Herausgeber

H. SCHULTE

F. HOFFMANN

R. MIKUT

Berlin | 23. – 24. November 2023

PROCEEDINGS **33. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

H. Schulte, F. Hoffmann, R. Mikut (Hrsg.)

Proceedings. 33. Workshop Computational Intelligence

Berlin, 23. – 24. November 2023

PROCEEDINGS **33. WORKSHOP** COMPUTATIONAL INTELLIGENCE

Berlin, 23. – 24. November 2023

Herausgegeben von
H. Schulte
F. Hoffmann
R. Mikut

KIT Scientific Publishing

Impressum

KIT Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu

Print on Demand 2023 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-1324-7
DOI 10.5445/KSP/1000162754

Inhaltsverzeichnis

Cognitive Architecture for Artificial Intelligence: Evaluating Realworld Applicability and the Significance of Online Machine Learning

Richard Schulz[1], Alexander Hinterleitner[1], Lukas Hans[1], Aleksandr Subbotin[1], Nils Barthel[1], Noah Pütz[1], Martin Rosellen[1], Thomas Bartz-Beielstein[1], Christoph Geng[2], Phillip Priss[2]

[1]Institute for Data Science, Engineering, and Analytics, TH Köln
Steinmüllerallee 6, 51643 Gummersbach
E-Mail: {richard.schulz, alexander.hinterleitner, lukas.hans, aleksandr.subbotin, nils.barthel, noah.pütz, martin.rosellen, thomas.bartz-beielstein}@th-koeln.de

[2] TH OWL
Campusallee 12, 32657 Lemgo
E-Mail: {christoph.geng, phillip.priss}@th-owl.de

1 Introduction

As the integration of hardware and software continues to evolve, production systems are becoming increasingly intricate, now often referred to as Cyber-Physical Production Systems (CPPS). Particularly, Artificial Intelligence (AI) can be instrumental in improving processes such as anomaly detection, optimization, or predictive maintenance. However, at the moment, incorporating AI algorithms into these systems is far from straightforward; it demands substantial time, financial resources, and expertise. Adopting a standardized architecture could facilitate the integration of AI technologies, especially for small and medium-sized enterprises, empowering them to remain competitive. For this reason the Cognitive Architecture for Artificial Intelligence (CAAI) was introduced in [1] as cognitive architecture for AI in CPPS. The goal of the system is to reduce the implementation effort by creating a standard architecture. The core of the CAAI is a cognitive module that processes the user's declarative goals, selects suitable models and algorithms, and creates a configuration for the execution of a processing pipeline on a big data platform. During the revision of the CAAI project, it became obvious that there are existing limitations in automating the algorithm pipeline development process for all environments and use cases. This is mainly due to rapidly changing software interfaces and transfer complexities. Additionally, it became apparent that the architecture in the use case for CPPS provides a good environment for the implementation of online machine learning (OML) algorithms. This can be explained by the continuous data streams produced by the system's machines. This abstract assesses the potential advantages of OML algorithms through an analysis of a real-world application in slitting machines.

Section 2 roughly describes the concept of OML. Furthermore, it includes a description of the experimental setup, including its real-world application. In Section 3 the results of the experiment are discussed. Finally, a short conclusion is presented at the end of the abstract.

2 Materials and Methods

2.1 Online Machine Learning

The amount of data generated from various sources has increased enormously in recent years ("Big Data"). Technological advances have enabled the continuous collection of data. Web, social media, share prices, search queries, but also sensors of modern machines produce continious streams of data. It becomes more and more challenging to store and process these infinite streams. Traditional batch machine learning is the common strategy to train machine learning models. It basically boils down to the following steps[2]:

1. Loading and pre-processing the train data;
2. Fitting a model to the data;
3. Calculating the performance of the model on the test data;

In modern OML approaches, we do not train the model with the entire dataset at once; instead, we update it incrementally with the newly arriving data. This way, we avoid storing large amounts of data by discarding it after the updating process. This procedure might be beneficial in terms of time and memory consumption. Additionally, this continuous updating allows OML methods to better address structural changes within the data, known as concept drift.The introduction of different methods of incremental learning has been quite slow over the years, but the situation is changing at the moment [3] [4] [5].

To compare OML with the classical batch learning method in our experiments, a Hoeffding Tree regressor (HTR) from the Python 'river' library [3] is used for online learning, while a Decision Tree regressor from the 'scikit-learn' [8] package is used for the classical approaches. In online machine learning, Hoeffding trees are preferred because they do not rely on previously used instances, instead they await the arrival of new instances [7]. Given their incremental learning capacity, Hoeffding trees are more adept at handling a data streaming context compared to traditional methods.

2.2 Experiment Setup: Slitting Machines

In the experiments discussed in this work, data was collected using a test setup for winding stations from "Kampf Schneid- und Wickeltechnik GmbH & Co. KG", a company that specializes in building machines for slitting and winding web-shaped materials. The idea of the experiment is to use the motor torque and revolution values of a slitting machine to predict the vibration level. All features are available in form of time series with with measuring intervals of 10 ms.

For our experiment, we divide the data into a training and a test set. The goal is to compare the prediction performance over an evaluation horizon that is subdivided into segments of 150 data points. Additionally, we analyze the calculation time and memory consumption during the evaluation. To compare OML with classical approaches, and to assess their respective strengths and weaknesses, we utilize four different approaches in our experiment. For all approaches, we train the initial model based on the training set before we start with the evaluation process. Three of the algorithms belong to the group of batch machine learning techniques. The first is the classical batch learning approach, where the model is trained only once on the training set and subsequently evaluated on the test set. Another method is the landmark approach, where we add the data from the current horizon to the training set after each evaluation step, and then train the model from scratch. The final batch learning method is the shifting window approach. Unlike the landmark approach, the algorithm is not trained on the entire set of observed data, but rather on a moving window of data. In our case, this window is the size of the initial training set, meaning we add 150 data points with each evaluation step and remove the earliest 150 data points. The last evaluation technique is the pure OML approach where we update the model incrementally after each prediction step with the new data. The implementations of all evaluation strategies can be found under the following link: `https://github.com/sequential-parameter-optimization/spotRiver`.

3 Results

In the following part, we compare the different approaches as introduced in Section 2.2.

The performance of the different approaches is visualized in the top graph of Figure 1. It shows how the MAE evolves over the evaluation horizon. All batch learning evaluation methods produce comparable results. Initially, performance degrades slightly and then improves continuously. The OML approach comparatively achieves constant results and outperforms the batch evaluations over the entire horizon.

The second diagram in Figure 1 shows a comparison of the computation times of the different methods. As assumed, the landmark and shifting-window methods show a continuous increase in computation time due to the models need to be retrained at each evaluation iteration. In contrast, the conventional batch learning approach exhibits a much lower processing time because of its singular model training phase. On the other hand, the OML algorithm

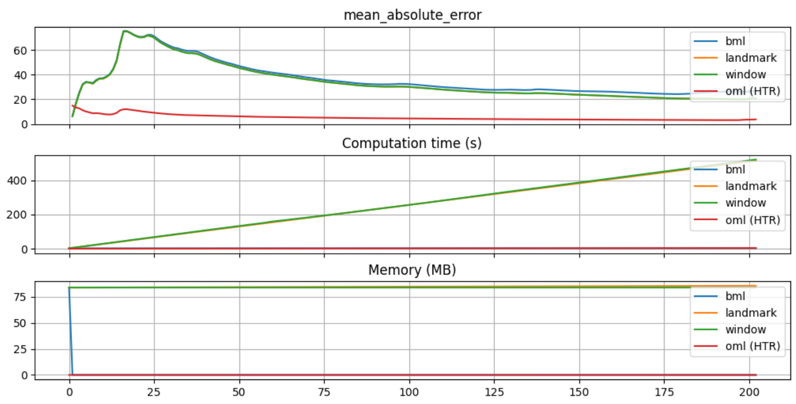


Figure 1: The MAE, computation time, and memory consumption of different approaches for each evaluation step. The MAE plot shows an overlap of the graphs of all bml methods. In the plots for computation time and memory consumption, the curves of landmark and shifting window, as well as the OML and the classical batch method overlap.

achieves time efficient results as well. This is because OML updates models incrementally, rather than training from scratch with each evaluation.

The lowest graph of Figure 1 shows the memory consumption. Here, the OML approach also delivers comparable results to the basic batch approach. However, it should be emphasized again that the batch approach's memory consumption only takes place during the training step, and the remaining consumption is negligible. This fact is also visualized by the graph. In the first evaluation step, the memory consumption for the classic batch method drops towards zero. The shifting window and the landmark approach perform comparably poorly. This is mainly due to the generated model, which must be built again in each iteration.

Furthermore, the assertions regarding the superior performance of the OML algorithms have been statistically substantiated by a one-sided t-test. This test demonstrates that the average deviation of predictions made by the OML algorithm from the actual values is significantly smaller than that observed in predictions from all batch learning approaches.

4 Conclusion and Discussion

The OML algorithms outperformed the classical approaches not only in terms of memory consumption and computation time, but they also achieved significantly better results in terms of prediction accuracy. This can be attributed to the algorithms' enhanced responsiveness to concept drift, a decisive advantage especially in the domain of production machinery. Further improvements to the results presented here could be realized through additional experiments, particularly with surrogate model based optimization of the hyperparameters. This evidence suggests that OML algorithms should undoubtedly be considered in the development of CAAI for CPPS.

Acknowledgement

This research work is funded by the German Federal Ministry for Economic Affairs and Climate Action as part of the project "*IMProvT_II* – Intelligente Messverfahren zur energetischen Prozessoptimierung von Trinkwasserbereitstellung und -verteilung II", funding code 03EN2086A.

References

[1] A. Fischbach, J. Strohschein, A. Bunte, J. Stork, H. Faeskorn-Woyke, N. Moriz, and T. Bartz-Beielstein. "CAAI—a cognitive architecture to introduce artificial intelligence in cyber-physical production systems". The International Journal of Advanced Manufacturing Technology, 111:609–626. Springer. 2020.

[2] E. Bartz, T. Bartz-Beielstein, M. Zaefferer, and O. Mersmann. "Hyperparameter Tuning for Machine and Deep Learning with R: A Practical Guide". Springer Nature. 2023.

[3] J. Montiel, M. Halford, S.M. Mastelini, G. Bolmier, R. Sourty, R. Vaysse, A. Zouitine, H.M. Gomes, J. Read, T. Abdessalem, et al. “River: machine learning for streaming data in Python”. 2021.

[4] A. Bifet, R. Gavalda, G. Holmes, B. Pfahringer. “Machine Learning for Data Streams with Practical Examples in MOA”. MIT Press. 2018. `https://moa.cms.waikato.ac.nz/book`.

[5] V. Losing, B. Hammer, H. Wersing. “Incremental on-line learning: A review and comparison of state of the art algorithms”. Neurocomputing, 275, 1261–1274. 2018. `https://doi.org/10.1016/j.neucom.2017.06.084`.

[6] A. Bifet, R. Gavalda. “Adaptive learning from evolving data streams”. In Proceedings of the Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8. Springer. 2009, pp. 249–260.

[7] P.M. Domingos, G. Hulten. “Mining high-speed data streams”. In Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000; R. Ramakrishnan, S.J. Stolfo, R.J. Bayardo, I. Parsa, Eds. ACM. 2000, pp. 71–80. `https://doi.org/10.1145/347090.347107`.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. “Scikit-learn: Machine Learning in Python”. Journal of Machine Learning Research. 2011, 12, pp. 2825–2830.

Active Learning for Regression Problems with Ensemble Methods

Bjarne Jaster, Martin Kohlhase

Hochschule Bielefeld, Center for Applied Data Science Gütersloh
Interaktion 1, 33619 Bielefeld
E-Mail: {bjarne.jaster, martin.kohlhase}@hsbi.de

1 Introduction

Traditional machine learning paradigms depend on the availability of labeled data, a luxury that is not often the reality in real-world scenarios. In domains such as industry, healthcare, autonomous systems and finances a massive amount of unlabeled data is produced every day. As the demand for accurate and robust models to deal with this data grows, the inefficiency and the cost of manual labeling motivates the research field active learning [1].

Active learning describes an efficient and effective way of selecting the most valuable data samples for labeling. The value of a sample is defined by a criterion which is unique to each active learning strategy. This selection reduces the amount of manual labeling, optimizes resource allocation, and enhances model performance in real-world scenarios. This makes active learning a pivotal tool for domains where labeled data is scarce or costly. A common criterion for active learning method is the informativeness of a data sample. This is measured by the uncertainty of a Machine Learning (ML) model in its prediction [2]. The uncertainty estimation is dependent on the ML-model, which motivates this work to explore the quality of different ML-models and their uncertainty estimation methods for active learning. Additionally, the computational effort of different uncertainty estimators is explored, because there often is a trade-off between the accuracy and the computational effort of an uncertainty estimation method. For instance, probabilistic techniques such as *Fully Bayesian*

Gaussian Processes (FB-GPs) and *Bayesian Neural Networks* provide accurate uncertainty estimates at the cost of computational complexity due to parameter sampling from an intractable distribution, commonly done with *Markov Chain Monte Carlo (MCMC)* Sampling [3]. *Neural Networks* and their softmax-layer probabilities, on the other hand, provide less accurate and often overconfident uncertainty estimates [4] but add none additional computational effort.

2 Related Work

Active learning describes the process of selecting unlabeled data samples. An active learner is characterized by a predefined budget, representing the quantity of data samples it can actively select, and a criterion quantifying the value of a given data sample. The active learner chooses the most valuable data samples, measured by the criterion, and requests labels for these selections. The criterion typically relies on either the spatial distribution of data samples [5] to maximize the training set diversity or utilizes uncertainty estimates [2] to minimize regions in the input-space characterized by high predictive uncertainty. Leveraging uncertainty estimates requires the use of a ML-model capable of quantifying predictive uncertainty. By iteratively selecting new samples, the active learner enhances its performance, often achieving better results with fewer labeled examples compared to traditional passive learning methods.

The research field of active learning is divided into three subfields [6]: *Pool-based Sampling* is the most common subfield of active learning. It involves selecting instances for labeling from a fixed pool of unlabeled data. The algorithm ranks instances within this pool based on its selection criterion. The selected instances are then labeled and added to the training set. *Membership Query Synthesis* involves generating label-queries synthetically based on its current knowledge, instead of selecting instances from an existing dataset. These queries are designed to be informative and help the model to learn more effectively. *Stream-based Selective Sampling* focuses on scenarios where data arrives in a continuous stream, and labeling resources are limited. In this subfield, the active learning algorithm processes data instances one by one as they arrive. It decides on-the-fly whether to label the current instance or wait for a more valuable one based on the selection criterion. This work focuses on the pool-based sampling approach to active learning.

The concept of uncertainty, which can be used as an active learning criterion, is commonly divided into two parts: Epistemic uncertainty refers to a systematic uncertainty and results from incomplete or missing knowledge. This uncertainty is caused by factors like small data sets and other influences arising from an incomplete and potentially faulty data source, as well as incomplete knowledge about the process being modeled. Aleatoric uncertainty represents the part of uncertainty that cannot be reduced, including statistical relationships such as noise or fundamentally random connections in the data [7]. When using uncertainty as a criterion for active learning, especially the epistemic uncertainty is of interest, because data samples with high epistemic uncertainty carry the information that can increase the performance of the model [8].

2.1 Gaussian Processes

One machine learning method that provides an uncertainty estimate is a *Gaussian Process (GP)*. GPs are a good model choice for active learning because they are well-suited for smaller datasets, a crucial characteristic due to the limited amount of training samples during the active learning process. However, there are two issues associated with GPs, when used for active learning. Firstly, they rely on proper hyperparameter selection, which poses a challenge as the costly labeling of data samples makes it difficult to justify withholding samples for testing and hyperparameter tuning. Consequently, hyperparameters must often be chosen based on heuristics or expert knowledge, which may not be available or applicable in all cases. Secondly, GPs' uncertainty estimate do not differentiate between epistemic and aleatoric uncertainty.

These issues have motivated Riis et al. [2] to explore the use of FB-GPs for active learning. The concept involves sampling the hyperparameters, commonly noise and lengthscale, from a posterior distribution conditioned on the training samples. This directly addresses the first issue and enables the creation of an ensemble of GPs. Combined with the law of total variance $V(y|x) =$

$V(E[y|x]) + E[V(y|x)]$, this approach allows for the decomposition of total uncertainty into epistemic $V(E[y|x])$ and aleatoric uncertainty $E[V(y|x)]$. This results in a significantly more accurate and useful uncertainty estimate for active learning.

2.2 Random Forests

Random Forests are a widely used ensemble learning technique, that uses multiple decision trees to make accurate and robust predictions. They are able to model diverse datasets without requiring extensive hyperparameter tuning [9]. Additionally, [10] shows that they are the best ML-method for small to medium sized real-world datasets. These features make Random Forest a good candidate for an ML-model in active learning problems.

In [11] different ways to estimate the uncertainty of a prediction of a Random Forest are described. First, they argue that the standard approach of estimating uncertainty for an ensemble, taking the variance of the individual predictions, is not suitable for Random Forest. This is due to the different training sets and feature selections in the individual trees. Then, two suitable methods for estimating the uncertainty are presented: The first method, denoted as the *Jackknife Estimate*, calculates the Leave-One-Out Error implicitly [12] and uses its average as the uncertainty. This is done by computing the difference between the average prediction made by trees not trained on a particular sample and the average prediction generated by the entire ensemble. The second method, referred to as the *Infinitesimal Jackknife estimator* [13], introduces a novel approach. It down-weighs each training sample by an infinitesimal amount and computes the variance of a prediction over all training samples. The variance can be interpreted as the uncertainty. To enhance the reliability and accuracy of uncertainty estimates from both the Jackknife and the Infinitesimal Jackknife estimator, the authors present unbiased versions of these methods. These enhancements correct the inherent upward bias observed in the initial estimations, ultimately resulting in better-calibrated uncertainty assessments. These estimates do not distinguish the two parts of uncertainty. Although a method exists to differentiate between aleatoric and epistemic uncertainty specifically for Random Forests [14], it is designed for classification tasks and does not easily extend to regression problems.

2.3 Neural Networks

Neural Networks are widely used models, especially for large datasets [15]. They also provide probabilities in their predictions when trained on a classification problem, which can be used as an uncertainty estimate. However, the uncertainty (entropy of the predicted class probabilities) of these predictions is both poorly calibrated [4] and not applicable to regression problems, as Neural Networks trained on regression problems only provide point predictions. One approach to obtain uncertainty estimates is by training an ensemble of Neural Networks [16], each initialized with different weights. This results in different Neural Networks converging to various local minima of the loss function. However, training multiple Neural Networks can be computationally expensive. Thus, implicit ensembling techniques that require the training of only a single Neural Network and yield well calibrated uncertainty estimates are presented.

The first technique, known as *Dropout*, is a well-established regularization technique applied during the training [17]. Dropout operates by randomly setting the output of a fraction of neurons in each layer to zero with a specific probability, effectively excluding their contribution to the networks output. Importantly, this random dropout of neurons varies during each training iteration, encouraging the development of redundant representations and mitigating the risk of overfitting. Notably, during testing or inference, Dropout is deactivated to allow the model to utilize its full predictive capability. However, when Dropout is retained and applied during testing, it can be demonstrated that the mean and variance of multiple forward passes approximate the behavior of Bayesian Neural Networks [18]. The underlying concept of the uncertainty estimation is that the model has formed redundant representations for test samples closely aligned with the training data, resulting in a lower variance in the predictions. Conversely, for samples more distant from the training data, the model lacks this redundancy, leading to a higher variance in predictions as they become heavily reliant on specific neurons.

Table 1: Synthetic Datasets from [23]

Name	No. of Features	Noise	Input Space
Gramacy1d	1	0.1	$[0.5, 2.5]$
Higdon	1	0.1	$[0, 20]$
Gramacy2d	2	0.01	$[-2, 6]^2$
Branin	2	11.32	$[-5, 10] \times [0, 15]$
Ishigami	3	0.187	$[-\pi, \pi]^3$
Friedman	5	0.1	$[0, 1]^5$
Hartmann	6	0.01	$[0, 1]^6$

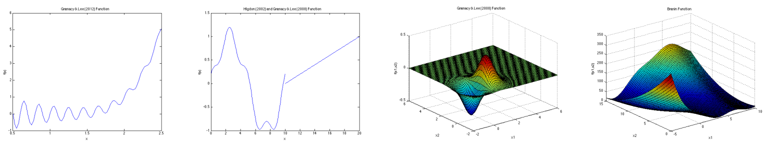
The second uncertainty estimation method utilizes *DropConnect*, which is conceptually similar to Dropout. Initially developed as a regularization technique for Neural Networks [19] too, DropConnect sets connections between neurons to zero, instead of the output of the neurons. The authors of [20] experimentally show that the variance of multiple DropConnect-Neural Networks predictions provides better calibrated uncertainty estimates than the variance of a Dropout-Neural Network.

The third method, *Local Ensembles* [21], approximates the variance of an ensemble of equally competent predictors without explicitly constructing the ensemble. To achieve this, the weights of the Neural Networks are perturbated in the direction of the smallest eigenvectors of the Hessian of the loss function. These eigenvectors represent directions of low curvature, indicating flat regions on the loss landscape, where weight-perturbations have minimal impact on the loss. To efficiently approximate these eigenvectors, *Lanczos iteration* is employed [22].

3 Experimental Procedure

3.1 Datasets

We use a comprehensive set of datasets, comprising seven synthetic functions (Tab. 1) and eleven real-world datasets (Tab. 2). Six of the seven synthetic



(a) Gramacy1d (b) Higdon (c) Gramacy2d (d) Branin

Figure 1: Visualizations of the 1d and 2d synthetic functions (taken from [23])

Table 2: Real World Datasets from [26]

Name	No. of Samples	No. of Features
auto-mpg	397	7
concrete-data	1030	8
cps-wages	534	18
housing	452	13
no2	500	7
pm10	397	7
real-estate-valuation	414	6
slump-test-slump	103	7
slump-test-flow	103	7
slump-test-compressive-strength	103	7
winequality-red	1599	11
winequality-white	4898	11
yacht-hydrodynamics	308	6

functions, described in [23], are employed in benchmarking FB-GP based active learning methods [2]. These synthetic functions serve to demonstrate the models' capabilities in addressing common challenges: The function *Gramacy1d* (Fig. 1a) assesses the models' ability to distinguish noise from signal. *Higdon* (Fig. 1b) and *Gramacy2d* (Fig. 1c) feature both linear and non-linear regions. The other synthetic functions are of higher dimension and some are characterized by strong non-linear behavior, enabling the evaluation of the models performance in complex scenarios. The *Friedman* function has a well-established reputation in regression problems, having been previously employed by Friedman et al. [24] and Breiman [25]. The datasets for every function are created by drawing 2000 samples uniformly from the defined input space and by adding Gaussian white noise to the function output.

While synthetic datasets offer valuable insights into specialized problem scenarios, the evaluation of active learning methods on real-world datasets is also relevant, because real-world dataset either combine multiple synthetic scenarios or exhibit behavior not covered with synthetic functions. To this end, eleven real-world datasets that are consistent with those used by Wu et al. [5] are incorporated. These datasets are sourced from the UCI and CMU StatLib repository [26]. An overview over the datasets and their number of data samples and features is given in Tab. 2

The inputs and outputs of all datasets are normalized to have zero mean and standard deviation one. Categorical features are one-hot-encoded for compatibility with the used ML-models. In the *slump-test* dataset, three target values are present. To deal with that each target is viewed as an individual dataset.

3.2 Active Learning Methods

Our evaluation focuses on three primary ML-models: FB-GPs, Random Forests, and Neural Networks. For FB-GPs, three distinct uncertainty estimation approaches are employed:

- Mean of the predicted variances: Represents aleatoric uncertainty
- Variance of the predicted means: Represents epistemic uncertainty
- Combination of the mentioned criteria: Represents total uncertainty

Random Forests are evaluated using the Jackknife and Infinitesimal-Jackknife estimators, along with their bias-corrected counterparts. Neural Networks undergo assessment with three presented uncertainty estimation methods: Dropout, DropConnect, and Local Ensembles. Additionally, a passive learning baseline for each machine learning method is provided, which randomly selects the same amount of data samples as the active learning methods.

Each method starts with an initial training set. The size of this set is equal to the dimensionality of the dataset. In case of a 1d- or 2d-dataset it consists of three samples. The samples of the initial set are selected as follows: The first sample is selected as the one closest to the mean of the input of the data. Subsequently, the remaining initial training samples are chosen based on their maximum distance to their nearest training sample. This is done to provide a deterministic initial training set with good diversity to the ML-models. This way, the starting conditions are equal and not dependent on random selection of initial data samples. Next, we train an ML-model and compute uncertainties for each unlabeled data sample. The sample with the highest uncertainty is then added to the training set, and the ML-model is retrained. This selection process is iterated 50 times.

3.2.1 Training of ML-Methods

Hyperparameter tuning for machine learning methods typically relies on a separate test or validation set. However, in the active learning scenario with limited data availability, this approach is impractical. Therefore, hyperparameters are chosen based on established rules of thumbs or taken from other work that dealt with similar problems.

For FB-GPs, the hyperparameters lengthscale and noise are sampled from a distribution, requiring the user to specify only the ensemble size, which is set to 800 to manage computational resources. For Random Forests, hyperparameters such as tree size and the number of input variables considered in each split are configured in alignment with Breiman's original Random Forests paper [9], which shows minimal sensitivity to the second hyperparameter. Neural Networks depend on a multitude of hyperparameters, with the network-size being the most critical. Given the limited number of training samples in active learning (at most 50 to 70), we chose one-hidden-layer networks with 50 neurons, inspired by Tohme [27]. The hyperbolic tangent is used as the activation function, and networks are trained with the ADAM optimizer (learning-rate: 0.01). To promote stability of the training results, network weights from the previous active learning iteration are used for initialization.

To mitigate the risk of overfitting, regularization techniques are employed. Specifically, Dropout and DropConnect are incorporated, which are already integrated into their respective uncertainty estimation methods. Dropout is also applied to the Neural Networks that are used for the Local Ensembles-method. The proportion of neurons or connections that are dropped is set to 0.05, similar to findings from [17]. Additionally, we implement early stopping based on training error to accelerate the training process when convergence is reached. This approach is particularly advantageous when combined with the weight initialization from the preceding iteration, as the model is likely near a local minimum.

3.3 Evaluation Process

In other work the quality of an active learning method is commonly assessed by providing learning-curves, which illustrate the final performance of the ML-model as well as the speed and the stability of the learning process. Due to space constraints we only show the most important metric, which is the quality of the ML-model after the final amount of training samples is acquired. For regression problems, the quality is a commonly assessed by the Root Mean Square Error (RMSE). We use the normalized-RMSE which allows comparisons over multiple dataset:

$$\text{normalized-RMSE} = \frac{\sqrt{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}}{\text{Var}(y)}$$

The normalized-RMSE is computed over the remaining unlabeled data samples from the pool, demonstrating the generalization capabilities of the model. The normalized-RMSE is also computed for the passive learning baselines, which randomly select the same amount of training samples as the active learning methods. This allows for an evaluation of the uncertainty based selection criterion. All active and passive learning methods are run ten times per dataset.

In addition to comparing the quality of the different active learning methods, the computational effort of each method is considered. The average time it takes to acquire the final amount of data samples per method is calculated, ensuring uniform hardware conditions for all experiments to allow a comparison.

Table 3: normalized-RMSE after the final active learning iteration on real-world datasets, averaged over 10 runs. The best method for each dataset is printed bold. The last row shows the average score over all datasets (RF - Random Forest, NN - Neural Network)

	FB-GP		**RF**		**NN**	
	Random	Best	Random	Best	Random	Best
auto-mpg	0.18	**0.12**	0.18	0.16	0.23	0.25
concrete	0.28	**0.25**	0.40	0.48	0.33	0.41
cps-wages	1.04	0.90	0.85	**0.78**	1.06	1.04
housing	0.27	0.12	0.17	**0.08**	0.29	0.39
no2	0.74	**0.60**	0.64	0.65	0.72	0.74
pm10	1.06	0.88	0.84	**0.81**	1.11	1.12
real-estate	0.46	**0.38**	0.41	**0.38**	0.59	0.60
slump-slump	0.83	0.58	0.75	**0.45**	0.69	0.76
slump-flow	0.63	0.51	0.63	**0.48**	0.67	0.68
slump-strength	0.01	**0.00**	0.38	0.49	0.12	0.10
wine-red	0.93	0.91	0.76	**0.75**	0.98	1.11
wine-white	0.97	0.94	**0.80**	0.83	1.04	1.23
yacht	0.01	**0.00**	0.19	0.05	0.05	0.03
average	0.57	0.48	0.54	0.49	0.61	0.65

While assessing complexity using $\mathcal{O}$-notations is of interest, it poses challenges for the MCMC-methods, which are used for FB-GPs, as shown in [28].

4 Results

4.1 Real-World Datasets

Tab. 3 shows the normalized-RMSE of the model after the final active learning iteration for the real-world datasets. For every ML-method the normalized-RMSE of the passive learning baseline (Random) and the best uncertainty estimator per dataset are shown. This is done to make the results more clear and to enable the comparison of the maximum potential of each ML-model for active learning. In the last row the average RMSE per method is shown for an overall comparison.

Table 4: Average RMSE per uncertainty estimator on real-world datasets (J - Jackknife estimator, IJ - Infinitesimal Jackknife estimator)

FB-GP			
aleatoric	epistemic	total	Random
0.57	0.51	0.49	0.57

RF				
J	unbiased J	IJ	unbiased IJ	Random
0.52	0.52	0.51	0.51	0.54

NN			
DropConnect	Dropout	LocalEnsemble	Random
0.74	0.73	0.69	0.61

The average score illustrates a comparable performance between active learning with Random Forests and FB-GPs, while active learning with Neural Networks performs worst. The RMSE per dataset also reinforces this observation, as for no dataset the Neural Networks perform best, whereas Random Forests performs best roughly on the same amount of datasets as FB-GPs. The performance of both methods is also very similar for every datasets, with two major exceptions: For the *concrete* and the *slump-strength* dataset the FB-GPs outperform the Random Forests severely. This could be a result of poor hyperparameter choice or because Random Forests are a bad model choice for those particular datasets. Additionally, the random baseline (passive learning) of Random Forests performs better in both cases, indicating sub-optimal uncertainty estimates unable to pinpoint regions benefiting from additional data samples. This trend of poor uncertainty estimates is further evident for Neural Networks, where, barring three datasets, the passive learning consistently outperforms the best active learning strategy. Because the papers introducing these methods show that they produce reliable uncertainty estimates, this suggests the inadequacy of the use of these uncertainty estimators for active learning. A potential explanation for this unsuitability could be the general poor generalization of the Neural Networks due to suboptimal hyperparameter selection. However, to proof the hypothesis that a poorly fitted model causes poor uncertainty estimates, dedicated experiments need to be conducted.

