation results, and ML methods for the analysis and optimization of manufacturing processes in the context of Industry 4.0.

## 3.5 High-Precision Wireless Localization

*Janis Tiemann*

**Abstract:** Recent developments in Ultra-Wideband (UWB) wireless communication enable wireless localization as a link between the digital and the physical world. With the technological advances, the achievable precision and accuracy is increased dramatically such that novel applications exploiting this precise cyber-physical link become feasible. Autonomous swarms of robots, precise and scalable tracking of goods or safety applications are within the reach of this potential. However, the increase in communication required for such capabilities to become feasible is constrained by bounds of channel utilization, energy consumption, and intelligent information distribution. Therefore, novel approaches for maximizing information and localization throughput while minimizing channel utilization and power consumption and maintaining precise localization results are crucial to overcoming technology barriers and ultimately enabling a connected cyber-physical world. Promising approaches are novel localization-specific protocols to coordinate channel access among localization targets in order to achieve reliable data rates while minimizing actual power consumption. Further, intelligent approaches are required to increase the achievable accuracy for these resource-efficient localization approaches such as Time-Difference of Arrival (TDOA). In those cases, additional parameters of the radio channel can be exploited to obtain quality indicators for measurement and mitigate outliers or generally improve the localization accuracy through adequate estimation. In the following, the requirements of several applications are analyzed, and an overview of the solution space in terms of channel utilization, energy efficiency, and accuracy is given. Based on these requirements, solution approaches are presented to improve both channel utilization and energy efficiency. Further, approaches to increase the achievable accuracy in challenging environments are illustrated and evaluated. It is shown that novel approaches for high-precision wireless localization enable novel applications by employing localization-specific protocols and methods to improve the accuracy despite challenging conditions.

### 3.5.1 Introduction: Precise, yet Scalable Wireless Localization

Wireless localization is seen as an enabler for many applications requiring a link between the physical and the digital world. Many approaches exist to achieve this connection utilizing a wide range of technologies. An overview of the capabilities in terms of accuracy and range of those technologies is given in Figure 3.22. The diagram illustrates the difference between widely adapted communication technologies, such as cellu-

lar networks and consumer-grade wireless standards, and more application-targeted standards, such as UWB. Early methods for localization were solely based on cell-id and/or sector differentiation. However, novel approaches are capable of increasing the accuracy for tracking and guiding applications. More dedicated localization systems like Global Navigation Satellite Systems (GNSS) can achieve exact localization results but are limited to outdoor Line of Sight (LOS) operations, as they suffer severely from multi-path fading. Here, UWB technology is key to overcoming the technology barrier, enabling highly precise indoor localization through precise Time Of Arrival (TOA) estimation. For localization methods enabled by 5G mmWave communications, see Section 5.5.

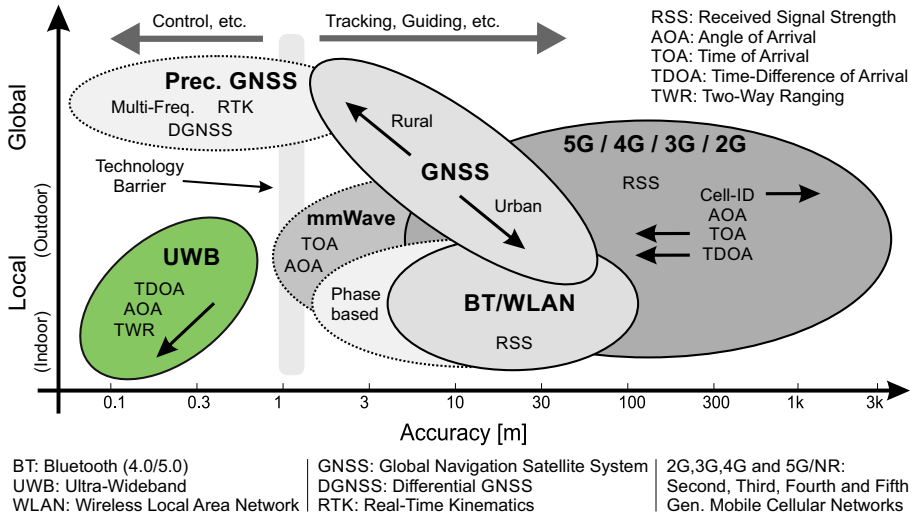



**Fig. 3.22:** Illustration of the field of localization capabilities of wireless communication technologies in terms of accuracy, range, and place of usage.

Hence, a significant amount of research is challenged by this newly available technology. Integrated UWB solutions enabled a new degree of accuracy in wireless localization. In contrast to many signal strength-based methods, these TOA-based approaches emerged as the most promising candidate for accurate and reliable measurements in the centimeter range. Due to the usage of high bandwidths that enable sharp pulse-based modulation, these UWB systems are capable of resolving many multi path-induced errors in TOA estimation [228].

Due to this reason, research in several areas of application arose utilizing this newly available connection between the digital and the analog world. One particular area of interest is the field of massively scalable and low-power localization for logistics

in the Industry 4.0. Here, low-power localization devices are key to monitoring and optimizing the whole production process, as illustrated in Figure 3.23.

For large-scale deployment, however, the scope of the most current research is insufficient as it neglects scalability and multi-user interference by utilizing localization approaches that require the exchange of many messages. This introduces not only significant resource usage in terms of channel utilization, but also requires substantial usage of energy for message exchanges. In summary, the addressed points of this section are listed as:

- **Massive Multi-User Scalability**
- **Minimal Energy Consumption at the Mobile Units**
- **High Accuracy Suitable for Control-Grade Applications**

### 3.5.2 Related Work: Evolution of wireless localization within CRC

The methodological continuity of the work within the CRC is illustrated by several publications:

- **UWB Indoor Positioning for UAVs (Unmanned Aerial Vehicles)** [682] uses Two-Way Ranging to enable UAV Indoor Navigation. Limitations in multi-user scalability motivated further research.
- **Multi-User Interference Analysis** [685] is a detailed look into multi-user interference for UWB systems and wireless clock synchronization. Further analysis is conducted in [687] which provides an analytical model for the interference.
- **ATLAS - TDOA-Based Localization** [684] overcomes multi-user scalability limitations and presents an open-source approach.
- **Scalable Multi-UAV Indoor Navigation** [688] demonstrates the scalability and accuracy for control-grade systems.

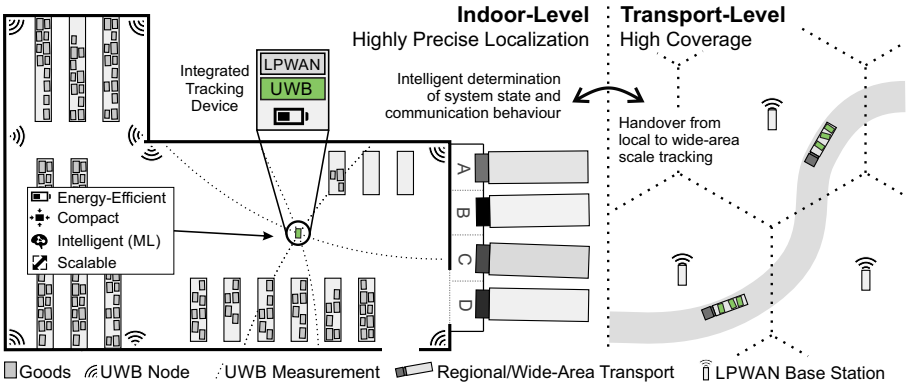



**Fig. 3.23:** Illustration of a potential usage scenario.

- **Enhanced UAV Indoor Navigation** [681] improves the precision and range for control-grade systems.
- Further work provides an open-source extension **ATLAS FaST** to the ATLAS approach in order to improve energy efficiency and reliability [678, 679, 687].
- Finally, the PhD thesis **Scalability, Real-Time Capabilities and Energy Efficiency in Ultra-Wideband Localization** incorporates and extends the key findings of the research, see [683].

### 3.5.3 Approaches: Scalable, Real-Time Capable Energy Efficient Localization through UWB

The following sections present approaches for high precision wireless localization based on the work in [680, 683, 687]. The approaches are tailored to find a sweet spot in the trade-off between multi-user scalability, energy efficiency, and achievable accuracy for wireless localization. In a first step, the underlying ATLAS localization system is highlighted briefly.

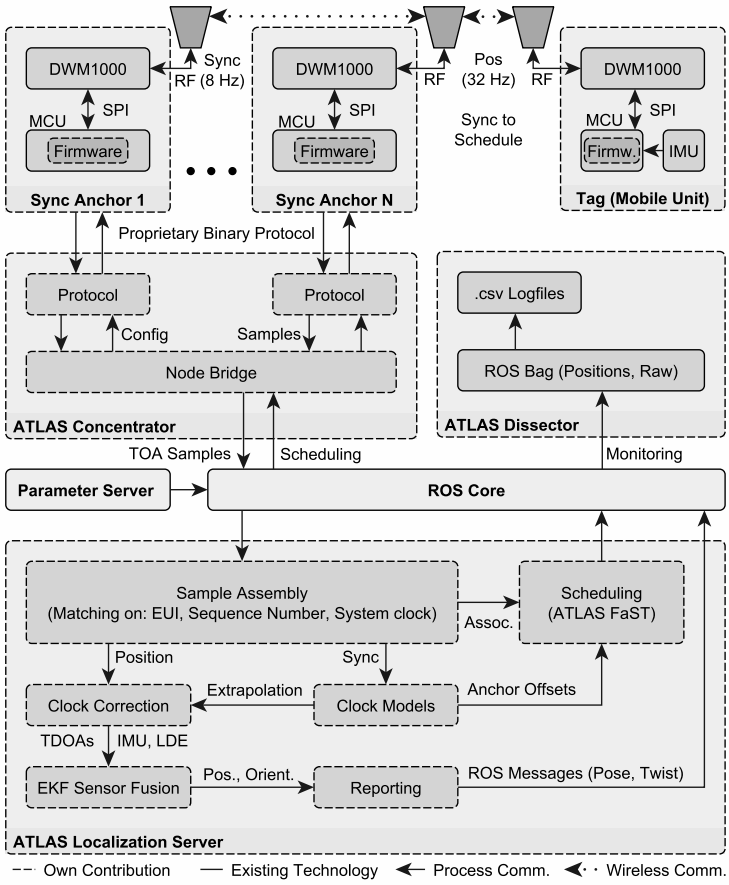



**Fig. 3.24:** System architecture of the ATLAS RTLS utilizing the FaST scheduling approach. The approach is based on the robot operating system (ROS) to allow for flexible and modular design as well as seamless integration with mobile robots.

### 3.5.3.1 ATLAS: Open-Source TDOA-Based Localization

In the context of the ATLAS Real-Time Localization System (RTLS), we propose building upon the TDOA topology in which the mobile nodes transmit a single message and the infrastructure receives (R-TDOA). Wireless clock synchronization is used to achieve a common time base among the clocks of the static infrastructure-based anchor nodes. Yet, random access is incapable of providing guaranteed update rates required by many robotic applications. Furthermore, with an increase of mobile nodes, the quality of the localization will degrade due to missed synchronization frames, as pointed out in [685]. This means that the real-time requirements for control-grade applications such as indoor UAV navigation cannot be met at scale by random access.

### 3.5.3.2 ATLAS FaST: Lightweight Scheduling for Control-Grade Applications

Based on previous work [684], we thus propose a novel lightweight scheduling protocol that seamlessly integrates with the wireless clock synchronization needed for proper operation, see [687]. As depicted in Figure 3.23, our approach aims to support scalable, low-power, reliable, and real-time capable wireless localization. In the following, we will document the approach and experimentally evaluate its capabilities in an industrial setting. The open-source implementation provided will enable the usage of scalable UWB localization in the robotics and automation community, see [678, 679].

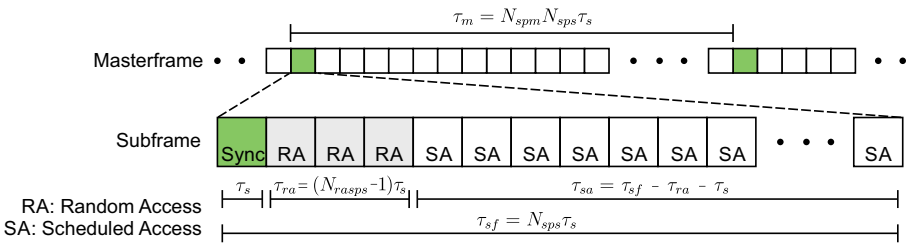



**Fig. 3.25:** Temporal structure of the ATLAS FaST localization specific Time Division Multiple Access (TDMA) scheme. The structure utilizing subframes enables multiple spatially distributed synchronization cells.

The publication [687] illustrates the continuity of the development of wireless localization building upon the widely used Robot Operating System (ROS). In contrast to previous work, which focuses mainly on plain localization aspects, a significant increase in scalability and real-time capabilities is provided, presenting a method that allows for scalable localization without degradation in the performance of the localization results.

The proposed system architecture is depicted in Figure 3.24. The ATLAS Concentrators are capable of connecting to multiple anchors and synchronize anchors through direct connection.

The chosen TDOA topology used in our system design enables our mobile nodes to be very energy efficient, as pointed out in Section 3.5.3.3. Only a single frame needs to be transmitted in order to obtain a full localization result. However, if a guaranteed update rate is desired, bi-directional communication, synchronization, and association with the system's scheduler are required. Through the modular system architecture, the individual components can be developed independently, which significantly reduces the time required to integrate application-specific features or competition-specific constraints. As mentioned before, the modularity of this concept is illustrated in Figure 3.24.

To lower the overall system complexity, the synchronization request is the same as any other positioning frame transmitted over the UWB channel. It consists of the tags Extended Unique Identifier (EUI) and a sequence number that is increased with

every new message. Depending of the application this message can be extended using Inertial Measurement Unit (IMU) data or battery status.

After accessing the channel, the tag immediately goes back to sleep and waits for the next sync frame. In the meantime, the anchor nodes receive and process the requests. Successful random access requests are propagated to the scheduling engine that has a database of pre-known period configurations and priorities from which it can prepare a response through the requesting tags EUI. This procedure simplifies the configuration overhead as the system is infrastructure-based. It also allows for seamless graceful degradation of the update rate if the overall system capacity limit is reached.

#### 3.5.3.3 ATLAS Low-Power Timekeeping under Resource Constraints

For random access based schemes, proper absolute time-keeping is not required. For the proposed scheduled access though, knowing the correct absolute time is of the essence. However, maintaining the high-frequency clock of the transceiver modules is associated with significant loss of power and therefore, battery life. Due to this reason, a concept of time-keeping, synchronization, and constant calibration should be employed for proper low-power operation in the scheduled scheme. This basic concept can be utilized for real-world implementations in combination with ATLAS FaST to support future low-power applications.

The transceiver should run in a mode that does not enable the high-frequency clock sources and, thus, loses its absolute time knowledge. Therefore, the proposed scheme employs intelligent switching between different clock sources to maintain the transmission schedule in the mobile nodes.

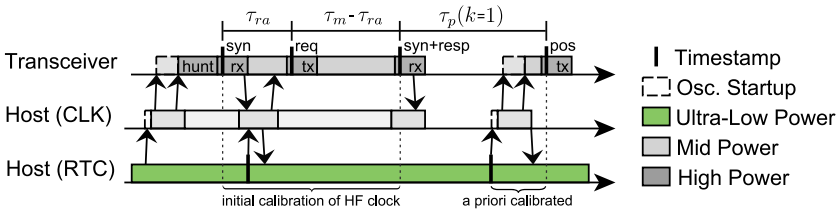



**Fig. 3.26:** An approach for low-power association with the system controller.

The initial association procedure at the mobile node is depicted in Figure 3.26. $\tau_{ra}$ is the duration between the sync frame and random access, $\tau_p(k)$ the duration between the response frame and the $k^{\text{th}}$ positioning frame, $\tau_m$ the duration of the master frame, and $k$ is the localization sample iterator. There are three main components of drawing power from the battery: the transceiver, the host controller, and a Real-Time Clock (RTC) within the host controller. Here, the internal RTC is used as the keeper of the overall system clock. The transceiver oscillator is calibrated using two consecutive

UWB synchronization frames. The resolution of the RTC is insufficient to calibrate upon initial association. Here, the re-association should be utilized as it provides a sufficient time-span enabling RTC drift calibration.