Tab. 4 shows the average scores per uncertainty estimator for the real-world datasets. This enables the comparison between the different uncertainty estimators of the specific ML-models. The top section emphasizes the advantage of distinguishing the uncertainty into its epistemic and aleatoric components, because active learning with FB-GPs based on the epistemic uncertainty demonstrates better results than the aleatoric-based approach. Interestingly, the total uncertainty criterion proves most effective for active learning, even though both aleatoric and epistemic uncertainties contribute equally.

Comparing random baselines indicates Random Forests as the optimal model choice for real-world datasets, consistent with findings by [10]. However, FB-GP uncertainty estimates exhibit a greater performance increase (compared to the passive learning baseline) than Random Forests. This indicates that active learning with Random Forests could benefit from a better uncertainty estimate that differentiates between epistemic and aleatoric uncertainty.

For Random Forests one can see that the unbiased versions of the Jackknife and Infinitesimal Jackknife do not achieve increased performance regarding active learning compared to the non-bias-corrected versions. This is an expected result as the bias correction is achieved by dividing the uncertainty estimate by a constant. For active learning, the data sample with the highest uncertainty is added to the training data, which does not change when all values are divided by a constant. The difference between the Jackknife and the Infinitesimal Jackknife is not significant on average.

The uncertainty estimator for Neural Networks are all, as already mentioned, not suitable for active learning without further research and adaptation. The average results show that Local Ensembles outperform Dropout and DropConnect, which have similar results, which is explainable due to their similar nature.

4.2 Synthetic Datasets

The outcomes obtained from the synthetic datasets are presented in Tab. 5. They showcase substantial distinctions from the results observed in real-world datasets. Notably, Random Forests and Neural Networks demonstrate similar

Table 5: normalized-RMSE after the final active learning iteration on synthetic datasets, averaged over 10 runs. The best method for each dataset is printed bold. The last row shows the average score over all datasets.

	FB-GP		**RF**		**NN**	
	Random	Best	Random	Best	Random	Best
Gramacy1d	0.07	**0.02**	0.05	0.05	0.08	0.09
Higdon	0.06	**0.04**	0.06	0.05	0.09	0.09
Gramacy2d	0.46	**0.02**	0.62	0.79	0.79	1.01
Branin	0.13	**0.08**	0.37	0.50	0.16	0.31
Ishigami	**0.42**	0.54	0.57	0.60	0.67	0.91
Friedman	0.04	**0.02**	0.33	0.60	0.22	0.28
Hartmann	0.58	**0.46**	0.86	0.81	1.16	0.86
average	0.25	0.17	0.41	0.49	0.45	0.51

performances, while active learning based on FB-GPs significantly outperforms both. FB-GPs exhibit superior performance for each dataset, solidifying their efficacy. Furthermore, the passive learning approach outperforms the active learning approaches for Random Forests and Neural Networks, indicating poor uncertainty estimates. Conversely, active learning with FB-GPs achieves superior results compared to passive learning for all datasets except the *Ishigami* dataset, which is characterized by strong non-linearities. This suggests that complex problems for which an ML-Model achieves poor generalization, may introduce uncertainty estimates that are not useful for active learning.

Analyzing the passive learning scores shows that, for the synthetic datasets, FB-GPs emerge as the optimal model choice. This deviates from the results observed with real-world datasets. This difference can be attributed to distinct characteristics of the synthetic datasets, notably the presence of homoscedastic noise and the continuous nature of their underlying functions. These features inherently favor FB-GPs, as they are able to model continuous output well due to their probabilistic distribution over functions. Additionally, FB-GPs employ a single noise parameter for the entire input space, predicated on the assumption of homoscedastic noise. While this assumption contributes to strong performance when homoscedastic noise prevails, it poses challenges in

Table 6: Average RMSE per uncertainty estimator on synthetic datasets

FB-GP				
aleatoric	epistemic	total	Random	
0.34	0.17	0.19	0.25	
RF				
J	unbiased J	IJ	unbiased IJ	Random
0.53	0.59	0.56	0.52	0.41
NN				
DropConnect	Dropout	LocalEnsemble	Random	
0.69	0.56	0.58	0.45	

effectively modeling heteroscedastic noise. However, it is essential to note that these favorable characteristics of synthetic datasets are not necessarily given for real-world datasets, requiring further experiments to validate the hypothesis that the performance of FB-GPs relies on the characteristics of the dataset.

The average results per uncertainty estimator (Tab. 6) strengthen the suitability of epistemic uncertainty as an active learning criterion over aleatoric uncertainty, with the latter performing worse than passive learning. In contrast to the real-world datasets, the total uncertainty does not result in a performance improvement. The uncertainty estimator performance for Random Forests and Neural Networks deviates from the results observed on real-world datasets. For Random Forests, the Infinitesimal Jackknife yields better results, with a slight advantage for the unbiased version, while the unbiased version of the Jackknife estimator outperforms the standard version. In the previous section, we argued that the unbiased and non-bias-corrected uncertainty estimates for Random Forests should theoretically yield similar results. However, due to the highly stochastic nature of Random Forests, caused by bootstrapping and random feature selection, the sample size of ten active learning runs might not suffice to ensure comparable results. This is further underscored by the observation that, for the Jackknife estimator, the unbiased version performs worse than the biased one, whereas for the Infinitesimal Jackknife estimator, the unbiased version fares better. If a significant difference in performance resulted from the bias correction, one would expect the difference to be consistent across both

Table 7: Average Time per uncertainty estimator in seconds

	FB-GP	RF	NN
Real-World	2851	6	137
Synthetic	1579	9	202

estimators, which is not the case. Regarding Neural Networks, DropConnect performs notably worse, while Dropout and Local Ensembles deliver roughly equivalent performance.

4.3 Computational Effort

As emphasized in the introduction, computational efficiency is as well an aspect as predictive performance when comparing various active learning methods. Tab. 7 outlines the time in seconds required to select a specific quantity of data samples, in this case fifty, averaged across all active learning runs for each respective ML-model. The results are quite apparent, with active learning using Random Forests proving to be the fastest. In contrast, Neural Networks are approximately 20 times slower, and FB-GPs are notably slower, ranging from approximately 100-500 times slower depending on the dataset. Although these results may vary with the amount of parallelization or optimized implementations, the trend of the computational effort for the different ML-methods is evident given the substantial differences observed.

The difference between the real-world and synthetic datasets is shown in Tab. 7, because it is notable with approximately 50%. For Random Forests and Neural Networks, the synthetic datasets are slower, whereas for FB-GPs, the real-world datasets display slower computation. This difference can be attributed to the pool size; the synthetic datasets comprise 2000 data samples, exceeding the size of most real-world datasets. The ML-model's predictions are important for uncertainty estimation, and processing becomes slower with a larger pool size. While this holds true for FB-GPs, their slower performance on real-world datasets is primarily due to the higher dimensionality of these datasets. The FB-GPs sample one lengthscale-parameter for each input dimension resulting in more parameters drawn by the overall time-consuming process of MCMC-sampling.

5 Conclusion and Future Work

Our work compares active learning strategies with three Machine Learning models and different uncertainty estimates for them. We apply a comprehensive set of datasets with real-world and synthetic problems to present individual strengths and weaknesses of the ML-models and their uncertainty estimates.

A key results is the superior reliability of uncertainty estimates provided by the FB-GPs. Their ability to differentiate aleatoric and epistemic uncertainty contributes to a high-quality active learning performance without the need for hyperparameter selection. The excellent results on different synthetically created problems, like noise-signal differentiation, makes them a good model choice when time and computational resources are not of the essence. Random Forests demonstrate performance comparable to FB-GPs across real-world datasets and present a significant speed advantage over them. Additionally, Random Forests are robust regarding the choice of hyperparameters, an important feature for active learning models due to the expensive or not feasible tuning process. This makes them the best model-choice for efficient active learning on real-world data. Conversely, Neural Networks are the least favorable ML-model, emphasizing the crucial role of hyperparameters that are challenging to select heuristically and ultimately impact the uncertainty estimation quality. Despite their relatively efficient processing, they fall behind Random Forests in terms of computational efficiency.

Moreover, our investigation underscores the importance of epistemic uncertainty for active learning. This motivates further research particularly for uncertainty estimates of Random Forests as there is to the best of our knowledge no method that differentiate epistemic and aleatoric uncertainty for regression-problems. Lastly, further research into the interplay between model characteristics and dataset features is motivated by the fact that FB-GPs excel significantly on synthetic data.

Acknowledgements

The author acknowledges financial support by the project “SAIL: SustAInable Life cycle of Intelligent Socio Technical Systems” (Grant ID NW21 059B), which is funded by the program “Netzwerke 2021” of the Ministry of Culture and Science of the State of Northrhine Westphalia, Germany.

References

[1] Tharwat, A. & Schenck, W. Balancing Exploration and Exploitation: A novel active learner for imbalanced data. *Knowledge-Based Systems*. 210 pp. 106500 (2020,12)

[2] Riis, C., Antunes, F., Boe, F., Carlos, H., Azevedo, L. & Pereira, F. Bayesian Active Learning with Fully Bayesian Gaussian Processes. *Advances In Neural Information Processing Systems*. pp. 12141-12153 (2022)

[3] Lampinen, J. & Vehtari, A. Bayesian approach for neural networks - review and case studies. *Neural Networks*. 14, 257-274 (2001)

[4] Gal, Y. & Others Uncertainty in deep learning. (PhD thesis, University of Cambridge,2016)

[5] Wu, D., Lin, C. & Huang, J. Active Learning for Regression Using Greedy Sampling. *Information Sciences*. 474 pp. 90-105 (2019)

[6] Tharwat, A. & Schenck, W. A Novel Low-Query-Budget Active Learner with Pseudo-Labels for Imbalanced Data. *Mathematics*. 10, 1068 (2022,3)

[7] Hüllermeier, E. & Waegeman, W. Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*. 110, 457-506 (2021,3)

[8] Nguyen, V., Destercke, S. & Hüllermeier, E. Epistemic Uncertainty Sampling. *Discovery Science: 22nd International Conference, DS 2019*. pp. 72-86 (2019,10)

[9] Breiman, L. Random Forests. *Machine Learning*. 45 pp. 5-32 (2001)

[10] Fernández-Delgado, M., Cernadas, E., Barro, S. & Amorim, D. Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?. *Journal Of Machine Learning Research*. 15 pp. 3133-3181 (2014)

[11] Wager, S., Hastie, T. & Efron, B. Confidence Intervals for Random Forests: The Jackknife and the Infinitesimal Jackknife. *Journal Of Machine Learning Research*. 15 pp. 1625-1651 (2014)

[12] Efron, B. Jackknife-after-bootstrap standard errors and influence functions. *Journal Of The Royal Statistical Society Series B: Statistical Methodology*. 54, 83-111 (1992)

[13] Efron, B. Estimation and Accuracy After Model Selection. *Journal Of The American Statistical Association*. 109, 991-1007 (2014,7)

[14] Shaker, M. & Hüllermeier, E. Aleatoric and Epistemic Uncertainty with Random Forests. *Advances In Intelligent Data Analysis XVIII*. pp. 444-456 (2020,4)

[15] Sharma, P. & Singh, A. Era of deep neural networks: A review. *2017 8th International Conference On Computing, Communication And Networking Technologies (ICCCNT)*. pp. 1-5 (2017,7)

[16] Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *31st Conference On Neural Information Processing Systems (NIPS 2017)*. (2017)

[17] Srivastava, N., Hinton, G., Krizhevsky, A. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal Of Machine Learning Research*. 15 pp. 1929-1958 (2014)

[18] Gal, Y. & Ghahramani, Z. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *33rd International Conference On Machine Learning, ICML 2016*. 3 pp. 1651-1660 (2015,6)

[19] Wan, L., Zeiler, M., Zhang, S., Lecun, Y. & Fergus, R. Regularization of Neural Networks using DropConnect. *International Conference On Machine Learning*. pp. 1058-1066 (2013)

[20] Mobiny, A., Yuan, P., Moulik, S., Garg, N., Wu, C. & Nguyen, H. DropConnect is effective in modeling uncertainty of Bayesian deep networks. *Scientific Reports*. 11, 5458 (2021,3)

[21] Madras, D., Atwood, J. & D'Amour, A. Detecting Extrapolation with Local Ensembles. *International Conference On Learning Representations*. (2019)

[22] Lanczos, C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. (United States Governm. Press Office Los Angeles, CA,1950)

[23] Surjanovic, S. & Bingham, D. Virtual Library of Simulation Experiments: Test Functions and Datasets. (Retrieved September 12, 2023, from http://www.sfu.ca/ ssurjano)

[24] Friedman, J. Multivariate adaptive regression splines. *The Annals Of Statistics*. 19, 1-67 (1991)

[25] Breiman, L. Bagging predictors. *Machine Learning*. 24 pp. 123-140 (1996)

[26] Kelly, M., Longjohn, R. & Nottingham, K. The UCI Machine Learning Repository. , https://archive.ics.uci.edu

[27] Tohme, T., Vanslette, K. & Youcef-Toumi, K. Reliable neural networks for regression uncertainty estimation. *Reliability Engineering & System Safety*. 229 pp. 108811 (2023,1)

[28] Matamoros, I. An introduction to computational complexity in Markov Chain Monte Carlo methods. *ArXiv Preprint ArXiv:2004.07083*. (2020,4)

AI-based automated active learning for discovery of hidden dynamic processes: A use case in light microscopy

Nils Friederich[1,2] Angelo Yamachui Sitcheu[1], Oliver Neumann[1], Süheyla Eroğlu-Kayıkçı[2], Roshan Prizak[2], Lennart Hilbert[2,3], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics
Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
[2] Institute of Biological and Chemical Systems
Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
[3] Zoological Institute
Karlsruhe Institute of Technology
Fritz-Haber-Weg 4
76131 Karlsruhe

Abstract

In the biomedical environment, experiments assessing dynamic processes are primarily performed by a human acquisition supervisor. Contemporary implementations of such experiments frequently aim to acquire a maximum number of relevant events from sometimes several hundred parallel, non-synchronous processes. Since in some high-throughput experiments, only one or a few instances of a given process can be observed simultaneously, a strategy for planning and executing an efficient acquisition paradigm is essential. To address this problem, we present two new methods in this paper. The first method, Encoded Dynamic Process (EDP), is Artificial Intelligence (AI)-based and represents dynamic processes so as to allow prediction of pseudo-time values from single still images. Second, with Experiment Automation Pipeline for Dynamic Processes (EAPDP), we present a Machine Learning Operations (MLOps)-based pipeline that uses the extracted knowledge from EDP to efficiently schedule acquisition in biomedical experiments for dynamic processes in practice. In a first experiment, we show that the pre-trained State-Of-The-Art (SOTA) object segmentation method Contour Proposal Networks (CPN) works reliably as a module of EAPDP to extract the relevant object for EDP from the acquired three-dimensional image stack.

1 Introduction

For the imaging-based assessment of dynamic processes in biomedical settings, objects of interest must be identified and relevant events that characterize the dynamic process must be recorded during their time of occurrence. Commonly, a human operator controls the imaging instrument to examine a biomedical sample using a microscope and relevant objects are found in the sample by inspection. Alternatively, the operator estimates for each object of interest the time at which an event of interest is expected to occur based on previous experience and triggers the recording of the event at that time. Nevertheless, many contemporary experiments provide several hundred relevant objects that can, in principle, be imaged in parallel. Events of interest, however, are non-synchronous and the estimation of future event times requires extensive human effort, is prone to error, and not necessarily time-efficient. These obstacles can result in unnecessarily large amounts of irrelevant data, unnecessary experimental repeats, or experimental biases inflicted by additional light exposure of the sample [18]. To address these obstacles, we present two new methods for the automated, real-time planning and execution of such experiments.

EDP. The traditional method of capturing all data or relying on human experience to predict future events is outdated and inefficient. Instead of relying on human experience, an accurate and comprehensive model of the dynamic process should be created. This model should be capable of uniquely identifying a relevant object state through a process known as fingerprinting, similar to how humans do it. In biomedicine, it is crucial that this fingerprint remains consistent despite contextual changes such as noise, brightness changes, or affine transformations. By modeling the relationship between the fingerprints, the dynamic process can be represented orderly. This representation suits as an approximation of relative progress within the dynamic process. This relative progress can also be interpreted as a relative time, known as pseudo-time. By adopting this method, we can achieve efficiency in predicting future events and a deeper understanding of the dynamic process.

EAPDP To unlock the full potential of the EDP, a well-designed pipeline is an absolute must. Such a pipeline should be able to recognize specific states in the real world, identify relevant objects, and then calculate a pseudo-time for those states using the EDP. Armed with this knowledge, the pipeline can automatically plan and execute a new state capture for any significant event that occurs. Due to the uncertain nature of the EDP predictions, the pipeline must be able to respond to unsuccessful recordings and learn from them. This is where MLOps [1] comes in. By retraining an existing production model in accordance with the live context, MLOps ensures that the pipeline always uses a current and accurate model, resulting in a potentially better outcome in real-world experiments.

2 Related Work

Object Extraction. Basically, there are different possibilities in Computer Vision (CV) like object detection and object segmentation, to identify individual objects in an image [10] and thus extract them. In the biomedical context, many methods mostly focus on segmentation [42] with SOTA methods like StarDist [38] and CPN [46].

Pseudo-time predictions. A first approach for pseudo-time predictions with classical, non Deep Learning (DL) methods was presented in [14]. For extracting relevant objects from the acquired image, thresholding is used as a classical CV segmentation method. Then the object's fingerprint is generated linearly with a Principal Component Analysis (PCA) [19, 20]. However, biological processes are usually not linear [5]. Therefore, recently, non-linear encodings of the dynamic processes using DL methods have become popular [17, 9, 21, 32]. For example, [21] and [9] present DL approaches to encode cell cycles and derive predefined cell phases. However, this classification-based approach does not allow for deriving continuous relations like the pseudo-time to each other directly. This continuous relation was modeled with DeepCycle in [32]. The training of DeepCycle is performed supervised. For this purpose, virtual labels are calculated based on the fluorescence intensity in specifically labeled channels of the cells. These classes can then be used as anchor points during training to determine a (relative) cell state as a pseudotime. It is important to note that the assumption that a correlation of fluorescence intensity to cell phase can be used is not always true in the biological context. A DL method that follows a comparable pseudo-time approach to the given constraints, in this paper, was presented in [17]. In [17], an autoencoder (AE) approach for pseudo-time approximation is used as a Self-Supervised Learning (SSL) approach. A DL model is used as the Variational Autoencoder (VAE) [23] encoding, from whose Hierarchical Agglomerative Clustering (HAC) and Minimum Spanning Tree (MST) code the pseudo-time is then determined. However, this approach also has a few limitations. First, a recording necessarily contains exactly one relevant object in one acquired image. Second, the entire dataset was acquired under comparable acquisition conditions, which also only contain identical positioned and oriented objects and are not able to learn affine transformations [3] between objects. Both constraints are generally not satisfied for microscopic images, such as in [27, 36]. Furthermore, this pseudo-time method was not designed as an End-to-end (E2E) model, which deprives the DL model of the ability to internally bind affine transformed objects.

AutoEncoder. Autoencoders are SSL methods to learn a representation from a given suitability, such as an image [49, 47]. For example, the autoencoder can be represented by a Conventional AutoEncoder (CAE) [49] and/or a VAE [23]. Especially recently, masked autoencoders (MAEs) [15] have become more popular than CAE because of their ability for a better visual representation learning [49], either using a transformer-based approach [15] or the Convolutional Neural Network (CNN)-based approach [47]. However, since in [47] the higher efficiency of ConvNeXt V2 is shown, this model is chosen for this work. In addition to the pure learning of a visual representation in the form of a fingerprint, relations between the fingerprints can also be learned, e.g. with VAE [17].

SSL To train the DL model in a supervised manner, labeled data are generally rare in the biomedical domain [13]. There are DataBases (DBs) like BioMedImage.io [30] or several challenges with own datasets [26, 2, 28]. However, especially for biological datasets with sometimes hundreds of relevant objects in an image, the datasets are often limited to the 2D case. Furthermore, in the context of this work, labeling the relevant events with pseudo-time stamps is only approximate, demanding, error-prone and time-consuming. Therefore, unsupervised or SSL methods are often used in the biomedical context [43]. Thereby, Active Learning (AL) [41] is used to selectively integrate expert knowledge into the learning process. Since AL aims to keep the number of interactions to a minimum [7, 33], data-efficient learning is preferable. For example, existing datasets from a related context can be identified and leveraged to train more robust models in transfer learning [11, 29, 48]. In order to be able to use possibly directly existing pre-trained models from similar contexts, a new concept was developed in [48].

3 Methodology

3.1 EDP

For the modeling of the dynamic process, a new concept is introduced with EDP. The new concept of EDP is based on an AE and is visualized in Figure 1. The basic idea of the new concept is to separate the generation of the fingerprint from the learning of the relation as a representation between all states. The fingerprint generation is done using a MAE as an evolution of CAE.

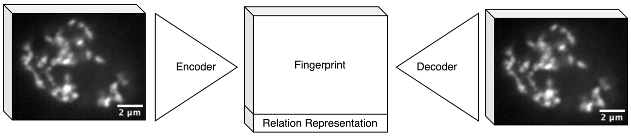


Figure 1: Visualization of the EDP model. A 3D object is transformed by an encoder into a fingerprint (like MAE) and into a relation representation between the fingerprints (like VAE). The example object is a recording of the DNA channel of a nucleus from an internal zebrafish embryo dataset. A scale bar of $2\mu m$ is indicated at the bottom of the input/output image.

Specifically, the SOTA MAE-based method ConvNeXt V2 is chosen. During the learning process, a maximum recovery of the encoded image is aimed in accordance with SSL. According to the challenge posed by biological objects, a context-independent representation is required. For this purpose, the images can be modified during the learning process through Data Augmentation [16] techniques like Rotations, reflections, contrast adjustments or noise additions. In addition to the fingerprint, the relation must also be learned as the actual modeling of the dynamic process. For this purpose, a VAE-like modeling is used by learning the uncertainty v in addition to the circle angle α. The assumption is that objects succeeded each other in the dynamic process with the relative distance t corresponding to this relative distance and differ in the same ratio in the circle representation. Such an exemplary circle representation is shown in Figure 2 using a cell division process of the zebrafish embryo. The state of the cell after cell division is visualized at 00 o'clock and up to the state of the cell just before cell division at 11 o'clock. This corresponds to a relative distance of $\sim$0.92 (normalized between [0,1)). This temporal difference must also be valid in reality for the temporal distance according to the model statement.

3.2 EAPDP

The new EDP module is integrated as a module into the new MLOps-based pipeline EAPDP. The pipeline concept is visualized in Figure 3 and contains nine other modules besides the EDP module. Each of these ten modules is

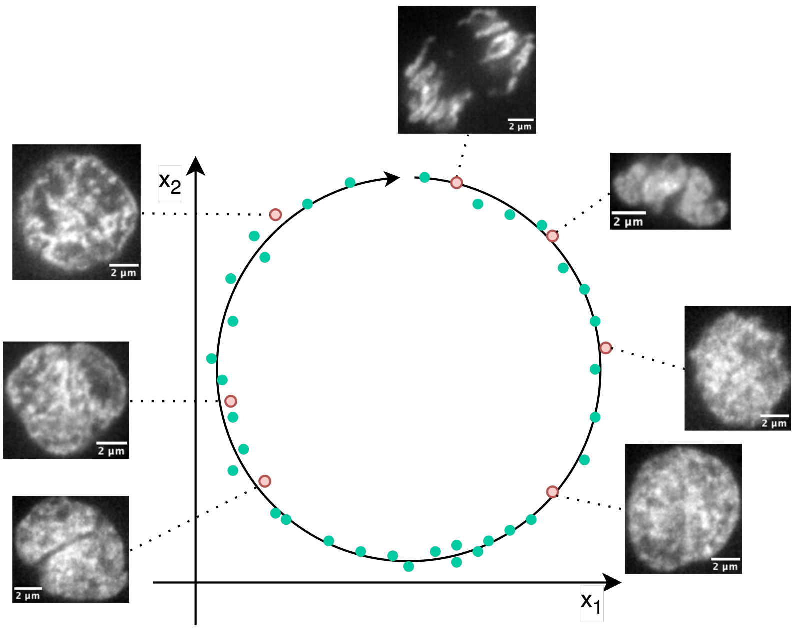


Figure 2: Example of a 2D feature space representation for an encoded dynamic process. Each point represents an encoded image. The circle serves as an estimation for the positioning of points in its vicinity. It's worth noting that due to the presence of uncertainty, the points may not always be precisely on the circle, but rather in its proximity. For the seven red dots, example images of cell nuclei from zebrafish embryos at various stages of cell division are shown. A scale bar of 2 μm is indicated at the bottom of each example image. The images are from an internal dataset.

briefly described below. The explanation of the modules and their relationships to each other is based on the pipeline visualization of Figure 3.

Microscope setup. In the EAPDP, the microscope is used as an actuator to the real-world environment represented by a biomedical sample. For this purpose, all microscope components relevant to image acquisition and the microscope accessories, such as lasers in the case of a laser scanning microscope, must be controllable via appropriate interfaces of the specific microscope setup. In addition, the microscope must be able to react on given com-

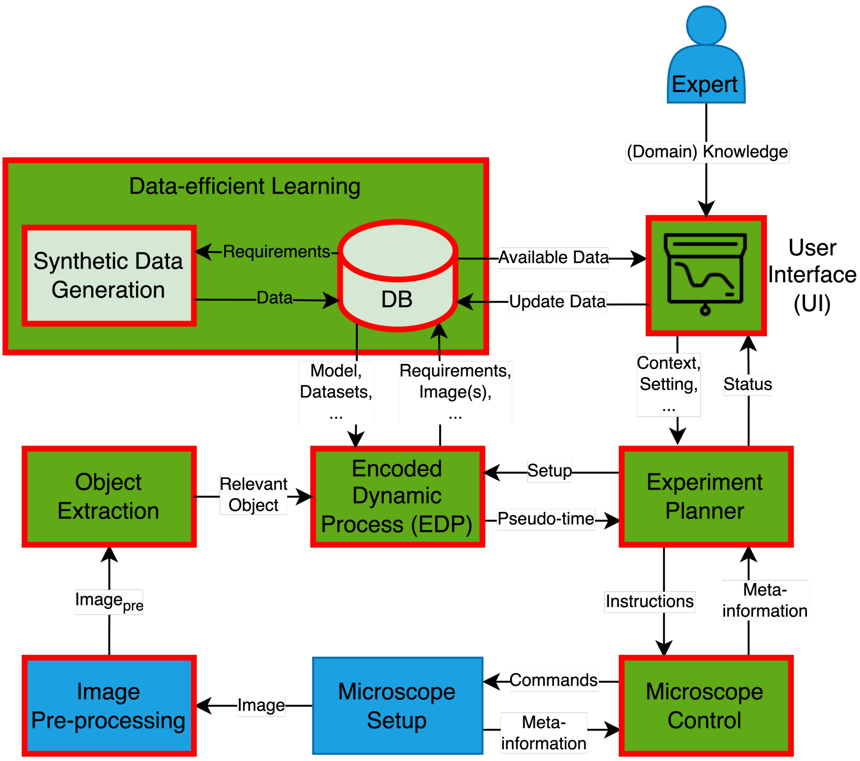


Figure 3: MLOps pipeline with the new EDP module. AI-based MLOps modules are marked with a green background, non AI-based ones with a blue background. Additionally, all modules marked with a red border must be newly developed or only partially adapted from existing methods.

mands like image acquisition or requested meta-information like the objective position in a standardized way.

Image Pre-processing. To optimize the analysis of dynamic processes in the biomedical environment, the raw images acquired through experiments must be pre-processed according to the microscope setup and the context of the targeted event. This may involve methods such as cropping, contrast adjustment, or denoising. Various libraries, such as Albumentations [4], offer pre-processing methods that can be used to improve the quality of the images and optimize their analysis by other modules in the machine learning operation pipeline.

Object Extraction. During an acquisition, the relevant object and the surrounding context are captured. In order to better analyze the object, it is necessary to separate it from the surrounding context. The extraction from the whole image is done via pre-trained segmentation methods. To find a suitable method, we compare the SOTA cell segmentation algorithms StarDist and CPN using a microscopic dataset in a first experiment. Importantly, the actual pseudo-time determination cannot be performed if both methods' segmentation is insufficient. Therefore, this submodule is of particular importance. Because (well) labeled data are generally scarce in the biological context, this work evaluates generalization performance during inference with already pre-trained models on new, unknown images. Since there are only pre-trained weights for 2D segmentation for both methods, the dataset was split into 2D images along the z-axis.

EDP. The EDP module gets the extracted object and should pass the pseudo-time to the experiment planner. In order to do this, the module is equipped by the Experiment planner before with the appropriate experiment setup. With the setup, the EDP model can then query according to its existing knowledge like pre-trained models in the context of data-efficient learning. If no weights are available, training can also be done with/without AL as specified by the expert. After successful training, the model is passed to the DB with the appropriate required metadata for possible further use. Then, when the inference with the original extracted relevant object has been determined, the results are passed to the expert planner accordingly. The recorded inference image is also sent to the Data-efficient Learning module and stored in its DB.

Experiment planner. The Experiment planner is the central module of the experiment automatization. As input it gets the pseudo-times for the recorded objects. Based on the experiment context, including interesting events, the Experiment planner can plan future experiments with utmost precision. Once the plan is set, the Experiment planner gives the microscope the command to ensure that the image captures the object's state at the right time, leaving no room for errors. Additionally, it can query the state of the microscope to ensure that there was no hardware drift, such as when moving to the object position. All the information about the experiment's state is then passed on to the User Interface (UI), ensuring that all aspects of the experiment are under control.