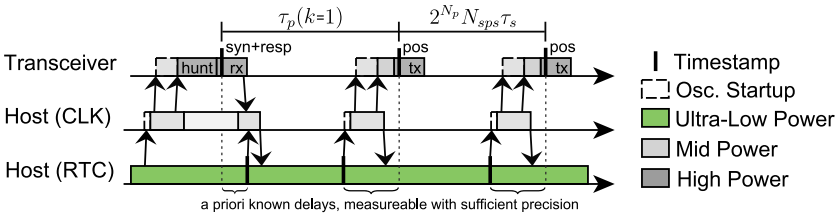



**Fig. 3.27:** Approach for low-power re-association with the system controller.

The typical re-association procedure, including the following transmissions, is depicted in Figure 3.27 where $N_p$ is the positioning period exponent for a mobile node, $N_{sps}$ the number of slots per subframe and $\tau_s$ the slot duration. The transceiver and the host-controller can be configured to a sleep state. An RTC timer can then be configured to wake up the host controller for the next positioning frame. Here, the delays for waking up the host controller and the transceiver need to be calibrated in the implementation phase. The processing times and oscillator start-up times need to be accounted for. Based on this, the RTC wake-up time can be configured. Although limited precision is given by the RTC, which mostly runs at 32.768 kHz, the available resolution of around 30.52 µs is sufficient to stay within the margins of the scheduling scheme. However, the potential error through clock-traversal needs to be accounted for in the slot-duration dimensioning for the scheduled scheme.

#### 3.5.3.4 Wireless Signal Assessment to Improve Overall Accuracy

Due to the utilization of a TDOA-based localization scheme, another variable in the localization solution is introduced. Instead of rangings as with TWR-based localization, TDOA utilizes the time difference of the arriving signal. Hence, the localization is inherently more challenging. Therefore, countermeasures such as introduced in [680] are required to obtain precise localization results.

One of the main benefits of UWB is the availability of additional information of the received signal. Here, methods can be applied that leverage this information in order to weigh the individual measurements. Figure 3.28 illustrates this concept. Here, the ratio between the first path and the remaining energy of the channel impulse response is taken in order to weigh the measurements in the Extended Kalman Filter (EKF) used to obtain the localization results.

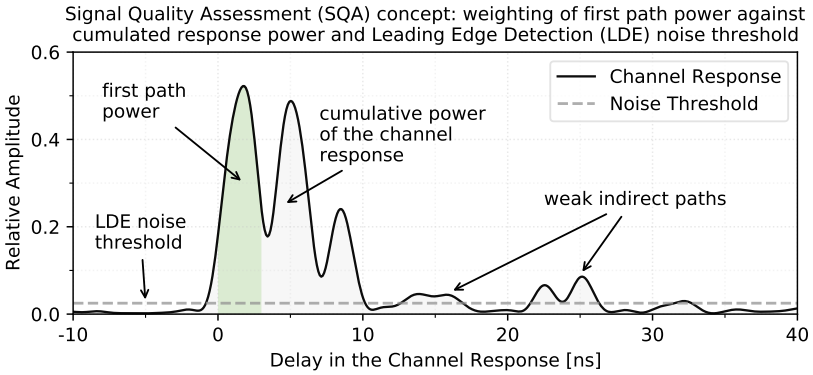



**Fig. 3.28:** An approach for signal quality analysis to improve accuracy. Note that the depicted channel response is interpolated from ten experimentally recorded sets in which the resolution is 1 ns per sample.

There are concepts that may be capable of extracting even more information such that even passive localization or simultaneous localization and mapping in certain scenarios are feasible. An initial evaluation of machine learning on these channel responses can be found in [686].

### 3.5.4 Results

In order to evaluate the performance characteristics of the presented approaches, selected evaluations based on the work in [683, 687] are discussed.

#### 3.5.4.1 Scalability of the ATLAS FaST Approach

One of the main benefits of random access is the simplicity of implementation due to the lack of coordination. Tags can be implemented in a transmitter-only design, allowing for a long battery lifetime through intermediate sleep modes. The main down-side, however, is the lack of predictability. For many applications, especially in real-time control of autonomous systems, defined update rates are required. Here, a higher overall throughput is not directly useful, if long inter-arrival times are expected. Therefore, systematic channel access is desirable.

With increased loads, low energy, battery-powered applications suffer from the non-reception of localization frames. Therefore, the effective energy per position ratio increases, leading to decreased efficiency. The ATLAS FaST scheme is designed to overcome these issues. In the following, the achievable inter-arrival times and reliability are analyzed. $\hat{\tau}_{nf}$ considers a processing time of $\tau_{proc}$ and the partial preamble reception effect.

The resulting throughput is the superposition of these individual effects that depend highly on the implementation of the receiving side in practical systems. Due to the unique characteristics of the UWB PHY, a non-destructive R-TDOA scheme is feasible as the throughput does not degenerate with increased loads in the available ranges. However, due to the non-reception of frames, the real-time capabilities for the localization systems degrade as a defined update rate cannot be guaranteed. This is especially severe for applications, requiring tight real-time constraints.

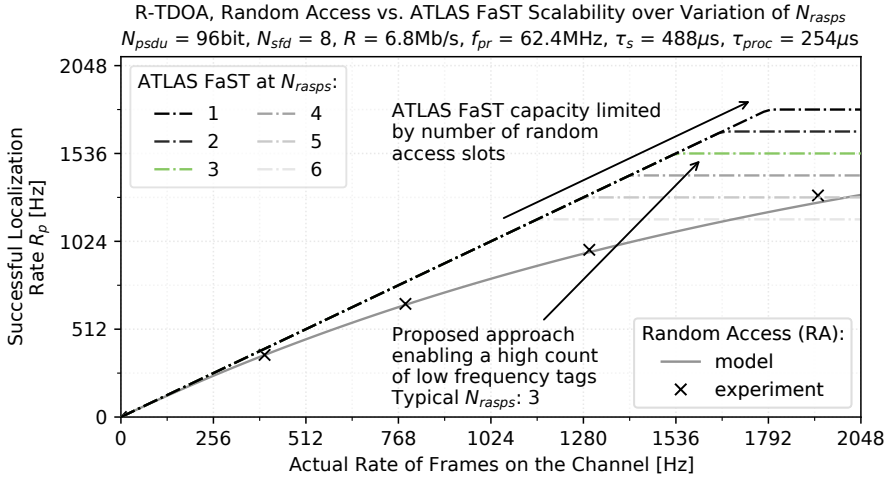



**Fig. 3.29:** Experimental vs. analytical throughput analysis for R-TDOA-based localization in which the infrastructure estimates the location.

ATLAS FaST overcomes this issue. Due to the slotted approach and the competition-free channel access in the scheduled slots, the successful positioning throughput scales linearly with the positioning frame load, as depicted in Figure 3.29. Here $N_{psdu}$ is the number of transmitted payload bits per packet, $N_{sfd}$ the UWB start of frame delimiter size in symbols, $R$ the effective data rate, and $f_{pr}$ the mean pulse repetition frequency on the physical layer. The slot duration $\tau_s$ is chosen such that there are 2048 available slots per second. Non-reception due to noise-induced frame error rates were neglected in this analysis as this will highly depend on the link budget the wireless localization system planner will provide for the given setup. However, the variation of the number of random access slots will define the upper bound of the system's capacity.

Alongside the throughput of the proposed FaST approach, Figure 3.29 depicts an analytical model for R-TDOA obtained in a scaled experiment. Here we can compare the multiuser scalability of the proposed approach with the previous, random access-based R-TDOA.

It is clear that the scheduled approach allows greater throughput than that produced by the same underlying positioning load using random access-based R-TDOA. Also, the analytical model for the R-TDOA throughput matches the experiment closely. The current implementation supports only static adjustment of the amount of random access slots $N_{rasps}$; for optimal performance a dynamic adjustment would be necessary. However, even when using a static set of six random access slots, FaST supports more than 1000 mobile units at 1 Hz considering a high re-association interval.

#### 3.5.4.2 Energy Utilization of Localization Schemes

In order to provide a simplified comparison between the energy consumption at the mobile unit for different channel access schemes, the bar chart in Figure 3.30 summarizes the results of this work relative to the mostly used Symmetrical Double-Sided Two-Way Ranging (SDS-TWR)-based topology now considered state of the art.

Here, a minimal set of four anchors is the baseline for all topologies. Multiple different ranging or localization schemes are considered: R-TDOA with random access as introduced in the early ATLAS implementations [684]; T-TDOA as illustrated by [256, 378] with transmitting anchors and a receiving-only mobile unit, similar to GNSS; typical single-sided Two-Way Ranging (TWR), which requires a message exchange of two messages per ranging; TWR with Multiple Acknowledgments (TWR-MA), utilizing a repeated response to estimate clock offset; combined TWR, which orchestrates a TWR exchange by pre-defined individual response time offsets for a set of anchors; and, finally, symmetric double-sided TWR (SDS-TWR), which utilizes symmetric response times to cancel out clock drift during ranging. The last one requires three messages for basic SDS-TWR, but it can also be configured to report calculated ranges in SDS-TWR-R, requiring another message at the end of each ranging. For the SDS-TWR topology no reporting from the anchor side is considered. So the range information is available only at the infrastructure side as it is using R-TDOA or FaST. But SDS-TWR-R considers reporting and therefore consumes even more energy than plain SDS-TWR.

Keep in mind that the energy consumption of the TWR- and T-TDOA-based topologies increases linearly with the number of anchors in the setup, while the energy consumption of R-TDOA-based localization is independent of the anchor count. Therefore, it can be stated that even at the worst-case scenario the proposed topology outperforms traditional TWR-based schemes in terms of scalability and energy usage.

By calculating the power per transmitted and received UWB frame for the transceiver system and the fundamental requirements for the localization topologies, er can determine an idealized maximal battery life. An exemplary positioning rate of 1 Hz, a re-association period of 300 s, and a reliability look-ahead $N_r = 1$ were chosen.

The power consumption is calculated based on the datasheet of the DW1000 transceiver resulting in $E_{tx}$=40.7 mJ and $E_{rx}$=76.9 mJ. Note that the different PSDU (Physical layer Service Data Unit) sizes, that are required for the different schemes

are simplified to the lowest size of the R-TDOA scheme due to the implementation-dependency of this value. The discrepancy between TDOA- and TWR-based schemes would be even greater.

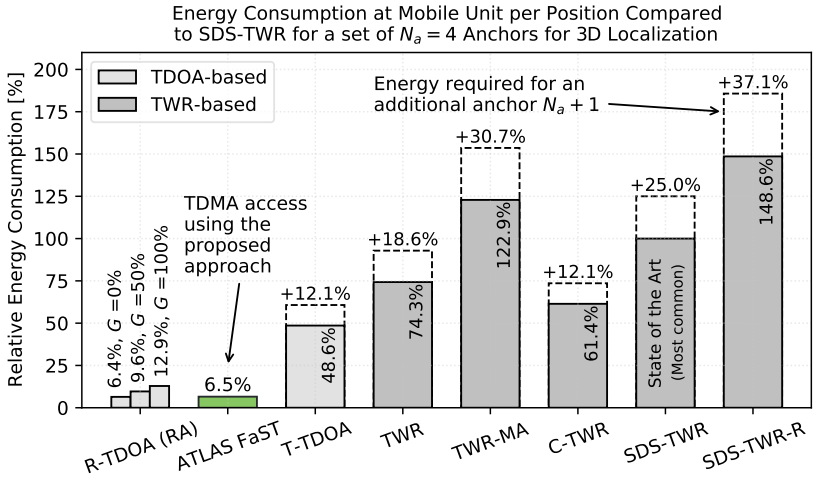



**Fig. 3.30:** Exemplary analytical evaluation of energy utilization per localization approach. Note that the main factor influencing the energy consumption is the amount of messages required per localization result.

The resulting battery lifetime is depicted in Figure 3.30. The battery lifetime of the T-TDOA- and TWR-based approaches is lower by orders of magnitude due to the dependency on the number of anchor nodes $N_a$ inherent in those topologies. For the R-TDOA-based approaches, the FaST results are close to those of random access, which is the baseline for low power consumption at the mobile node because there is no additional overhead. Therefore, it can be stated that through the planned scheduling with infrequent synchronization, FaST is capable of tracking many low-power devices without interfering with the real-time requirements for critical applications.

### 3.5.4.3 Accuracy

For the evaluation of the accuracy, several tracks were followed. One of the main contributions were lab experiments evaluating the accuracy of the signal quality assessment under motion capture tracking as published in [680]. To achieve internationally comparable results, the participation in competitions is key, as it is the only way to benchmark localization results comparably. In the following, based on [683], the participation in two international competitions is highlighted briefly.

The EvAAL competition alongside the *Seventh International Conference on Indoor Positioning and Indoor Navigation (IPIN2016)* in Alcalá de Henares, Madrid, Spain is used to provide a comparable basis for localization systems in the context of robotic tracking. In the fourth track, *Indoor Mobile Robot Positioning*, six teams registered. While four teams registered *in-track*, two teams were *out-of-track* from within the organization team and were only evaluated for comparison. In [507, 549] the organization and results of this competition are covered in detail.

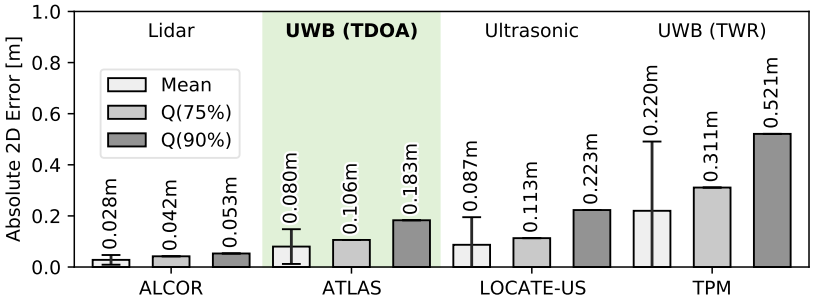



**Fig. 3.31:** Reconstructed competition results for the EvAAL 2016 as a bar chart. The official competition metrics were based on the third quartile Q(75 %).

The goal of the competition was to localize a mobile robot following a predetermined track that has to be obtained by the evaluated system. A set of four poles to mount the system under evaluation was provided by the organizers to cover the 12 m × 6 m evaluation area. The systems were evaluated sequentially so that one of the challenges of this competition was a maximum set-up time of 30 min that was extended to 45 min during the competition. The metric used for evaluation by the organizers is the third quartile of the Euclidean distance to the track, due to the missing temporal component of the evaluation setup.

A bar chart of the results is shown in Figure 3.31. In addition to the Q(75 %) quantile used for ranking, the mean absolute Euclidean error and the Q(90 %) error are depicted. It should be noted, that through the use of TDOA-based localization an additional unknown error is introduced, which generally lowers the accuracy of these systems. This has to be considered when comparing the ATLAS results and accuracy with the TWR-based system of the TPM team.

It can be seen that the ATLAS approach, although utilizing TDOA instead of TWR, can provide accurate robot tracking results and is, therefore, usable in the context of real-time robotic movement tracking.

To evaluate the localization approach in a more challenging environment, we participated in the fifth iteration of the *Microsoft Indoor Localization Competition* (MILC) co-located with the CPS-Week 2018 in Porto, Portugal. Previous competitions featured

broad participation from science and industry, see [422, 423]. In this competition, 34 teams submitted abstracts, 26 systems officially registered, and 25 systems showed up in Porto. However, only 22 systems could provide data and were evaluated.

The second category that used custom infrastructure such as UWB, was required to report 3D locations. Up to ten anchor nodes were allowed in the evaluation area. The teams had a time slot of 8 hours to set up and calibrate their systems.

The teams were evaluated using a mobile laser scanner-based ground-truth system. The organizers allocated a 15 min evaluation slot per team and fixed order. However, during the competition, the handling was more dynamic so that teams with a non-functioning system during their slot had the chance to debug their system and evaluate it later. Furthermore, since the competition area was the main staircase for the attendees of the four conferences held during the evaluation, the systems had to cope with the obstruction and interference of visitors and observers.