UI. The UI is the interface between the expert and the MLOps pipeline. On the one hand, simple interactions can be provided, such as displaying meta-information or adapting the experimental context, e.g. the cell classes that occur. On the other hand, much more complex interactions such as result justifications of DL models can be represented through Explainable Artificial Intelligence (XAI) [34] or expert knowledge can be brought into the pipeline within the context of AL. With XAI, the expert should be able to understand better the processes in the DL models used and why decisions were made, e.g. for event detection. This helps the expert to eliminate potential errors like unfavorable experiment settings at an early stage. Such XAI methods can be realized using a library like PyTorch Captum [25]. For AL, only if the expert can capture the actual state in the best possible way, the expert can transfer his domain knowledge to the method in the best possible way and support the method. For example, points, boxes, or entire segmented regions can be passed to the method as hints. For this purpose, a custom segmentation module can be developed based on exiting AL labeling platforms like ObiWanMicrobi [40] or Karlsruhe Image Data Annotation (KaIDA) [37].

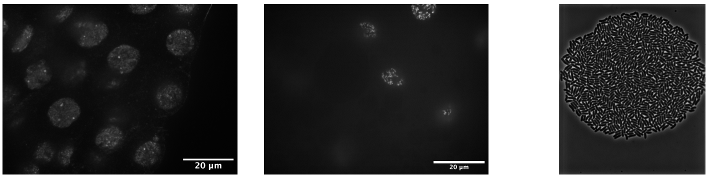
Expert (Domain) Knowledge. The domain knowledge contributed by the expert to the MLOps pipeline can take several forms. For example, the context of the experiment with a specific cell class can improve a more efficient event detection module. Furthermore, knowledge can be injected, e.g. by labeling in the context of AL. For this, the expert must ensure the quality of the injected domain knowledge with maximum correctness. Incorrect information can affect the learning processes in the network.

Data-efficient Learning. To minimize AL interactions with the human expert, as much existing knowledge as possible is reused. To this end, building on [48], a new AE-based fingerprinting approach for datasets and Machine Learning (ML) models is being implemented to reuse as much knowledge as possible. For this, from a DB, context-given requirements can query existing knowledge. If no data is available, it can be created synthetically e.g. with biophysical simulations like Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) [44].

Microscope control. In order for the planned experiment to be automated and performed in real-time, a corresponding software library is needed to control the microscope. Since the first release in 2010, μManager has been used for this purpose as one of the SOTA open-source solutions [6, 22, 31, 45]. Therefore, this is also used in this work.

3.3 Exemplary Use Cases

Example use cases for the presented EAPDP with the EDP are presented using record extracts in Figure 4 below. A first use case is shown in Figure 4a and represents the temporal sorting of RiboNucleic Acid (RNA) Polymerase II (Pol II) clusters that occur in the nuclei of pluripotent zebrafish embryos. A method for this use case has already been presented in [14]. A comparison of the pipeline based on classical ML methods with our DL-based EDP method allows a direct statement about limitations or improvements of our approach. The second use-case in Figure 4b is the recording of cell divisions in pluripotent zebrafish embryos, where the time of reaching a new division stage and thus the regions of an event of interest need to be extrapolated. A final biological application from the field of microbiology is presented in Figure 4c. In this example, one interesting event could be the state at which n microbes reach the recording region. For this purpose, a modeling of the cell division process with EDP can be used to plan the experiment accordingly and automatically record the event of interest at a time t. The modeling of the cell division process with EDP can be used for this purpose.



(a) [27] (b) *Internal dataset* (c) [39]

Figure 4: Three exemplary images from biological datasets for dynamic processes. Figure 4a shows cell nuclei of zebrafish embryos with marked Pol II Ser5P clusters. Figure 4b shows the DeoxyriboNucleic Acid (DNA) of zebrafish embryos nuclei. In these first two images, a scale bar of $20\mu m$ is shown in the lower left. The last Figure 4c shows a microbial cell division state.

In addition to these biological use cases, other use cases are also possible, e.g. in medicine. For example, by modeling a tumor accordingly, a prediction can be made about the relative stage. Consequently, a therapy concept such as surgery or medication can be tailored to the patient.

4 Experiments

The comparison of segmentation algorithms was performed on Helmholtz AI COmpute REssources (HAICORE) resources equipped with Intel Xeon Platinum 8368 Central Processing Units (CPUs) and an Nvidia A100-40 Graphics Processing Unit (GPU) [24]. The operating system utilized was Red Hat Enterprise Linux (RHEL) version 8.6.

4.1 Dataset

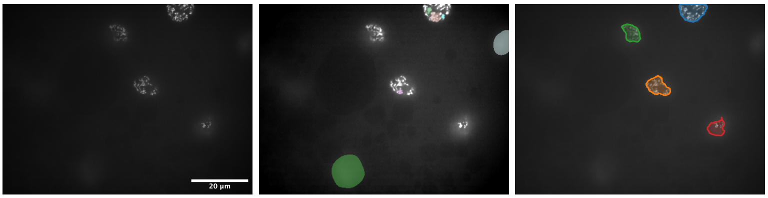
The internal microscope dataset from Figure 4b is used to compare the segmentation algorithms. This dataset was chosen over the other two example datasets from Figures 4a and 4c because of the challenging, frayed structure of the nuclei as the relevant image object. This is because the fibrillar structure of the nuclei sometimes deviates strongly from their typical ellipsoidal shape as in Figure 4a due to individually advanced cytokinesis. This poses a challenge because contiguous pixel regions are not trivially identifiable and correct boundary segmentation is a challenge. With microbeSEG [35], a working SOTA solution for microbes like in Figure 4c also already exists.

For the dataset in Figure 4b, zebrafish embryo DNA was imaged. DNA was stained with 1:10000 5'-TMR Hoechst in TDE or glycerol. Confocal z-sections were obtained using a commercial instant SIM microscope (iSIM, VisiTech). A Nikon 100x oil immersion objective (NA 1.49, SR HO Apo TIRF 100xAC Oil) and a Hamamatsu ORCA-Quest camera were used for image acquisition. In accordance with a common problem in biology, no labels exist for this dataset. According to the desired 2D segmentation, the 3D images are split into 2D images along the z-axis.

4.2 Object extraction

These 2D images were then segmented using each of the two methods. In the following, the results are evaluated qualitatively because of the non-existent labels. Therefore, the results are shown in Figure 5. The comparison of the original image in Figure 5a with the StarDist prediction in Figure 5b, shows that StarDist cannot well segment semantically related objects as the nucleus in the upper area. For the method designs with center prediction, StarDist focused primarily on segmenting ellipsoidal objects from [12] and was trained only on these. The cell detection was designed to be more flexible and additionally trained on a more heterogeneous set of non-elliptical cells such as *MCF7* from the dataset [8]. This leads to better generalization and results in qualitatively evaluated good initial segmentation performance on this most challenging of our datasets from Figure 4.

Thus, we could show that CPN is a good pre-trained SOTA approach for extracting the relevant objects from the 2D decomposition. The 2D segmentations can be reassembled back to 3D segmentations in post-processing, e.g. using Nearest Neighbor. Based on this, the further submodules of EDP can be developed in future work and the presented MLOps pipeline can be built upon it.



(a) Original (b) StarDist [38] (c) CPN [46]

Figure 5: Comparison of segmentation predictions for the two SOTA methods StarDist [38] and CPN [46]. Figure 5a represents the original image, duplicated from Figure 4b. In Figure 5b and Figure 5c, the predictions of pre-trained StarDist or CPN are then shown. The predictions are highlighted differently for better visual differentiation depending on the method used.

5 Conclusion and Further Work

In this work, we motivated that due to the large number of parallel non-synchronous dynamic processes, a novel concept for automated planning and execution of two novel DL-based approaches is essential. First, the EDP was introduced to model dynamic processes and derivate a pseudo-time for a given object state. The pseudo-time prediction can then be used with the EAPDP for real-time experiment automation. We explained the EDP realized within the MLOps pipeline by an AE and trained using SSL with AL. At the same time, the key advantage of higher execution speed and lower human cost while minimizing user interactions with data-efficient learning was highlighted. Finally, as a first Proof of Concept (PoC), we showed the necessary pre-processing step for the EDP to extract the relevant objects based on good inference results of CPN.

However, the lack of pre-trained weights for 3D segmentation was a drawback of the segmentation experiments. However, since the fragmented objects are partially reconnected along the z-axis, this could simplify the problem and improve accuracy. This will be done as soon as appropriate weights are available. In addition, a suitable affine-invariant 3D AE needs to be developed for use within the EDP method. In this context, further research is needed to investigate whether the ConvNeXt V2 is suitable for 3D segmentation, also from an efficiency perspective. Of course, the modules of the MLOps pipeline must be implemented accordingly.

Acknowledgments

This project is funded by the Helmholtz Association under the Program Natural, Artificial and Cognitive Information Processing (NACIP) and the Helmholtz Association's Initiative & Networking Fund through Helmholtz AI. All experiments were performed on the HAICORE. We sincerely thank all of them for supporting our research.

The authors have accepted responsibility for the entire content of this manuscript and approved its submission. We describe the individual contributions of N. Friederich (NF), A. Yamachui Sitcheu (AYS), O. Neumann (ON), S. Eroglu-Kayıkçı (SEK), R. Prizak (RP), L. Hilbert (LH), R. Mikut (RM): Conceptualization: NF, LH, RM; Methodology: NF, LH, RM; Software: NF; Investigation: NF; Resources: SEK, RP, LH; Writing – Original Draft: NF; Writing – Review & Editing: NF, AYS, ON, SEK, RP, LH, RM; Supervision: LH, RM; Project administration: RM; Funding Acquisition: LH, RM

References

[1] S. Alla, S. K. Adari, S. Alla, and S. K. Adari. What is MLOps? *Beginning MLOps with MLFlow: Deploy Models in AWS SageMaker, Google Cloud, and Microsoft Azure*, pages 79–124, 2021.

[2] M. Aubreville, C. Bertram, K. Breininger, S. Jabari, N. Stathonikos, and M. Veta. MItosis DOmain Generalization Challenge 2022. `https://doi.org/10.5281/zenodo.6362337`, 03 2022.

[3] E. Begelfor and M. Werman. Affine Invariance Revisited. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2087–2094, 2006.

[4] A. V. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin. Albumentations: fast and flexible image augmentations. *CoRR*, abs/1809.06839, 2018.

[5] J. Carballido-Landeira and B. Escribano. *Nonlinear dynamics in biological systems*. Springer, 2016.

[6] C. Conrad, A. Wünsche, T. H. Tan, J. Bulkescher, F. Sieckmann, F. Verissimo, A. Edelstein, T. Walter, U. Liebel, R. Pepperkok, et al. Micropilot: automation of fluorescence microscopy–based imaging for systems biology. *Nature Methods*, 8(3):246–249, 2011.

[7] A. M. Das, G. Bhatt, M. M. Bhalerao, V. R. Gao, R. Yang, and J. Bilmes. Continual Active Learning. `https://openreview.net/forum?id=GC5MsCxrU-`, 2023.

[8] C. Edlund, T. R. Jackson, N. Khalid, N. Bevan, T. Dale, A. Dengel, S. Ahmed, J. Trygg, and R. Sjögren. LIVECell—A large-scale dataset for label-free live cell segmentation. *Nature Methods*, 18(9):1038–1045, 2021.

[9] P. Eulenberg, N. Köhler, T. Blasi, A. Filby, A. E. Carpenter, P. Rees, F. J. Theis, and F. A. Wolf. Reconstructing cell cycle and disease progression using deep learning. *Nature Communications*, 8(1):463, 2017.

[10] X. Feng, Y. Jiang, X. Yang, M. Du, and X. Li. Computer vision algorithms and hardware implementations: A survey. *Integration*, 69:309–320, 2019.

[11] P. Godau and L. Maier-Hein. Task Fingerprinting for Meta Learning inBiomedical Image Analysis. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part IV 24*, pages 436–446. Springer, 2021.

[12] A. Goodman, A. Carpenter, E. Park, jlefman nvidia, Josette-BoozAllen, Kyle, Maggie, Nilofer, P. Sedivec, and W. Cukierski. 2018 Data Science Bowl. `https://kaggle.com/competitions/data-science-bowl-2018`, 2018.

[13] H. A. Gündüz, M. Binder, X.-Y. To, R. Mreches, B. Bischl, A. C. McHardy, P. C. Münch, and M. Rezaei. A self-supervised deep learning method for data-efficient training in genomics. *Communications Biology*, 6(1):928, 2023.

[14] H. Hajiabadi, I. Mamontova, R. Prizak, A. Pancholi, A. Koziolek, and L. Hilbert. Deep-learning microscopy image reconstruction with quality control reveals second-scale rearrangements in RNA polymerase II clusters. *PNAS nexus*, 1(3):pgac065, 2022.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[16] A. Hernandez-Garca and P. König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.

[17] J. Hong, S. K. Kang, I. Alberts, J. Lu, R. Sznitman, J. S. Lee, A. Rominger, H. Choi, K. Shi, and A. D. N. Initiative. Image-level trajectory inference of tau pathology using variational autoencoder for Flortaucipir PET. *European Journal of Nuclear Medicine and Molecular Imaging*, 49(9):3061–3072, 2022.

[18] J. Icha, M. Weber, J. C. Waters, and C. Norden. Phototoxicity in live fluorescence microscopy, and how to avoid it. *BioEssays*, 39(8):1700003, 2017.

[19] J. E. Jackson. *A user's guide to principal components*. John Wiley & Sons, 2005.

[20] I. T. Jolliffe. *Principal component analysis for special types of data*. Springer, 2002.

[21] A. Jose, R. Roy, D. Moreno-Andrés, and J. Stegmaier. Automatic Detection of Cell-cycle Stages using Recurrent Neural Networks. *bioRxiv*, pages 2023–02, 2023.

[22] R. A. Kellogg, R. Gómez-Sjöberg, A. A. Leyrat, and S. Tay. High-throughput microfluidic single-cell analysis pipeline for studies of signaling dynamics. *Nature Protocols*, 9(7):1713–1726, 2014.

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[24] KIT. HAICORE - Hardware Overview. `https://www.nhr.kit.edu/userdocs/haicore/hardware/`, Sep 2022.

[25] N. Kokhlikyan, V. Miglani, M. Martin, E. Wang, B. Alsallakh, J. Reynolds, A. Melnikov, N. Kliushkina, C. Araya, S. Yan, and O. Reblitz-Richardson. Captum: A unified and generic model interpretability library for PyTorch. *CoRR*, abs/2009.07896, 2020.

[26] K. Lee, H. Byun, and H. Shim. Cell Segmentation in Multi-modality High-Resolution Microscopy Images with Cellpose. In *Proceedings of The Cell Segmentation Challenge in Multi-modality High-Resolution Microscopy Images*, volume 212 of *Proceedings of Machine Learning Research*, pages 1–11. PMLR, 28 Nov–09 Dec 2023.

[27] I. Mamontova, A. Pancholi, R. Prizak, and L. Hilbert. Microscopy data: interaction of the gene klf2b with RNA polymerase II clusters during early zebrafish embryo development. `https://doi.org/10.5281/zenodo.5268833`, 08 2021.

[28] M. Maška, V. Ulman, P. Delgado-Rodriguez, E. Gómez-de Mariscal, T. Nečasová, F. A. Guerrero Peña, T. I. Ren, E. M. Meyerowitz, T. Scherr, K. Löffler, et al. The Cell Tracking Challenge: 10 years of objective benchmarking. *Nature Methods*, pages 1–11, 2023.

[29] M. Molina-Moreno, M. P. Schilling, M. Reischl, and R. Mikut. Automated Style-Aware Selection of Annotated Pre-Training Databases in Biomedical Imaging. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2023.

[30] W. Ouyang, F. Beuttenmueller, E. Gómez-de Mariscal, C. Pape, T. Burke, C. Garcia-López-de Haro, C. Russell, L. Moya-Sans, C. de-la Torre-Gutiérrez, D. Schmidt, et al. BioImage Model Zoo: A

Community-Driven Resource for Accessible Deep Learning in BioImage Analysis. *bioRxiv*, pages 2022–06, 2022.

[31] H. Pinkard, N. Stuurman, K. Corbin, R. Vale, and M. F. Krummel. Micro-Magellan: open-source, sample-adaptive, acquisition software for optical microscopy. *Nature Methods*, 13(10):807–809, 2016.

[32] L. Rappez, A. Rakhlin, A. Rigopoulos, S. Nikolenko, and T. Alexandrov. DeepCycle reconstructs a cyclic cell cycle trajectory from unsegmented cell images using convolutional neural networks. *Molecular Systems Biology*, 16(10):e9474, 2020.

[33] P. Ren, Y. Xiao, X. Chang, P.-Y. Huang, Z. Li, B. B. Gupta, X. Chen, and X. Wang. A Survey of Deep Active Learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

[34] W. Saeed and C. Omlin. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263:110273, 2023.

[35] T. Scherr, J. Seiffarth, B. Wollenhaupt, O. Neumann, M. P. Schilling, D. Kohlheyer, H. Scharr, K. Nöh, and R. Mikut. microbeSEG: A deep learning software tool with OMERO data management for efficient and accurate cell segmentation. *PLOS ONE*, 17(11):1–14, 11 2022.

[36] T. Scherr, J. Seiffarth, B. Wollenhaupt, O. Neumann, M. P. Schilling, D. Kohlheyer, H. Scharr, K. Nöh, and R. Mikut. microbeSEG dataset. `https://doi.org/10.5281/zenodo.6497715`, 04 2022.

[37] M. P. Schilling, S. Schmelzer, L. Klinger, and M. Reischl. KaIDA: a modular tool for assisting image annotation in deep learning. *Journal of Integrative Bioinformatics*, 19(4):20220018, 2022.

[38] U. Schmidt, M. Weigert, C. Broaddus, and G. Myers. Cell Detection with Star-Convex Polygons. In *Medical Image Computing and Computer Assisted Intervention - MICCAI 2018 - 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part II*, pages 265–273, 2018.

[39] J. Seiffarth, L. Blöbaum, K. Löffler, T. Scherr, A. Grünberger, H. Scharr, R. Mikut, and K. Nöh. Data for - Tracking one in a million: Performance of automated tracking on a large-scale microbial data set. `https://doi.org/10.5281/zenodo.7260137`, 10 2022.

[40] J. Seiffarth, T. Scherr, B. Wollenhaupt, O. Neumann, H. Scharr, D. Kohlheyer, R. Mikut, and K. Nöh. ObiWan-Microbi: OMERO-based integrated workflow for annotating microbes in the cloud. *bioRxiv*, pages 2022–08, 2022.

[41] B. Settles. *Active Learning Literature Survey*. University of Wisconsin-Madison Department of Computer Sciences, 2009.

[42] P. Shrestha, N. Kuang, and J. Yu. Efficient end-to-end learning for cell segmentation with machine generated weak annotations. *Communications Biology*, 6(1):232, 2023.

[43] R. Sun, J. Wu, Y. Miao, L. Ouyang, and L. Qu. Progressive 3D biomedical image registration network based on deep self-calibration. *Frontiers in Neuroinformatics*, 16:932879, 2022.

[44] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in ’t Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton. LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Comp. Phys. Comm.*, 271:108171, 2022.

[45] S. Tosi, A. Lladó, L. Bardia, E. Rebollo, A. Godo, P. Stockinger, and J. Colombelli. AutoScanJ: A Suite of ImageJ Scripts for Intelligent Microscopy. *Frontiers in Bioinformatics*, 1:627626, 2021.

[46] E. Upschulte, S. Harmeling, K. Amunts, and T. Dickscheid. Contour proposal networks for biomedical instance segmentation. *Medical Image Analysis*, 77:102371, 2022.

[47] S. Woo, S. Debnath, R. Hu, X. Chen, Z. Liu, I. S. Kweon, and S. Xie. ConvNeXt V2: Co-designing and Scaling ConvNets with

Masked Autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16133–16142, 2023.

[48] A. Yamachui Sitcheu, N. Friederich, S. Baeuerle, O. Neumann, M. Reischl, and R. Mikut. MLOps for Scarce Image Data: A Use Case in Microscopic Image Analysis. In *Proceedings 33. Workshop Computational Intelligence*, volume 1, 11 2023. Submitted manuscript.

[49] C. Zhang, C. Zhang, J. Song, J. S. K. Yi, K. Zhang, and I. S. Kweon. A survey on masked autoencoder for self-supervised learning in vision and beyond. *arXiv e-prints*, pages arXiv–2208, 2022.

Machine learning based model fitting concept for energy system components in energy management

Martin Winkelkotte[1], Steffi Naumann[2], Sebastian Flemming[2], Peter Bretschneider[1]

[1]Technische Universität Ilmenau
Gustav-Kirchhoff-Str. 5, 98693 Ilmenau
E-Mail: martin.winkelkotte@tu-ilmenau.de
E-Mail: peter.bretschneider@tu-ilmenau.de

[2]Fraunhofer IOSB-AST
Am Vogelherd 90, 98693 Ilmenau
E-Mail: steffi.naumann@iosb-ast.fraunhofer.de
E-Mail: sebastian.flemming@iosb-ast.fraunhofer.de

1 Introduction

The heterogenous nature of energy systems is a big challenge for the large-scale implementation of energy management systems (EMS). The different energy systems can be small energy communities, factories, or large-scale supply areas. A consistent challenge throughout the various energy system forms is the accurate modelling of the generation and storage units as well as the prediction of consumer behaviour, all what is needed to optimize which energy source to use. While significant work has been put into the automated prediction of consumption timeseries, the modelling of generation and storage units is still mostly done “by hand” and is reliant on external information sources like datasheets. Models created using this data basis are extremely vulnerable to discrepancies between the external information they are based on and the actual unit properties. Such inaccuracies often go unnoticed in the initial model generation, until they cause large errors in the optimization which then require time consuming analysis and finetuning to correct. In this paper an initial approach to automate the modelling of generation and storage units is presented. After that the approach is tested on its sensitivity to different training data sets since a dependency on high quality training data would severely limit its transferability.

2 Concept

The idea of the approach is to create a library of simplified generic models for the most common types of generation and storage technologies. These models can then be fitted to match the properties of the target units using key parameters whose values are identified by an artificial neural network (ANN). For this purpose, the ANN is trained on timeseries data that has been created using numerus simulated units with different properties of one unit type. The defining properties of the simulated units have been distributed to reflect the entire feasible parameter range. The trained ANN is then used on measured data of the target unit to identify the relevant parameter values to modify the generic model. The resulting fitted model should more accurately reflect the properties and performance limits of the target than a model based on generic datasheet information. Additionally, it would be easy to implement a periodic refitting of the model on the most recently measured data. This would allow the model to reflect changes due to aging and could even be used as an indicator for maintenance scheduling.

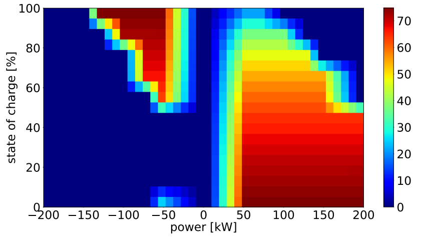
3 Generation of training data

The performance of ANN is dependent on the relation between the data used for its training and the target. Since the goal is the recognition of the physical parameters describing the behaviour of the target unit, the input timeseries must be related to / influenced by, said parameters. Unfortunately, high resolution timeseries measurements of the target unit are rarely available which forces engineers to install temporary measurement equipment which then can only provide data for a short time range.

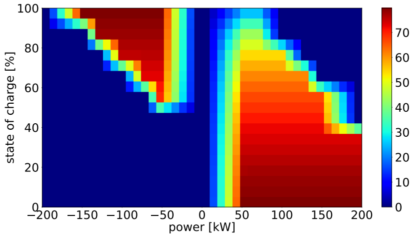
For this paper a vanadium redox flow battery storage, build as part of the research project: "Smart Region Pellworm" [1], was used as target unit. First a model for this unit type was chosen from the available literature [2]. This model was further simplified until its behaviour was defined by 6 variable parameters. In the next step plausible parameter ranges were set and discretized into steps, resulting in 125 000 possible parameter combinations. These combinations provide the labels for the training data. As input variables, timeseries for the voltage (U), current (I) and state of charge (SoC) were chosen.
In the next step an initial SoC was selected for each model configuration after which they were run through specific load profiles to generate the required timeseries for I, U and SoC. Here 3 different sets of load profiles were used to test the ANN's dependency on data that is highly correlated to the target unit. In set 1, each model configuration was given one of 4 historic measured load profiles and related initial SoC's of the target unit to create the training data. In set 2, 10 historic measured load profiles and SoC's were used and in the last, set 3, each model configuration was given a randomly generated load profile and starting SoC. This resulted in 3 separate training data sets.

4 Parameter recognition

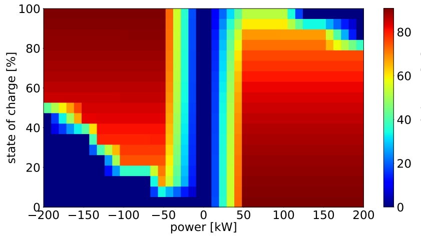
The ANN used in this work is a basic MLP [3] with 3 hidden layers. All layers used the tanh activation function except the output layer for which a linear activation function was used. The width of the layers was arranged in descending size with the intention of compressing the information contained in the input timeseries, with 3 times 288 timesteps, down to the 6 target parameters. The timeseries and labels were normalized between -1 to 1 and 0 to 1 respectively, before being used in the training. During training overfitting was limited by an early stopping function with a patience of 10 steps.
The resulting trained ANN were first tested solely on the historic data belonging to the load profiles used in the creation of their respective training data. It became apparent that the parameter values identified by the two ANN trained on historic load profiles of the target united, differed significantly form the values of the ANN trained on randomized load profiles. Especially the parameters total vanadium concentration and maximum flow speed of the electrolyte are



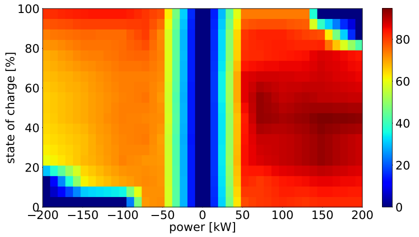
(a) Fitted by ANN trained on data based on 4 historic load profiles



(b) Fitted by ANN trained on data based on 10 historic load profiles



(c) Fitted by ANN trained on data based on random load profiles



(d) Generated based on detailed measurements of the target Battery

Figure 1: Heatmaps depicting the efficiency of the battery system models in realation to the SoC and charging (positiv) or discharging (negative) power

lower than expected in the historic data trained ANN. These parameters, in combination with the SoC, are responsible for determining how much usable vanadium ions can be pumped into the cell, which in turn defines the maximum charge and discharge power of the battery.

To allow for a better comparison between the parameter sets a second test was performed during which the ANN where each given historic data of 50 days and the resulting parameters where averaged and put into individual sets. To evaluate the accuracy of the identified parameters sets they were then used to fit 3 generic models to the target battery. The fitted models were then used to simulate a test scenario and from the resulting data the efficiency across the full range of possible operation situations was calculated.

The results are presented in the heatmaps of Figure 1 which show the efficiency across the full range of possible SoC's and all charge and discharge speeds.

The heatmap of Figure 1d was generated during experiments with the target unit and represent the real battery properties.

The heatmap in Figure 1a was generated by the model fitted with dataset 1 and shows a very early discharge power restriction, these restrictions can be seen in the heatmaps as 0 % efficiency zones. According to this model it would be impossible to discharge the battery further than roughly 50 %. Additionally, it shows an artifact near a SoC of 10 to 0 % which would indicate that a further discharge would suddenly become possible again. The charging side however is much better represented, while the charging power restriction near a full SoC is significantly overestimated, the remaining field is much closer to the reality than the discharge field.

The heatmap in Figure 1b, generated using data set 2, looks very similar to Figure 1a even though its training data was based on a larger set of load profiles. The most notable improvements are the removal of the artifact near the bottom of the discharge field and a slight widening of the possible discharge zone on the top. The charging side however shows no improvement and instead exhibits a slight expansion of the charging power restriction in the top right corner.

Figure 1c shows the results of the model fitted with data set 3, generated with the ANN using randomized training data. It produces by far the most accurate representation of the target properties. The largest improvement can be seen on the discharge side, that now, while still overestimating the discharge power restriction, shows a much more plausible range of usage. A similar improvement is visible on the charging side where the restricted zone has been significantly reduced and now closely resembles the restriction shown by the target unit. Besides the estimated power restrictions another modelling error is visualised in the heatmaps.

All 3 models exhibit a clear overestimation of the battery efficiency. This overestimation is more severe during discharge processes. This error would cause the model SoC to drift away from the actual SoC over time and thus require frequent refreshing with a measured value.

5 Conclusion

The paper proposed a data driven approach to fit generic models to the target generation or storage units. The approach was tested on the example of a vanadium redox flow battery system and further evaluated on its sensitivity regarding the required training data. The results show the best performance can be achieved using randomized load profiles and initial SoC's for the training data generation instead of historic load profiles of the target unit. This has some positive implications for the transferability of the approach on different target units of the same type. While the generation of training data using historic load profiles would have required a new set of training data and retraining of the ANN for each target unit, the randomized approach could allow the usage of one pretrained ANN for all target units of the same type, no retraining necessary. This would also justify putting more resources into the training of the ANN since it would be possible to not only create a library with generic models for each unit type but also create a library of pretrained, ready to use, ANN to fit them to the desired target. Thus significantly reducing the expertise required from users. Further research is required to determine if a single ANN can really fit any unit of a single type regardless of its scale and to test if other ANN architectures could offer higher accuracy. There is also the possibility that completely random load profiles might not be the best approach to training data generation since some of the load profiles might end up being nonsensical and their resulting data devoid of useful information for training.

References

[1] Forschungsprojekt: "Smart Region Pellworm". Gefördert durch BMWi, Förderkennzeichen: 0325498A-F

[2] C. Blanc. "Modeling of a Vanadium Redoxflow Battery Electricity Storage system". École Polytechniqe Fédérale de Lausanne, Suisse 2009.

[3] I. Goodfellow, Y. Bengio, A. Cousville “Deep Learning - das umfassende Handbuch”. mitp-Verlag, Frechen 2018.

Zum Einsatz von KI-Methoden zur Lösung von Problemen der Regelungstechnik

Hanns Sommer, Andreas Kroll

Fachgebiet Mess- und Regelungstechnik, Universität Kassel
34128 Kassel, Mönchebergstr.7
E-Mail: {hans.sommer, andreas.kroll}@mrt.uni-kassel.de

Kurzfassung

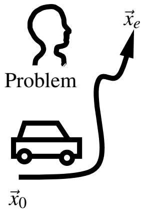
Es wird eine allgemeine Entwurfsmethode für Heuristiken zur Lösung von Problemen der Regelungstechnik vorgestellt. Diese Methode ist mit nur sehr geringen Kenntnissen der Regelungstechnik benutzbar. Mittels diesem Beispiel werden die Vor- und Nachteile einer Verwendung von KI-Methoden in den Ingenieurswissenschaften erörtert.

1 Einführung

Wir betrachten wohlgestellte Probleme, bei denen alle Daten zur Bestimmung einer Lösung mit der Aufgabenstellung gegeben sind und fragen:
„Was macht ein Problem schwierig?“ Man findet Versuche in der Literatur, die Schwierigkeit von Problemen zu definieren [1], aber diese Versuche erscheinen in der Praxis wenig sinnvoll, da das Lösen von Problemen verschiedenartige Fähigkeiten erfordert. Im Bild 1 sind diese Fähigkeiten dargestellt:

1. Das Problem muss aus seiner umgangssprachlichen, praktischen Formulierung in eine formale Sichtweise überführt werden.

2. Die Berechnung des Gesuchten erfolgt aus der formalen Sichtweise.



$$\left\{\begin{array}{l}\text{Formale Sichtweise}\\[6pt] \dot{\vec{x}} = A\vec{x}+B\vec{u}\\[6pt] \vec{x}(t_a)=\vec{x}_0,\ \vec{x}(t_e)=\vec{x}_e\\[6pt] \int\limits_{t_a}^{t_e}||\vec{u}||\mathrm{d}t = \text{minimal}\end{array}\right.^{(1)} \left.\vphantom{\begin{array}{l}a\\a\\a\\a\end{array}}\right\}^{(2)} \begin{array}{l}\text{Lösungs-}\\\text{berechnung}\\[6pt]\text{Lösung}\end{array} \qquad (1)$$

Bild 1: Schritte beim Problemlösen

Die formale Sichtweise stellt einen Verstehenshorizont dar, innerhalb dem dann Berechnungen möglich werden. Das Problem der Suche nach einer Steuerungsfunktion, die $\vec{x}_0$ in $\vec{x}_e$ überführt, wird so in ein mathematisches Problem überführt, das im Rahmen der Matrizenrechnung gelöst werden kann. Es ist klar, dass Fähigkeit (2) durch Rechnerunterstützung erheblich erhöht wird und heute vielfach ganz dem Computer übertragen werden kann (vgl. [2]), aber die Ziele der Künstlichen Intelligenz (KI) sind wesentlich weiter, es soll auch Fähigkeit (1) erweitert werden. Da der Mensch immer derjenige bleibt, der dem Computer, in einer diesem verständlichen Weise, sagen muss, was zu tun ist, wird er niemals vollständig vom Computer ersetzt werden, die Kommunikation mit dem Computer ist dagegen vereinfachbar. Mittels einer **Bereitstellung von Berechnungsleistungen für sehr allgemeine Sichtweisen** macht es die KI möglich, dass der Mensch spezielle Einzelheiten seiner Fragestellung nicht mehr erfassen muss und diese, in einer für ihn sehr einfachen Form, an den Computer übergibt. Dadurch soll es dem Menschen ermöglicht werden, ohne tiefere Fachkenntnisse auch kompliziertere Aufgabenstellungen zu lösen.

In unserem Beitrag soll dieses „Versprechen der KI“ an Aufgabenstellungen aus der Regelungstechnik erprobt werden. Dafür wird in Kapitel 2 eine Überführung einer großen Klasse von Aufgaben der Regelungstechnik in Suchprobleme vorgestellt. Nach einer Diskussion der Schwierigkeiten beim Lösen von Suchproblemen in Kapitel 3, werden dann in Kapitel 4 Listen von Prinzipien zum schnellen Lösen von Suchproblemen und zur Koordinierung der Prozessoren bei Mulit-Prozessor-Computern vorgestellt. Mittels diesen Listen und einem in Kapitel 5 dargestellten Schema zur Konzipierung eines Suchverfahrens ist es dann Studierenden, ohne tiefere Kenntnisse aus der Regelungstechnik möglich, Aufgaben dieses Fachgebiets zu lösen. Die Vor- und Nachteile dieser Methode gegenüber den klassischen Lösungsverfahren können damit erkannt werden. Die KI-Methode hat insbesondere das Problem der Validierung. Dieses Problem wird in Kapitel 6 diskutiert, mittels eines Vergleichs der dargelegten Methode mit ChatGPT. Kapitel 7 resümiert die Ideen zur Konzipierung eines allgemeinen Problemlösers.

2 Umformung eines allgemeinen Problems in ein Suchproblem

Mit der Methode der Generalisierung von Fragestellungen, lassen sich die Aufgabenstellungen der Regelungstechnik vereinheitlichen. Da diese Aufgabenstellungen mathematisch wohlgestellte Probleme sein müssen, erfüllen sie die folgende **Definition:**

Definition: Ein Problem heißt **mathematisch wohlgestellt**, wenn folgende Fragen beantwortet werden können:

1. Wo wird gesucht?	$\longrightarrow$	S	Suchraum
2. Was wird gesucht?	$\longrightarrow$	$P(s)$	Beschreibung des Gesuchten

Da für Probleme der Regelungstechnik der Suchraum S und die Beschreibung des Gesuchten, das Prädikat $P(s)$, bekannt sind, ist eine Überführung in ein Suchproblem möglich.

Im Bild 2 sind in der ersten Spalte die Problemtypen und in der zweiten Spalte die zugehörigen Spezifikationen von S und $P(s)$ angegeben. Die dritte Spalte zeigt die vereinheitlichende Schreibweise, die alle diese Probleme erfasst. Insbesondere bei Mehrgrößensystemen wird die regelungstechnische Lösung sehr komplex.

Das Suchproblem soll mit einem **einheitlichen Lösungsprozess**, ohne spezielle Fachkenntnis aus der Regelungstechnik, gelöst werden. Im nächsten Abschnitt, wird zunächst die Schwierigkeit dieses Problems untersucht.

$$\left.\begin{array}{ll}
\text{Regelungstechnik:} & S = \text{Funktionen } \vec{u}(t) \\
\text{Steuerungsproblem} & \text{mit } t \in [ta, t_e] \\
 & \text{z.B. Treppenfunktionen,} \\
 & \text{Polynome, Splines, } \ldots \\
 & P : \vec{u}(t) \text{ überführt } \vec{x}_0 \text{ in } \vec{x}_e \\
 & \text{und Nebenbedingungen} \\
 & \\
\text{Polvorgabe-} & S = \{F \in \mathbb{R}^{p \times n}\} \\
\text{Regler} & P{:}p(\lambda){=}\det(\lambda I{-}(A{+}BF)) \\
\text{Berechnung} & \text{hat Nullstellen} \\
 & \lambda_1, \lambda_2, \ldots, \lambda_n \\
 & \text{und Nebenbedingungen} \\
 & \\
\text{Beobachter-} & \ldots\ldots \\
\text{Konstruktion} & \\
\vdots &
\end{array}\right\}
\begin{array}{l}
\text{Suchproblem} \\
\Pi : s \in S? \quad P(s)
\end{array}$$

Bild 2: Überführung von Aufgaben der Regelungstechniker in ein Suchproblem

3 Über die Schwierigkeit von Suchproblemen

Eine Problemlösung wird schrittweise erreicht. Bild 3, stellt das Problem der „Suche nach dem Wirtshaus“ dar.

Zunächst könnte das Wirtshaus irgendwo sein, und es müsste der gesamte Suchraum danach abgesucht werden. Gibt es aber wegweisende Schilder, so müssen wir nur die, in unserem lokalen Sichtbereich entdecken und können somit das Problem einfach lösen. Wir definieren:

Definition: Probleme, die mit einer **lokalen Sichtweise** gelöst werden können, heißen **einfache Probleme**, Probleme zu denen keine lokale Sichtweise existiert, heißen **schwierige Probleme**.

Um diese Definition zu rechtfertigen, überlegen wir und kurz den Rechenaufwand zur Lösung eines Optimierungsproblems. Ohne jede Problemvereinfachung müssten wir jedes Element aus S mit jedem anderen Element verglei-

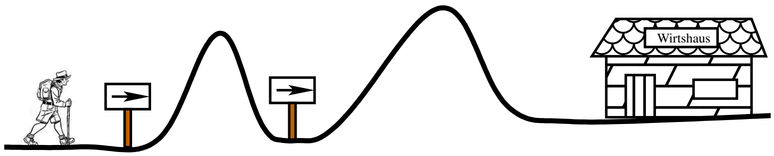


Bild 3: Die Suche nach dem Wirtshaus

chen, was bei N Elementen in S eine Größenordnung von $\exp(N)$ Vergleichen ergibt. Kann die Anzahl der Vergleiche aber auf kleine Untermengen von S mit u Elementen eingeschränkt werden, so sind hierfür nur größenordnungsmäßig $\exp(u)$ Vergleichsoperationen notwendig, was eine Rechenzeitvereinfachung um den Faktor $\frac{\exp(n)}{\exp(u)}$ ergibt.

Wir erkennen somit, dass unsere Definition konzeptionell der Unterscheidung zwischen P-schwierig und NP-schwierig entspricht. Der Vorteil unserer Definition ist jedoch, dass diese den folgenden Satz beweisbar macht:

Satz: Es gibt schwierige Probleme.

Beweis. Die Unlösbarkeit des **Consensus Problems** (vgl. [3]) zeigt, dass nicht jede globale Zwangsbedingung durch lokale Zwangsbedingungen ersetzt werden kann. Daher ist es nicht immer möglich, eine globale Kennzeichnung des Gesuchten in eine lokale Kennzeichnung umzuformen, die bereits in einem lokalen Bereich erkennbar wäre. □

Dass ein Problem schwierig ist, erkennen wir daraus, dass es widersprüchliche, lokal unvereinbare Lösungsstrategien gibt. Im Bild 4 ist **Verpackungs-Problem** dargestellt. Für das Verpackungs-Problem gibt es zwei Strategien, die sich gegenseitig widersprechen und die nicht in einer einzigen lokal formulierbaren Anweisung vereinigbar sind:

Strategie (1): Fülle jedes Paket möglichst vollständig!

Strategie (2): Verteile die großen Schachteln zuerst!

In analoger Weise haben wir für das **Suchproblem** die unvereinbaren Strategien:

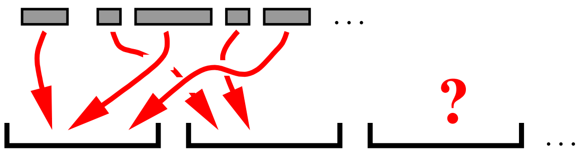


Bild 4: Verpacke die oberen Schachteln in möglichst wenigen der unten gezeigten Standard-Pakete

(1) Exploitation: Nutze Dein Wissen, um nicht alles absuchen zu müssen!

(2) Exploration: Berücksichtige alle Teile des Suchraums!

Für schwierige Probleme gilt das

„No Free Lunch“ Theorem: Zu schwierigen Problemen existiert nicht ein bestes Lösungsverfahren für die gesamte Problem-Klasse. Ein jeweils bestes Verfahren muss an das spezielle Problem angepasst sein.

4 Effiziente Suchverfahren

4.1 Schnelle Ein-Agenten-Suchverfahren

Um effiziente Heuristiken zu erstellen, benötigen wir Prinzipien, mittels denen wir die Suche schneller machen können. Dazu müssen wir einen Grund in der Problemstellung finden, der ein Prinzip zur Suchbeschleunigung möglich macht. Alle diese Prinzipien können wir in einer, im folgenden angegebenen, Liste zusammenstellen. Die Vollständigkeit dieser Liste ist natürlich nicht mathematisch beweisbar, sondern nur argumentativ begründbar.

- Eine Untersuchung aller uns bekannten Heuristiken ergab keine weiteren Prinzipien.
- Das Argument „Was sonst?“. Man kann überlegen, welche Möglichkeiten zur Beschleunigung einer Suche prinzipiell bestehen und daraus erkennen, dass keine weiteren Prinzipien denkbar sind.

In der Liste sind angegeben:

- der Grund, der beim Problem erfüllt sein muss, damit das Prinzip anwendbar ist,
- das **Lösungsprinzip**
- und Beispiele von Verfahren, in denen das Prinzip realisiert ist.

Um die Liste zur Konzipierung von **Ein-Agenten-Suchverfahren** zu verwenden, muss der Anwender zunächst prüfen, ob der Grund für die Anwendung eines Prinzips in seinem speziellen Such-Problem erfüllt ist. Falls dies bejaht wird, so kann er das entsprechende Prinzip mittels einer Unterroutine, die dieses realisiert, in seine Heuristik aufnehmen. Am einfachsten ist es, ein in den Beispielen angegebenes, realisiertes Verfahren zu kopieren und in die eigene Heuristik einzubauen. Eine Übersicht über alle Methoden, die in der Literatur angegeben werden, ist hier natürlich nicht möglich. Die Anzahl der Beispiele könnte beliebig ergänzt werden. Weiter ist zu beachten, dass in vielen Verfahren oft auch mehrere Prinzipien realisiert sind, d.h. eventuell auch solche, die beim vorliegenden Problem keine Verringerung der Suchzeit bewirken.

Prinzipien schneller Lösungsverfahren:

- Es gibt eine Nachbarschaftsstruktur $N(s)$ für $s \in S$ auf dem Suchraum, die zum Problem passt.
  $\Longrightarrow$ Prinzip: **„Suche das Bessere nahe dem Guten!“**

  Beispiele: Gierige Heuristik (Gradientenverfahren), Nachbarschaftssuche, variable Nachbarschaftssuche.

- Es gibt eine Bewertung der Elemente $s \in S$ bezüglich Ihrer Lösungsqualität, ohne dass $P(s)$ explizit berechnet werden muss.
  $\Longrightarrow$ Prinzip: **„The most promissing first!“**

  Beispiele: A^*-Algorithmus, Fred Glovers Methode: Ändere den bisherigen Lösungsweg an der Stelle der vielversprechendsten Alternative!

- Es gibt Teile des Lösungsvorschlags $s \in S$, die besser bestätigt sind als andere.
  $\Longrightarrow$ Prinzip: **„Lass Sicheres ungeändert und ändere Unsicheres.“**

Beispiele: Backbone-Methode, No-Goods-Methode, Ameisenverfahren, Kernal Search, Auswahlverfahren der Nachbarschaftsstruktur $N_k(s)$ bei der variablen Nachbarschaftssuche.

- Vermeidung von Doppelt-Durchsuchungen.
  $\Longrightarrow$ Prinzip: "**Ordnung hat, wer weiß, wo er erst gar nicht suchen muss!**"

  Beispiele: TABU-Search, Branch & Bound-Methode.

- $\Pi : s \in S? \quad P(s) \qquad$ mit $s = (s_1, s_2, \ldots s_K) \in S_1 \times S_2 \times \ldots \times S_K \equiv S$
  ist darstellbar durch die Teilprobleme:
  $\Pi_k : s_k \in S_k? \quad P_k(s_k) \qquad$ mit $k = 1, 2, \ldots K$
  $\Longrightarrow$ Prinzip: **„Zerlege das Problem in Teilprobleme“**

  Beispiele: Gauß-Algorithmus, Crossover-Operator beim Genetischen Algorithmus.

- $\Pi : s \in S? \quad P(s)$ ist äquivalent zur Folge der Probleme:
  $\Pi_1 : s_1 \in S_1? \quad P_1(s_1)$,
  $\Pi_2 : s_2 \in S_2? \quad P_2(s_2|s_1), \ldots,$
  $\Pi_K : s_K \in S_K? \quad P_K(s_K|s_1, s_2, \ldots s_{K-1})$
  $\Longrightarrow$ Prinzip: **„Verfolgung von Zwischenzielen!“**

  Beispiel: Schrittweise Approximationsmethoden.

- $\Pi_k : s_k \in S? \quad P_k(s)$, mit $k = 1, \ldots K$ sei eine Folge von Problemen mit:

  (α) Alle Π_k haben den selben Suchraum S.

  (β) Pi_1 ist einfach lösbar.

  (γ) Pi_k ist ähnlich zu Pi_{k+1}.

  $\Longrightarrow$ Prinzip: **„Homotopie-Suchverfahren“**
  Suche Lösung s_1 zu Π_1,
  Suche Lösung s_2 zu Π_2 in Umgebung $N(s_1)$,
  Suche Lösung s_3 zu Π_3 in Umgebung $N(s_2)$,
  ....
  Suche Lösung s_K zu Π_K in Umgebung $N(s_{K-1})$.

Tabelle 1: Koordinierungsstrategien

Intensivierung	**Diversifikation**
Agenten lernen vom Verhalten anderer Agenten.	Die Agenten teilen sich die Aufgabe auf, und verhalten sich möglichst unterschiedlich.

- Die Menge der möglichen Lösungsverfahren ist klein.
  $\Longrightarrow$ Prinzip: **„Überführung eines Problems in sein Meta-Problem“:**

$$\begin{aligned} \Pi &: s \in S? \quad P(s) \qquad \Longleftrightarrow \\ \Pi_\Pi &: \text{Lösungsverfahren} \in \text{Menge der Lösungsverfahren zu } \Pi? \\ & \text{ mit Lösungsverfahren löst } \Pi. \end{aligned}$$

4.2 Koordinierungsmethoden bei Multi-Agenten-Suchverfahren

Multi-Agenten-Verfahren sind dann sinnvoll, wenn die Arbeit mehrerer Agenten koordiniert werden kann. Da bei Multi-Prozessor-Rechnern die Agenten auf die Prozessoren verteilt werden können, ergeben sich in diesem Falle große Erhöhungen der Rechengeschwindigkeit. Das **Ziel der Koordinierung** sind die beiden Strategien in Tabelle 1. Diese Strategien können, falls die entsprechenden Voraussetzungen zu den Regeln auf dem betrachteten, speziellen Problem erfüllt sind, in der Heuristik zur Koordinierung der Agenten eingesetzt werden (Tabelle 2).

5 Ein generelles Lösungsverfahren für die Probleme der Regelungstechnik

In diesem Kapitel wird zunächst ein Ansatz vorgestellt, mittels dem die im vorherigen Kapitel bereitgestellten Prinzipien und Regeln zur Konzipierung eines

Tabelle 2: Strategien der Koordinierung

Koordinierung von Agenten	**Verfahren die diese realisieren**
„Lerne vom Besten!“	Partikelschwarm-Optimierung
„Lerne beim Test jedes Agenten zur Verbesserung der gesamten Agenten-Gruppe“	Nelder-Mead-Verfahren
„Bilde das Bessere aus guten Teilen“	Genetische Algorithmen
„Lass viele Gruppen versuchen ein Problem zu lösen und wähle die effizienteste Gruppe aus“	Neuronale Netze
„Eliminiere ineffiziente Agenten unnd füge neue Agenten in effizienter Weise ein“	Neustart-Methode, Identifikation von vielversprechenden Gebieten in S.
Kombination vieler Regeln	Fred Glovers „Multi-Agenten-Suchverfahren“

Lösungsverfahrens für Suchprobleme realisiert werden können. Mit diesem Ansatz konnten Studierende Erfahrungen sammeln und regelungstechnische Probleme lösen. Dies ermöglicht einen Vergleich der verwendeten KI-Methode mit den klassischen Lösungsansätzen.

Als Ausgangspunkt wurde angenommen, dass die Studierenden praxisnah, ein einzelnes anfallendes Problem möglichst schnell lösen sollten, d.h. ohne tiefere Einarbeitungszeit in die zugehörige Mathematik. Zur Lösung wird die hier eingeführte Methode, mittels der Anleitung von Abschnitt 5.1, verfügbar gemacht. Die Aufgabe gilt als erfolgreich gelöst, wenn eine geeignete Lösung zum Problem gefunden wurde. Eine darüber hinausgehende Bewertung soll aus den folgenden Gründen nicht erfolgen:

1. In der industriellen Praxis ist es in der Regel nicht von Interesse, nachzuweisen, dass eine bereits gefundene Lösung auch einfacher auffindbar gewesen wäre.

2. Eine Bewertung des Lösungsverfahrens hängt, wegen des „No Free Lunch“-Theorems, immer auch von dem oder den betrachteten Problemen ab. Die in einer solchen Bewertungsaussage vorkommenden Begriffe: Schnelligkeit, Zugehörigkeit zur Klasse des speziellen Problems, Erfolgssicherheit des Lösungsansatzes und einige weitere, sind nur schwer fassbar. Es ist, zum Beispiel für SAT-Probleme, einer Untermenge der Klasse der Suchprobleme, bekannt, dass die besten Löser für schwierige Probleme nicht optimal für industrielle Probleme sind.

3. Da die Validierung ein eigenes Gebiet ist, das spezielle Betrachtungsweisen erfordert, soll darauf erst im Kapitel 6 eingegangen werden. In Kapitel 5 beschränken wir uns bei der Validierung auf den Nachweis der Korrektheit der erhaltenen Lösung.

5.1 Konzipierung eines Suchverfahrens

Gegeben ist ein Problem $\Pi : s \in S? \quad P(s)$ und Zusatzwissen.

(1) Prüfe, ob Voraussetzungen für schnelle Suchverfahren erfüllt sind, indem die folgenden Fragen beantwortet werden:

- Gibt es Umgebungsstrukturen auf S, die zum Problem passen?
- Gibt es „vielversprechende Elemente“ in S?
- Gibt es Kriterien, um Gebiete von S auszuschließen?
- Gibt es Wichtiges und Unwichtiges bez. S oder bez. seiner Elemente?
- Bestehen die Elemente $s \in S$ aus mehreren Teilen?
- Sind „sichere oder gute Teile“ in Elementen $s \in S$ erkennbar?
- Ist das Problem in mehrere Teilprobleme zerlegbar?
- Gibt es Zwischenziele, die angestrebt werden können?

- Kann das Problem aus der „Verbiegung“ eines einfachen Problems erhalten werden?
- Sollten die Agenten eine Gruppe bilden um größere Bereiche zu überblicken?

(2) Wähle einige der Methoden zu den Prinzipien aus, deren zugehörige Testfrage in (1) mit „ja“ beantwortet wurde.

(3) Entscheide, ob mehrere Agenten zur Realisierung der Methoden von (1) koordiniert werden können. Falls ein Multi-Agenten-Verfahren sinnvoll erscheint, wähle die Regeln für die Koordinierung.

(4) Realisiere die ausgewählten Methoden in einer Heuristik.

(5) Validiere die erhaltene Heuristik.

Bemerkung: In (1) bis (3) werden die Voraussetzungen der Prinzipien geprüft, (4) erfordert softwaretechnische Realisierungsschritte, die in (5) überprüft werden. Die Prinzipien ergeben eine „grobe“ Charakterisierung der verwendeten Verfahren. Die Effizienz der Heuristik hängt auch stark davon ab, wie gut die einzelnen Verfahren in der Heuristik realisiert sind.

5.2 Anwendung und Test der Methode zur Konzipierung von Heuristiken

Um dem Leser einen Eindruck zu vermitteln, welche Aufgabenstellungen durch die Studierenden gelöst wurden, werden diese im Folgenden kurz vorgestellt. Alle Aufgaben konnten innerhalb der Zeit, die für eine Masterarbeit verfügbar ist, gelöst werden. Auf eine weitere Bewertung des Lösungsverfahrens wurde, wegen der oben angegebenen Schwierigkeiten, verzichtet.

5.2.1 Polvorgaberegler für ein lineares Multi-Input-System

Aufgabe: Gesucht ist eine Matrix aus dem Suchraum $S = \{F \in \mathbb{R}^{n\times p}\}$ mit:

$$p_A(\lambda) = \det(\lambda I - (A + BF)) \text{ hat die Nullstellen: } \lambda_1, \lambda_2 \ldots \lambda_n \qquad (2)$$

und der **Nebenbedingung** $||F||_2$ minimal.
Lösungsansatz: Das Problem ist sehr komplex, da der Suchraum sehr viele lokale Optima hat. Es wurden Tests mit verschiedenen Topologien über dem Suchraum durchgeführt. Schließlich konnten für Systeme bis zur Dimension 6 mit dem **Nelder Mead-Verfahren** befriedigende Ergebnisse erzielt werden. Da das Nelder Mead-Verfahren eine ganze „Wolke von Agenten" über dem Suchraum koordiniert, wobei die Bewertung jedes einzelnen Agenten die gesamte Wolke beeinflusst, erschien diese Methode passend zu sein, zum Problem der sehr vielen lokalen Optima.
Lösungsumsetzung: Mit den Bezeichnungen:
$\sum_{k=0}^{n} a_k(F)\lambda^k := \det(\lambda I - (A+BF))$ und $\sum_{k=0}^{n} \bar{a}_k\lambda^k := \Pi_{k=1}^{n}(\lambda - \lambda_k)$
wird der Wert $\alpha_0 = \sum_{k=0}^{n-1}(a_k(F) - \bar{a}_k)^2$ definiert.
Die Heuristik minimiert $\alpha_0 + \rho||F||_2$ wobei ρ mit ansteigender Suchzeit verkleinert wird. Die Suche gilt als erfolgreich, falls im Rahmen der Suchzeit $\alpha_0 < \varepsilon = \frac{1}{10}$ erreicht werden konnte.

5.2.2 Multi-Input-Steuerungsproblem

Aufgabe: Gesucht wird eine zeitdiskrete Steuerungsfunktion $(u_1(k), u_2(k))$, die das im Bild 5 gezeigte Fahrzeug mit den in (3) angegebenen Systemgleichungen vom Zustand $\vec{x}_0$ in den Zustand $\vec{x}_e$ überführt.

$$\begin{pmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \\ x_4(k+1) \\ x_5(k+1) \end{pmatrix} = \begin{pmatrix} x_1(k) + T_a(x_4(k) \cdot \cos(x_3(k))) \\ x_2(k) + T_a(x_4(k) \cdot \sin(x_3(k))) \\ x_3(k) + T_a \cdot x_5(k) \\ x_4(k) + T_a \cdot u_1(k) \\ x_5(k) + T_a \cdot u_2(k) \end{pmatrix} \tag{3}$$

Lösungsansatz: Eine Steuerfunktion besteht aus mehreren Abschnitten, daher wurde die Lösung mit dem Prinzip der Zerlegung in Teilprobleme gesucht und ein **Lösungsansatz mit einem genetischen Verfahren** gewählt.
Lösungsumsetzung: Da die Teilabschnitte von unterschiedlicher Wichtigkeit sind, wurden die Operatoren des Genetischen Algorithmus um Zusatzfunktionen erweitert, die weitere Prinzipien realisieren (Tabelle 3).

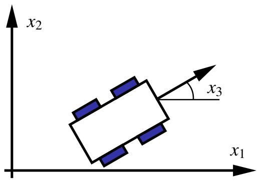


Bild 5: Zu steuerndes Fahrzeug

Wird den Routinen, welche die Prinzipien realisieren, eine gewisse Wirkungsstärke und ein gewisser Anteil an der Gesamtrechenzeit der Heuristik zugeordnet, so kann aus dem Vergleich der Größe dieser Werte mit der verbleibenden Fehlergröße auf die Relevanz des Prinzips geschlossen werden.

5.2.3 Single-Input Steuerungsproblem

Aufgabe: Gesucht ist eine Steuerfunktion $u(t)$, die für das im Bild 6 dargestellten Dreitanksystem mit den Systemgleichungen (4) $\vec{x}_0$ in $\vec{x}_e$ überführt.

$$\begin{aligned} \dot{x}_1(t) &= -\rho\sqrt{x_1(t)} + \frac{1}{A_b}u(t) \\ \dot{x}_2(t) &= -\rho\sqrt{x_2(t)} + \rho\sqrt{x_1(t)} \\ \dot{x}_3(t) &= -\rho\sqrt{x_3(t)} + \rho\sqrt{x_2(t)} \end{aligned} \tag{4}$$

Lösungsansatz: Da eine Linearisierung $\dot{\vec{x}}(t) = A\vec{x} + \vec{b}u$ in $\vec{x} = \vec{0}$ bestimmt werden konnte und da das System, wie Gleichung (5) zeigt, stetig vom linearen System in das nichtlineare System überführt werden kann, wurde eine Lösung mit der **Homotopie-Suchmethode** ermittelt.

$$\dot{\vec{x}}(t) = \vec{x}(t) + \vec{b}u(t) + \lambda\left(f(\vec{x}(t), u(t)) - A\vec{x}(t) + \vec{b}u(t)\right) \tag{5}$$

$$\text{mit } f(\vec{x}(t), u(t)) = \begin{pmatrix} -\rho\sqrt{x_1(t)} + \frac{1}{A_b}u(t) \\ -\rho\sqrt{x_2(t)} + \rho\sqrt{x_1(t)} \\ -\rho\sqrt{x_3(t)} + \rho\sqrt{x_2(t)} \end{pmatrix} \tag{6}$$

Tabelle 3: Zusatzfunktionen der Operatoren des Genetischen Algorithmus

Operatoren	**Zusatzprinzipien**
Crossover-Operator	Sichere Gene werden mit hoher Wahrscheinlichkeit erhalten. TABU-Liste (vermeidet zu große Ähnlichkeit der Elemente). Effiziente Agenten werden öfter berücksichtigt.
Mutation	Variable Mutationsstärke entsprechend der Effizienz des Agenten.
Agenten-Elimination	Löschen schlechter Agenten. Tendenz-Abschätzung (Agenten mit schlechter Prognose werden eliminiert.)
Neue Agenten	Diese werden in bisher unbesetzten Gebieten erzeugt.

Lösungsumsetzung: Zunächst wurde eine Steuerfunktion für das zugehörige lineare System bestimmt und dann die Homotopie-Suchmethode durchgeführt. Das Ergebnis für den erreichten Vektor $\vec{x}(t_e)$ zeigt eine hinreichend gute Übereinstimmung mit dem Soll-Wert $\vec{x}_e$, siehe (7).

$$\vec{x}_e = \begin{pmatrix} 7,04223167 \\ 6,07614977 \\ 5,1288758 \end{pmatrix} \qquad \vec{x}(t_e) = \begin{pmatrix} 7,21634809 \\ 6,13234234 \\ 5,07650415 \end{pmatrix} \tag{7}$$

5.3 Methoden-Vergleich

Tabelle 4 zeigt die Vor- und Nachteile der angewandten Methode gegenüber den klassischen Verfahren. Das größte offene Problem bei der Entwicklung eines generellen Problemlösers mit KI-Methoden ist die Durchführung einer

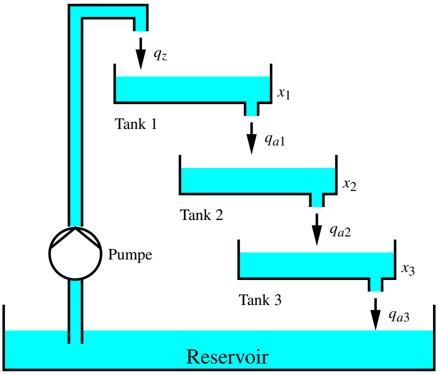


Bild 6: Nichtlineares Dreitanksystem

Validierung. Auf dieses Problem soll im Kapitel 6 genauer eingegangen werden.