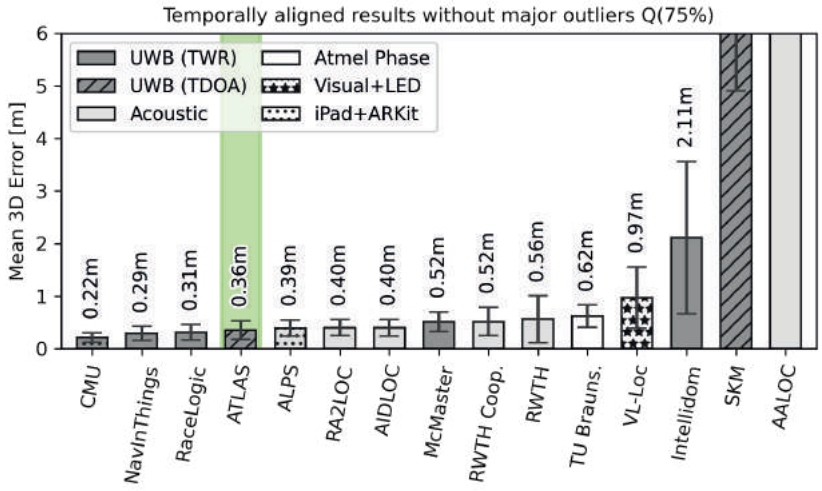



**Fig. 3.32:** Reconstructed bar chart of the mean 3D accuracies from the MILC 2018 data after temporal alignment and elimination of the 25 % largest errors for each team. Note the diversity of sensors for the individual localization methods.

The organizers chose the mean Euclidean error as the metric for evaluation. This error, however, is strongly influenced by the large outliers at the beginning of the evaluation run. Furthermore, a temporal offset of around 1 s between the ground-truth and the ATLAS-team result trajectory is observed. Due to this, additional errors were introduced into the evaluation.

Note that with improved temporal alignment, the mean error of the ATLAS team improves. Since it was desirable to compare the performance of the properly initialized

ATLAS results from the rest of the teams, the mean of the best 75 % of the temporally aligned results was evaluated. In order to provide a fair basis for comparison, the best 75 % of all teams were considered. Therefore, the analysis does not merely remove the outliers but improves the results of all teams as depicted in the lowest bar chart.

As depicted in Figure 3.32, when considering the aforementioned points, the actual performance of the ATLAS system is significantly better than indicated by the official results. Even though the ATLAS system uses a TDOA-based approach, which has significant downsides in accuracy, due to the proposed implementation, the results are comparable and often better than the other TWR-based approaches.

### 3.5.5 Conclusion

This section presented the basic capabilities, challenges, and novel solutions for high precision wireless localization. Here, the bounds in channel utilization, energy consumption, and achievable accuracy are limiting factors for the feasibility of a wide range of applications.

Therefore, novel approaches for maximizing information and localization throughput while minimizing channel utilization and power consumption and maintaining precise localization results were shown to overcome technology barriers and ultimately enable a connected cyber-physical world. The ATLAS FaST scheduling scheme and approaches for improving the achievable accuracy are highlighted.

It could be shown that based on these requirements, solution approaches are capable of improving both, channel utilization and energy efficiency. Further, approaches to increase the achievable accuracy in challenging environments can be successfully applied to improve TDOA localization accuracy.

In future work, challenges such as reducing the amount of required anchors, ad hoc configuration, and in-depth signal quality assessment with machine learning is envisioned. The potential for high precision localization enabled by the characteristics of novel communication solutions such as new UWB standards along with 5G and future 6G systems is huge.

### 3.5.6 Acknowledgments

In addition to the CRC 876, part of this work has been supported by the federal state of North Rhine-Westphalia and the “European Regional Development Fund” (EFRE) 2014-2020 in the course of the “CPS.HUB/NRW” project under grant number EFRE-0400008.

## 3.6 Indoor Photovoltaic Energy Harvesting

*Mojtaba Masoudinejad*

**Abstract:** Advancement in the field of electronics has enabled devices with ultra-low power (ULP) demands. However, Industry 4.0 devices made of such components will be empowered for long operational periods, specifically in remote or hardly accessible environments. Scavenging energy from the environment is a very common technique to tackle this energy supply issue. Different energy harvesting principles are available that exchange energy from distinct forms into electric power. However, we focus on Photovoltaic (PV) energy harvesting because it is the most mature technique.

In addition to the small size and weight limitation of Industry 4.0 and IoT devices, which require constraints on the size of a PV cell, they are applied mostly in indoor environments. Hence, the specific behavior of PV modules for indoor applications under artificial lighting is analyzed here. Using a systematic data acquisition procedure, typical PV models are adapted for the ULP harvesting environments. A normalization procedure is introduced during parameter tuning because common techniques are not applicable on PV modules when operating in low and ultra-low lighting conditions. Guidelines are provided to assure the numerical stability of parameter tuning of models.

The second layer of a two-fold model represents the relation of tuned curves with the environmental factors. Using a relative representation according to the highest light intensity, these models can be applied for different conditions. Performance of the overall model is evaluated on an extra dataset collected from a new environment showing model errors less than 6 % in the worst-case condition.

Parts of this section are taken from [434] with the consent of the author.

### 3.6.1 Introduction: Energy Harvesting

Energy is available in nature in a wide range of forms, from heat and mechanical energy to the energy stored in electromagnetic waves and light photons. Any method which enables scavenging these energies can be called *energy harvesting* and a transducer that converts them into the desired form is an energy-harvesting device, or *harvester* for short. Energy harvesting has a long history, since windmills, which convert wind energy into mechanical energy to mill grains, date back to the 9th century. Nonetheless, modern energy harvesting converts energy into electricity.

Wind, solar, Photovoltaic (PV), piezoelectric, thermal, radio frequency and tidal energy harvesting are only some examples of modern energy-harvesting techniques. However, techniques based on the conversion of light (specifically solar energy) into

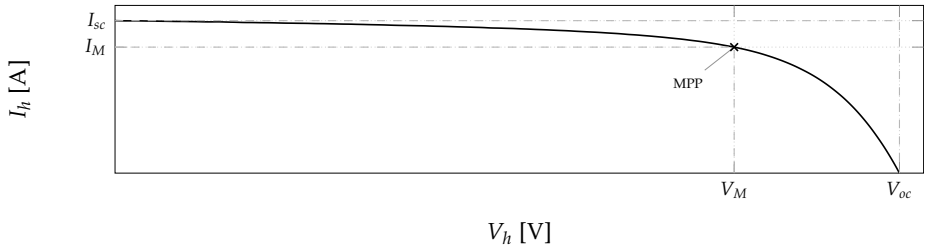



**Fig. 3.33:** I-V curve of a Sanyo/AM-1464 PV module measured under florescent light.

electricity are more mature than others due to long research and applications in diverse fields. While the term *solar* is commonly known, it includes different principles for converting the sunlight into the electricity. Nevertheless, PV is a specific form that uses semiconductor materials and technologies converting light into electric energy. In a simplified version, a PV transducer can be considered to use the inverse principle of a Light Emitting Diode (LED). In addition to the maturity of PV harvesting, their integration into the ULP Industry 4.0 hardware is the main reason we focus on them below.

#### 3.6.1.1 PV Energy Harvesting

"A PV transducer is a semiconductor device generating electrical power when illuminated with photons [434]". These semiconductors have electrons in their valance energy band, which is weakly bounded. This bound can be broken by any photon that has higher energy than the band gap and causes movement of the electron to the conduction band. As long as enough photons are illuminated on the semiconductor surface, a photon's energy is converted into the flow of electrons to convert light into electric energy. Due to the nature of this conversion, PV can generate Direct Current (DC). Consequently, a PV harvester is an electric source, but not an ideal one. According to the operational condition of the PV module it can act as either a voltage or a current source. The common I-V behavior of a PV module is shown in Figure 3.33.

As can be seen in Figure 3.33, for a large portion of the voltages (mainly in the lower range) the PV module acts as a current source while for most current values (in a small voltage range) it acts as a voltage source. However, none of these sections are ideal because they are pure vertical or horizontal lines. The bending point where the behavior between source form changes is critical because the maximum power can be extracted from the module at this specific point. Hence, it is called Maximum Power Point (MPP) and techniques used to keep the operational point at this point are called Maximum Power Point Tracking (MPPT). When $V_h$ and $I_h$ describe the harvested voltage and current subsequently, MPP can be found mathematically from Equation 3.37.

$$\frac{\partial P_h}{\partial V_h} = \frac{\mathrm{d}I_h}{\mathrm{d}V_h} \cdot V_h + I_h = 0, \qquad \text{at: } V_h = V_M \tag{3.37}$$

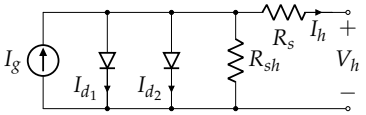



**Fig. 3.34:** Equivalent 2 diodes circuit model of a PV transducer including parasitic resistances.

In addition to the MPP, there are two other critical points for a PV module including the crossing from voltage and current axes: *open-circuit* and *short-circuit* values.

### 3.6.2 PV Transducer Model

While some applications require only knowledge of these three specific points, most utilizations of PV systems require a more descriptive explanation of the whole I-V curve. Based on the Shockley diode equation [62] as an outcome of work by Hall [254] and Shockley et al. [604], a multi-diode model of this behavior is explained as:

$$I_h = I_g - I_{d_1} - I_{d_2} - \frac{V_h + I_h \cdot R_s}{R_{sh}}, \tag{3.38}$$

when $I_g$ is the photo-generated current, $R_s$ and $R_{sh}$ are subsequently series and parallel resistances. Diode's current $I_d$ is defined as in Equation 3.39 when $\star$ is the diode number.

$$I_{d\star} = I_{s\star} \cdot \left[ \exp\left( \frac{V_h + I_h \cdot R_s}{n_\star \cdot V_t} \right) - 1 \right] \tag{3.39}$$

In Equation 3.39, $I_s$ and $n$ are saturation current and ideality factor of each diode. In addition, $V_t$ describes the *thermal voltage* from $V_t = B \cdot T/q$ when $B$ is the Boltzmann's constant, $T$ is the temperature in Kelvin, and $q$ is the electric charge.

From the formulation in Equation 3.38, an Equivalent Circuit Model (ECM) can be made for the reproduction of the source behavior of a PV module. This ECM shown in Figure 3.34 is able to replicate characteristics of the PV cell, including its non-idealities and resistances. However, according to the desired accuracy of the I-V curve replication, it is possible to reduce the number of diodes into one. But some applications utilize higher number of diodes to increase the degree of freedom for a better replication of the curve.

Regardless of the number of diodes in a PV transducer's model, this model has to be fitted into the real curve of each module and light type. Consequently, the number of parameters to tune differs according to the number of diodes. Moreover, a system designer may even remove some parameters from the model in some applications. Therefore, a general vector $\boldsymbol{\psi}$ will be used to represent these unknowns.

#### 3.6.2.1 Parameter Tuning

There is a large body of research on methods for tuning $\boldsymbol{\psi}$ for each PV module. However, they can be simply divided into *numerical* tuning and *Algebraic Equation Set* (AES)

tuning. In numerical methods, a set of points on the I-V curve is measured and directly used to solve an optimization problem to minimize the error. For an ideal case when the whole I-V curve is measured, this can be formulated as error minimization when $\hat{I_h}$ shows the current from the tuned model:

$$\min_{\boldsymbol{\psi}} \ell(\boldsymbol{\psi}) = \int_0^{V_{oc}} \parallel I_h(V_h) - \hat{I}_h(V_h, \boldsymbol{\psi}) \parallel_2 \, \mathrm{d}V_h \tag{3.40}$$

While this method can be explained in a simple way, its application has different challenges. At first, a large set of measurements from the PV system is necessary in a constant environmental condition. It can be especially challenging to keep the light intensity constant for the whole measurement duration. Other issues are related to the computational aspects of this method. On the one hand, a roughly computationally intensive optimization has to be solved to reach a set of reliable values for parameters in $\boldsymbol{\psi}$. On the other, these methods are sensitive to the initial values used for these parameters. Another issue that adds complexities to this utilization is finding the current value for each voltage in the model. As seen in the model's formulation, Equation 3.38 describes an implicit relation between voltage and current. Therefore, the calculation of current requires either an explicit relation or has to be done in a numerical (iterative) way. For the single-diode model, Femia et. al. [204] gives an explicit relation between parameters using the Lambert function as:

$$I_h = \frac{R_{sh} \cdot (I_g + I_s) - V_h}{R_{sh} + R_s} - \frac{n}{R_s} \cdot \mathcal{W}(\theta_i) \tag{3.41}$$

where Lambert W function is presented with $\mathcal{W}$, and $\theta_i$ is:

$$\theta_i = \frac{R_{sh} \cdot R_s}{n \cdot (R_s + R_{sh})} \cdot I_s \cdot \exp\left[\frac{R_{sh} \cdot R_s \cdot (I_g + I_s) + R_{sh} \cdot V_t}{n \cdot (R_{sh} + R_s)}\right] \tag{3.42}$$

Although this simplifies the single-diode finding of the current in the model, the two diodes model (which is more accurate) still requires a numerical solution for each voltage value. Therefore, there is a numerical iterative function solving inside the optimization in Equation 3.40, which is also solved in a numerical iterative sense itself.

The computational complexity of the numerical method is the main reason for most researchers to overcome this challenge by finding alternative routes for tuning parameters in $\boldsymbol{\psi}$. Using physical knowledge from the model is a common way to build an AES that can be solved with less computation. From the general I-V relation and basic knowledge about a PV system, several equations can be formulated:

$$\text{at SC: } V_h = 0 \Rightarrow I_{sc} = I_g - I_s \left[\exp\left(\frac{I_{sc} \cdot R_s}{n \cdot V_t}\right) - 1\right] - \frac{I_{sc} \cdot R_s}{R_{sh}} \tag{3.43a}$$

$$\text{at OC: } I_h = 0 \Rightarrow 0 = I_g - I_s \left[\exp\left(\frac{V_{oc}}{n \cdot V_t}\right) - 1\right] - \frac{V_{oc}}{R_{sh}} \tag{3.43b}$$

$$\text{at MPP: } I_M = I_g - I_s \left[\exp\left(\frac{V_M + I_M \cdot R_s}{n \cdot V_t}\right) - 1\right] - \frac{V_M + I_M \cdot R_s}{R_{sh}} \tag{3.43c}$$

It has to be noted that these equations are for a single-diode model but can be expanded for the double-diode model as well. While open circuit and short circuit can be easily measured, finding the MPP without any prior knowledge is not possible. However, it is known that the MPP is mostly proportionate to the open circuit voltage ($V_{oc}$) and short circuit current ($I_{sc}$). Hence, it is possible to simply measure the parameters at these specific proportional values and use them in Equation 3.43c, though it introduces a marginal error. The above equations are only a mathematical representation of the exact points from the curve, which is somehow similar to the numerical method. Hence, using the limited knowledge from this AES can provide a minimal fit to the model on these specific points. Furthermore, even a single-diode model has a $\boldsymbol{\psi}$ with 5 values. Consequently, this AES is under-determined and requires either more equations or a reduction of some parameters. Although few researchers have experimented on models with only 3 or 4 parameters, it is a well-established principle to use the derivation of the model on keypoints to expand the AES. These equations for a single-diode model are:

$$\left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{I_h=0} = -I_g \cdot \left[\frac{1}{n \cdot V_t} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{I=0} \cdot R_s\right) \cdot \exp\left(\frac{V_{oc}}{n \cdot V_t}\right)\right] - \frac{1}{R_{sh}} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{I_h=0} \cdot R_s\right) \tag{3.44}$$

$$\left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{V_h=0} = -I_g \cdot \left[\frac{1}{n \cdot V_t} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{V_h=0} \cdot R_s\right) \cdot \exp\left(\frac{I_{sc} \cdot R_s}{n \cdot V_t}\right)\right] - \frac{1}{R_{sh}} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{V_h=0} \cdot R_s\right) \tag{3.45}$$

$$\left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{\mathrm{MPP}} = -I_g \cdot \left[\frac{1}{n \cdot V_t} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{\mathrm{MPP}} \cdot R_s\right) \cdot \exp\left(\frac{V_M + I_M \cdot R_s}{n \cdot V_t}\right)\right] - \frac{1}{R_{sh}} \cdot \left(1 + \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{\mathrm{MPP}} \cdot R_s\right) \tag{3.46}$$

Furthermore, the derivation of power according to the voltage is zero at MPP, which leads to Equation 3.47.

$$\left(\frac{\partial P_h}{\partial V_h}\right)_{\mathrm{MPP}} = V_h \cdot \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{\mathrm{MPP}} + I_h = 0 \Rightarrow \left(\frac{\mathrm{d}I_h}{\mathrm{d}V_h}\right)_{\mathrm{MPP}} = -\frac{I_M}{V_M} \tag{3.47}$$

Substituting this in Equation 3.46 adds Equation 3.48 as an additional equation to the AES.

$$-\frac{I_M}{V_M} = -I_g \cdot \left[\frac{1}{a \cdot V_t} \cdot \left(1 - \frac{I_M}{V_M} \cdot R_s\right) \cdot \exp\left(\frac{V_M + I_M \cdot R_s}{a \cdot V_t}\right)\right] - \frac{1}{R_{sh}} \cdot \left(1 - \frac{I_M}{V_M} \cdot R_s\right) \tag{3.48}$$

From all these equations, an AES with 7 equations can be made that are all explicit. Nevertheless, in addition to the complexity of measuring MPP, the challenging task of

finding the curve's slope at keypoints is mandatory. Some simplified solutions suggest calculating the slope from the slope of a line between keypoints, though it adds error to the tuning performance. Therefore, further measurements are necessary to improve the calculation of the slopes. Moreover, these equations fit the model's formula to the curve only for the measured keypoints and their derivation and cannot provide high accuracy on other points. However, it is possible to measure any further point from the curve and add it as a simple equation in the AES. As can be seen, both parameter tunings have their advantages and disadvantages.