6 Validierung

6.1 Anwendung des generativen Sprachmodells ChatGPT

Um das Problem der Validierung zu veranschaulichen, wird ein Vergleich des betrachteten Verfahrens mit dem Sprachmodell ChatGPT durchgeführt. ChatGPT wird zur Zeit in der KI-Gemeinde intensiv diskutiert und mancher KI-Forscher stellt sich die Frage, ob seine Forschung noch aktuell ist, oder ob er sich auf dieses neue Gebiet umstellen sollte. Auch Fachdidaktiker der Mathematik nehmen ChatGPT zur Kenntnis und stellten diesem die Frage [4]:

> **Frage an ChatGPT:** Max ist 78 cm groß, Wenn er auf eine 20 cm hohe Kiste steigt, ist er genauso groß wie Klaus, Paul ist 15 cm kleiner als Klaus. Wie groß ist Paul?

Sie erhielten zunächst die falsche Antwort von ChatGPT: „Paul ist 43 cm groß.“

Tabelle 4: Vor- und Nachteile der Verfahren

Klassische Verfahren	**KI-Verfahren**
Wissen aus der Regelungstechnik ist notwendig.	Problemlösung ist bei geringem Wissen aus der Regelungstechnik möglich.
Jedes Problem erfordert eine spezifische Lösung.	Alle Probleme sind mit einer einzigen Lösungsanleitung lösbar.
Die entwickelten Regler erfordern geringe Rechenzeit und wenig Energie	Die entwickelten Regler erfordern hohen Rechenaufwand und viel Energie.
Der Datenaufwand ist gering	Der Datenaufwand kann hoch sein, bei einem großen Suchraum.
Die gefundene Lösung ist direkt einsetzbar und hat die gewünschten Eigenschaften.	Die gefundene Lösung erfordert eine ausgiebige **Validierung**.

Der **Zusatz zur Frage an ChatGPT:** „Lass uns Schritt für Schritt vorgehen.“ Ergab jedoch die richtige Antwort: Paul ist 83 cm groß.

Die Frage ist nun: **Wie erhält man aus der Validierung eine Erklärung des Fehlers, die eine Verbesserung der Heuristik möglich macht?**

Als Nicht-Fachdidaktiker würde man nicht auf die Idee kommen, den obigen Zusatz zu geben, da angenommen würde, dass jemand, der überhaupt eine Antwort findet, diese immer nur schrittweise finden kann.

Das Problem von ChatGPT ist, dass es als Wissens-Hintergrund das Internet verwendet, das aus einer völlig unstrukturierten und auch häufig widersprüchlichen Informationsflut besteht. Bezüglich dieses Wissens-Chaos ist keine zielgerichtete Fehler-Grund-Suche möglich!

Dass die KI Erklärungen für ihre Ergebnisse liefern sollte, ist seit über 25 Jahren ein Thema und die Aufgabe einer „Explainable Artificial Intelligence“.

In der neueren KI-Literatur findet man viele Hinweise darauf, dass in diesem Zweig der KI Fortschritte wesentlich langsamer erhalten werden, als dies zunächst erhofft wurde. In speziellen Anwendungsfällen steht jedoch die Möglichkeit einer zielgerichteten Validierung zur Verfügung.

6.2 Zielgerichtete Validierung

Die Methode, mit der in unserem Fall eine Validierung durchgeführt werden kann, ist der „Means-end account of Explainable Artificial Intelligence“ [5]: **Eine zielgerichtete Ermittelung der Nicht-Korrespondenzen zwischen dem Problem und der Heuristik.** Wie in Bild 7 dargestellt ist, können wir nicht nur Tests auf die üblichen Fehlerarten (Bias-Fehler, Overfitting und Overtraining usw.) durchführen, sondern auch gezielt die Effizienz der in der Heuristik realisierten Prinzipien und Koordinierungsregeln testen (analog, wie dies von Studierenden im Beispiel in Kapitel 5.2 durchgeführt wurde). Je nach Testergebnis kann dann die Wirkung eines Prinzips verstärkt oder abgeschwächt, bzw. ein Prinzip durch ein anderes ersetzt werden.

Da unsere Prinzipien- und Koordinierungslisten endlich sind, könnten diese (zur Zeit noch manuell ausgeführten) Tests automatisiert werden. Die Gültigkeit eines Prinzips oder einer Regel, (mit den zugehörigen Parametern,) kann als Hypothese aufgefasst werden, deren Gültigkeits-Wahrscheinlichkeit von einem Bayes-Schätzer mit den Ergebnissen aus der Validierung berechnet wird.

7 Schematische Darstellung eines allgemeinen Problemlösers

Wir gehen von einem Problem aus und machen uns zunächst allgemeine Überlegungen. In unserem Fall haben wir mit diesen Überlegungen einen Verstehenshorizont verfügbar gemacht, der von den Listen der Prinzipien und Koordinierungsregeln gebildet wird. Mit dem Verstehenshorizont können wir dann die Prinzipien zur Bildung eines Lösungsverfahrens zusammenstellen, dessen

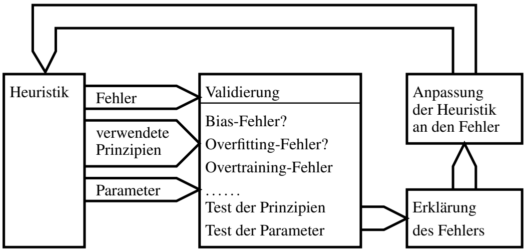


Bild 7: Validierung einer Heuristik, für den hier konzipierten Lösungsansatz.

Formalisierung eine Lösungs-Heuristik zur Berechnung einer Lösung liefert (vgl. [6] bezüglich einer analogen Vorgehensweise.)

Die Validierung der Lösung vor dem Hintergrund der verwendeten Prinzipien und Koordinierungsregeln ergibt eine Erklärung des Fehlers und damit die Möglichkeit zur Verbesserung der Heuristik. Bei allgemeinen KI-Systemen wie ChatGPT (oder auch z.B. Fahrerassistenz-Systemen) steht die Validierung zur Zeit vor dem Problem eines unerfassbaren und unstrukturierten Verstehenshorizonts, bezüglich dem der „Begriff der Erklärung“ nicht spezifizierbar ist. Der Attention-Mechanismus von ChatGPT erzeugt erst dessen Verstehenshorizont, so dass dieser dem Validierungs-Prozess nicht verfügbar ist. Die Validierungs-Aufgabe ist somit, im Gegensatz zu unserem Ansatz, mathematisch nicht vollständig spezifiziert. Analog zur „Strukturierten Programmierung“ bei der über die strikt mathematische Ebene der Programme eine argumentative Ebene gesetzt wird, welche die Einhaltung von Ordnungsprinzipien garantiert, wurde hier, oberhalb der Ebene der reinen Heuristiken, eine argumentative Ebene zur Konstruktion dieser Heuristiken eingefügt, um deren Auswahl und Organisation begründet durchführen zu können.

Literatur

[1] A. Arana und W. Stafford. „On the difficulty of discovering mathematical proofs“. Synthese, http://doi.org/10.1007/s11229-023-04184-5. 2023.

[2] W. Flannery. „A revolution in physics education was forecast in 1989, why hasn't it happened? What will it take?", Am. J. Phys., Vol.91, No4, 256-257, April 2023.

[3] M. Lund und R.K. Belew, „No perfect two-state cellular automata for density classification exists". Physical Review Letters 74 (25):5148-5150, 1995

[4] S. Schorcht, L. Baumanns, N. Buchholtz, J. Huget F. Peters und M. Pohl. „Ask Smart to Get Smart: Mathematische Ausgaben generativer KI-Sprachmodelle verbessern durch gezieltes Prompt Engineering". GMD-Mitteilungen 115, S.12-23, 2023.

[5] O. Buchholz. „A Means-End Account of Explainable Artificial Intelligence". Synthese, http://doi.org/10.1007/s11229-023-04260-w. 2023.

[6] A. Marques Del Valle, R. Gomes Mantovani und R. Cerri. „A systematic literature review on AutoML for multi-target learning tasks". *Artificial Intelligence Review*, https://doi.org/10.1007/s10462-023-10569-2. 2023.

Physical Interpretability of Data-Driven State Space Models

Hermann Klein, Max Schüssler, Oliver Nelles

Universität Siegen, Department Maschinenbau,
Institut für Mechanik und Regelungstechnik – Mechatronik
Paul-Bonatz-Str. 9-11, 57068, Siegen, Germany
E-Mail: {hermann.klein,oliver.nelles}@uni-siegen.de
research@maxschuessler.de

1 Introduction

Nonlinear state space models are powerful model architectures for system identification. They provide the necessary flexibility for the description of nonlinear dynamic processes while still maintaining in a quite compact respresentation. Typically, the data-driven state space models are black-box models, a fact that causes shortcomings regarding interpretability [9]. Since the modeling performance is satisfying and competitive to recurrent neural networks [5], we strive for an increase in interpretability. Interpretability in terms of physical insights can be achieved by the incorporation of prior process information. This leads to a gray-box identification strategy. Eventually, the goal of this contribution is to combine both modeling power and the interpretability of its parameters.

Gray-box methods require prior process knowledge besides measurement data. Prior knowledge can be available in different forms. This work studies the case in which the process structure is a priori known in form of a physical equation. If a structured identification is carried out, the model is forced into an explainable form. Then, both the modeling task and the interpretability problem are solved. The central idea of the proposed method is the reduction of the model parameters down to the physically required number.

An advantageous architecture for the structured identification is the Local Model State Space Network (LMSSN), developed by Schüssler [6]. Its local linear behavior supports the desire for interpretability, because the well-known foundations of linear system theory can be applied.

2 Gray-Box Modeling with the Local Model State Space Network

LMSSN is an extension of the linear time-discrete state space for modeling nonlinear processes. Detailed information about LMSSN can be found in [6]. On the one hand, it can be seen as a state space framework in which the multidimensional nonlinear functions in the state and output equations are implemented with two Local Model Networks (LMN) [1]. On the other hand, it can be seen as a deep neural network consisting out of one recurrent layer and a dense layer. Expressed with the above mentioned LMNs, a single-input single-output LMSSN of order n_x with the input $u(k)$, the state $\hat{x}(k)$ and the output $\hat{y}(k)$ is defined by

$$\begin{aligned} \underline{\hat{x}}(k+1) &= \sum_{j=1}^{n_{\text{state}}} \left(\underline{o}_j^{[\text{d}]} + \underline{A}_j^{[\text{d}]} \underline{\hat{x}}(k) + \underline{b}_j^{[\text{d}]} u(k)\right) \cdot \Phi_{\text{state},j}(k) \\ \hat{y}(k) &= \sum_{j=1}^{n_{\text{out}}} \left(p_j + \underline{c}_j^{\text{T}} \underline{\hat{x}}(k) + d u(k)\right) \cdot \Phi_{\text{out},j}(k). \end{aligned} \tag{1}$$

Here, $\underline{o}_j^{[\text{d}]}$ is the offset of the state equation, $\underline{A}_j^{[\text{d}]}$ and $\underline{b}_j^{[\text{d}]}$ can be interpreted as the slopes of the j-th model of the state equation. Accordingly, $\underline{p}_j^{[\text{d}]}$ is the offset of the output euqation and $\underline{c}_j^{[\text{d}]}$ and $\underline{d}_j^{[\text{d}]}$ are the slopes of the j-th model of the output equation. (The superscript $(\cdot)^{[\text{d}]}$ marks the discrete-time description.) There are altogether n_{state} superposed affine models in the state equation network and n_{out} in the output equation network. The basis functions $\Phi_j(k)$ express the j-th local validity. They are realized with normalized Gaussians and generate a global nonlinear function by superposition of the local affine models. Due to the fact that these local state space models are fully-parameterized, LMSSN is a black-box model.

In the following, the steps of structured identification are presented. The procedure requires a novel initialization technique for LMSSN models. Normally, LMSSN is initialized as a linear state space model, before it is divided into serveral local models with the help of the Local Linear Model Tree (LOLIMOT) or the Hierachical Local Model Tree (HILOMOT) algorithm [1]. The initial model for this tree-construction algorithm is the Best Linear Approximation (BLA) of the process estimated from input-output data $\{u(kT_0), y(kT_0)\}$. It is generated via a subspace-based system identification method [2] and results in a linear fully-parameterized model. If the model is restructured and restricted, it leads to a linear gray-box model. With the help of LOLIMOT, a nonlinear gray-box model is then derived by splitting the extended input space $\underline{\tilde{u}} = [\underline{\hat{x}}, u]^{\mathrm{T}}$ and adding local models. Note that the structure of the initial model is kept as splitting progresses and it is passed to the local models generated by LOLIMOT. The restructuring step will be described more closely in the following. Figure 1 shows the workflow for gray-box structured identification.

Canonical state space forms like the Canonical Controllable Form (CCF) are easy to achieve via similiarity transformations [8]. An arbitrary gray-box structure requires a more sophisticated restructuring strategy because the transformation can not be calculated directly. Therefore, three possible methods are stated. A nonlinear unconstraint optimization with an additional penalty term, called Penalty Method (PM) [7] can force the free parameters to their desired values. Alternatively, a classical gray-box technique is the Prediction Error Method (PEM) [6]. It uses hard constraints for implementation of the known parameters $\underline{\theta}_{\mathrm{con}}$ by placing them in the model. Here, the fully-parameterized black-box model based on BLA is only necessary for initialization. Another alternative is to estimate a specific transformation matrix that leads to the desired state space form [4].

After the restructuring step, only the free parameters will be optimized while the constrained parameters are kept "frozen". In the following step, LOLIMOT generates a nonlinear global model by partitioning $\underline{\tilde{u}}$. Finally, the described procedure yields a nonlinear model containing the desired gray-box structure. As a post-processing step, the physical parameters can be extracted, which is useful for analysis and gives insights into process.

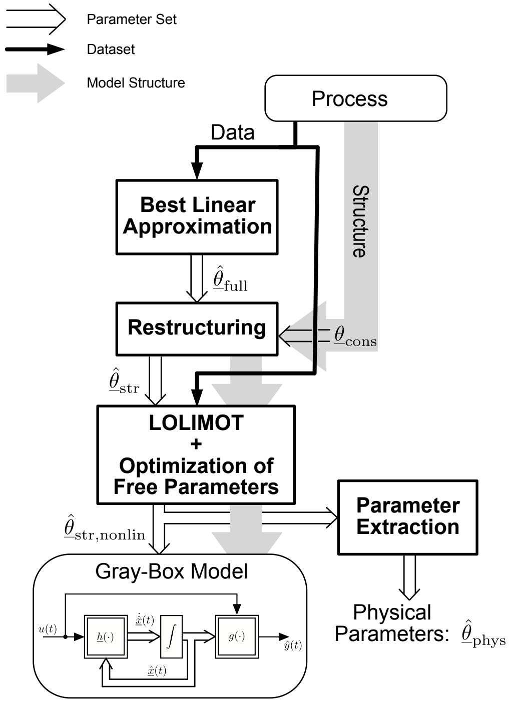

Parameter Set
Dataset
Model Structure
Process
Data
Structure
Best Linear Approximation
$\hat{\underline{\theta}}_{\mathrm{full}}$
Restructuring
$\underline{\theta}_{\mathrm{cons}}$
$\hat{\underline{\theta}}_{\mathrm{str}}$
LOLIMOT + Optimization of Free Parameters
Parameter Extraction
$\hat{\underline{\theta}}_{\mathrm{str,nonlin}}$
Gray-Box Model
$u(t)$
$\underline{h}(\cdot)$
$\dot{\hat{\underline{x}}}(t)$
$\hat{\underline{x}}(t)$
$g(\cdot)$
$\hat{y}(t)$
Physical Parameters: $\hat{\underline{\theta}}_{\mathrm{phys}}$

Figure 1: Structured identification procedure. The BLA yields the parameter vector $\hat{\underline{\theta}}_{\mathrm{full}}$. For restructuring the constraint parameters $\underline{\theta}_{\mathrm{cons}}$ and their indexes are used. The structured parameters are in the vector $\hat{\underline{\theta}}_{\mathrm{str}}$ while the parameters of the nonlinear model are in $\hat{\underline{\theta}}_{\mathrm{str,nonlin}}$. The parameter extraction decodes $\hat{\underline{\theta}}_{\mathrm{str,nonlin}}$ to the interpretable parameters $\hat{\underline{\theta}}_{\mathrm{phys}}$.

3 Test Process

The proposed structured LMSSN is demonstrated on simulated data from a mechanical system which is a moving body with a single degree of freedom. The system's input $u(t)$ is the excitation force acting on the center of gravity while the output $y(t)$ is its position:

$$M\ddot{y}(t) + D\dot{y}(t) + \underbrace{F_{\mathrm{S}}(y)}_{=F_{\mathrm{off}}(e^{\gamma y}-1)} = u(t). \tag{2}$$

Here, the body's mass is M and the linear damping ratio is D. A static progressive spring curve $F_S(y)$ leads to a nonlinear equation of motion. The stiffness characteristic is parameterized with the curve offset F_{off} and the exponential stiffness rate γ. For the excitation of the system, a step-like signal containing 96 events with random levels in the interval $[0\,\text{N}; 1\,\text{N}]$ is chosen. The numerical integration of the equation of motion, necessary for output calculation, is carried out with the Euler-forward method. After data generation, the output signal was artifically disturbed with additive white Gaussian noise with an signal-to-noise ratio of 49 dB.

Next, we state the specific prior knowledge of the process (see (2)) required for gray-box identification. In the present case, the gray-box knowledge is the information that the process can be approximated by a second order system whose numerator equals one (PT_2 system[1]). This knowledge is applied to the model. A favorable structure for the above mentioned task is a Nonlinear Controllable Form (NCF). Compared to CCF, the initial NCF has an additional offset in the state equation. The NCF is written as

$$\begin{aligned}\dot{\underline{x}}(t) &= \underbrace{\begin{bmatrix} 0 & 1 \\ \theta_{\text{free},1} & \theta_{\text{free},2} \end{bmatrix}}_{\underline{A}(\underline{\theta}_{\text{free}})} \underline{x}(t) + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{\underline{b}} u(t) + \underbrace{\begin{bmatrix} 0 \\ \theta_{\text{free},3} \end{bmatrix}}_{\underline{o}(\underline{\theta}_{\text{free}})}, \\ y(t) &= \underbrace{\begin{bmatrix} \theta_{\text{free},4} & 0 \end{bmatrix}}_{\underline{c}^{\text{T}}(\underline{\theta}_{\text{free}})} \underline{x}(t) + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{d} u(t) + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{p}.\end{aligned} \tag{3}$$

For the sake of completeness, the linear case relations are stated as

$$\begin{aligned}\theta_{\text{free},1} &= -\frac{C}{M}, \quad \theta_{\text{free},2} = -\frac{D}{M}, \\ \theta_{\text{free},3} &= 0, \quad \theta_{\text{free},4} = -\frac{1}{M}.\end{aligned} \tag{4}$$

Here, $C = \text{const.}$ is the stiffness of a hypothetical linear spring.

[1] A PT_2 system is a second order transfer function without zeros.

4 Parameter Extraction

The final step is the interpretation of the model parameters in a physical manner, compare Fig. 1. Regarding the nonlinear equation, these are the body's mass, damping ratio and stiffness. The parameter extraction is able to deliver the spring force $\hat{F}_{\mathrm{S}}(y)$ as a function dependent on the position y. Since we modeled with local affine functions, we find $\hat{F}_{\mathrm{S}}(y)$ as the weighted sum of n_{LM} local affine stiffness functions $\hat{F}_{\mathrm{affine},i}(y)$, as

$$\begin{aligned}\hat{F}_{\mathrm{S}}(y) &= \sum_{i=1}^{n_{\mathrm{LM}}} \hat{F}_{\mathrm{affine},i}(y)\Phi_i(y)\\ &= \sum_{i=1}^{n_{\mathrm{LM}}} (\hat{C}_{\mathrm{lin},i}y + \hat{C}_{\mathrm{off},i})\Phi_i(y).\end{aligned} \tag{5}$$

Here, $\hat{C}_{\mathrm{lin},i}$ is the linear stiffness and $\hat{C}_{\mathrm{off},i}$ the offset of the i-th local stiffness model. With (3), we can extract $\hat{C}_{\mathrm{lin},i}$ and $\hat{C}_{\mathrm{off},i}$ from the estimated model parameters

$$\hat{C}_{\mathrm{lin},i} = -\frac{\hat{\theta}_{1,i}}{\hat{\theta}_4}, \quad \hat{C}_{\mathrm{off},i} = -\hat{\theta}_{3,i}. \tag{6}$$

Alternatively, the function $\hat{\theta}_1(y)$ which describes the variation of the parameter $\hat{\theta}_1$ with $y(t)$ can be constructed from (5) and (6) as weighted sum,

$$\hat{\theta}_1(y) = \sum_{i=1}^{n_{\mathrm{LM}}} \hat{\theta}_{1,i}\Phi_i(y). \tag{7}$$

The left side of Fig. 2 visualizes the extracted validity functions and $\hat{\theta}_1(y)$. Additionally, on the right is the true spring curve plotted, which has been fitted accurately by the LMN.

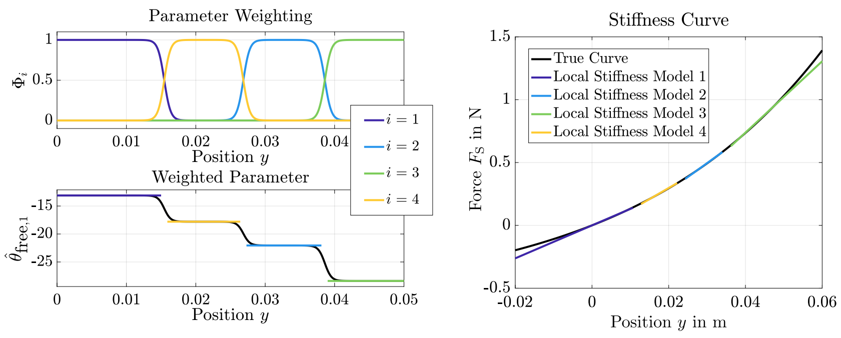


Figure 2: Local Linear Stiffness

5 Conclusion

This contribution emphasized that a nonlinear data-driven state space model with physically inspired structure is able to combine interpretability with high performance. Furthermore, computational resources can be preserved compared to an unstructured black-box model because the gray-box model contains less parameters that have to be optimized.

We are able to demonstrate our apporach on a widely known but simple example process. Next, our gray-box method shall be expanded on more complex processes like the Bouc-Wen hysteresis benchmark.

References

[1] O. Nelles. “Nonlinear System Identification: From Classical Approaches to Neural Networks, Fuzzy Models, and Gaussian Processes”. Springer International Publishing. 2020.

[2] T. McKelvey, H. Akcay, L. Ljung. “Subspace-based multivariable system identification from frequency response data”. In: *IEEE Transactions On Automatic Control* (**41**, 960-979). 1996.

[3] M. Schüssler. “Machine learning with Nonlinear State Space Models”. In: *Schriftenreihe Der Arbeitsgruppe Mess- Und Regelungstechnik - Mechatronik, Department Maschinenbau*. 2022.

[4] G. Mercère, O. Prot, J. Ramos. “Identification of Parameterized Gray-Box State-Space Systems: From a Black-Box Linear Time-Invariant Representation to a Structured One”. In: *IEEE Transactions On Automatic Control*. (**59**, 2873-2885). 2014.

[5] M. Schüssler, T. Münker, O. Nelles. “Deep Recurrent Neural Networks for Nonlinear System Identification”. 2019.

[6] L. Ljung. “System Identification: Theory for the User.” Prentice-Hall. 1986.

[7] J. Nocedal, S. Wright. “Numerical Optimization”. Springer. 2006.

[8] B. Friedland “Control System Design: An Introduction to State-Space Methods”. Dover Publications. 2012.

[9] P. Parrilo, L. Ljung. “Initialization of Physical Parameter Estimates”. In: *IFAC Proceedings Volumes*. (**36**, 1483-1488). 2003.

Nonlinear Modeling and Control of 3-DOF Gyroscope Actuated Furuta Pendulum by Takagi-Sugeno Fuzzy Systems

Johannes Brunner[1], Nis Keilhauer[1], Vincent Vellguth[1], Heide Brandtstädter[1], Horst Schulte[1]

[1] HTW-Berlin University of Applied Sciences
E-Mail: brunner@htw-berlin.de, keilha@htw-berlin.de, vellguth@htw-berlin.de, h.brandtstaedter@htw-berlin.de, schulte@htw-berlin.de

Abstract

This paper presents the derivation of the equations of motion of a 3-DOF gyroscope with a pendulum attachment through the Euler-Lagrange approach, followed by a conversion into a Takagi-Sugeno Fuzzy Model. First, suitable coordinate frames and generalized coordinates are defined, followed by the definition of the kinetic energy of each body frame of the gyroscope. Next, the kinetic and potential energy of the pendulum attachment is described. The derived equations of motion are then validated by simulation and compared to the behavior of a testbed system (see Figure 1). The conversion to the Takagi-Sugeno fuzzy model is done by a weighted combination of locally valid linear models. A set of adequate premise variables and membership functions represent the nonlinear system. Finally, a controller synthesis through parallel distributed compensation and LMIs satisfying local quadratic Lyapunov functions is conducted and validated by simulation.

1 Introduction

The aim of this paper is to derive and validate the equations of motion of a "Control Moment Gyroscope" (CMG) with 3 degrees of freedom, extended by

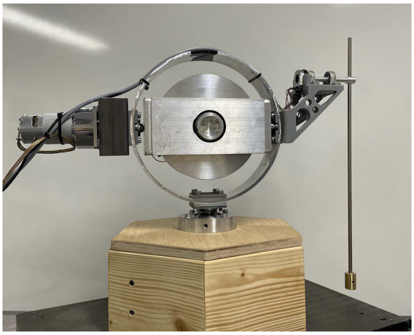


Figure 1: Testbed system

a pendulum attachement, similar to the testbed used in [1]. For the derivation, the modeling approaches of the CMG from [2] and the modeling approach of the Furuta pendulum in [3] are combined. The derived equations of motion are augmented with friction terms and validated using measurement data from the real testbed. Subsequently, the developed equations of motion are transformed into a Takagi-Sugeno formulation, and a Parallel Distributed Compensation (PDC) controller is derived using Linear Matrix Inequalitys (LMIs) to enforce closed-loop-dynamic constraints.

2 Methods

2.1 Nomenclature

The Notation in this Paper is as follows. Vectors are written italic with an underline $\underline{x}$, Matrices $\mathbf{A}$ bold and scalars s italic. Furthermore, $\prec$ and $\succ$ indicate negative and positive definiteness, respectively. $\mathbf{E}$ indicates the identity Matrix.

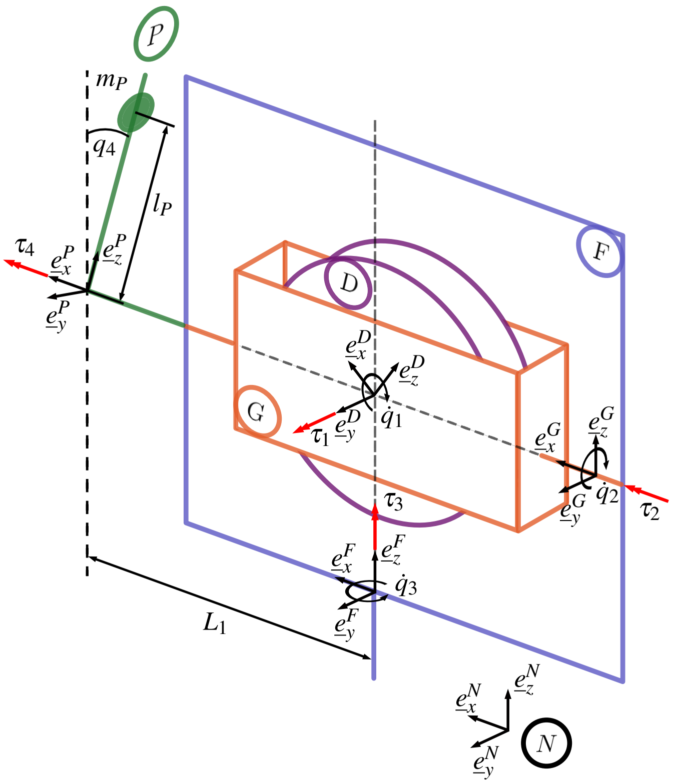


Figure 2: System sketch of the CMG Furuta Pendulum with 3 Degrees of freedom, the body frames, D, G, F, P and the reference frame N. For clarity the origins of the coordinate systems (except the Pendulum) are depicted with an offset.

2.2 Modelling

The CMG pendulum system in Figure 2 is described using five body frames, namely Disk (D), Gimbal (G), Frame (F), Pendulum (P), and the fixed reference frame (N). The coordinate systems are defined using the normal vectors $\underline{e}_i^j$ with $i = x, y, z$ as axes and $j = D, G, F, P, N$ as the associated body frame. The reference system N is chosen to be stationary, so that relative movements of the other reference systems in N constitute to an absolute velocity. The torques acting on the body frames are denoted as τ_n with $n = 1, 2, 3, 4$ along the corresponding rotational axis following the right hand rule.