#### 3.6.2.2 Environmental Factors

An I-V curve represents a PV module in a specific environmental condition according to light intensity ($E$) and temperature ($T$). While the overall form of a curve remains, its details, including keypoints and slopes, shift with these parameters. Therefore, a further level of model is required to explain the effect of environmental factors. Some initial models explain the changes in keypoint values due to the deviation in environmental elements. Unfortunately, these models are not consistent in the literature and there are different formulations for a single parameter. However, there is a similarity in the methodology of these models due to their origin from PV operation in the solar light. In these models, environmental factors are described in a relative sense according to a reference condition, which is mostly the AM1.5 condition. It is described under a single sun at 1000 W/m$^2$ with a perpendicular line of sight to the PV cell at 25 °C.

Few relations are available in the literature, which explains some of the parameters in $\boldsymbol{\psi}$ according to the environmental factors. However, most of them use some kind of simplification assumption and mostly explain a specific application case study. Masoudinejad [434] reviews the state-of-the-art formulation for these parameters.

#### 3.6.2.3 Indoor PV Energy Harvesting

Despite the maturity of PV energy harvesting and the availability of diverse techniques for analysis and modeling, most of the available methods are based on the applications under sunlight. By contrast, modern use cases, especially within the IoT and Industry 4.0 realm, are in indoor areas with artificial lighting. Consequently, a revision of the available methodologies and validation of their solutions is required. One of the major difference in these fields is the scale of light intensity. Figure 3.35 provides an overview. As can be seen, the light intensity in some industrial applications such as PhyNetLab [197] as an industrial warehouse is multiple orders less than solar-based applications. In addition to the light-intensity level, the indoor light spectrum has many more forms and can differ according to the light source, building materials, and surrounding environment. This difference in the spectrum between solar light and typical indoor light sources can be seen in Figure 3.36.

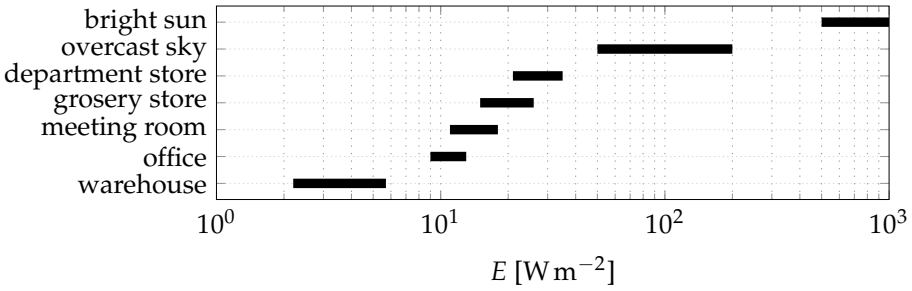



**Fig. 3.35:** Light intensity range in some common conditions. Reproduced from [146].

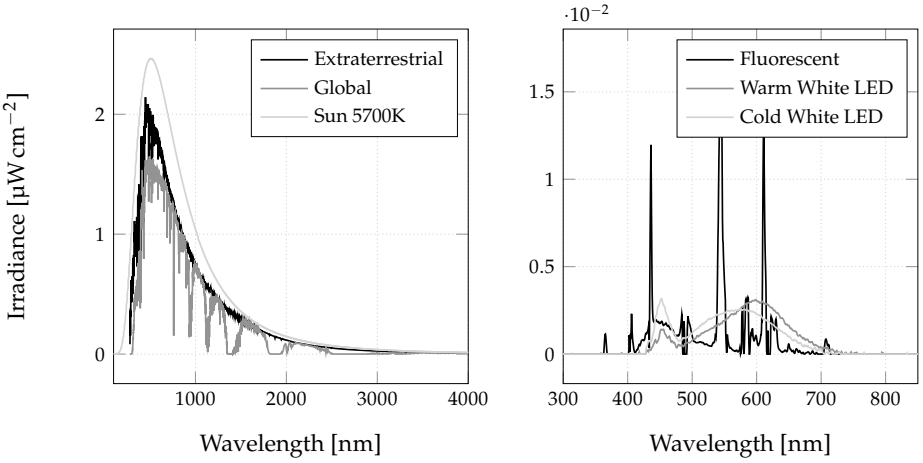



**Fig. 3.36:** Left: outdoor solar light spectrum [479] and sun's black body radiation at 5700 K [703]. Right: measured indoor light spectrum of three different artificial lighting.

Differences between light sources directly affect the I-V curve of a PV module. Figure 3.37 provides an example. Although both light sources are from the same manufacturer with the same specification and power, they produce dissimilar curves.

The only difference between these two sources is their color temperature, which can be seen in their spectrum in Figure 3.38.

As can be seen, although integrative light intensity of both sources is equal at 248 lx, their color temperature difference is a consequence of a discrepancy in the form of spectrum. Due to the non-uniform sensitivity of PV modules to each wavelength, the produced I-V curve will be different for each specific condition. Consequently, not only is an analysis of the PV behavior and modeling under artificial lighting necessary; careful consideration is required as well.

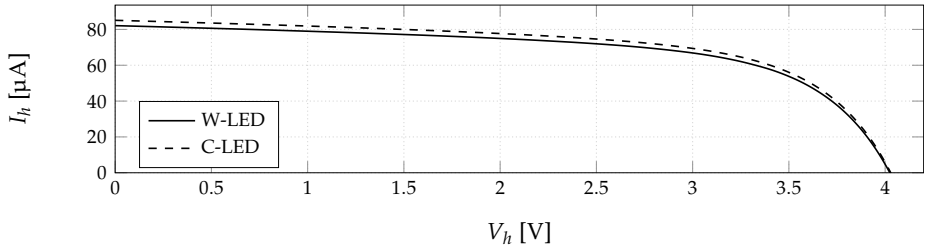



**Fig. 3.37:** I-V curve of a Solems PV module measured under cold and warm LED light. Both sources are from the same manufacturer measured at: $E$ =248 lx and $T$ =299 K.

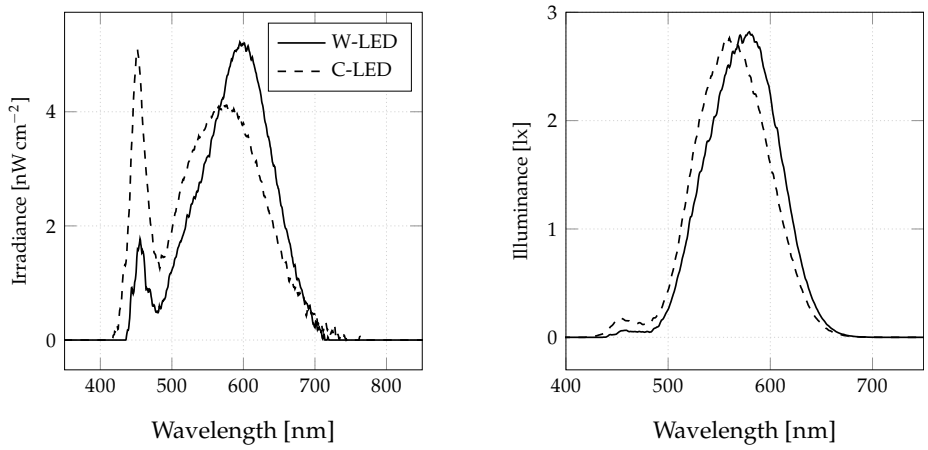



**Fig. 3.38:** Light spectrum measured under cold and warm white LED light both at 248 lx. In spite of equal integrative value, the spectrum is different. Left: irradiance, Right: illuminance.

### 3.6.3 Indoor PV Modeling

The diversity of the indoor lighting types and the lack of reliable data from PV behavior under indoor artificial lighting demand the collection of representative datasets for them. This data can be used later to analyze the I-V curve for such lighting, for parameters extraction through curve tuning and for formalizing the relation of them according to the environmental factors.

The required information for each measurement of the PV in an indoor area includes the I-V curve and light-intensity information in addition to the temperature. For measuring the I-V curve, a variable impedance has to be connected to the PV cell. Starting from a very large value this impedance will decrease till the open circuit voltage is reached. During this impedance sweep, voltage and current have to be measured simultaneously the whole time. This can be used to reproduce the curve. Such a procedure can be applied by using a Source Measurement Unit (SMU).

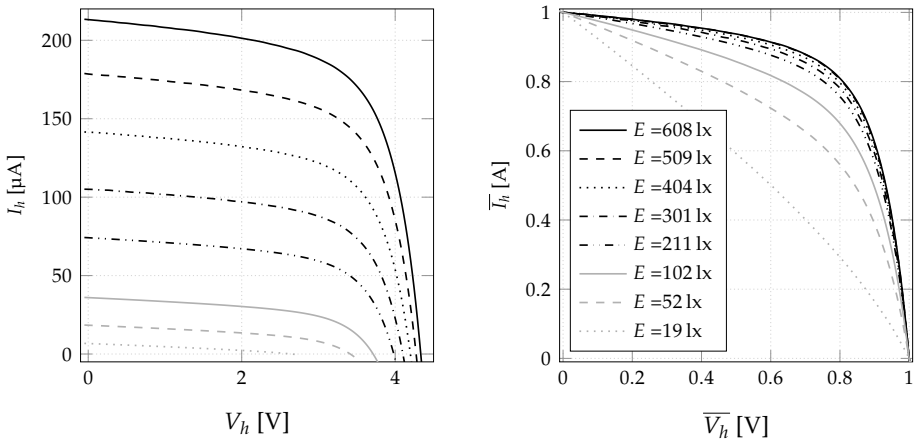



**Fig. 3.39:** Examples of measured I-V curves. Left: in normal space. Right: in PVNS.

There are multiple systems and devices for light intensity measurement. To have combined information, both integrative measurement and spectrometry are used here. Moreover, a closed environment with controlled light intensity is used to assure reliable and reproducible lighting. Using a system with detailed explanation from [432, 433, 435] different datasets are collected, which are publicly accessible from [433].

These datasets will be used below to develop the models and explore indoor PV behavior. However, there are two preliminary topics that need to be discussed before modeling.

### 3.6.3.1 PV Normalized Space

Looking at the available indoor datasets, as can be seen in the example in Figure 3.39, reveals that signal ranges are in different magnitudes. This diversity can cause calculation errors and add complexity to the numerical method calculations.

Therefore, *PV Normalized Space* (PVNS) is introduced to scale all curves into a similar range and avoid numerical problems. This conversion can be simply applied by converting the voltage and current of each curve according to its maximum referred to as $V_{oc}$ and $I_{sc}$, respectively. It has to be noted that resistances have to be scaled as well, though because the parameter $n$ does not have units it does not require any scaling. All in all, the relation of all scaled parameters (shown by $\overline{\bullet}$) is:

$$\overline{\boldsymbol{P}} = \boldsymbol{P} \times \left[ \frac{1}{V_{oc}}, \frac{1}{V_{oc}}, \frac{I_{sc}}{V_{oc}}, \frac{I_{sc}}{V_{oc}}, \frac{1}{I_{sc}}, \frac{1}{I_{sc}}, \frac{1}{I_{sc}} \right], \tag{3.49}$$

when:

$$\boldsymbol{P} = \left[ V_h, V_t, R_s, R_{sh}, I_h, I_g, I_s \right]^{\mathsf{T}}. \tag{3.50}$$

The effect of this scaling on the I-V curve can be seen in Figure 3.39 on the right. This scaling highlights the differences on the form of the I-V curve as a consequence of

**Tab. 3.7:** Bound suggestions for the PV model's parameters in the PVNS.

| Parameter | Lower limit | Upper limit | Unit |
|---|---|---|---|
| $I_g$ | 0.9 | 1.1 | [A] |
| $R_s$ | $\epsilon$ | $(V_{oc} - V_M)/I_M$ | [$\Omega$] |
| $R_{sh}$ | $V_M/(I_{sc} - I_M)$ | $\infty$ | [$\Omega$] |
| $I_s$ | $\epsilon$ | 1.1 | [A] |
| $n$ | 1 | 10 | [-] |

different light ranges, which are not easily detectable on the original curve. Moreover, it brings all available data into a uniform scale which helps to reduce the sensitivity to the signal values in numerical methods.

#### 3.6.3.2 Evaluation Criteria

Similar to the modeling procedures, an evaluation factor is required to quantify the performance of a model. When replication of the I-V curve is desired, relative performance factors are preferred due to the large signal range. Nonetheless, division to zero at the open-circuit point can hinder calculating a relative factor without removing this point. Unfortunately, open circuit is a keypoint and plays a critical role in any model. Consequently, the Mean Absolute Normalized Error (MANE) is defined here as in Equation 3.51. It normalizes the percentage absolute error according to the $I_{sc}$ to find the mean value. This can be explained as the Mean Absolute Error for the I-V curve in the PVNS as well.

$$\text{MANE} = \frac{100}{I_{sc} \cdot V_{oc}} \cdot \int_0^{V_{oc}} |\delta(I_h)| \, \mathrm{d}V_h = 100 \cdot \int_0^1 |\delta(\overline{I_h})| \, \mathrm{d}\overline{V_h} \tag{3.51}$$

or for the case of discrete measured data with $m$ points:

$$\text{MANE} = \frac{100}{I_{sc} \cdot m} \cdot \sum_{i=1}^{m} |\delta(I_h)| = \frac{100}{m} \cdot \sum_{i=1}^{m} |\delta(\overline{I_h})|. \tag{3.52}$$

#### 3.6.3.3 IV Curve Parameter Tuning

In the first step of the modeling, $\boldsymbol{\psi}$ parameters have to be tuned for each I-V curve. For this purpose, SWL, SCL, and IWL datasets from [433] are used. For the numerical tuning method, as discussed in 3.6.2.1, the initial value used for each parameter plays a critical role. After a large set of experiments, a general guideline can be provided as in Table 3.7 for the bounds on these parameters in the PVNS. Using these bounds, parameters for all curves in datasets are tuned using least square method. This process is repeated for both single- and double-diode models using 200 equidistant points along the voltage axis, for each curve. Distribution of error for all cases is presented in Figure 3.40. As

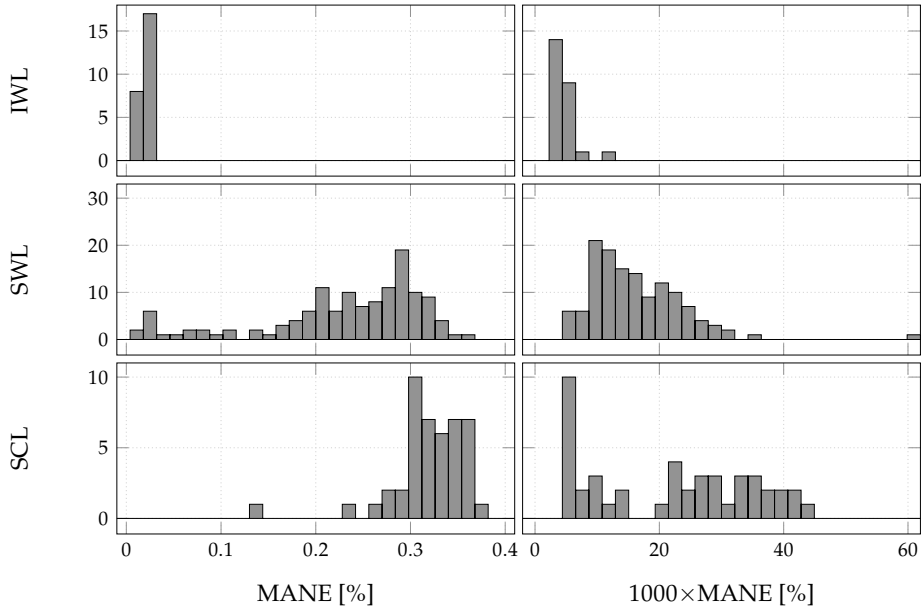



**Fig. 3.40:** Density of MANE distribution for all datasets using, the single-diode model (left) and double-diode model (right).

could be estimated, double-diode model has a much better performance compared with the single-diode counterpart. This can be simply argued because of extra tuning parameters.