The modelling of the CMG part of the overall system follow [2]. Therefore the generalized coordinates are chosen as:

$$\begin{aligned} \underline{q} &:= [q_1,\, q_2,\, q_3,\, q_4]^\top \\ \underline{\dot{q}} &:= [\dot{q}_1,\, \dot{q}_2,\, \dot{q}_3,\, \dot{q}_4]^\top \\ \underline{\ddot{q}} &:= [\ddot{q}_1,\, \ddot{q}_2,\, \ddot{q}_3,\, \ddot{q}_4]^\top , \end{aligned} \tag{1}$$

whereas q_1 represents the disk position about $\underline{e}_y^D$, q_2 represents the gimbal position about $\underline{e}_x^G$, q_3 represents the frame position about $\underline{e}_z^F$, and q_4 represents the pendulum position about $\underline{e}_x^P$. The controllable inputs of the system are the torques τ_1 and τ_2. Disturbance torques are τ_3 and τ_4. All torques are combined into the vector

$$\underline{\tau} := [\tau_1,\, \tau_2,\, \tau_3,\, \tau_4]^\top \tag{2}$$

The positive rotation directions of all coordinate systems follow the right-hand rule. Friction terms will be added at a later time. The center mass of the coordinate systems describing the CMG are assumed to be at the center of the disk. The mass center of the pendulum, denoted as m_P, has an effective pendulum length of l_P. The distance from the pendulum coordinate system's rotation axis to the frame's rotation axis is L_1. The moments of inertia of the bodies around different coordinate axes are represented in tensor form to calculate the kinetic energies for the Lagrange function.

$$\begin{aligned} \mathbf{I}_D^D &= \begin{bmatrix} J_{Dxx} & 0 & 0 \\ 0 & J_{Dyy} & 0 \\ 0 & 0 & J_{Dzz} \end{bmatrix}, \; \mathbf{I}_G^G = \begin{bmatrix} J_{Gxx} & 0 & 0 \\ 0 & J_{Gyy} & 0 \\ 0 & 0 & J_{Gzz} \end{bmatrix} \\ \mathbf{I}_F^F &= \begin{bmatrix} J_{Fxx} & 0 & 0 \\ 0 & J_{Fyy} & 0 \\ 0 & 0 & J_{Fzz} \end{bmatrix}, \; \mathbf{I}_P^P = \begin{bmatrix} J_{Pxx} & 0 & 0 \\ 0 & J_{Pyy} & 0 \\ 0 & 0 & J_{Pzz} \end{bmatrix} \end{aligned} \tag{3}$$

For the later description of the rotational positions of the bodies relative to each other rotation matrices are used. By combining the rotation matrices it is possible to represent all positions of the bodies in relation to the reference coordinate system. The rotation matrices $\mathbf{R}_i$ around the coordinate axes $i = x, y, z$ of the reference coordinate system are defined as follows:

$$
\begin{aligned}
\mathbf{R}_x(q_j) &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(q_j) & -\sin(q_j) \\ 0 & \sin(q_j) & \cos(q_j) \end{bmatrix} \\
\mathbf{R}_y(q_j) &= \begin{bmatrix} \cos(q_j) & 0 & \sin(q_j) \\ 0 & 1 & 0 \\ -\sin(q_j) & 0 & \cos(q_j) \end{bmatrix} \\
\mathbf{R}_z(q_j) &= \begin{bmatrix} \cos(q_j) & -\sin(q_j) & 0 \\ \sin(q_j) & \cos(q_j) & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{4}
$$

Here, the notation $\mathbf{R}^i_j$ is introduced, which describes the rotation of the coordinate system j with respect to i.

$$
\begin{aligned}
\mathbf{R}^G_D &= \mathbf{R}_y(q_1) \\
\mathbf{R}^F_G &= \mathbf{R}_x(q_2) \\
\mathbf{R}^N_F &= \mathbf{R}_z(q_3) \\
\mathbf{R}^F_P &= \mathbf{R}_x(q_4)
\end{aligned}
\tag{5}
$$

By multiplying the rotation matrices accordingly, it is possible to describe the rotation of any desired coordinate system of the CMG pendulum relative to the reference coordinate system.

$$\mathbf{R}^N_D = \mathbf{R}^N_F \mathbf{R}^F_G \mathbf{R}^G_D \tag{6}$$

With the rotation matrices, it is possible to represent the angular velocities of the different bodies in the reference coordinate system.

$$\underline{\omega}^N_{N,F} = [0^{3\times1}\ 0^{3\times1}\ \underline{Z}^N_N\ 0^{3\times1}]\underline{\dot{q}} \tag{7}$$

$$\underline{\omega}^N_{N,G} = [0^{3\times1}\ \underline{X}^N_F\ \underline{Z}^N_N\ 0^{3\times1}]\underline{\dot{q}} \tag{8}$$

$$\underline{\omega}^N_{N,D} = [\underline{Y}^N_G\ \underline{X}^N_F\ \underline{Z}^N_N\ 0^{3\times1}]\underline{\dot{q}} \tag{9}$$

with the rotational descriptions

$$\begin{aligned} \underline{Z}_N^N &= \underline{e}_z^N \\ \underline{X}_F^N &= \mathbf{R}_F^N \underline{e}_x^N \,. \\ \underline{Y}_G^N &= \mathbf{R}_G^N \underline{e}_y^N \end{aligned} \tag{10}$$

The matrices describing the rotations in (7), (8), and (9) are hereafter denoted as $\mathbf{J}_{N,k}^N$. Here examplary for the disk.

$$\mathbf{J}_{N,D}^N = [\underline{Y}_G^N \, \underline{X}_F^N \, \underline{Z}_N^N \, 0^{3\times 1}] \tag{11}$$

Additionally, the set of coordinate systems $\mathscr{S} = D, G, F$ is defined. From this, the kinetic energy $T_{\text{CMG}}(\underline{q}, \underline{\dot{q}})$ of the CMG part of the overall system can be defined as:

$$T_{\text{CMG}}(\underline{q}, \underline{\dot{q}}) = \frac{1}{2} \sum_{k \in \mathscr{S}} \underline{\dot{q}}^\top \left[\left(\mathbf{J}_{N,k}^N\right)^\top \mathbf{R}_k^N \mathbf{I}_k^k \left(\mathbf{R}_k^N\right)^\top \mathbf{J}_{N,k}^N \right] \underline{\dot{q}} \tag{12}$$

The description of the kinetic energy of the pendulum follow [3]. The linear velocity $\underline{v}_\text{P}$ and angular velocity $\underline{\omega}_\text{P}$ of the Pendulum are described separately and then combined. First the angular velocity of the pendulum arm is determined as follows:

$$\underline{\omega}_P = \mathbf{R}_P^F \cdot \begin{bmatrix} 0 \\ 0 \\ \dot{q}_3 \end{bmatrix} + \begin{bmatrix} \dot{q}_4 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \dot{q}_4 \\ -\dot{q}_3 \sin(q_4) \\ \dot{q}_3 \cos(q_4) \end{bmatrix} \tag{13}$$

The linear velocity $\underline{v}_\text{P}$ of the pendulum arm is composed of the translational velocity of the pendulum joint

$$\underline{v}_2 = \mathbf{R}_P^F \left(\underline{\omega}_{N,F}^N \times [L_1,\, 0,\, 0]^\top \right) = \begin{bmatrix} 0 \\ \cos(q_4) L_1 \dot{q}_3 \\ \sin(q_4) L_1 \dot{q}_3 \end{bmatrix} \tag{14}$$

and the translational velocity of the pendulum center mass

$$\underline{v}_{b,m} = \underline{\omega}_P \times [\,0,\ 0,\ l_P]^\top = \begin{bmatrix} -\sin(q_4) l_P \dot{q}_3 \\ -l_P \dot{q}_4 \\ 0 \end{bmatrix} \tag{15}$$

which results in

$$\underline{v}_P = \underline{v}_2 + \underline{v}_{b,m} = \begin{bmatrix} -\sin(q_4) l_P \dot{q}_3 \\ \cos(q_4) L_1 \dot{q}_3 - l_P \dot{q}_4 \\ \sin(q_4) L_1 \dot{q}_3 \end{bmatrix}. \tag{16}$$

The kinetic energy of the pendulum is then given by

$$T_P(\underline{q}, \underline{\dot{q}}) = \frac{1}{2}\left(\underline{v}_P^\top (m_P \mathbf{E}^{3\times 3}) \underline{v}_P + (\underline{\omega}_P)^\top \mathbf{I}_P^P \underline{\omega}_P \right), \tag{17}$$

where $\mathbf{E}$ represents the identity matrix. Since the pendulum has potential energy, and the reference height is chosen at $q_4 = \pi$ (hanging pendulum), the potential energy of the Pendulum V_P is

$$V_P(\underline{q}) = g \cdot m_P \cdot l_P (1 + \cos(q_4)) \tag{18}$$

With the kinetic and potential energy of all bodies, the Lagrangian can be formulated as

$$\mathscr{L}(\underline{q}, \underline{\dot{q}}) = T_{CMG}(\underline{q}, \underline{\dot{q}}) + T_P(\underline{q}, \underline{\dot{q}}) - V_P(\underline{q}). \tag{19}$$

Now, the equations of motion can be derived using the Euler-Lagrange formalism:

$$\frac{d}{dt}\left(\frac{\partial \mathscr{L}(\underline{q}, \underline{\dot{q}})}{\partial \underline{\dot{q}}} \right) - \frac{\partial \mathscr{L}(\underline{q}, \underline{\dot{q}})}{\partial \underline{q}} = \underline{\tau} \tag{20}$$

Due to the total number of bodies, four moving equations are derived describing the acceleration of the different bodies.

For the formulation of the Lagrangian from (19), the Symbolic Toolbox in MATLAB is used, and the Euler-Lagrange equations were derived according to (20) using the community function "EulerLagrange".

2.3 Takagi-Sugeno System

The TS-Model is constructed via local linearization; therefore, consider the nonlinear system

$$\begin{aligned} \dot{\underline{x}} &= \underline{f}(\underline{x},\underline{u}) \\ \underline{y} &= \underline{h}(\underline{x}) \end{aligned} \tag{21}$$

Through the first-order Taylor Series Expansion, we obtain the matrices

$$\mathbf{A}_i = \left.\frac{\partial \underline{f}}{\partial \underline{x}}\right|_c \qquad \mathbf{B}_i = \left.\frac{\partial \underline{f}}{\partial \underline{u}}\right|_c \qquad \mathbf{C}_i = \left.\frac{\partial \underline{h}}{\partial \underline{x}}\right|_c , \tag{22}$$

where c describes the linearization point. We then formulate the TS-System, where all linear submodels are blended into each other by the membership functions $h_i(\underline{z})$

$$\begin{aligned} \dot{\underline{x}} &= \sum_{i=1}^{N_r} h_i(\underline{z})\left(\mathbf{A}_i\underline{x} + \mathbf{B}_i\underline{u} + \underline{a}_i\right) \\ \underline{y} &= \sum_{i=1}^{N_r} h_i(\underline{z})\mathbf{C}_i\underline{x} + \underline{c}_i \end{aligned} \tag{23}$$

where $\underline{z}$ denotes the vector of premise variables that determine which model is active at any given time, N_r is the number of linearization points [4]. The membership functions $h_i(\underline{z})$ are chosen to be triangular and fulfill the properties $1 \geq h_i(\underline{z}) \geq 0$ and $\sum_{i=1}^{N_r} h_i(\underline{z}) = 1$. Each submodel i represents the nonlinear system at the linearization point c to 100%, i.e., if $h_1 = 1$, only the submodel $\dot{\underline{x}} = \mathbf{A}_1\underline{x} + \mathbf{B}_1\underline{u} + \underline{a}_1$ would be active and represent the current dynamics fully [5]. The affine terms $\underline{a}_i$ and $\underline{c}_i$ of non-equilibrium linearization points are computed as follows:

$$\begin{aligned} \underline{a}_i =& \underline{f}(\underline{x}_c,\underline{u}_c) - \mathbf{A}_i\underline{x}_c - \mathbf{B}_i\underline{u}_c \\ \underline{c}_i =& \underline{h}(\underline{x}_c) - \mathbf{C}_i\underline{x}_c \end{aligned} \tag{24}$$

It is noted here that, for simplicity reasons, the affine terms are neglected for controller synthesis.

2.4 Parallel Distributed Compensation

The fuzzy controller formulation is done in a similar manner as the TS-Model by utilizing local controller gains for each linearization point c and blending these together with the membership functions $h_i(\underline{z})$ of the model as the premise variables $\underline{z}$ change [6].

$$\underline{u} = -\sum_{i=1}^{N_r} h_i(\underline{z})\mathbf{K}_i\underline{x} \tag{25}$$

Augmenting (25) into (23) and neglecting the affine term $\underline{a}_i$ yields the closed-loop TS-System of the form:

$$\dot{\underline{x}} = \sum_{i=1}^{N_r}\sum_{j=1}^{N_r} h_i(\underline{z})h_j(\underline{z})\{\mathbf{A}_i - \mathbf{B}_i\mathbf{K}_j\}\underline{x} \tag{26}$$

Which can be written in compact form:

$$\dot{\underline{x}} = \sum_{i=1}^{N_r}\sum_{j=1}^{N_r} h_i(\underline{z})h_j(\underline{z})\mathbf{G}_{ij}\underline{x} \tag{27}$$

2.5 Controller Synthesis via LMIs

For controller synthesis, the local quadratic Lyapunov functions are applied:

$$V_i(\underline{x}) = \underline{x}^\top \mathbf{P}_i\underline{x} \tag{28}$$

$$\dot{V}_i(\underline{x}) = \dot{\underline{x}}^\top \mathbf{P}_i\underline{x} + \underline{x}^\top \mathbf{P}_i\dot{\underline{x}} \tag{29}$$

where $\mathbf{P}_i$ is a symmetric, positive definite matrix. To find controller gains, we augment (29) with (27) and define $\mathbf{P}_i = \mathbf{X}_i^{-1}$ and $\mathbf{K}_j = \mathbf{M}_i\mathbf{X}_i^{-1}$ as for each local controller j the LMIs are solved independently thus $i = j$ for controller synthesis. The requirements for asymptotic lyapunov stability:

$$\begin{aligned} V_i(\underline{x}) &> 0, \ \forall \underline{x} \neq 0 \\ V_i(\underline{x}) &= 0, \ \underline{x} = 0 \end{aligned} \tag{30}$$

$$\dot{V}_i(\underline{x}) < 0, \ \forall \underline{x} \neq 0 \tag{31}$$

can then be expressed in LMIs which constrain convex sets in the complex plane [7]. The so-called $\mathscr{D}$-Region LMI constraint then results in the formulation:

Find a matrix $\mathbf{X}_i = \mathbf{X}_i^\top \succ 0$ and $\mathbf{M}_i$ for a desired $\alpha \geq 0$, $r > \alpha$, and $\theta > 0$ under the constraint:

$$\mathbf{X}_i\mathbf{A}_i^\top + \mathbf{A}_i\mathbf{X}_i - \mathbf{M}_i^\top\mathbf{B}_i^\top - \mathbf{B}_i\mathbf{M}_i + 2\alpha\mathbf{X}_i \prec 0, \tag{32}$$

$$\begin{bmatrix} -r\mathbf{X}_i & \mathbf{A}_i\mathbf{X}_i - \mathbf{B}_i\mathbf{M}_i \\ \mathbf{X}_i\mathbf{A}_i^\top - \mathbf{M}_i^\top\mathbf{B}_i^\top & -r\mathbf{X}_i \end{bmatrix} \prec 0, \tag{33}$$

$$\begin{bmatrix} \sin\theta(\mathbf{X}_i\mathbf{A}_i^\top + \mathbf{A}_i\mathbf{X}_i - \mathbf{M}_i^\top\mathbf{B}_i^\top - \mathbf{B}_i\mathbf{M}_i) & \cos\theta(\mathbf{A}_i\mathbf{X}_i - \mathbf{B}_i\mathbf{M}_i - \mathbf{X}_i\mathbf{A}_i^\top + \mathbf{M}_i^\top\mathbf{B}_i^\top) \\ \cos\theta(\mathbf{X}_i\mathbf{A}_i^\top - \mathbf{M}_i^\top\mathbf{B}_i^\top - \mathbf{A}_i\mathbf{X}_i + \mathbf{B}_i\mathbf{M}_i) & \sin\theta(\mathbf{A}_i\mathbf{X}_i - \mathbf{B}_i\mathbf{M}_i + \mathbf{X}_i\mathbf{A}_i^\top - \mathbf{M}_i^\top\mathbf{B}_i^\top) \end{bmatrix} \prec 0 \tag{34}$$

where α denotes the minimum required decay rate, r defines the radius of a half circle toward the complex left-hand open plane with the origin in the center of the complex plane, and θ defines the angle between the real axis and a cone restriction toward the complex open left-hand plane [8].

As the $\mathscr{D}$-Region constraint might not be ideal for the coupled dynamics of the system (all states underlie the same decay constraints α), an optimal controller design is pursued as well. The optimal controller LMI approach from [6] is used, utilizing the performance function:

$$J = \int_0^\infty \{\underline{y}^\top(t)\mathbf{W}\underline{y}(t) + \underline{u}^\top(t)\mathbf{R}\underline{u}(t)\}dt \tag{35}$$

The cost function (35) results in the minimization problem $J < \underline{x}^\top(0)\mathbf{P}_i\underline{x}(0) < \lambda$ with the following LMI constraints:

$$\min_{\mathbf{X}_i,\mathbf{M}_i,\mathbf{Y}_{0,i}} \lambda$$

subject to

$$\mathbf{X}_i \succ 0, \qquad \mathbf{Y}_{0,i} \succeq 0, \tag{36}$$

$$\begin{bmatrix} \lambda & \underline{x}^\top(0) \\ \underline{x}(0) & \mathbf{X}_i \end{bmatrix} \succ 0, \tag{37}$$

$$\hat{\mathbf{U}}_{ii} + (s-1)\mathbf{Y}_{3,i} \prec 0, \tag{38}$$

where $s > 1$

$$\hat{\mathbf{U}}_{ii} = \begin{bmatrix} \begin{pmatrix} \mathbf{X}_i\mathbf{A}_i^\top + \mathbf{A}_i\mathbf{X}_i \\ -\mathbf{B}_i\mathbf{M}_i - \mathbf{M}_i^\top\mathbf{B}_i^\top \end{pmatrix} & \mathbf{X}_i\mathbf{C}_i^\top & -\mathbf{M}_i^\top \\ \mathbf{C}_i\mathbf{X}_i & -\mathbf{W}^{-1} & \mathbf{0} \\ -\mathbf{M}_i & \mathbf{0} & -\mathbf{R}^{-1} \end{bmatrix} \tag{39}$$

$$\mathbf{Y}_{3,i} = \text{block-diag}(\mathbf{Y}_{0,i}, \mathbf{0}, \mathbf{0})$$

The optimization problem above is in a reduced form from [6], as stability is only demanded for the local models and no combination of $i \neq j$ as well as the relaxed stability condition where s is the maximum number of submodels that are active at the same time. The weighting matrices $\mathbf{W}$ and $\mathbf{R}$ are chosen constant and do not deviate for different local constraints.

For handling the LMIs, the YALMIP interface together with the solver MOSEK is used [9],[10].

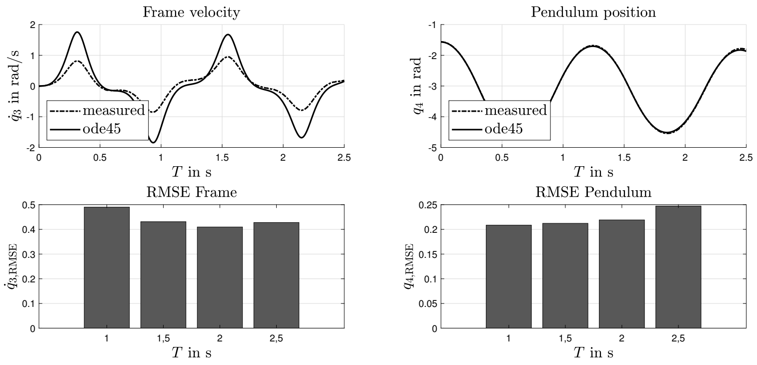


Figure 3: State comparison between the nonlinear CMG Furuta Pendulum equations evaluated using MATLAB's ode45 solver and measurements of the testbed system on the top, and RMSE errors at the bottom with $\dot{q}_1 = 0$ rad/s.

3 Results

3.1 Model Validation

The derived nonlinear equations of motion for the CMG Furuta Pendulum from the Euler-Lagrange approach (20) are simulated using the parameters provided in Table 1. The simulations are carried out using the MATLAB ode45 solver and are compared to measured values obtained from the testbed system. The comparison is evaluated through the Root Mean Squared Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y(k) - \hat{y}(k))^2} \tag{40}$$

for time intervals of 1, 1.5, 2, and 2.5 seconds. Two different sets of measurements are taken while the pendulum is falling from its upright position. For the first measurement, the disk is not spinning ($\dot{q}_1 = 0$ rad/s), and the states $\underline{x}_{\mathrm{meas},1} = [\dot{q}_3, q_4]^\top$ are measured, as depicted in Figure 3. At this point, the system essentially represents a Furuta Pendulum with the center of mass of the cantilever arm at the center of rotation of the frame.

Table 1: Systen Parameters of the CMG Furuta Pendulum testbed system

Variable	Value	Unit
J_{Dxx}	0.0027	kg m^2
J_{Dyy}	0.0048	kg m^2
J_{Dzz}	0.0027	kg m^2
J_{Gxx}	0.0014	kg m^2
J_{Gyy}	0.005	kg m^2
J_{Gzz}	0.005	kg m^2
J_{Fxx}	0	kg m^2
J_{Fyy}	0	kg m^2
J_{Fzz}	0.0414	kg m^2
J_{Pxx}	0.003	kg m^2
J_{Pyy}	0.003	kg m^2
J_{Pzz}	0	kg m^2
μ_1	0.8	kg m s^{-1}
μ_2	7.2	kg m s^{-1}
μ_3	0.7	kg m s^{-1}
μ_4	0.135	kg m s^{-1}
L_1	0.254	m
l_P	0.246	m
m_P	0.216	kg

The second measurement is taken with the disk spinning at its maximum speed, $\dot{q}_{1,\text{max}} = 28.8$ rad/s. In this case, the states $\underline{x}_{\text{meas},2} = [q_2, \dot{q}_3, q_4]^\top$ are measured to observe the effect of precession on the gimbal states and vice versa, as shown in Figure 4.

In the first measurement, the pendulum angle q_4 shows an RMSE of around 0.2 up to 2 seconds, whereas the frame angular velocity $\dot{q}_3$ exhibits an RMSE of around 0.4. The frame velocity measurements particularly deviate from the simulation at the peaks, where the frame is accelerated due to the falling pendulum. The second measurement shows that the measured and simulated states strongly differ from each other. It is presumed that this discrepancy is due to the neglect of the motor actuating the disk, which is attached along the y-axis of the gimbal coordinate frame. The motor has a gearbox, increasing its weight and potentially offsetting the center of mass of the gimbal. Therefore,

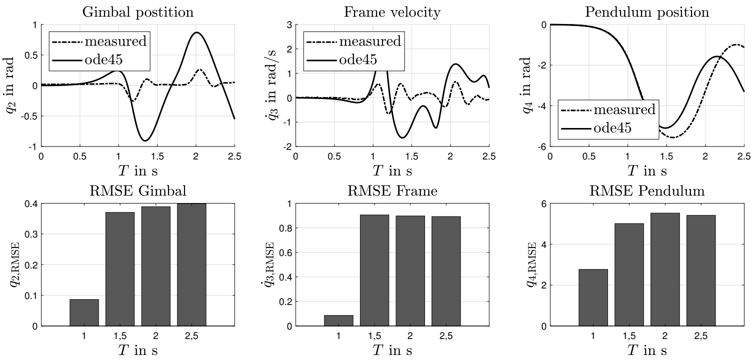


Figure 4: State comparison between the nonlinear CMG Furuta Pendulum equations evaluated using MATLAB's ode45 solver and measurements of the testbed system on the top, and RMSE errors at the bottom with $\dot{q}_1 = 28.8$ rad/s.

the assumption that all CMG center masses are located in the center of the CMG is not valid for the measured values.

Table 1 displays the system parameters for the testbed system, which were obtained through the CAD program Inventor, using an internal Finite Element Method (FEM) to calculate the inertia of the components.

3.2 Controller Design

The state feedback control structure is depicted in Figure 5. The controllable inputs of the system are $u_1 = \tau_1$ and $u_2 = \tau_2$. The torque τ_1 actuating the disk is not controlled and is kept at a constant value to ensure the disk spins at its highest angular velocity. Input u_2 is controlled via the PDC control law with controller parameters synthesized through the LMI formulations given in Section: Controller Synthesis. As premise variables, the current gimbal and pendulum angles $\underline{z} = [q_2, q_4]^\top$ are used. The disk position and velocity are neglected for the state feedback controller, leaving the states $\underline{x}_{\text{cntrl}} = [q_2, \dot{q}_2, q_3, \dot{q}_3, q_4, \dot{q}_4]^\top$ to be controlled via the PDC.

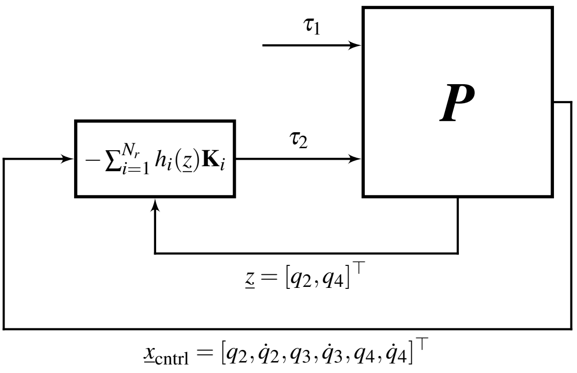

τ_1
$\boldsymbol{P}$
τ_2
$-\sum_{i=1}^{N_r} h_i(\underline{z})\mathbf{K}_i$
$\underline{z} = [q_2, q_4]^\top$
$\underline{x}_{\text{cntrl}} = [q_2, \dot{q}_2, q_3, \dot{q}_3, q_4, \dot{q}_4]^\top$

Figure 5: Basic Control Structure, $\boldsymbol{P}$ resembles the Plant, $\underline{z}$ the premise vector, τ_1 and τ_2 the inputs u_1 and u_2 into the system respectively and $\underline{x}_{\text{cntrl}}$ the controlled system states

The linearization points are chosen as $q_2 \in [-1.2,\ 1.2]$ rad for the gimbal angle and $q_4 \in [-0.5236,\ 0.5236]$ rad for the pendulum angle. The disk speed is set to $\dot{q}_1 = 28.8$ rad/s for Scenario 1 (Low-Speed Disk) and $\dot{q}_1 = 45$ rad/s for Scenario 2 (High-Speed Disk).

Scenario 1 yields the following submodels of the TS-System:

$$\mathbf{A}_{1,4,\text{cntrl}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -7.2 & 0 & 12.41 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0.115 & -0.89 & 0 & -0.7 & 2.47 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0.081 & -0.65 & 0 & 0 & 28.41 & -0.14 \end{bmatrix} \tag{41}$$

$$\mathbf{A}_{2,3,\text{cntrl}} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -7.2 & 0 & 12.41 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -0.115 & -0.89 & 0 & -0.7 & 2.47 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -0.081 & -0.65 & 0 & 0 & 28.41 & -0.14 \end{bmatrix} \tag{42}$$

Table 2: $\mathscr{D}$-Region constraints and Optimal Controller weighting for Low-Speed Disk (Scenario 1) and High-Speed Disk (Scenario 2) controller synthesis.

Scenario	$\mathscr{D}$-Region	Optimal Controller
1	$\theta = 0.5236$ rad $\alpha = 2$ $r = 10$	$\mathbf{W} = \mathrm{diag}(0.1, 0.001, 0.1, 0.0001, 1, 0.001)$ $\mathbf{R} = 0.3$ $\lambda = 0.2$ $\underline{x}(0) = [0, 0, 0, 0, 0.27, 0]^\top$
2	$\theta = 0.5236$ rad $\alpha = 2$ $r = 10$	$\mathbf{W} = \mathrm{diag}(0.1, 0.001, 0.1, 0.0001, 1, 0.001)$ $\mathbf{R} = 0.3$ $\lambda = 0.1$ $\underline{x}(0) = [0, 0, 0, 0, 0.37, 0]^\top$

$$\mathbf{B}_{1,3,\mathrm{cntrl}} = \begin{bmatrix} 0 & 0 \\ 0 & 264.61 \\ 0 & 0 \\ 16.6 & 0 \\ 0 & 0 \\ 12.1 & 0 \end{bmatrix}, \quad \mathbf{B}_{2,4,\mathrm{cntrl}} = \begin{bmatrix} 0 & 0 \\ 0 & 264.61 \\ 0 & 0 \\ -16.6 & 0 \\ 0 & 0 \\ -12.1 & 0 \end{bmatrix} \tag{43}$$

The parameters for controller synthesis in the different scenarios are listed in Table 2. The input $u_2 = \tau_2$ is limited to 2.5 Nm, which is the maximum torque of the actuating motor for the gimbal.

For scenario 1 we obtain the different gain sets for the $\mathscr{D}$-Region controller

$$\begin{aligned} \mathbf{K}_{1,4,\mathscr{D}} &= [-7.44, 0.075, -6.01, -8.15, -14.47, -1.99] \\ \mathbf{K}_{2,3,\mathscr{D}} &= [7.66, 0.076, 6.41, 9.8, -14.32, -3.49] \end{aligned} \tag{44}$$

and the optimal controller:

$$\begin{aligned} \mathbf{K}_{1,4,\mathrm{opt}} &= [-0.098, 0.13, -0.004, 0.102, -15.48, -2.87] \\ \mathbf{K}_{2,3,\mathrm{opt}} &= [-0.087, 0.15, 0.051, 0.17, -17.48, -3.32] \end{aligned} \tag{45}$$

Table 3: Initial conditions for Low Speed Disk (Scenario 1) and High Speed Disk (Scenario 2) simulation studies.

Scenario	$\mathscr{D}$-Region	Optimal Controller
1	$x_0 = [0, 28.8, 0, 0, 0, 0, 0.2, 0]^\top$	$x_0 = [0, 28.8, 0, 0, 0, 0, 0.27, 0]^\top$
2	$x_0 = [0, 45, 0, 0, 0, 0, 0.2, 0]^\top$	$x_0 = [0, 45, 0, 0, 0, 0, 0.37, 0]^\top$

3.3 Simulation Studies

In total four simulation studies are conducted to find the maximum initial angle q_4 of the pendulum the controller is able to return into the unstable equilibrium $q_4 = 0$ rad.