The application of the AES-based method on these datasets is computationally simpler while it is very sensitive on the way that the slope of the curve is calculated. However, comparing the results with the numerical method, AES-based tuning has lower performance because of the extensive number of points in the numerical method. Hence, tuned parameters from the numerical method continue to be used. Their distribution according to the light intensity for the single- and double-diode models can be seen subsequently in Figure 3.41 and Figure 3.42.

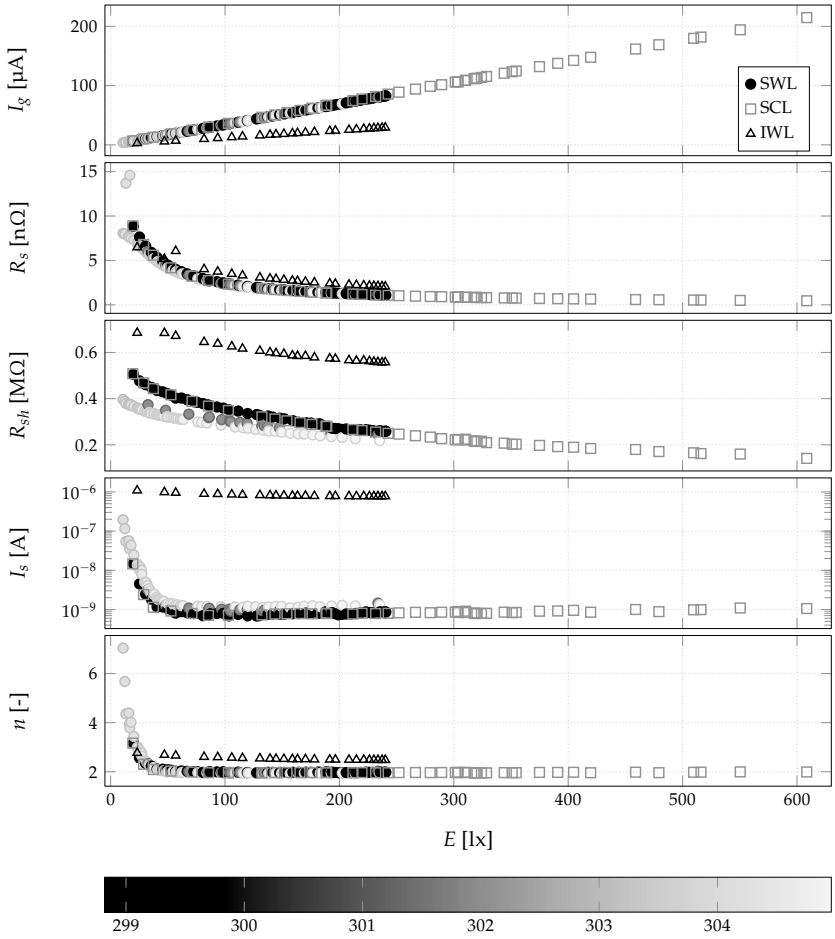



**Fig. 3.41:** Changes in single-diode model parameters according to the light intensity for all datasets. Color of SWL points shows temperature in K as in the color bar.

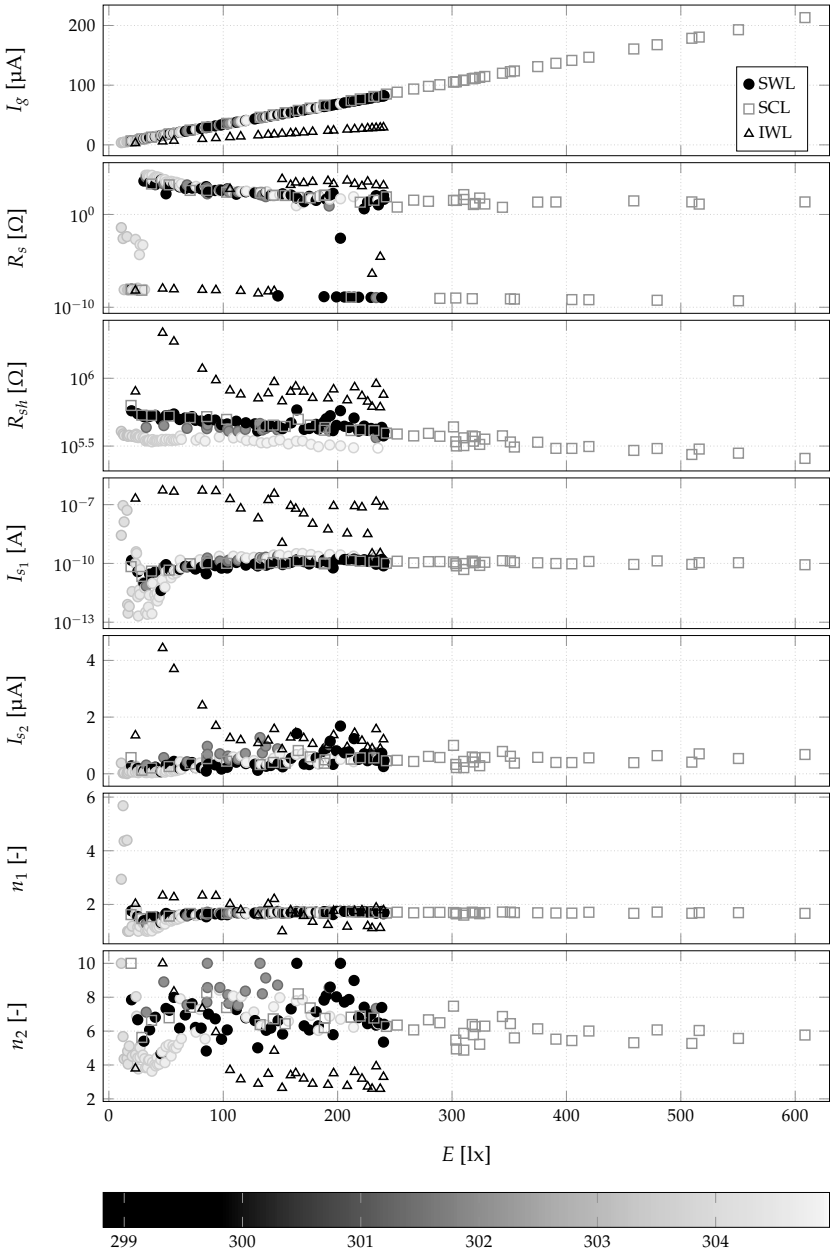



**Fig. 3.42:** Changes in double-diode model parameters according to the light intensity for all datasets. Color of SWL points shows temperature in K as in the color bar.

As can be seen from these figures, the single-diode model parameters have a clear relation to the light intensity and temperature. Although this holds for some parameters of the double-diode model, others such as resistors and saturation currents show abnormal behavior. This can hint at an over-dimensioning of the model with a second diode that introduces parameters that are not actually part of the physical system. Therefore, a hypothesis can be made that a secondary diode (or at least some of its parameters) is not part of the physical model of the PV system under artificial lighting. Hence, it improves the performance of the tuning but is no physically significant at the higher level. This behavior is somehow similar to the principle of over-fitting in machine learning, where the model focuses too much on the data, so that the overall form gets lost. Nevertheless, this hypothesis cannot be proven at this stage (using available data) without any further physical insight. Consequently, the single-diode model will continue to be used here for the remaining part of the modeling of the indoor PV system.

#### 3.6.3.4 Modeling Effect of Environmental Factors

After tuning all parameters for the single-diode model, the next step is to formulate the relation of each parameter with the environmental factors. However, since this is a purely data-based modeling approach, the resulting models are empirical and will not assure any physical insight. Yet, by building these models for all datasets that include different PV technologies and light sources, the generalization of the model can be kept to some extent. After testing diverse function types on the available data, these relations are formulated as:

$$\widetilde{I_g} = \frac{I_g}{I_g^\star} = \alpha_{g1} \cdot \widetilde{E} + \alpha_{g2} \cdot \Delta T \tag{3.53a}$$

$$\widetilde{R_s} = \frac{R_s}{R_s^\star} = \frac{\alpha_{s1} + \alpha_{s4} \cdot \Delta T}{\alpha_{s2} + \alpha_{s3} \cdot \widetilde{E}} \tag{3.53b}$$

$$\widetilde{R_{sh}} = \frac{R_{sh}}{R_{sh}^\star} = \frac{\alpha_{p1} + \alpha_{p6} \cdot \Delta T}{\alpha_{p2} + \widetilde{E}} + \ln\left(\alpha_{p3} \cdot \widetilde{E} + \alpha_{p5}\right) \tag{3.53c}$$

$$\widetilde{I_s} = \frac{I_s}{I_s^\star} = \alpha_{i1} + \alpha_{i2} \cdot \widetilde{E} + \frac{1}{\alpha_{i3} \cdot \widetilde{E}^{\left(\alpha_{i4} + \alpha_{i5} \cdot \widetilde{T}\right)}} + \alpha_{i6} \cdot \Delta T \tag{3.53d}$$

$$\widetilde{n} = \frac{n}{n^\star} = \alpha_{n1} + \alpha_{n2} \cdot \widetilde{E} + \frac{1}{\alpha_{n3} \cdot \widetilde{E}^{\left(\alpha_{n4} + \alpha_{n5} \cdot \widetilde{T}\right)}} + \alpha_{n6} \cdot \Delta T \tag{3.53e}$$

In these equations, parameter $\alpha$ shows a tuning factor for each dataset, while $\widetilde{\bullet}$ is a representation of a relative parameter. Each relative parameter under a reference condition is similar to the AM1.5 for the solar case. However, contrary to the solar case with a unique reference point, there is no constant condition that can be defined for all indoor environments. Hence, the point with maximum light intensity in each dataset is used as a reference point and all its parameters are used as a base condition. Despite

parameters whose relative value is found from a ratio, the difference of temperature is more suitable for these formulations and is used in Equation 3.53.

Using these formulations, $\boldsymbol{\alpha}$ is tuned for each dataset. As an example, the output parameter for each light intensity in the SWL dataset is presented in Figure 3.43. It is clear that these provided empirical models are able to replicate the behavior of each parameter with a good accuracy.

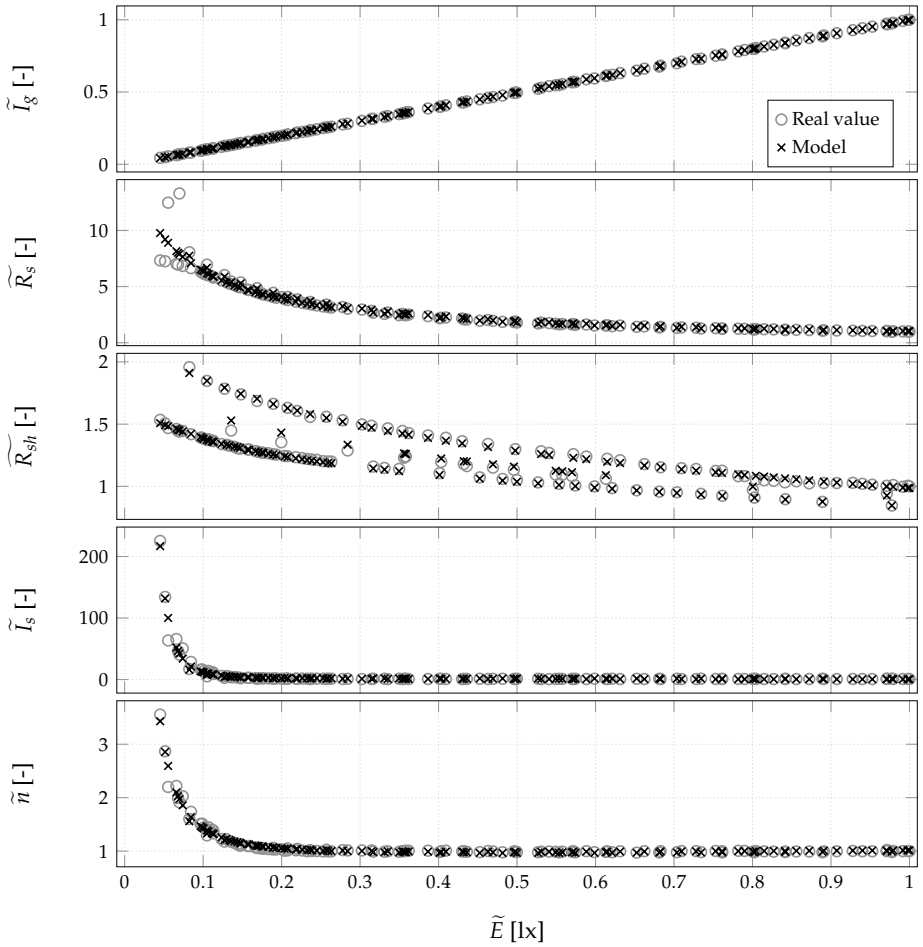



**Fig. 3.43:** Performance of models for parameters at different environmental conditions on the SWL dataset including temperature effect.

It has to be noted that parameters of the I-V curve with maximum light intensity are used here as the reference point. Also, it is advisable to remove the temperature factor

for a dataset when the deviation of temperature is minimal and can be ignored. In this way, the model can be simplified without reducing its performance.

### 3.6.4 Indoor PV Model Evaluation

So far, three available datasets have been used for the parameter tuning and environmental factor modeling. Since the overall behavior of the models has been checked on the same dataset, it can not guarantee a reliable performance for other conditions. Therefore, an extra evaluation on a new dataset is beneficiary. Consequently, a new set of data has been collected in the PhyNetLab, including 120 samples collected in different positions, heights, and temperatures from different days. This set includes light intensities between 244 lx to 494 lx and temperatures in the range of 298 K to 302 K.

To avoid error due to the training during the tuning of $\boldsymbol{\alpha}$ factors, 30 % of samples are selected randomly in a uniform distribution of the light intensity. Parameters for this smaller set is extracted and used in a least square method to tune $\boldsymbol{\alpha}$ in Equation 3.53. Similar to the model itself, the highest light intensity in this subset is used as the reference condition. The resulting formulations are used to find $\boldsymbol{\psi}$ parameters for the remaining 70 % of the data. Using these parameters, the resulting I-V curve is compared with the real measured value to find the MANE presented in Figure 3.44.

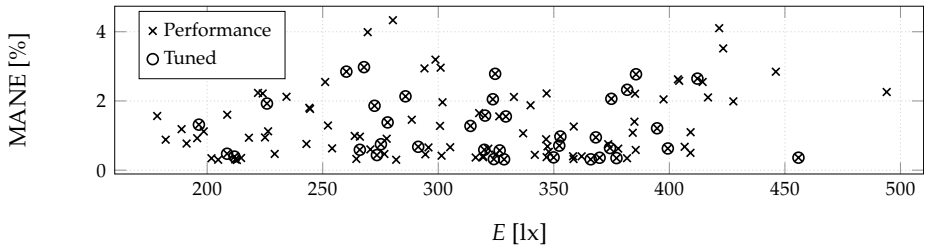



**Fig. 3.44:** MANE of model on the evaluation dataset. Only 30 % of data (randomly selected, shown with circles) was used for parameter tuning.

As can be seen, all errors are in a very small range, which shows reliable performance from the empirical abstract level model on a new indoor environment. When repeating the random selection subset, the worst case MANE is always less than 6 %. Hence, the overall modeling principle can be accepted and applied to other environments.

### 3.6.5 Conclusion

This contribution has discussed the basic principles of the PV energy harvesting and their modeling. After a short review of available PV models, we introduced tuning and adapting a model for a PV transducer. We then discussed the differences between PV behavior under solar light and artificial sources (used for indoor environments). Next we determined the need for model development and evaluation specifically for indoor PV harvesting.

It was shown that each light source and indoor environment has its particular specifications due to differences in the light spectrum seen by a PV module. Therefore, a new setup was presented to assure reliable, high accuracy, and reproducible indoor PV behavior data. Using this data, a PV normalized space was introduced to enable a unified modeling strategy, regardless of large differences in the signals within the indoor environment. Furthermore, the mean absolute normalized error was introduced as a non-compromised evaluation factor for model performance of each I-V curve.

Using the collected data, guidelines were provided for the boundary of tuning parameters in I-V curve fitting. While parameters were tuned for single- and double-diode models of a PV system, it has been shown that some parameters in the double diodes model lack specific relation to the environmental factor. This led to a hypothesis that the double-diode PV model includes more parameters than its real physical factors. Therefore, these parameter(s) improve the fitting but with the price of loosing the physical meaning. Hence, single-diode model data was used to develop empirical models for each parameter according to the environmental factors. These models use relative values according to a reference point which is simply selected as the data point with the highest light intensity.

Finally, these models were tested using all available datasets, and a new set of data was collected in a real-case scenario including 120 samples. The environmental condition of this new scenario was tuned into the model using only 30 % of data points selected randomly. The application of this tuned model showed promising performance with MANE of less than 6 % in the worst-case scenario. Therefore, we showed that adaptability of the developed models on new real world environments and their good performance on predicting the I-V curve of a PV module in new environments.

## 3.7 Micro-UAV Swarm Testbed for Indoor Applications

*Nils Gramse*
*Moritz Roidl*
*Shrutarv Awasthi*
*Christopher Reining*

**Abstract:** This contribution describes a testbed for the development of resource-constrained micro-UAV (Unmanned Aerial Vehicle) swarms . The testbed architecture combines external observation, visualization of logistic scenarios, and simulation systems with a drone control unit. Swarm algorithms are implemented on the drones to control their movements and enable their cooperation. A drone learns to perform a warehousing task using reinforcement learning. In combination with swarm algorithms, this behavior is extendable to a drone swarm. This work describes how drones can be deployed to solve tasks in industrial settings. In addition, an automatic charging station extends the runtime of the swarm.

### 3.7.1 Introduction

Drones are more formally known as unmanned aerial vehicles (UAVs) or unmanned aircraft systems. They are battery-powered devices, that can be as large as an aircraft or as small as the palm of a hand. A drone is an intelligent flying system with sensors and actuators that can be remotely controlled or fly autonomously. Due to its high adaptability, the use of drones is increasing across many civil application domains. Over the last decade, extensive research has been performed on consumer and commercial drones. Due to the advances in microelectronics, sensors are steadily getting lighter, smaller, more economical, smarter, and more accurate. As a result, drones are getting smaller, more energy-efficient, and easier to operate [521]. Moreover, there is a growing need to build small and resource-constrained drones that fit perfectly into the Industry 4.0 setting, where all intelligent devices are networked and can exchange their data with each other [52].

### 3.7.2 Drones in Logistics

Drones have shown high potential in the logistics industry. Some have projected that this market will grow by $29 billion by 2027 with an annual growth rate of almost 20 % [721]. A 2018 study found that electric drone delivery was more efficient than trucks, vans, passenger cars, and gasoline drones [650]. Electric drones are environmentally

friendlier than other aerial vehicles when comparing $CO_2$ emissions. Drones are a potential substitute for traditional delivery methods, such as door-to-door or last-mile delivery using trucks. The difference is that they do not consume fuel and do not need infrastructure such as streets. However, drones still pose several safety hazards, limiting their use in industries and outdoors. Some common safety hazards include drones colliding with people, structures on the ground, and other drones [720]. Thus, more research is required to make drones safe to use in urban and industrial settings. Deploying drones for transporting goods to end users may transform the existing transportation methods in large cities and rural areas. They might not completely replace the traditional methods but will significantly impact this domain [521].

Warehouse management is an essential operation for most business activities nowadays. Manual inventory has been the only option for a long time, but it poses several challenges in terms of costs, inaccuracies, and safety [151]. Furthermore, in the EU, warehousing and storage represent up to 15 % of the current costs in logistics [193]. Thus, there is a growing need to automate warehouse operations while ensuring the warehouse flexibility and adaptability. Such automation is something automated storage solutions made of mechanical conveyors such as high-bay warehouses and automated small parts storage cannot achieve. However, drones could potentially cater to this ever-increasing demand. There has been extensive research on drone use in transporting goods outdoors, and the researches are available commercially [418, 642]. However, large-sized drones cannot reach small, constricted spaces such as warehouses. The key reasons are that GPS-based navigation systems are unavailable, and the tolerances regarding collision avoidance and the time available for decision-making are drastically lower. By contrast, small-sized drones can operate in the interiors of warehouses and the narrow and high rows of shelves. Thus, one can integrate small-sized drones into the supply chain to automate various intra-logistics operations. [525].

Research on deploying autonomous drones and robots for human-machine interaction in warehousing is still in its early stages. However, industries have pioneered drones as extensions to their IoT environments or complement other data-gathering processes. Most IoT devices have a limited battery life, are stationary, and thus can collect the data of a specific location for a limited time. However, resource-constrained small drones can gather data from dangerous areas and other fixed IoT devices. Subsequently, they can transmit all the collected data to a central station. Drones can also aid in recharging IoT devices wirelessly or remotely. Thus, drones can be essential in connecting IoT devices to the whole IoT ecosystem [21, 26, 417]. This work introduces a testbed to replicate and experiment with any industrial scenario involving small-sized resource-constrained drones working alongside humans. In addition, an automatic charging station has been proposed, to ensure the continuous operation of drones.

### 3.7.3 Application

A possible logistics application for resource-constrained drone swarms, especially those that can recharge independently, is their use as a mobile surveillance system. For instance, this drone swarm can be used in warehouses to protect the stored goods from theft. The drones can then perch like birds on the existing trusses, high racks, and steel beams and observe their surroundings. From their high positions, they can gather a larger amount of data instead of low-power stationary IoT devices equipped with a battery. A charging station enables the continuous operation of the drone swarm. If a drone is low on battery, it automatically flies into the charging station, and another fully charged drone takes over its tasks.

### 3.7.4 Swarm Algorithms

Swarm algorithms create a group dynamic similar to the swarm behavior of animals, such as a flock of birds or a school of fish. Craig Reynolds developed a simulated swarm behavior in 1987 with the help of these observations [526]. In logistics systems, a swarm algorithm is similar to traffic rules, which enable a smooth traffic flow by observing the local environment with as little communication as possible between the vehicles. One advantage of using swarm algorithms in logistics is that they do not require fixed routes but can use all available spaces. This allows them to easily change the layout and react to unpredictable events or disruptions. Furthermore, due to the modular design of swarm algorithms, new behaviors can be added without altering the existing ones. The paradigm of swarm behavior aims to enable several independent robots to collaborate towards achieving a collective goal, acting as a swarm.

Swarm behavior, established by Reynolds, is implemented by three basic rules: the cohesion rule, the alignment rule, and the separation rule [153]. Figure 3.45 shows the rules schematically. The active agent in each case is the highlighted black triangle with its current velocity vector. The remaining triangles represent other agents. The highlighted velocity vector indicates the resulting vector adapted to satisfy the requirements of that rule. The white circles visualize the effective range of local perception of the active agent. The black circle represents the center of all agents located in the local perception.

The cohesion rule aims to move toward the swarm's center. Thus, an agent tries to point its velocity vector to the center of its local perception. The alignment rule ensures that all agents within the range of local perception aim for the same direction of motion. Finally, the separation rule states that a minimum distance to all neighbors [153] must be maintained.

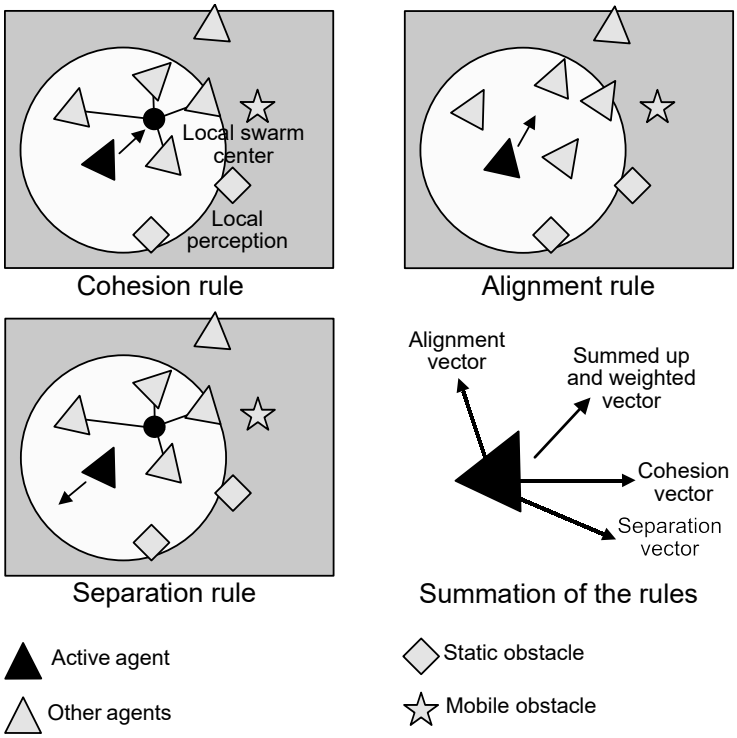



**Fig. 3.45:** Schematic representation of the three basic rules for simulated swarm behavior [153]. Mobile obstacles are people moving with headgear, and static obstacles could be the charging station, walls, or shelves.

Weighting the individual rules influences the appearance of the swarm behavior. For example, if collisions must not occur under any circumstances, the separation rule receives a high weighting. After assigning appropriate weights, the calculated velocity vectors are summed up to a control vector, which determines the final movement of an agent, as can be seen in Figure 3.45. The above three basic rules can also be supplemented by other rules as seen in [483]. One of the advantages of swarm control is that there is no need for the centralized control of each drone. Depending on the implementation, each drone in the swarm can perceive its local environment by itself and avoid collisions. Thus, it is possible to manoeuver an arbitrary number of drones simultaneously without calculating a separate trajectory for each drone centrally. This decentralized control makes the swarm implementation scalable.

### 3.7.5 System Architecture of Drones

The drones used in this work are a custom design using the Crazyflie 2.0 as a base platform [509]. Figure 3.46 shows the UAV used in this work. A single drone has a size of 103x103x29 mm and a total weight of 37.85 g. It can carry a payload of 64.85 g (compared with 15 g for an original Crazyflie drone). Thus, drones can use larger batteries which increases the flight time. Furthermore, additional sensors such as cameras and Lidar can also be mounted on the drone.

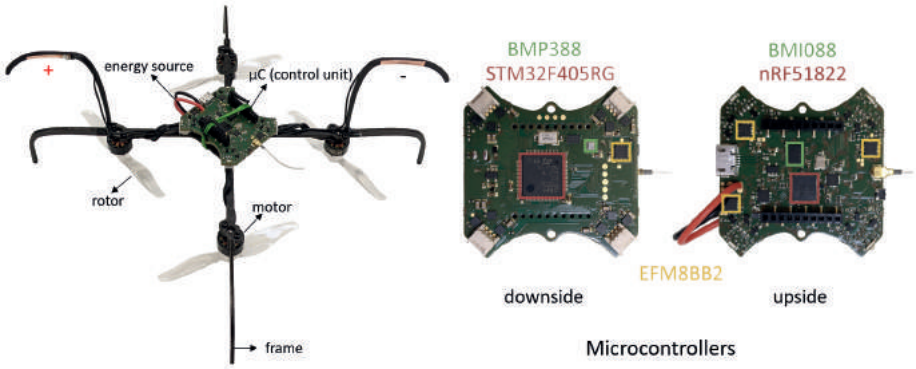



**Fig. 3.46:** Crazyflie 2.0 drones used in this work.

Four brushless DC motors on which four rotors with a diameter of 60 mm are mounted in reverse, comprise the propulsion system. Thus, the flow characteristics of the propellers are not influenced negatively by the frame. In addition, the frame structure itself is lightweight and built using fibre composite material.

The computing hardware consists of two microcontrollers. An STM32F405RG from STMicroelectronics [648] is used as the main computing unit. It performs all computationally intensive calculations and control tasks of the drone. For this purpose, a 32-bit ARM Cortex M4 with a clock frequency of 168 MHz and a floating-point arithmetic unit are used on the microcontroller. The SRAM has a capacity of 192 kB, and the flash memory holds 1 MB of program code. An EEPROM of 8 kB is connected to the microcontroller via the $I^2C$ bus to store static information.

A second microcontroller (nRF51) from Nordic Semiconductor is used for wireless communication and as a power manager. This microcontroller uses a 32-bit ARM Cortex M0, which clocks at 32 MHz and has a 32 kB SRAM and a 128 kB flash memory. It has a low idle power consumption of 3 µW [591]. The two microcontrollers communicate via a UART interface.

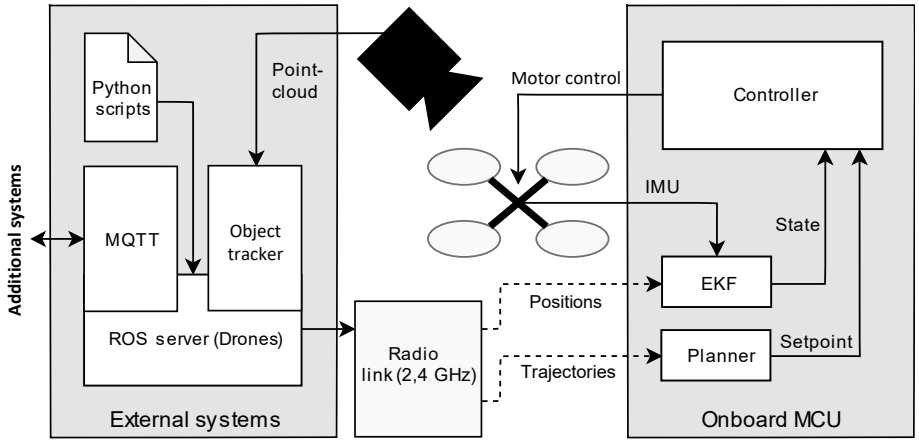



**Fig. 3.47:** System architecture for a drone swarm as a platform for data acquisition. The main components with their interfaces and the interaction between all components are shown [509].

Functionalities of the entire system architecture of the drone swarm are realized on various external systems, as seen in the Figure 3.47. Due to the loose interconnection of the multiple systems in the architecture, they can be easily replaced.

The Robot Operating System (ROS) server is well suited for inter-process communication [457]. Therefore, it is a part of the high-level application layer for controlling Crazyflie 2.0. Data from the MoCap system determines the absolute position and the attitude of the drones. A unique marker configuration consisting of four markers is pasted on every drone to identify each drone uniquely. The captured point cloud from the MoCAP system generates set points for the drones and compensates for the drift of the inertial measurement unit onboard the drone. All parameters converge in a ROS server and are forwarded to the drone.

The main computation task performed onboard the drones is the calculation of the flight parameters. A trajectory is computed from the set point and the state estimates in a specified time. The set-point values are obtained through an Extended Kalman Filter (EKF) to achieve a higher overall control accuracy. Trajectories can also be calculated externally and transmitted directly to the drone.

Communication between the external systems and the drone happens via the 2.4 GHz ISM band. The CRTP (Crazyflie Real-Time Protocol) is designed to send data packets without much overhead, thus minimizing the latency. The transmission speed is configured to 250 kbit/s to prevent interference caused by the metal walls of our research center and enable the transmitted signals to be decoded reliably by the drones. The MQTT server transmits drone information from the ROS server to other subsystems.

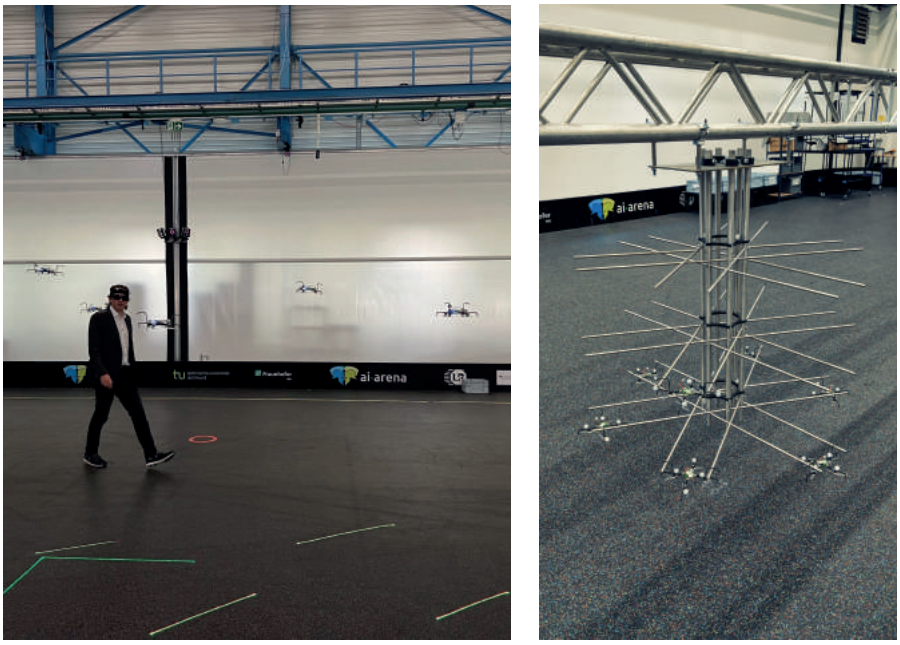
#### 3.7.5.1 Recharge Station

Another adaptation is having the drones charged in a drone recharge station. Therefore, two holders with which the drones can hang in the station for charging, are attached to the drone frame. The charging current for the drone then flows via these holders. Thus, a battery exchange is no longer necessary, enabling continuous operation.