1. Disk spinning at $\dot{q}_1 = 28.8$ rad/s, $\mathscr{D}$-Region constraints
2. Disk spinning at $\dot{q}_1 = 28.8$ rad/s, Optimal controller design
3. Disk spinning at $\dot{q}_1 = 45$ rad/s, $\mathscr{D}$-Region constraints
4. Disk spinning at $\dot{q}_1 = 45$ rad/s, Optimal controller design

The simulation setups allow for comparison between the two different controller synthesis procedures. The increase in angular velocity of the Disk from $\dot{q}_1 = 28.8$ to $\dot{q}_1 = 45$ rad/s shows if control performance can be improved by increasing the dynamic of the actuator. As the actuating torque moving the Frame through precession is defined as

$$\tau_3 = \dot{q}_2 \cdot \dot{q}_1 \cdot J_{Dyy} \cdot \cos(q_2) \tag{46}$$

As the goal of the Simulations is to find the maximum angle q_4 the controller is able to recover the initial conditions for the different scenarios are listed in Table 3. The results show, that the maximum initial angle for the $\mathscr{D}$-Region controller is $q_4 = 0.2$ rad for Scenario 1 and Scenario 2. The optimal controller was able to recover the Pendulum from $q_4 = 0.27$ for Scenario 1 and $q_4 = 0.37$ rad for Scenario 2.

The simulation results are depicted in Figures 6, 7, and 8. Two main findings can be obtained. Firstly the simulations indicate a limited stability region

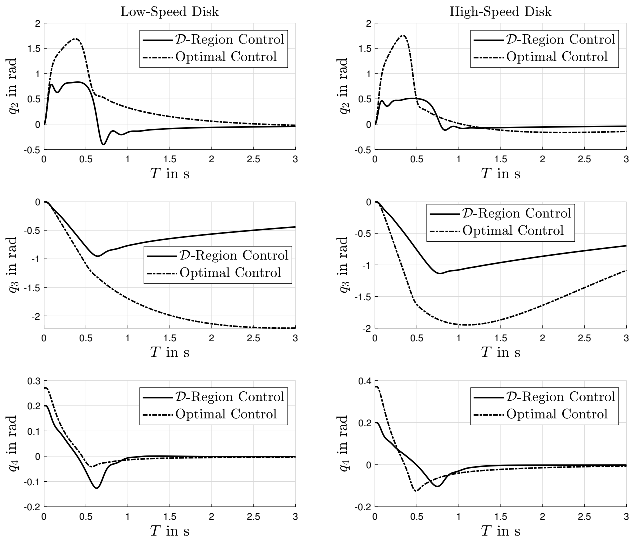


Figure 6: Gimbal, Frame, and Pendulum angles q_2, q_3 and q_4 respectively for Low-Speed Disk and High-Speed Disk scenarios as well as a comparison between $\mathscr{D}$-Region and Optimal Controller

independent of controller synthesis due to various factors. The torque τ_3 is limited by the velocity of the disk $\dot{q}_1$, and the maximum torque τ_2 enforced by the motor moving the gimbal limits $\dot{q}_2$ as seen in (46). Most notable is the $\cos(q_2)$ term from the precession in (46) if the gimbal is moved towards the angle $q_2 \to \pm\pi/2$ the effect on the frame through actuating the gimbal decreases significantly.

Secondly, the simulations indicate that the decay constraints imposed by the $\mathscr{D}$-Region have a negative impact on the maximum recoverable initial pendulum angle q_4. This is likely induced by the coupled dynamics of the system, as the body acting on the pendulum is the frame, which is actuated by the gimbal movement.

Increasing the Disk angular velocity to 45 rad/s for Scenario 2 also increases the angular momentum of the disk proportionally, therefore increasing τ_3 by actuating the Gimbal with τ_2.

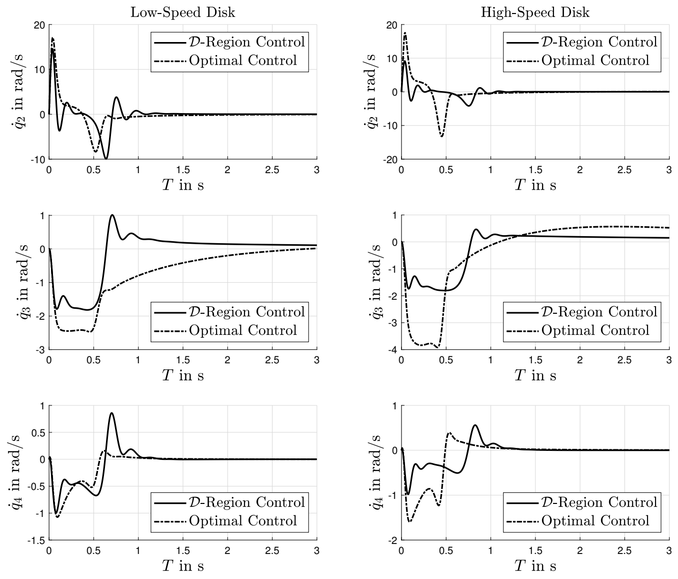


Figure 7: Gimbal, Frame and Pendulum angular velocities $\dot{q}_2$, $\dot{q}_3$ and $\dot{q}_4$ respectively for Low-Speed Disk and High-Speed Disk scenarios as well as a comparison between $\mathscr{D}$-Region and Optimal controller

Figure 8 displays the torque τ_2 demanded by the controller, which acts on the gimbal. It can be observed that at the beginning of the simulation, the recovery of the pendulum angle q_4 has the most significant impact on the demanded torque. The $\mathscr{D}$-Region controller and the optimal controller exhibit different behaviors when the pendulum angle is recovered and stabilized. This difference is due to the less conservative decay constraints for the optimal controller on the states, excluding the pendulum position, which is also evident in Figures 6 and 7.

4 Discussion

The results presented in this paper demonstrate the capability of a Takagi-Sugeno Parallel Distributed Compensation (PDC) fuzzy controller to stabilize a highly nonlinear Control Moment Gyroscope (CMG)-actuated Furuta

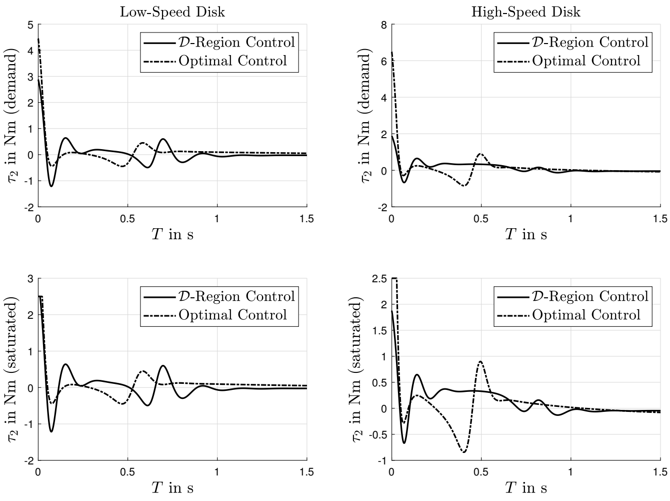


Figure 8: Demanded and saturated input torque $u_2 = \tau_2$ for Low Speed Disk and High Speed Disk scenarios as well as comparison between $\mathscr{D}$-Region and Optimal controller

Pendulum at an unstable equilibrium point. It is found that the dynamics of the torque generated through precession, in combination with the controller design, have an impact on the maximum initial pendulum angle the controller is able to recover to its unstable equilibrium. In an optimal controller design, where the closed-loop dynamics of each state can be weighted independently, the controller can exceed a common decay rate constraint through LMIs, especially when the actuator dynamics are high, and the system dynamics are strongly coupled, as in the testbed system. Shortcomings primarily lie in the identification of the system behavior when the disk is spinning. Therefore, an adjustment in the hardware and weight distribution of the gimbal might yield better results. Furthermore, the current testbed system has cables running off the motor that actuates the gimbal, as seen in Figure 1, introducing random friction terms due to the current cable positions. An adaptation to slip rings might be advantageous, eliminating random friction terms.

Furthermore, the current linearization points are chosen to include a wide range of angles the pendulum and gimbal can have during the recovery of the pendulum angle. Performance might be improved by increasing the number of linearization points.

It is noted, that the $\mathscr{D}$-Region constraints were not further optimized after showing acceptable performance therefore the $\mathscr{D}$-Region controller might show some room for improvement.

The angles the controllers are able to recover are quite small indicating a limitation of the dynamics of the system under current actuating forces. Therfore the current Hardware is subject to improvement to increase overall dynamics of the system.

5 Conclusion

The control of a testbed system similar to the one investigated in this paper is, to the knowledge of the authors, only conducted in [1] where an LPV approach is pursued. [1] does focus on the swing up of the system and does not include an investigation of the stabilizing controller for the unstable equilibrium. It is only stated, that the switching between the swing-up controller and stabilizing controller is executed at a Pendulum angle $|q_x| < 0.15$ rad and the LPV scheduling region ranges from $q_2 \in [-60°,\ 60°]$ and $\omega_1 = \dot{q}_1 \in [30\ \text{rad/s},\ 60\ \text{rad/s}]$. This paper presents an alternative control concept for the stabilizing controller through a Fuzzy TS approach and LMIs which in Scenario 2 achieved a stable recovery of the Pendulum from an initial angle of $q_4 = 0.37$ rad under ideal conditions and Single Input only actuating the Gimbal through the controller.

As the system parameters in [1] differ from the one in this work it is to be investigated, if the Fuzzy controller is able to show similar performance when adapting the system parameters to match the system in [1].

It is mentioned here, that for the system parameters [1] refers to [11] where the inertia Tensores might not be correctly given in comparison to the system sketch. [12] is dated later than [11] where the inertia Tensors are corrected. [11] and [12] refer to the manual of the testbed system ECP 750 (Educational Control Products) which is to the knowledge of the authors not accessible publicly.

6 Future Work

Future work could include an extension to a multi-input system where the torque τ_2 is actuated through the controller as well as conducted in [1]. The premise variables could be extended to incuding the current Disk velocity $\dot{q}_1$. Also the $\mathscr{D}$-Region and optimal controller constraints can be altered for each linearization point increasing flexibility for controller design. As the number of linearization points is quite small, the controller synthesis could also be extended to find a global lyapunov function.

Another object of future work is the design of a Fuzzy TS swing-up controller, where further linearization points as well as an extension of the premise vector might yield satisfying results.

A significant improvement of the testbed system is the adaptation of the Hardware, real life tests can then be performed to show the performance of the controller on the Hardware itself. The controller synthesis therefore can be extended to an optimal-robust approach to ensure robustness against model uncertainty.

References

[1] Patrick J. W. Koelewijn, Pablo S. G. Cisneros, Herbert Werner, Roland Toth. "LPV Control of a Gyroscope with Inverted Pendulum Attachment". In: *2nd IFAC Workshop on Linear Parameter Varying Systems LPVS 2018*. vol. 51, no. 26, pp. 49-54, 2018. IFAC-PapersOnLine.

[2] Tom Bloemers, Roland Toth. "Equations of Motion of a Control Moment Gyroscope", 2019.

[3] Benjamin Seth Cazzolato, Zebb Prime. "On the Dynamics of the Furuta Pendulum". In: *Journal of Control Science and Engineering*, 2011.

[4] Zsofia Lendek, Thierry Marie Guerra, Robert Babuska, Bart De Schutter, "Stability Analysis and Nonlinear Observer Design Using Tagaki-Sugeno Fuzzy Models", 2011.

[5] M. Johansson, A. Rantzer, K.-E. Arzen, “Piecewise Quadratic Stability of Fuzzy Systems”, In: *IEEE Transactions on Fuzzy Systems*, vol. 7, no. 6, pp. 713-722, 1999.

[6] Kazuo Tanaka, Hua Wang. “Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach”. 2001.

[7] Carsten Scherer, Siep Weiland. “Linear Matrix Inequalities in Control”. Lecture Notes. 2015.

[8] Florian Pöschke, Eckhard Gauterin, Horst Schulte, “Chapter 2 - LMI Region-Based Nonlinear Disturbance Observer With Application to Robust Wind Turbine Control”, In: *New Trends in Observer-based Control*, Academic Press, pp. 35-75, year 2019, Series: Emerging Methodologies and Applications in Modelling.

[9] J. Lofberg, “YALMIP: A Toolbox for Modeling and Optimization in MATLAB”, In: *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, year 2004, pp. 284-289.

[10] MOSEK ApS, “The MOSEK Optimization Toolbox for MATLAB Manual. Version 10.0.”, year 2023, URL: `http://docs.mosek.com/10.0/toolbox/index.html`.

[11] Hossam S. Abbas, Ahsan Ali, Seyed M. Hashemi, Herbert Werner. “LPV Gain-Scheduled Control of a Control Moment Gyroscope”. In: *2013 American Control Conference*, Washington, DC, USA, 2013, pp. 6841-6846.

[12] Hossam Abbas, Ahsan Ali, Seyed Hashemi, Herbert Werner, “LPV State-Feedback Control of a Control Moment Gyroscope”, In: *Control Engineering Practice*, vol. 24, pp. 129-137, year 2014.

Data-driven identification of disturbances using a sliding mode observer

Ricarda-Samantha Götte, Jo Noel Klusmann, Julia Timmermann

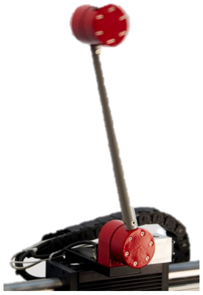
Heinz Nixdorf Institute, Paderborn University
Fürstenallee 11
E-Mail: rgoette@hni.upb.de

1 Introduction

Sliding mode observers (SMOs) provide a robust tool for state estimation and give additional information about disturbances and model uncertainties [1]. Thus, they are frequently deployed for fault detection and analysis. However, analysis often contains only low-pass filtering without any further identification scheme [2]. Yet, characterizing disturbances may be advantageous not only for disturbance control to prevent any harm to the plant and maintain its desired behavior, but also to ensure a longer life cycle of mechanical components, e.g. by actively compensating for disturbances with eigenfrequencies. While our previous work [3, 4] focused on the joint estimation of states and model uncertainties in general, this contribution transfers the concepts to robust estimation. In particular, we demonstrate how to efficiently and even automatically receive dynamical representations for disturbances by a SMO, while also delivering correct state estimates. Ultimately, this insight can be utilized for disturbance control and model adaption.

2 Sliding Mode Observer

For the purpose of this contribution, a SMO is designed for the control of an inverted pendulum on a cart. The set up of the pendulum is displayed in Fig. 1a and its parameters are shown in Tab. 1b.


(a) Set up at our laboratory

Parameter	**Value**	**SI Unit**
mass m	0,654	kg
gravity g	9,81	$\mathrm{m/s^2}$
length a	0,267	m
inertia J	0,0101	$\mathrm{kg/m/s^2}$
damping d	0,001	N m s

(b) Table of parameters

Figure 1: Pendulum on a cart and its characteristics

Its dynamics are described by the following:

$$\begin{pmatrix} \dot{\varphi} \\ \ddot{\varphi} \\ \dot{s} \\ \ddot{s} \end{pmatrix} = \begin{pmatrix} \dot{\varphi} \\ \dfrac{am\cos(\varphi) \cdot u + mga\sin(\varphi) - d\dot{\varphi}}{J + ma^2} \\ \dot{s} \\ u \end{pmatrix}, \tag{1}$$

with φ denoting the angle, s the position of the cart and $\dot{\varphi}, \dot{s}$ the velocities, respectively. However, for simplicity we consider a second-order system in its nonlinear observability canonical form with dynamics f [5] to describe a SMO, since it can be easily adapted towards the pendulum on a cart. Then, the corresponding SMO takes the following form

$$\begin{aligned} \begin{pmatrix} \dot{\hat{x}}_1 \\ \dot{\hat{x}}_2 \end{pmatrix} &= \begin{pmatrix} \hat{x}_2 + \nu_1(e_y) \\ \hat{f}(\hat{x}_1, \hat{x}_2, u) + \nu_2(e_y) \end{pmatrix}, \\ \hat{y} &= \hat{x}_1, \\ e_y &= y - \hat{y} = x_1 - \hat{x}_1 = e_1, \end{aligned} \tag{2}$$

with $\hat{x}_1, \hat{x}_2$ denoting the estimated states, $\hat{f}$ representing the model of the system and e_y indicating the measurement error. Hence, the error dynamics with

$e_1 = x_1 - \hat{x}_1$ and $e_2 = x_2 - \hat{x}_2$ are deduced with Eq. (2) by

$$\begin{aligned} \dot{e}_1 &= e_2 + \nu_1(e_y), \\ \dot{e}_2 &= \Delta f + \nu_2(e_y). \end{aligned} \tag{3}$$

Δf is hereby the deviation between the system f and the model $\hat{f}$. The parameters k_i of the injection terms $\nu_i(e_y) = -k_i \operatorname{sign}(e_y)$ control the stability, effectiveness and convergence rate of the estimation. Especially, k_2 needs to be chosen such that $k_2 > |\Delta f|$ holds [1, 2] but guessing the maximal model deviation correctly often remains a challenge.

3 Data-driven disturbance identification

Since these injection terms $\nu_i(e_y) = -k_i \operatorname{sign}(e_y)$ are available at any time, we can utilize them for identifying the model deviation Δf. Assuming that the SMO is roughly well parameterized with design parameters k_i and has reached its sliding phase, we can not only expect $\dot{e}_y \to 0$ but also $\dot{e}_2 \to 0$. Thus, we receive $\Delta f = -\nu_2(e_y)$. Now instead of low-pass filtering Δf [2], which is usually the way to track potential disturbances, we seek for a physically interpretable representation of the disturbances besides capturing their dynamics. By this, we gain more insight into the disturbances and are able to e.g. compensate for these actively or analyze their effects regarding the life cycle of affected components such as actuators. Moreover, this information can be utilized for model adaption. Simply, assume a linear combination of n_θ suitable, physics-based terms stored within a library $\Psi \in \mathbb{R}^{n_\theta}$ that incorporate one's hypotheses which characteristics the disturbances may exhibit. Therefore, the following holds for Δf's approximation by the parameters $\theta \in \mathbb{R}^{n_\theta}$:

$$e_\theta = \Delta f - \theta^T \Psi(\hat{x}, u). \tag{4}$$

For useful insights into Δf, the interpretation error e_θ must tend towards zero, ideally for $t \to \infty$. Hence, the optimal $\hat{\theta}$ is found by minimizing

$$\arg\min_{\hat{\theta}} \int_0^t e_\theta^2 \mathrm{d}\tau = \int_0^t \left(-\nu_2(e_y) - \theta^T \Psi(\hat{x}, u)\right)^2 \mathrm{d}\tau, \tag{5}$$

whose solution is given by [6]

$$\hat{\theta} = \left(-\int_0^t \nu_2(e_y)\Psi(\hat{x},u)^T \mathrm{d}\tau\right)\left[\int_0^t \Psi(\hat{x},u)\Psi(\hat{x},u)^T \mathrm{d}\tau\right]^{-1}. \tag{6}$$

By using an efficient, dynamic calculation for Ψ [6], the inversion of the library does not need to be computed completely within every time update. To account for changing characteristics and time-dependent behavior, the cost function can be averaged by a time factor if necessary.
However, choosing terms ψ_i for Ψ is difficult beforehand if no prior knowledge regarding the disturbances' characteristics is available. A solution to this challenge is to collect information regarding the disturbances, e.g. by a Fourier transformation for oscillations. Fig. 2 illustrates the Fourier transformation for the example used in Sec. 5, which identifies the three most important frequencies. Thus, using the Fourier transformation helps to identify the main frequencies that are then automatically included by trigonometric terms within Ψ as characteristics of Δf.

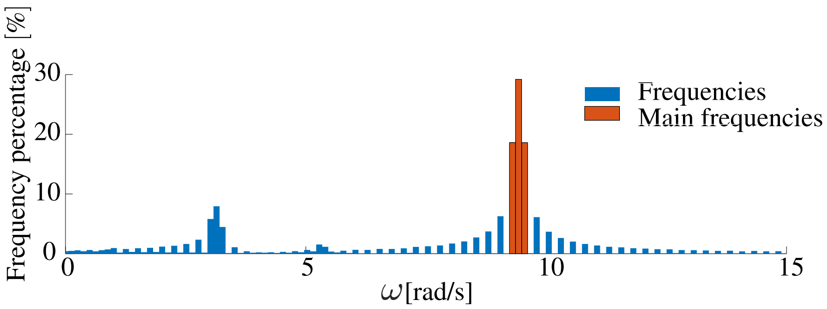


Figure 2: Fourier transformation to identify the main frequencies for choosing library terms ψ_i automatically

4 Results and outlook

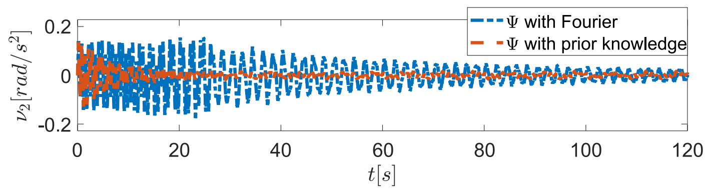
To illustrate the effects of our proposed method, we present results from an open-loop scenario since it is easier to account for the outcomes without the

controller's influence. However, similar results have been obtained for closed-loop behavior using a linear quadratic controller combined with a DMOC optimal trajectory [7].
Forcing an external disturbance $\rho(t) = 4\sin(3\pi t + \pi/2)$ additional to the excitation $u(t) = \sin(\pi t + \pi/2)$ on the test bench at our laboratory, that affects the cart's position, we check if the proposed SMO automatically identifies the additional dynamics. Therefore, we compare two SMOs with libraries that are constructed differently. First, a library is set up by prior knowledge that contains the dynamics of $\rho(t)$. Thereafter, a library is constructed by the Fourier transform whose identified frequencies are utilized within it. Both libraries finally exhibit identical terms ψ_i to compare their performance, namely

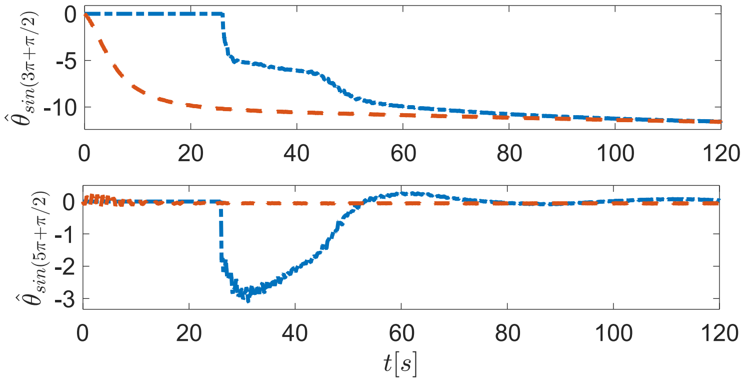
$$\Psi(\hat{x},u) = (\sin(\hat{\varphi}), \hat{\dot{\varphi}}, \mathrm{sign}(\hat{\dot{\varphi}}), \sin(\pi t + \pi/2), \sin(3\pi t + \pi/2), \sin(5\pi t + \pi/2))^T. \tag{7}$$

As Fig. 2 depicts the frequency of $\rho(t)$, namely $\omega_1 = 3\pi$, is identified correctly by the Fourier transformation. It also recognizes the frequency of the excitation $u(t)$ at $\omega_2 = \pi$. Using this information, Fig. 3 then shows excerpts regarding the convergence of the parameters $\hat{\theta}$ and the model deviation expressed by $v_2(e_y)$. If the library Ψ is set up by prior knowledge, the orange dashed signal in Fig. 3a shows that the deviation reduces much faster compared to when relevant terms for Ψ first need to be determined by a Fourier transformation which is illustrated by the blue dashed signal. However, in cases when we do not have any information regarding $\rho(t)$, this enables a fully automated identification of disturbances and features only slightly more convergence time due to the necessary collection of data that lasts in this case around $26s$. Note that it ultimately arrives at a similar level of error compared to when prior knowledge is used directly.

Considering the course of the parameters $\hat{\theta}$, both strategies show strong convergence rates, only varying in speed due to the data collection and analysis of the Fourier transformation. Yet, both converge to the same value and deliver consistent results, e.g. identifying the term $\psi(t) = \sin(3\pi t + \pi/2)$ as present within the disturbances and neglecting the term $\psi(t) = \sin(5\pi t + \pi/2)$ by convergence towards zero. However, it can be noticed that the identified parameter $\hat{\theta}_{\sin(3\pi+\pi/2)}$, which both strategies converge to, does not coincide



(a) Convergence of $\nu_2(t)$: It converges more slowly if no prior knowledge is used due to the necessary data collection for the Fourier transformation (blue) compared to when prior knowledge is applied (red).



(b) Convergence of selected $\hat{\theta}$: Parts of $\rho(t)$ are identified correctly, although convergence rates vary due to how the library terms are determined.

Figure 3: Excerpts from the identification of Δf by prior and automatically chosen Ψ

with the amplitude of $\rho(t)$. This results from the effect that $\rho(t)$ acts directly on the control input $u(t)$. As Eq. (1) describes it holds $am\cos(\varphi)(J+ma^2)^{-1} \cdot u$. Due to the angle's oscillation around $-\pi$ as it can be seen later in Fig. 5, which results in $\cos(\varphi) \approx -1$, the factor tends to $am\cos(\varphi)(J+ma^2)^{-1} \approx 3$. Thus, both SMOs identify the overall amplitude of $\rho(t)$ with 12, assuming the factor rather belongs to the disturbance than to the control input.

Moreover, in addition to the decrease of $\nu_2(t)$, we verify if the SMOs identify the disturbance $\rho(t)$ correctly. Hence, Fig. 4 shows an excerpt of a comparison between the disturbance $\rho(t)$ and its approximation by the linear combination $\hat{\theta}^T\Psi(\hat{x},u)$ with the Fourier transformation once the parameters converged. It reveals that the approximation captures the disturbance well although some minor deviations can be recognized. Note that $\rho(t)$ is displayed with the factor

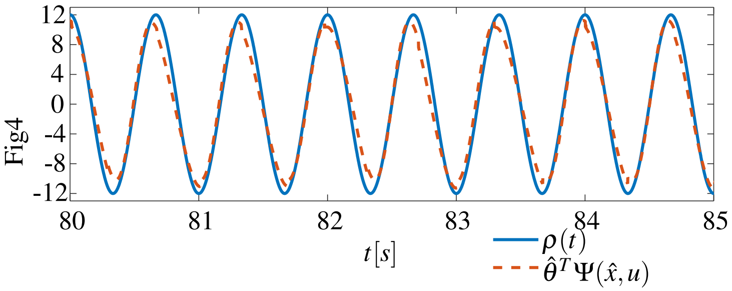


Figure 4: Excerpt from comparison of disturbance $\rho(t)$ and its approximation by $\hat{\theta}^T\Psi(\hat{x},u)$

acting on the control input, since the SMO assumes it acting on the disturbance.

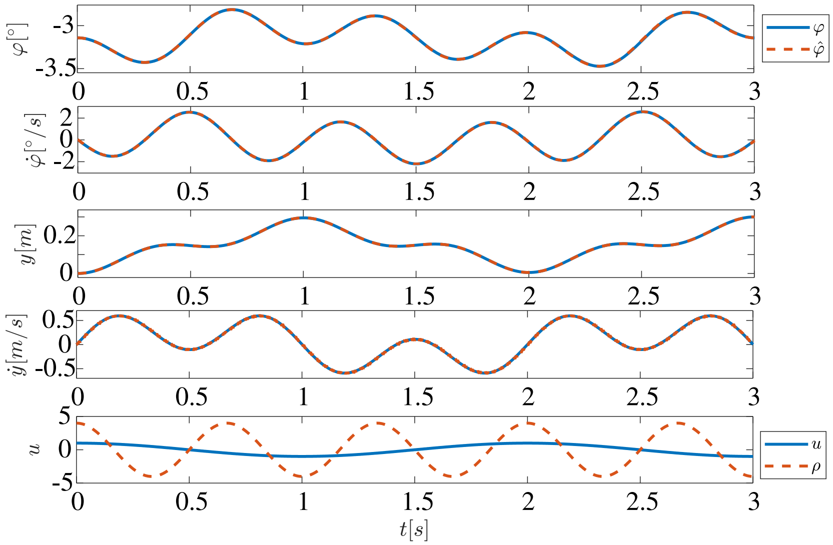


Figure 5: State estimation when the pendulum is excited by $u(t)$ and the observer gets additional disturbance $\rho(t)$

Since the injection term $v_2(t)$ decreases significantly over time and is already very low in the beginning of the estimation, the quality of the state estimation is expected to be high throughout. Fig. 5 confirms this impression by presenting the trajectories of the pendulum over time. Due to its good parameterization with k_i the sliding mode observer captures the pendulum's dynamical behavior right from the beginning very well without any major estimation errors even though the disturbance is present and not yet fully identified.

In conclusion, this contribution showed the concept of joint estimation deployed within a sliding mode observer. It highlighted the advantages that result from disturbance identification and additionally presented the option to automatically receive candidate functions for the library by the Fourier transformation. Further, it convinced with a high quality of state estimation, while gaining more insight into present disturbances. Future research allows the usage of those strategies for intelligent fault management and provides a tool for online model adaption.

Acknowledgment

This work was developed in the junior research group DART, Paderborn University, and funded by the Federal Ministry of Education and Research of Germany under the funding code 01IS20052. The responsibility for the content of this publication lies with the authors.

References

[1] S. K. Spurgeon, "Sliding mode observers: a survey," *International Journal of Systems Science*, vol. 39, no. 8, pp. 751–764, 2008.

[2] Y. Shtessel, C. Edwards, L. Fridman, and A. Levant, *Sliding mode control and observation*, 1st ed. New York and Heidelberg: Birkhäuser, 2013.

[3] R.-S. Götte and J. Timmermann, "Estimating states and model uncertainties jointly by a sparsity promoting ukf," *IFAC-PapersOnLine*, vol. 1, no. 56, pp. 85–90, 2023.

[4] ——, "Approximating a laplacian prior for joint state and model estimation within an ukf," in *Presented and Published at IFAC World Congress 2023*, 2023.

[5] J. Adamy, *Nonlinear Systems and Controls*, 1st ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2022.

[6] J. Davila, L. Fridman, and A. Poznyak, "Observation and identification of mechanical systems via second order sliding modes," in *International Workshop on Variable Structure Systems, 2006. VSS'06.* IEEE, 2006, pp. 232–237.

[7] J. Timmermann, S. Khatab, S. Ober-Blöbaum, and A. Trächtler, "Discrete mechanics and optimal control and its application to a double pendulum on a cart," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 10 199–10 206, 2011.