The drone recharge station represents another external system. It is constructed out of stainless steel rods and currently allows charging up to 32 drones distributed over three floors. A 200 W 5 V/20 A power supply feeds power to the recharge station. The recharge station can be hung from the roof or fixed in another safe place. When the drones land on the station, the charging of the drones starts automatically. Therefore, the 24/7 operation of the drone swarm is achievable. If a drone is low on battery, it automatically flies into the charging station, and another fully charged drone takes over its tasks.

### 3.7.6 Testbed

The testbed was developed at the Chair of Material Handling and Warehousing research center at TU Dortmund University. It is housed in a lightweight hall that is structurally identical to conventional industrial buildings in the logistics sector and follows the concept of a highly flexible development laboratory [478, 520]. Developing the testbed aims to create an environment for simulating real warehouse scenarios. The testbed is equipped with an infrastructure designed to prototype Cyber-Physical Systems (CPS) accurately. While the test area remains free of permanently installed infrastructure, the hall contains several permanently installed observation systems on the ceiling, walls, and floor.




(a) (b)

**Fig. 3.48:** General view of the testbed created in the research hall. (a) Person (wearing a headgear) standing between the drone swarm during a test scenario (b) Charging station hanging from the roof.

### 3.7.6.1 Architecture

Figure 3.49 shows the architecture used to create the testbed. The architecture is similar to the one used in [465]. The MoCap system consists of 46 infrared cameras manufactured by Vicon. It can track a substantial amount of appropriately marked objects with an accuracy of 0.3 mm and operates at a data transmission rate of up to 200 Hz with latencies of 4 ms to 15 ms. The observed experimental space is 22 m long, 15 m wide and up to 3.5 m to 4 m high. The localization data is accessible to multiple clients simultaneously over the network and provides the absolute position and attitude of the marked objects in a three-dimensional space. Figure 3.48 shows a general view of the testbed.

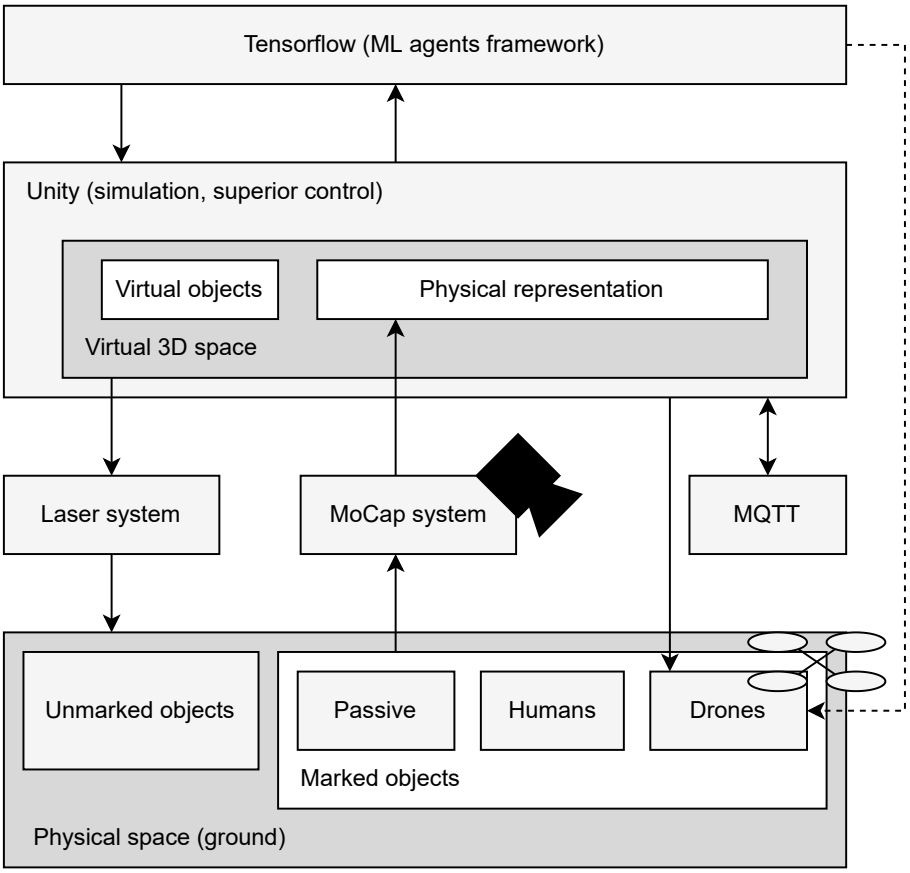



**Fig. 3.49:** Architecture used for creating the testbed [465].

The testbed uses MQTT (Message Queuing Telemetry Transport) [292] to distribute data between the subsystems in Figure 3.47. The simulation subsystem, designed as a development platform, consists of a programmable 3D modelling environment. In this work, Unity is used for 3D modelling. Unity generates virtual objects and scenes. All objects in a scene are mapped into an inheritance tree, which can be manipulated or extended using the C# programming language [307]. In addition to the common objects of the 3D modelling environment, such as cameras or light sources, the C# script creates custom objects.

The simulation subsystem comprises a laser projection system consisting of eight Kvant Clubmax FB4 laser projectors [373]. The laser system generates both static and dynamic projections of virtual objects from the simulation. Therefore, it is possible to visualize complex algorithms [436]. Visualization can be done for demonstration purposes and a better understanding of the complex behavior of an algorithm.

All marked objects in the physical space are mirrored into the Unity 3D simulation environment via the MoCap system. In addition to the representation of physical objects, the simulation can contain any number of virtual objects. Virtual objects with no physical representation are mirrored on the hall floor via the laser projection system. Unity sends simple motion setpoints to the UAVs and receives the resulting positions of the UAVs via the MoCap connection. The low-level control logic is performed onboard the UAVs. UAVs and their trajectories can be simulated at an accelerated time in simulation-only mode.

In this work, TensorFlow is used to develop machine learning algorithms. These algorithms are coupled to the simulation environment via the ML-Agents toolkit [307]. The toolkit implements reinforcement learning on a drone and uses Unity as a training environment. During training, the simulation is executed up to a hundred times faster. The learned behavior in neural networks controls the drone using Unity. The current system uses C# in the Unity simulation, Python and C++ for drone control based on the ROS, and plain C on the embedded systems of the drones.

#### 3.7.6.2 Drone Swarm Setup

The testbed described in the previous section simulates swarm control on a physical drone swarm. The current swarm at the research hall consists of up to 16 drones controlled by an extended version of the open-source project [509]. Based on the architecture in Figure 3.47 a transport scenario in a warehouse is replicated.

The drone swarm flies in a test area indicated by the laser projection system. Humans can enter the test area if they wear laser protection glasses and headbands with markers, so they get recognized by the swarm as a mobile obstacle. Humans can safely move within the swarm as long as they move at moderate speeds. After takeoff, the drones fly in the range between 1.5m and 2.6m in height.

Transport orders are created by the laser when a marked Frisbee disc gets thrown on the ground in the test area. The transport order is generated as a virtual packet projected by the laser system. The target for all orders is an area on the ground indicated by the laser. The drone with the shortest path to the packet flies to it at a low altitude and picks it up. The drone then delivers it to the target area at a low altitude and ascends back to join the swarm. Multiple orders can be picked up by multiple drones simultaneously.

### 3.7.7 Reinforcement Learning for Micro-UAV Swarm

After successful simulation of the drone swarm in the testbed, we extended the capabilities of the swarm by using machine learning to perform a warehouse task. This work uses Reinforcement Learning (RL) algorithms. In RL, the algorithm is not given examples of optimal outputs but instead discovers them by trial and error [70].

The goal in RL is to find suitable actions for a given situation to maximize the reward. In RL, an agent is an entity that acts based on a policy. The policy defines how an agent behaves based on the observations it perceives at a given time. An agent is embedded within an environment, and at a given instance, it is in a specific state. The value of a given state refers to how rewarding it is to be in that state. From the current state, the agent can take one of the sets of actions that can bring it to a new state, provide a reward, or both. The agent's main objective is to maximize the total cumulative reward that it receives over the long run [664].

Reinforcement learning with drones is being extensively used for various applications such as drone tracking, and following the leader drone in a swarm [14], achieving a decentralized control of a drone swarm [51], collision avoidance [518], and trajectory planning [318]. This work shows that machine learning techniques such as RL can be tested in the testbed environment created in the research hall.

#### 3.7.7.1 Scenario

In this work, the task of transporting an object to the target area is simulated using RL by a single drone as a proof of concept. A drone flying in the testbed is presented with a virtual object. At first, the drone wanders around in the testbed, unsure what to do. Eventually, it picks up the object and delivers it to the target area, getting a reward. After multiple training sessions, the drone learns that picking up and delivering an object is the best way to maximize the reward.

#### 3.7.7.2 Implementation

The ML agent toolkit creates simulated environments using the Unity Editor and interacts with them via a Python API [307]. The toolkit provides the ML-Agents SDK, which contains the necessary functionality to define an environment within the Unity Editor and the core C# scripts to build a learning pipeline. In this work, the task of transporting orders, as described in Section 3.7.6.2, is simulated. The environment is the testbed in the research hall, and the agent is the drone. Initially, the drone needs to learn how much to rotate the motors enabling it to move some specific distance in a particular direction. Then, the drone computes the relative distance between itself and the target to decide what action to take next. The action of picking up the order and delivering it to the target area earns the drone a positive reward. Drone receives a fixed penalty (negative reward) for every other action. The goal is to maximize the rewards and minimize the penalty. Thus, the drone learns to swiftly pick up and deliver an object to avoid a heavy time penalty.

A hierarchical approach integrates the swarm algorithms with RL algorithms. The swarm algorithms such as separation, cohesion, and obstacle avoidance have a higher weightage which ensures collision-free flight and the safety of the persons in the testing area.

An RL technique called Proximal Policy Optimization (PPO) is used in this work. PPO is a family of policy optimization methods that use multiple epochs of stochastic gradient ascent to perform each policy update [587]. PPO uses a neural network to approximate the ideal function that maps an agent's observations to the best action an agent can take in a given state. A simple neural network (developed in TensorFlow) with an input layer, one hidden layer and an output layer is used. The network's output is a vector indicating the direction the drone should fly in. Agents can ask for decisions from the policy either at a fixed or dynamic interval, as defined by the developer. The PPO algorithm is implemented in TensorFlow and runs in a separate Python process. Communication between Python and Unity takes place via a gRPC communication [718] protocol and utilizes protobuf messages.

After training the RL agent for an hour, the trained model is saved. The saved model computes the actions of the drone in the testing phase. While testing, it is observed that the Micro-UAV can perform the transport task successfully using RL. Moreover, a hierarchical approach allows the swarm algorithms to be used with the RL algorithm. The simulations were performed on a single drone, but can be extended to the entire swarm.

### 3.7.8 Conclusion

This contribution describes a testbed for the development of resource-constrained Micro-UAV swarms. The testbed architecture combines external observation, visualization, and simulation systems with a drone control unit. The architecture in Figure 3.47 successfully integrates external systems with drones and ensures fast data exchange using radio signals. Swarm algorithms formulate a collision-free path for each drone. The Micro-UAVs successfully perform a transportation task using reinforcement learning in combination with swarm algorithms. Therefore, it is possible to integrate machine learning into the current setup, which opens up new opportunities in the use of machine learning with resource-constrained drones. This work describes how drones can be integrated into a process environment and operated in a meaningful way. In addition, an automatic charging station extends the runtime of the swarm.

### 3.7.9 Future Work

Industrial use of micro-UAVs will increase as sensors become smaller and more efficient for the use on drones. Thus, on-board sensors will be used instead of a MoCAP system for accurate localization in the future. Future works will include autonomous exploration of an environment, which will involve the swarm creating a map from a safe starting point and updating it in subsequent flights. The recharge station described in this paper will form the basis for this work. Moreover, human-machine and machine-machine

interaction will be developed in the hall to create a safe environment for humans to work alongside drones.

Various ML algorithms could be implemented directly on the drones to enable them to perform complex tasks. For example, ML could be used to learn the complex drone trajectories and motion patterns and thus could help to reduce the dependence on some of the systems in Figure 3.49. In the future, the different research halls could be equipped with a 5G installation that allows the drone swarm to fly between halls. Connecting the swarm via 5G to an already existing high-performance cluster for ML will enable larger-scale field studies for further development of the underlying algorithms.

# 4 Smart City and Traffic

## 4.1 Inner-City Traffic Flow Prediction with Sparse Sensors

*Thomas Liebig*

**Abstract:** The current traffic situation in urban areas and its forecasting are of interest to various application scenarios, as cities become more crowded and jammed. But the observation and monitoring of traffic situations are expensive, and thus estimates need to be imputed for unobserved locations and predicted for future locations.

In this contribution, we focus on a situation-aware routing use case, which prevents traffic jams. This system needs to take into account real-time estimates of unobserved and future traffic and present several probabilistic approaches to estimating traffic quantities.

### 4.1.1 Introduction: Problem Understanding

Traffic congestions are crucial problems of urban traffic, both for logistics, and passenger traffic. Data-driven, dynamic control and a mobility shift to automated vehicles could potentially ease current problems and lead to a mobility change in urban environments.

However, traffic systems are complex real-time systems with multiple actors, so control is difficult. Moreover, the observation of this process by measurements is sparse and prone to local (spatio-temporal) validity. Several real-time imputation and prediction steps are therefore prerequired, before the computation of meaningful dynamic recommendations for control is possible.

However, if individual navigation could take the predictions of future traffic situations into account, one would be able to avoid congested road segments or to decide on the best mode of travel in advance. Moreover, since some hazards such as traffic jams often occur by excessively high traffic densities, situation-aware trip planning would also cause fewer traffic congestions, and the infrastructure could be used more efficiently.

The tasks posed by situation-dependent routing are

- the prediction of future traffic situations from sparse observations,
- the utilization of dynamic predictions in planning, and
- the evaluation and selection of individual actions.


https://doi.org/10.1515/9783110785982-004

In the following sections, we address these tasks one after the other and reflect the contributions we made to these questions within the CRC 876. The contribution concludes with a discussion that highlights future research directions.

### 4.1.2 Traffic Prediction from Sparse Observations

In data mining, learning a prediction model is a well-defined supervised learning task. Given labeled observations $x, y$ from a space $X \times Y$, we train a model $f : X \to Y$ such that the expected loss $l$ between prediction $f(x)$ and the truly observed label $y$ becomes minimal. With this model, the prediction in a certain situation can be obtained by the application of the model to new data $y = f(x)$.

However, the modeling process is highly dependent on the available data, and the suitability of this approach depends on the entanglement of the modeled process.

In traffic control, we have a dynamic spatial process, and observed data is valid only for a limited extent in space and time. Moreover, the basic assumption of the supervised learning approach that the process repeats and, therefore, that past observations are suitable to project the future, does not necessarily hold in general. Traffic control and individual decisions depending on the expectation of future traffic behavior are counter indicators of such assumptions.

Based on these difficulties, various approaches to model traffic exist for different model assumptions, modes of transportation, and granularities.

#### 4.1.2.1 Gas Kinetic Models

In contrast to the data-driven supervised learning approach, one could start by observing patterns and physical properties of traffic and representing these in models. This model-driven approach is subject to the physics of transport and traffic theory. One of the basic observable properties of traffic is that individual moving objects (cars, pedestrians, etc.) do not disappear; rather, over time the number of objects entering spatial regions equals the number of objects leaving this region.[1] By formulating of this observation in a conservation law and deriving a general description model, we obtain so-called gas-kinetic traffic models. Systems of differential equations model traffic similar to any liquid or gas. Prominent examples of these models are the *Burgers Turbulence* and the *Navier Stokes Equation*. In macroscopic traffic modeling (focusing on traffic at gross granularity), gas kinetic models are often applied. A possible numeric approximation is *Force Based Models*, which update individual states of particles based on impacting forces and inertia. The critics of these approaches to traffic modeling are manifold [271]:

1 Note that this property requires a sufficiently long observation interval. For example in a living house people tend to rest and stay at night, while in a car park vehicles are stored until departure.

- If vehicles interact, the impulse and the kinetic energy are usually not preserved. Thus, Newton's Third law of motion (actio=reactio) is not applicable.
- The temperature of a vehicle fluid cannot be matched directly, as it is the variance of the vehicle speed.
- Vehicular gases do not move on account of external pressure but are caused by the inner intention to move at a certain speed.
- Due to the various movement targets, separate flows in different directions occur and interact.
- Vehicular behavior is anisotropic.

#### 4.1.2.2 Cellular Automaton

Cellular automata are a widely used model of physical processes. Introduced by Neumann [473], a cellular automaton features discrete space and time, with transition rules defining the future state of a cell based on its previous state and the state of its neighbors. The advantage of cellular automatons over dynamics is their scalability. Accordingly, boundary conditions are often implemented in a cellular automata model because they have a natural interpretation at this level of description (e.g. particles bouncing back on an obstacle). For traffic modeling, the NAGEL-SCHRECKENBERG MODEL is widely used.

A cellular automaton models a Markov chain, and the conditional probabilities of the future state are completely described by the current state. This Markov assumption does not generally hold for traffic at all granularities. Consider the movement on a highway or the queue at a traffic light. Without additional individual information on the past, one could not tell whether a vehicle leaves the highway or continues, whether the queue will start moving, and in which direction the cars turn at traffic lights.

#### 4.1.2.3 Hierarchy of Motion and Dependency Models

Previous models describe traffic as a Markov process. Current traffic observation and a fixed number of past observations suffice to predict the future. This approach neglects the sociological and psychological aspects of traffic. Persons are traveling by purpose and follow a certain plan to achieve this. Hoogendoorn thus defines the hierarchy of motion [280], which represents the different aspects of trip planning.

Observed traffic combines individual traffic plans and thus has a complex dependency structure which is hard to capture. Tobler's first law of geography [689] states that in a spatio-temporal process, geographically close observations are more related than distant observations. However, in traffic processes, this does not hold, as individual movement paths often start in a living area, use a larger street, and branch back to a tiny street where the working place is situated. So, the information about driving on a highway might be less informative to predict the goal of a trip than considering the few starting locations of the trajectory.

The dependency structure of the traffic process can be represented in a model as follows. Given any evidence of the presence of a moving object at several locations $x_i, \ldots, x_j$, we may model the likelihood of being at other places $p(x_k, \ldots, x_l | x_i, \ldots, x_j)$. If these dependencies are formulated without any circular dependencies, the joint probability distribution over the presence of a moving object at all locations factorizes as:

$$p(x_1, \ldots, x_n) = \prod_{i=1,\ldots,n} p(x_i | \text{pa}(x_i))$$

where $\text{pa}(x_i)$ are the parents of $x_i$ in the above equation. This directed model is called a *Bayesian network* and can be represented by a *graphical model* (a graph consisting of vertices and connecting directed edges) encoding dependencies as arrows between the random variables (the vertices) $\text{pa}(x_i) \rightarrow x_i$. A Bayesian network consists of a structure, given by the previous equation and the associated conditional probability tables for each variable based on its ancestors.

The dependency model can be trained directly from data by comparing whether the trained dependency model represents the same distribution as the traffic observations using a suitable loss function to compare distributions, for example the Kullback-Leibler-divergence. But due to the vast amount of random variables (one per addressable location, e.g., a street segment), learning the model requires some relaxation and approximation. In [406, 407], we propose an algorithm to learn spatial *Bayesian networks* from traffic observations.

#### 4.1.2.4 Gaussian Processes

By the central limit theorem, we know that observations converge towards a normal distribution when repeated multiple times or given a sufficient observation time. In the case of multidimensional observations (e.g., traffic counts at various locations in a street network) this converges to a stationary multivariate normal distribution. Assuming these traffic observations are generated by a probabilistic process, we can use the knowledge of the joint probability distribution to impute observations for unobserved locations (similar to the previous section). Under the assumptions described above, we may assume the observations were generated by a multivariate Gaussian distribution.

$$P(\mathbf{f} \mid \mathbf{X}) = \mathcal{N}(0, K)$$

The generating process is completely defined by the covariance between the variables. Due to the finite number of traffic observations (e.g., measuring locations at street segments), we can denote their pairwise covariances in a kernel matrix $K$.[2] When we observe some of these locations, we may impute the value at the other locations using the kernel matrix.

2 For an infinite number of observations, a kernel function had to be applied.

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_u \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \hat{K}_{-u,-u} + \sigma^2 I & \hat{K}_{-u,u} \\ \hat{K}_{u,-u} & \hat{K}_{u,u} \end{bmatrix}\right)$$

The correlations in traffic values imposed by the traffic network can be represented in the kernel $K$. Assuming that a person randomly moves on the traffic network, she travels in the network from one location to another. The dependency structure generated by these random walks can be captured by the diffusion kernel [336], where $L$ is the combinatorial Laplacian of the adjacency matrix, and $\lambda$ is a hyperparameter.

$$K = \left[\sum_{m=1}^{\infty} \frac{\lambda^m}{m!} L^m\right]_{ij} = \exp(\lambda L)$$

This kernel models every route choice as equally likely. But knowledge of real trajectories can be used to weight the adjacency matrix, and the correlation model can be detailed further [409].

$$m = \hat{K}_{u,-u}\left(\hat{K}_{-u,-u} + \sigma^2 I\right)^{-1} \mathbf{y}$$
$$\Sigma = \hat{K}_{u,u} - \hat{K}_{u,-u}\left(\hat{K}_{-u,-u} + \sigma^2 I\right)^{-1} \hat{K}_{-u,u}$$

A handy property of predictions made by a Gaussian process is that these predictions are normal distributions and are not just expectations. Thus, we may quantify the uncertainty of the predictions and use them to find near-optimal sensor placement [408].

However, the calculation of predictions requires an expensive matrix inversion step. With some preprocessing, the data can be grouped into nearly independent local chunks and the calculations can be distributed [111].

#### 4.1.2.5 Markov Random Fields

In previously described time-dependent models (fluid dynamics and cellular automata), the assumption is that future states are dependent on previous ones. This property is called the Markov property. If we now consider a field of random variables, and the Markov property holds for these variables, it is called a *Markov Random Field (MRF)*. A Markov random field is similar to a Bayesian network in its representation of dependencies. The difference is that Bayesian networks are directed and acyclic, whereas Markov networks are undirected and may be cyclic. Thus, a Markov network can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies). By contrast, it may not be able to represent certain dependencies that a Bayesian network can (e.g., induced dependencies). The graph of a Markov random field may be finite or infinite.

Any positive Markov random field can be written as an *exponential family*, such that the full joined distribution can be written as

$$P(X = x) = C \cdot \exp < w_k, f_k >$$

A multivariate normal distribution forms a *Markov random field* with respect to a graph $G = (V, E)$ if a correlation of zero corresponds to missing edges in the graph.

A tailored method to model traffic with *Markov random fields* is the spatio-temporal random field [499], which penalizes complex structures in the learning process by regularization and thus hold a sparse representation of the underlying distribution. The Markov property holds over time: the next observation is completely defined by current sensor readings. An application of this model to traffic is presented in [475].

#### 4.1.2.6 Poisson Dependency Models

Empirical observations of spatial phenomena are often count values. For instance, density is the number of objects in a spatial area and flow is the number of objects passing a location in a given time interval. While previous probabilistic models primarily model categorical data or multivariate normal distributed observations, Poisson models seem to be a natural fit, as count values are neither binary nor continuous but are discrete with a right-skewed distribution over an infinite range [249]. A possible approach to combining graphical modeling with Poisson distributions is to use an ENSEMBLE of POISSON REGRESSION TREES [249], each modeling a conditional Poisson distribution. With this model, the underlying joint distribution is unknown and local distributions might be inconsistent. Thus, Pseudo Gibbs Sampling [265] is required to impute unobserved measurements from given evidence. This algorithm initializes the unobserved variables arbitrarily at random and then updates these values according to the conditional distribution given its parent variables. After a burn-in phase, which is highly dependent on the initial distribution, this algorithm draws samples from the joint distribution.

For the imputation of unobserved traffic values, this model outperforms exponential models in [247]. On a massive dataset, training these dependency models is challenging. In [451], the authors show how dependency networks can be trained on core sets, a compressed dataset that can be used as a proxy for the original data. For their algorithm, there is a proven guarantee that in the case of Gaussian dependency networks, the size of the coreset is independent of the size of the dataset. This property does not hold in general, i.e., for *Poisson dependency networks*, it does not hold.

#### 4.1.2.7 Conditional Sum-Product Networks

The need for tractable inference in graphical Poisson models led to the adoption of sum-product networks [506] in Poisson distributions [452]. The graphical structure of these sum-product networks consists of a tree having alternating sum and product nodes in the layers and Poisson-distributed random variables in the leaves. For inference, the structure just needs to be traversed once from the bottom to the root. For training the model, independences between sets of random variables are estimated. In case of no independence, similar objects are grouped into clusters, and a sum-node is introduced. In the case of independences, a product-node is constructed.

For modeling traffic, a model that combines the physical properties (as represented by a cellular automaton) with probabilistic data-driven models would be beneficial. In [597, 598], we show how to model Markov processes with sum-product networks by conditioning them on the previous time slice. The model can represent cellular automaton, is data driven, and outperforms previous Poisson dependency networks for traffic prediction.

#### 4.1.2.8 Differentially Private Learning from Label Proportions

Traffic data is usually collected in a centralized manner, which results in high data transfer and data protection risks. It is especially important that data protection risks are addressed by institutions, due to the introduction of GDPR in all EU countries in 2018 [231]. Organizations want to use this data in order to gain more information or predict future sensor states, e.g., "Will the traffic flow stay the same over the next 15–30 minutes?" Accordingly, they also have to be compliant with GDPR.

Therefore we extend the decentralized learning approach from [651, 655] by applying *differential privacy* to label proportions sent between the different decentralized sensor devices resulting in a privacy-preserving algorithm. In general, the `Learning from Label Proportions` (LLP) algorithm stays the same as proposed in [651]. Because it is important to know the structure and flow of the algorithm, we will briefly consider it further. There are $m$ wireless sensor nodes $(n_1, n_2, \dots n_m)$, which store their measurements in $D(i)$ $\forall i \in 1 \dots m$. Each row in $D(i)$ consists of $[t - w, t]$ measurements, where $t$ denotes a timestamp and $w$ is the *window size* of the last $w$ measurements. Each row is assigned a label, which is taken from a measured value from a future timestamp $t + r$. In the first place, those measurements are split into batches $B_1, \dots, B_h$ where $h = \lceil |D(i)|/b \rceil$ and $b$ denotes the size of the batches, in which $D(i)$ will be divided. The *batches* are then used to calculate *label proportions* for each batch. The generated *label proportions* are sent to the closest $c$ neighbors. Each node uses the received *label proportions* to train $c + 1$ models $f_j(k)$, where $k \in 1, \dots, c + 1$ and $j$ is the current node. The prediction is made by doing a majority voting of the $c + 1$ trained models. This approach has the advantage that we can make use of more than only local measured data point while keeping the bandwidth of transferred data low because only *aggregated data* is sent between the nodes. However, privacy cannot be guaranteed by this approach. Assuming we have traffic flow measurement values, with labels 0, 1, 2, 3, 4 and over a time frame of size $b$ only label 4 is present. Then, from the label proportion, it can be inferred that everyone drove that fast during the period.

We solve this issue by applying *differential privacy* to the label proportions. Firstly, we have to calculate the $l_1$-sensitivity function to know, how much influence a single data point can make on the output of a function $f : D \to R$:

$$\triangle f = \max_{\substack{D', D'' \in D, \\ ||D' - D''||_1 = 1}} ||f(D') - f(D'')||_1 \tag{4.1}$$

For this scenario, $D$ is the current batch $B_i$, and $R$ is the resulting *label count*. Considering that we have a simple counting query, a single data point can have at most the influence of 1 (see [185] example 3.1). Finally, we can use the *Laplace distribution* to generate noise, which can be added to the *label counts* to be privacy-compliant under $\epsilon$-differential privacy [185]:

$$lap(x, \sigma, \mu) = \frac{1}{2\sigma} e^{-\frac{|x-\mu|}{\sigma}} \tag{4.2}$$

$$lap(x, \frac{\triangle f}{\epsilon}, 0) = \frac{\epsilon}{2 \triangle f} e^{-\frac{|x-0|\epsilon}{\triangle f}} \tag{4.3}$$

In the formula above, the position parameter $\mu$ is set to 0, and the *scale* parameter is set to $\frac{\triangle f}{\epsilon}$. These parameters have to be set like this to be compliant with the *differential privacy* definition (proof can be found in [185] Theorem 3.6).

The modified algorithm for calculating label counts can be seen below. As mentioned before, the batches $B_i$ are already generated, and possible labels $Y$ are also known. The output $Q(j)$ contains differentially private label proportions for all batches.

**Algorithm 3:**

**Input:** $B_1, \ldots, B_h, Y$
**Output:** $Q(j)$

```
1  Q(j) ← matrix(h, |Y|);
2  for i in 1..h do
3      for j in 1..|Y| do
4          Q(j)_{i,j} ← sum(B_i == Y_j);
5      end
6      // adding noise to label counts
7      m ← sum(Q(j)_i);
8      for j in 1..|Y| do
9          Q(j)_{i,j} ← Q(j)_{i,j} + lap(e = 0, s = 1/ϵ);
10         clip Q(j)_{i,j} to bounds [0.001, m];
11         normalize Q(j)_i;
12     end
13 end
```

Initially, $Q(j)$ is created with dimensions count batches ($h$) and count possible labels ($|Y|$). Afterward, the label proportions are calculated iteratively for each batch as follows. First, the label counts (see lines 3–5) and the total sum (see line 7) are calculated. Then the Laplace noise, which is calculated by the sensitivity and $\epsilon$, is applied. Afterward, the new value is clipped to the maximum bounds to prevent values that are too large or negative. Finally, the label counts with noise are normalized. The resulting proportion is stored in $Q(j)$.

The proposed approach is based on the existing LLP algorithm to use the decentralized properties and extend this approach by applying differential privacy to the transferred data. This yields reduced data transfer and increased privacy [568].

### 4.1.3 Efficient Routing with Dynamic Predictions

In a changing world, geo-spatial data is subject to dynamic changes, and geo-information systems are required to incorporate real-time updates in their analysis and computations. In this section, we focus particularly on route-planning systems. While in a static world, many algorithms exist to compute the (shortest) path from a starting location to a target location efficiently (see Section 4.1.2), this problem becomes more difficult in the case of multi-modal trip planning, as with public transport, because temporal constraints, e.g., transit times and departure times, need to be incorporated. In the real world, these static schedules are not met, but delays occur [439], and deviations from the schedule can be observed. The incorporation of this dynamic information in route computation is beneficial, as it provides tractable travel recommendations to the public. The dynamic information on the delays can be achieved by monitoring the positions of the vehicles and by predicting future delays. This enables proactive trip computation.

In this section, we focus on the tractability of dynamic transit computation. Existing single-source shortest path computation algorithms for the dynamic transit problem suffer from their long computation time. `Transfer Pattern`, a very fast route planning algorithm for transit networks, does not guarantee soundness in case of real-time delay information. Our approach [411] overcomes these shortcomings and introduces dynamic transfer patterns, a data structure that encodes which novel transit possibilities are enabled due to the delays.

In comparison with existing dynamic transit-routing schemes in the city of Warsaw, we highlight the performance gain using our method. Our findings are implemented in the commonly used open-source trip planning framework OpenTripPlanner.

Here, we focus on the point-to-point shortest path problem [49], where in a graph $G = (V, E)$ a path between a source $s \in V$ and target $t \in V$ needs to be found such that the cumulative edgewise cost $l(u, v), with(u, v) \in E \subseteq V \times V$ along the path is minimized.

The standard solution to the problem is Dijkstra's algorithm [176]. Given the graph $G = (V, E)$ and $s, t \in V$, it initializes a queue of nodes $Q = V$ and a distance function over $V \times V$ with $dist(s, s) = 0$ and $dist(s, v) = \infty, \forall v \neq s, v \in V$. Until the queue is empty, the node $u$ with the smallest distance $dist(s, u)$ is picked and removed from $Q$. For each neighboring node of $u$, the distance is updated as follows: $dist(s, v) := dist(s, u) + l(u, v)$, if the latter is smaller than the former. Dijkstra's algorithm can be sped up by running it simultaneously from both $s$ and $t$ until a common node $u$ is hit. In the slightly modified version of Dijkstra's algorithm A$^*$ [258], the order in the priority-