Robust Training with Adversarial Examples on Industrial Data

Julian Knaup, Christoph-Alexander Holst, Volker Lohweg

inIT – Institute Industrial IT
Technische Hochschule Ostwestfalen-Lippe
Campusallee 6, 32657 Lemgo
E-Mail: {julian.knaup, christoph-alexander.holst, volker.lohweg}@th-owl.de

Abstract

In an era where deep learning models are increasingly deployed in safety-critical domains, ensuring their reliability is paramount. The emergence of adversarial examples, which can lead to severe model misbehavior, underscores this need for robustness. Adversarial training, a technique aimed at fortifying models against such threats, is of particular interest. This paper presents an approach tailored to adversarial training on tabular data within industrial environments.

The approach encompasses various components, including data preprocessing, techniques for stabilizing the training process, and an exploration of diverse adversarial training variants, such as Fast Gradient Sign Method (FGSM), Jacobian-based Saliency Map Attack (JSMA), DeepFool, Carlini & Wagner (C&W), and Projected Gradient Descent (PGD). Additionally, the paper delves into an extensive review and comparison of methods for generating adversarial examples, highlighting their impact on tabular data in adversarial settings.

Furthermore, the paper identifies open research questions and hints at future developments, particularly in the realm of semantic adversarials. This work contributes to the ongoing effort to enhance the robustness of deep learning models, with a focus on their deployment in safety-critical industrial contexts.

1 Introduction

In recent years, artificial intelligence (AI) has witnessed tremendous advancements, revolutionizing various domains and becoming an integral part of our daily lives. From computer vision systems [1, 2] to natural language processing [19, 4] and object detection [5] for autonomous vehicles, deep learning models have showcased remarkable capabilities, surpassing human performance in many complex tasks. In particular, AI experienced extreme media interest due to the capabilities of ChatGPT [4]. However, as AI systems become increasingly integrated into critical applications, ensuring their reliability and robustness becomes imperative.

One of the key challenges in the deployment of deep learning models is their vulnerability to adversarial examples (AEs). AEs are carefully crafted perturbations applied to input data, often imperceptible to humans, that can cause deep learning models to misbehave or produce incorrect predictions [6]. The existence of AEs has raised significant concerns about the reliability and security of AI systems, particularly in safety-critical domains such as healthcare, autonomous driving, and industrial automation.

Nowadays, industrial production plants are intelligent technical systems. These cyber-physical production systems can be severely affected by AEs, causing major financial or personnel damage. An attacker can either stop systems without an anomaly being present or allow them to continue operating even though a fault has occurred [7, 8]. Notably, in the industrial context, data exhibits high heterogeneity, diverging significantly from the limited value ranges typically encountered in image data, the origin of AEs. Additionally, industrial data can often be unstructured and accompanied by sparse labels. To effectively employ common AE generation algorithms, preprocessing of industrial data becomes a necessary step.

This paper delves into the practical application of AEs within the industrial landscape. Specifically, this paper encompasses the following key elements:

- an exploration of prevalent AE generation algorithms,
- practical insights into adversarial training techniques,

- preprocessing methodologies tailored for tabular data,
- a comparative analysis of diverse adversarial attacks, evaluating their suitability for adversarial training with tabular data drawn from the industrial context,
- identification of ongoing challenges and a prospective outlook on future research avenues.

2 Related Work and Preliminaries

This section provides background information and relevant methods for generating adversarial examples (AEs) and countermeasures to enhance robustness.

2.1 Adversarial Examples

The concept of AEs was initially introduced by *Szegedy et al.* [6] and *Biggio and Roli* [9]. In general they can be defined as:
Let $x \in \mathbb{R}^d$ be an input with true label y_0 and y_t is a (target) label different from y_0. An AE x' results from a mapping $\mathscr{A} : \mathbb{R}^d \to \mathbb{R}^d$ such that the modified input $x' = \mathscr{A}(x)$ is misclassified as y_t without changing its true class.

However, mapping $\mathscr{A}(\cdot)$ is often limited to a linear operation [10], so that an additive perturbation δ is introduced

$$x' = x + \delta.$$

To avoid changing the original class membership, δ must be small w. r. t. a distance metric. On image data, δ is commonly minimized in the literature [10] w. r. t. the L_p norm

$$||x' - x||_p = ||\delta||_p = \left(\sum_{i=1}^{n} |\delta_i|^p \right)^{\frac{1}{p}}$$

to create AEs that are visually indistinguishable to human observers. In particular, the L_0, L_2 and L_∞ norm are employed [11]. The L_0 norm represents the number of changed features or pixel, the Euclidean distance is measured with the L_2 norm and the L_∞ norm indicates the maximum change of a feature or pixel.

2.2 Adversarial Attacks

To generate AEs, a variety of approaches have been proposed, again with various modifications [10]. In the following, the most influential basic methods are presented, which will be compared later. Only white-box methods were considered, i. e. those that have complete knowledge of all parameters, as they allow for the strongest attacks [10].

Fast Gradient Sign Method

Szegedy et al. describe the generation of AEs as a constrained optimization problem [6]. They leverage the box-constrained limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm to obtain solutions. However, to reduce the computational cost, *Goodfellow et al.* introduce the Fast Gradient Sign Method (FGSM) [12]. Here, gradients ∇_x are calculated once for all input features. Each input feature is then modified in gradient ascent direction by a fixed step size ε to maximize the loss function $\mathscr{L}$

$$\delta = \varepsilon \cdot sign(\nabla_x \mathscr{L}(x, y_0)). \tag{1}$$

Since the stepsize ε is equal for all input features and they are all modified at once, the FGSM is optimized for the L_∞ norm. Furthermore, the FGSM is fast to compute but not an optimal solution. *Kurakin et al.* [13, 14] provide an iterative version of this attack, which leads to more sophisticated AEs.

Jacobian-based Saliency Map Attack

Papernot et al. introduced the Jacobian-based Saliency Map Attack (JSMA) [16]. They compute the Jacobian matrix for a specific target class w. r. t. its input features. Based on these partial derivatives a saliency map is constructed indicating the influence of each input feature. Subsequently, the most influential input is modified accordingly and checked whether an AE is present. This process is repeated until a predefined number of features has been altered or an AE has been found. Due to the successive nature of feature changes, the JSMA is optimized for the L_0 norm.

DeepFool

The basic idea of the DeepFool algorithm [15] is to view the model as an affine transformation, i. e. the authors linearize the models decision boundary around an input x. In binary classification the decision boundary becomes a hyperplane and in the multinomial case the decision boundaries around the input x are approximated with a polyhedron formed by each of the decision hyperplanes. They project the input orthogonal, i. e. with minimum distance, to the nearest hyperplane and push it slightly beyond it to craft an AE. Since the linearization is an approximation they just take a step in the direction of this projection and iterate this process until an AE is reached. In the original version the algorithm is optimized for the L_2 norm.

Carlini & Wagner

Carlini and Wagner [11] offer a variety of attacks with L_0, L_2, and L_∞ distance metrics. However, they claim their L_2 attack (C&W) to be the strongest one and in fact, the L_0 version leverages the L_2 attack. They iteratively optimize an objective function consisting of a misclassification term and a distance measure of the perturbation. Furthermore, they exploit a scaled and shifted *tanh* function with a variable exchange

$$\delta = \frac{1}{2}(tanh(w) + 1) - x \tag{2}$$

to let the perturbation map natively into the interval $[0,1]$. By eliminating the necessity of clip functions in this way, they are able to employ momentum-based optimizers such as Adam [17]. The C&W attack is one of the strongest attacks in terms of finding minimal perturbation and fooling the machine learning model [11]. Additionally, they overcame numerous defensive strategies [19], such as Defense Distillation [18], that existed at the time of release.

Projected Gradient Descent

As Gradient Descent is a standard way to solve an unconstrained optimization problem, Projected Gradient Descent (PGD) in general provides a way to solve constrained optimization problems. The PGD attack [20] leverages this approach to generate AEs. One starts from a random perturbation in an L_p ball around an input sample, takes a step in the gradient direction of the loss function w. r. t. its input data and, if necessary, projects the result back into the L_p ball. This procedure is repeated until convergence or exceeding the maximum number of iterations. Therefore, *Madry et al.* [20] reference the iterative FGSM as an L_∞ bounded PGD attack, where the projection is realized by the clipping function. The authors claim that the PGD method is probably the strongest first-order attack. They argue that AEs generated with it are more suitable for adversarial training, since models are also robust against weaker methods after training with these AEs.

2.3 Adversarial Defensives

The sequence of developments in countermeasures for adversarial examples is similar to the history of cryptography. After methods for defense are proposed, there are new attack strategies, which in turn overcome them [19]. Defensive approaches that do not require the secrecy of specific aspects, such as gradients, are therefore to be preferred here as well [21].

Adversarial training is a primary strategy for enhancing the adversarial robustness of neural networks. By introducing AEs during training [6, 12], models can be designed to be more robust to small perturbations. *Madry et al.* consider adversarial training as a saddle point or min-max problem [20]. On the one hand, the goal is to generate AEs that maximize the loss function and, on the other hand, to find model parameters that minimize this loss. Moreover, *Tsipras et al.* demonstrate that adversarial training can lead to more robust features, which, however, are obtained at the expense of accuracy [22]. A more detailed overview of adversarial training can be found in [23].

3 Approach

In this section, an approach is developed that facilitates cross-comparison of AE generation methods. To ensure comparability among the presented methods, appropriate metrics must be selected. However, there is no uniform definition of quantifiable adversarial robustness in the literature. Additionally, adversarial attacks are optimized w. r. t. different L_p norms, further complicating the assessment of AE quality. To address these challenges, we first empirically test whether adversarial training enhances model robustness against attacks using the same method as in training. To achieve this, we employ both the accuracy on the original data and the accuracy on the AEs as metrics. Subsequently, models trained using one method are evaluated against the remaining attacks.

Another critical consideration is the nature of the data. Humans have less intuition for tabular, numerical data compared to speech or images [24]. In the image domain, the L_p norm serves as an approximation for human perception. Visual inspection helps assess whether visible artifacts are present in the AEs. This allows to establish a budget for the adversarial attacks such that these artifacts are minimized while ensuring that the original class membership of the sample is maintained. However, this is not feasible for tabular data, so alternative constraints on the adversarial attacks are required. The specific limitation of the attack methods is detailed in the next section.

Moreover, various approaches exist for conducting adversarial training. In [20], the iterative training is exclusively performed on the AEs to reduce computational costs, arguing that AEs already offer greater diversity than the original data points. Conversely, *Specht et al.* compute AEs only once and not iteratively within the training, augmenting their original training dataset once with an equivalent amount of AEs [7, 8]. However, our initial tests failed to reproduce sufficient robustness when training solely on once-generated AEs. Given the trade-off between adversarial robustness and accuracy [22], we adopt a mixed approach. We include the original data in training to prioritize accuracy, but AEs are recalculated in each minibatch with the current model parameters. For each input, an AE is computed without applying a weighting parameter, ensuring that AEs and original data have equal influence on the loss.

Additionally, we introduce a one-epoch warm-up phase to stabilize the training process. During this phase, only the original data is utilized. Starting from epoch two, a mixture of AEs and original data is incorporated. The warm-up phase is essential as AEs, considered as worst-case inputs, are typically more challenging to learn than the distribution of the original data, which can potentially interfere with finding the appropriate parameters at the beginning. Eliminating the warm-up phase in later implementations resulted in some classes not receiving any predictions at all.

Furthermore, the heterogeneous nature of industry data requires adjustments to restrict the range of feature values. This prevents the emergence of unrealistic values and eliminates the need to adapt algorithms, as they naturally operate in the constrained range $x+\delta \in [0,1]^n$, originating from the image processing domain. Achieving this is straightforward through a min-max scaler. However, when scaling, it is important to consider the variance within the dataset. Special attention must be paid to extreme outliers that would distribute the majority of data into a significantly smaller interval.

4 Evaluation

4.1 Experimental Setup

Dataset

The experimantal results are obtained on the Sensorless Drive Diagnoses (SDD) dataset [25]. This dataset is derived from two-phase currents measured in a 425W permanent magnet synchronous motor, which is part of a modular demonstrator, as detailed in [26, 27]. The demonstrator itself is comprised of several components, including the test motor, measuring shaft, bearing module, flywheel, and load motor. To simulate various fault conditions, synthetic hardware corruptions can be introduced. The raw data from the demonstrator were preprocessed as described in [28] when creating the SDD dataset. Thereby, empirical mode decomposition was applied to determine three intrinsic mode functions and their residuals per phase. Subsequently, the mean, standard deviation, skewness, and kurtosis were calculated for each, resulting in a total set of 48 features.

The SDD dataset encompasses 58,509 samples and includes 11 distinct classes, maintaining a balanced distribution. In this context, class 1 signifies the fault-free state of the engine, while the remaining ten classes represent various fault cases stemming from issues such as shaft misalignment, axis inclination, or bearing failure. A summary of the classes and their respective fault cases is provided in Table 1. To facilitate model evaluation, an 80/20 split between the training and test sets was employed, resulting in 46,806 samples in the training set and 11,703 samples in the test set.

The SDD dataset is particularly suitable as it offers multiple defect classes with diverse characteristics in the area of industrial data, which facilitates AE generation. At the same time, it is not too high dimensional, which keeps the computation times within reasonable limits.

Table 1: Error indicators of the individual classes of the SDD dataset. Class 1 is error-free, and the remaining ten are error cases. Equal classes, such as class 4 and 5, are not identical; they differ in the level of error, for example, the angle of the axis inclination.

Class	1	2	3	4	5	6	7	8	9	10	11
Bearing Failure	0	0	0	0	0	1	1	1	1	1	1
Axis Inclination	0	0	1	1	1	0	1	1	0	1	1
Shaft Misalignment	0	1	0	1	1	1	0	1	1	0	1

Model Implementation

A deep neural network (DNN) with four hidden layers is deployed for evaluation. The input layer comprises 48 neurons, followed by hidden layers with 590, 1180, 2360, and 590 neurons, respectively, and an output layer with 11 neurons. The architecture is based on [7]. Although smaller DNNs can classify the SDD dataset [29], the selection should prevent a bottleneck of capacity as discussed in [20] and exclude this influence. After each hidden layer, Batch Normalization (BatchNorm) [31] is applied followed by the Rectified Linear Unit (ReLU) activation function. Dropout [30] with a dropout rate of 20% is utilized after each ReLU layer to prevent overfitting. The output layer employs a linear layer with Softmax for classification. Cross-entropy loss is used with the Adam optimizer [17], configured with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and a learning rate of 10^{-4}. The implementation is carried out using the PyTorch framework [32]. A min-max scaler is used for preprocessing to scale feature values to the range $[0,1]$.

The generation of AEs is performed using the adversarial-robustness-toolbox [33] and advertorch [34]. For the L_∞ attacks PGD and FGSM, the maximum perturbation is controlled by the explicit parameter ε, which is determined as described in the following section. The control parameter Γ for JSMA, regulating the proportion of features that can be altered, is set at 14.5% following [16]. As DeepFool and C&W do not have explicit attack budget parameters, they are limited to 10 iterations, equivalent to the number of iterations in the

Table 2: The perturbation budget influences adversarial robustness. A perturbation per feature of 1% of its range leads to adversarial robustness close to the accuracy of clean data. An increase in the attack budget significantly reduces adversarial robustness, suggesting a potential change in the true class.

Attack	Perturbation in %	Clean Accuracy	Adversarial Accuracy
PGD	1	0.99	0.92
PGD	2	0.96	0.74
PGD	3	0.93	0.60
PGD	4	0.99	0.22

PGD algorithm with the chosen ε. Further details, code, and parameter settings can be accessed here[1], allowing for result reproduction and further research.

4.2 Results

Attack Budget for L_∞ Norm Attacks

To establish an attack budget for L_∞ norm attacks, specifically FGSM and PGD, we selected the stronger of the two variants, i.e., PGD, and tested it with various parameter values. Table 2 presents the results of these tests. All the attacks listed in Table 2 were capable of reducing the accuracy of models without adversarial training by at least 60%. It was observed how long adversarial training remained effective. A significant drop in adversarial robustness can be interpreted as an indication that the true class membership has changed, rendering the model incapable of learning the data distribution. Based on the results in Table 2, a maximum perturbation of 1% of the feature range was set as the attack budget for the L_∞ norm attacks.

[1] https://ds-juist.init.th-owl.de/j.knaup/ciworkshop

Table 3: Accuracy results of the cross-comparison of different AE generation methods. Clean indicates the usage of only original training and test data, respectively. The remaining training methods utilize a mix of the data in the training phase, and the remaining attack methods are evaluated on the manipulated data exclusively.

Adversarial	Adversarial Attack Method					
Training Method	FGSM	JSMA	DeepFool	C&W	PGD	Clean
FGSM	0.92	0.21	0.08	0.74	0.91	0.99
JSMA	0.40	0.33	0.26	0.27	0.39	0.43
DeepFool	0.65	0.12	0.24	0.20	0.57	0.99
C&W	0.74	0.09	0.09	0.12	0.67	0.97
PGD	0.93	0.21	0.09	0.74	0.92	0.99
Clean	0.41	0.07	0.01	0.35	0.32	0.99

Cross-comparison of AE Generating Methods

Table 3 shows that training with JSMA generated AEs, significantly affects the accuracy on the original data. PGD and FGSM achieve almost identical values and are robust to themselves and each other. JSMA and DeepFool reduce the accuracies of the other models the most and C&W achieves an increased robustness against FGSM and PGD. A detailed discussion is provided in the next section.

5 Discussion

The results presented in Table 3 align with findings in the literature, where FGSM and PGD are commonly used for adversarial training on image data [23]. The fact that PGD differs only slightly from FGSM may be attributed to factors such as the number of iterations or the limited attack budget. JSMA, originally designed for gray-scale images like the MNIST dataset [35], poses certain challenges when applied to tabular data. By searching individual pixels and increasing or decreasing their values depending on the sign of the adjustment parameter, JSMA sets individual features here to 0 or 1, respectively. This leads to unrealistic inputs, which on the one hand are difficult to classify for other models, but on the other hand it is not reasonable to enrich the training set with them. DeepFool is more suitable in this respect, since the respective decision boundary is only slightly exceeded. The C&W attack, known for its high success rate in finding minimal AEs [11], may benefit from further hyperparameter tuning but at the cost of increased computation time.

However, this study's approach has yielded the expected results. The preprocessing enabled the application of various algorithms and the employment of the original data and the manipulated data prioritized the clean as well as adversarial accuracy. The warm-up phase added stability to the training process. A similar approach to this is curriculum-based learning [36], where attack strength adapts and increases as the training progresses.

Nevertheless, challenges persist in distinguishing between adversarial examples and points at which the ground truth has fundamentally changed. While approaches like [37] improve PGD by considering the proximity of each input to the decision boundary when applying perturbations, they still do not identify the true tipping point. Additionally, selecting an appropriate distance measure for this assessment remains an open question. Even though L_∞ norm attacks show promise, perturbations with the same L_p norm can have vastly different effects. Comparing methods that employ different distance metrics poses particular challenges. Furthermore, adversarial training defined as min-max problem inherently lacks robustness guarantees due to the non-convex nature of deep neural networks, which makes it intractable to find a global optimum [23].

Moreover, the adversarial mapping $\mathscr{A}(\cdot)$ in this paper has been limited to additive perturbations δ. Future research directions may involve exploring the use of generative adversarial networks (GANs) [38, 39] to create adversarial examples. This approach could generate AEs with semantic information, leading to more natural and meaningful adversarial examples, commonly referred to as semantic adversarials in the literature [41, 40].

6 Conclusion and Outlook

In this paper, we presented an approach that extends the application of adversarial attacks to tabular data for adversarial training. We began by providing an overview of various adversarial example generation methods, followed by the introduction of a straightforward preprocessing technique and training stabilization mechanisms. Subsequently, we conducted a comprehensive cross-comparison of popular attack methods, including FGSM, JSMA, DeepFool, C&W, and PGD, on an industrial dataset.

The results of our study validate existing findings in the literature, demonstrating the effectiveness of FGSM and PGD for adversarial training. However, our investigation also highlights the unique challenges posed by tabular data when employing methods like JSMA, which generate unrealistic inputs. The quest for a suitable distance metric remains a pivotal aspect of future research, as it not only determines the presence of adversarial examples but also serves as the foundation for method comparisons.

Looking ahead, the exploration of non-additive perturbations presents a promising avenue for the development of new adversarial example generation methods. Incorporating semantic contextual information into the generation process may yield more natural and meaningful adversarial examples, albeit with potentially higher L_p norm values. This shift toward semantically enriched adversarial examples could lead to advancements in the robustness of machine learning models, particularly in applications involving tabular data.

Acknowledgment

The authors acknowledge financial support by the project “SAIL: SustAInable Life-cycle of Intelligent Socio-Technical Systems” (Grant ID NW21-059C), which is funded by the program “Netzwerke 2021” of the Ministry of Culture and Science of the State of Northrhine Westphalia, Germany.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc. 2012.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In: *9th International Conference on Learning Representations (ICLR)*. 2021.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In: *Neural computation*, 9(8):1735–1780. 1997.

[4] OpenAI. GPT-4 technical report. *ArXiv*, 2023.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. Los Alamitos, CA, USA. June 2016.

[6] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In: *International Conference on Learning Representations (ICLR)*. 2014.

[7] Felix Specht, Jens Otto, Oliver Niggemann, and Barbara Hammer. Generation of adversarial examples to prevent misclassification of deep neural network based condition monitoring systems for cyber-physical production systems. In: *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, pages 760–765. 2018.

[8] Felix Specht and Jens Otto. Hardening deep neural networks in condition monitoring systems against adversarial example attacks.

In: *Machine Learning for Cyber Physical Systems*, pages 103–111. Springer, Berlin, Heidelberg. 2021.

[9] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 2154–2156. New York, NY, USA. 2018.

[10] Naveed Akhtar, Ajmal S. Mian, Navid Kardan, and Mubarak Shah. Advances in adversarial attacks and defenses in computer vision: A survey. *IEEE Access*, 9. 2021.

[11] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In: *IEEE Symposium on Security and Privacy*, pages 39–57. 2017.

[12] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In: *International Conference on Learning Representations (ICLR)*. 2015.

[13] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In: *5th International Conference on Learning Representations (ICLR), Workshop Track Proceedings*. 2017.

[14] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In: *5th International Conference on Learning Representations (ICLR)*. 2017.

[15] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582. 2016.

[16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. 2016.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In: *3rd International Conference on Learning Representation (ICLR)*. San Diego, CA, USA. May, 2015.

[18] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In: *2016 IEEE Symposium on Security and Privacy (SP)*, pages 582–597. San Jose, CA, USA. 2016.

[19] Nicolas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3-–14. Dallas, Texas, USA. 2017.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In: *6th International Conference on Learning Representations (ICLR),Conference Track Proceedings*. Vancouver, BC, Canada. April 30 - May 3, 2018.

[21] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 274–283. Stockholm, Sweden. 2018.

[22] Dimitris Tsipras, Shibani Santurkar, Logan G. Engstrom, Alexander M. Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In: *7th International Conference on Learning Representations (ICLR)*. New Orleans, Louisiana, USA. May, 2019.

[23] Tao Bai, Jinqi Luo, Jun Zhao Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness. In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4312–4321. 2021.

[24] Vincent Ballet, Xavier, Jonathan Aigrain, Thibault Laugel, Pascal Frossard, and Marcin Detyniecki. Imperceptible adversarial attacks on

tabular data. In: *NeurIPS 2019 Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness and Privacy*. Vancouver, Canada. 2021.

[25] Martyna Bator. Dataset for sensorless drive diagnosis. In: *UCI Machine Learning Repository*. https://doi.org/10.24432/C5VP5F . 2015.

[26] Detmar Zimmer, Christian Lessmeier, Kay Hameyer, Christelle Piantsop Mbo'o, and Isabel Coenen. Untersuchung von Bauteilschäden elektrischer Antriebsstränge im Belastungsprüfstand mittels Statorstromanalyse. In: *ant Journal - Anwendungsnahe Forschung für Antriebstechnik im Maschinenbau*, pages 8–13. 2012.

[27] Christian Lessmeier, Olaf Enge-Rosenblatt, Christian Bayer, and Detmar Zimmer. Data acquisition and signal analysis from Mmeasured motor currents for defect detection in electromechanical drive systems. In: *PHM Society European Conference*. 2014.

[28] Martyna Bator, Alexander Dicks, Uwe Mönks, and Volker Lohweg. Feature extraction and reduction applied to sensorless drive diagnosis. In: *22nd Workshop Computational Intelligence (VDI/VDE-Gesellschaft Mess- und Automatisierungstechnik (GMA)*, pages 163–177. 2012.

[29] Anton Pfeifer and Volker Lohweg. Classification of faults in cyber-physical systems with complex-valued neural networks. In: *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–7. Vasteras, Sweden. 2021.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In: *Journal of Machine Learning Research 15*, pages 1929-1958. 2014.

[31] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *Proceedings of the 32nd International Conference on Machine Learning*, pages 448–456. Lille, France. 2015.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance Deep Learning library. In: *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc. 2019.

[33] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian Molly, and Ben Edwards. Adversarial robustness toolbox v1.2.0. In: *Computing Research Repository (CoRR)*. 2018.

[34] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on PyTorch. *arXiv preprint arXiv:1902.07623*. 2019.

[35] Yann LeCun, Corinna Cortes, and Christopher Burges. MNIST handwritten digit database. Available: http://yann.lecun.com/exdb/mnist. 2010.

[36] Qi-Zhi Cai, Min Du, Chang Liu, and Dawn Song. Curriculum adversarial training. *arXiv preprint:1805.04807*. 2018.

[37] Yogesh Balaji, Tom Goldstein, and Judy Hoffman. Instance adaptive adversarial training: Improved accuracy tradeoffs in neural nets. *arXiv preprint:1910.08051*. 2019.

[38] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In: *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[39] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial

networks. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3905–3911. Stockholm, Sweden. 2018.

[40] Tommaso Dreossi, Somesh Jha, and Sanjit A. Seshia. “Semantic adversarial deep learning”. In: *Computer Aided Verification: 30th International Conference (CAV) Proceedings, Part I 30*, pages 3–26, Springer. Oxford, UK, July 14-17. 2018.

[41] Hossein Hosseini and Radha Poovendran. “Semantic adversarial examples”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2018.

An Intelligent Camera Tracking System for Live Stage Performances

Steffen Borchers-Tigasson, Erik Rodner

Hochschule für Technik und Wirtschaft (HTW) Berlin
Wilhelminenhofstraße 75A, 12459 Berlin
E-Mail: steffen.borchers@htw-berlin.de

1 Introduction

Cameras have become increasingly ubiquitous in our daily lives, whether they are positioned in public spaces, within our households, or conveniently nestled in our pockets via smartphones. They serve diverse real-world applications, including video surveillance of human activities, observing wildlife, facilitating home care, enabling optical motion capture, and enhancing multimedia experiences. These applications typically entail a sequence of tasks, beginning with the detection of moving objects, followed by tracking and recognition. Over the past three decades, the field of computer vision has dedicated substantial research efforts to the task of detecting moving objects, resulting in a wealth of publications (cf. [8] for a review). The number of techniques dedicated to addressing scenarios involving moving cameras is steadily increasing, and this subject matter has become a significant focus for in-depth investigation, as evident from recent comprehensive reviews [14, 15, 7].

Focussing on the visual control of moving objects, several approaches have been proposed using PTZ cameras, e.g. for surveillance [2], autonomous off-road navigation and mobile robots [4], and the filming industry [6]. However, available professional systems are still very limited in functionality and flexibility.

To this end, we develop an intelligent camera tracking system suitable for theater, dancing and performances. The system consists of a remote PTZ camera, a state-of-the art real-time object detection and tracking algorithm (Yolov8), and an user interface to direct and adjust tracking. In this paper, we sketch our efforts in developing the system including hardware setup (Sect. 2), application requirements (Sect. 3), and detection and tracking algorithms (Sect. 4). We conclude with an evaluation (Sect. 5) and discussion (Sect. 6).

2 Technical Configuration

The smart camera system consists of a Panasonic AW-UE 4K camera, which allows for a pan movement denoted by x with range $-175° \leq x \leq +175°$, a tilt movement denoted by y with range $-30° \leq x \leq +90°$ and an optical zoom termed z with $1 \leq z \leq 24$. For each of the three movement coordinates, the camera allows for adjusting the movements speed in a range from $-50 \leq \{x,y,z\}_v \leq 50$. The current position and the movement commands are send and received from the camera via an http API interface. The visual data in transferred to a dedicated GPU server via an SDI cable, and the cameras are connected to the server via LAN.

The camera is mounted on a tripod and the absolute camera position remains fixed for the play.

The GPU Server (Windows 10) is equipped with a Blackmagic Capture Card to access the video signal in real time (approx. 40 frames/sec). The server features a NVIDIA RTX Titan (24 GB memory) graphic card.

3 Requirements

In this section, we briefly outline the specific requirements for the camera system. The requirements can be structured into three main building blocks of the overall system.

3.1 Camera Control

We aim to track objects throughout the stage and over the course of a play or performance. The desired position of an object in the frame is the setpoint.

- We demand setpoint control of pan (x), tilt (y) and zoom (z), the respective setpoints are denoted by x_{SP}, y_{SP}, and z_{SP}. The pan and tilt setpoints denote the coordinates where an image object is to be situated. For zoom control, the objects size is estimated from the current frame (see perception) and compared with z_{SP}.
- Setpoint control is extended by a dead zone, i.e. if the object moves though remains within the specified dead zone, no feedback is applied.
- To ensure homogeneous control performance for close and distant objects which appear on the frame at same size, the distance of the object has to be taken into account.

3.2 Perception

Advanced perception capabilities are required for reliable tracking. A basic understanding of the scene and its actors as well as background, possible projection, and backstage people is demanded.

- To detect objects on the current frame, bounding boxes and masks are considered. A high frame rate and low total latency is desired.
- Detection classes are head, face, body. Detected objects are represented by the respective bounding boxes. These detection classes allow for different capture settings such as close-up, knee-up, whole body, and dialog capture.
- All visible persons on the stage shall be detected. Multiple actors may be present on stage. Detected objects have to be tracked and assigned with a tracking ID number.
- The objects distance is to be estimated.

- Track losses and switches, e.g. due to occlusion, have to be considered. Fallback options have to be elaborated.
- Basic association is required for the detected objects and tracks. Objects from the three detection classes shall be associated to persons.
- To re-identify a person across multiple cameras, face identification is employed.

3.3 Interface

A flexible and intuitive Human Machine Interface is required. In short, the interface allows for:

- visualizing the detections, tracks, and associations
- adjusting setpoints and the dead zone online
- changing the tracked object, optionally with smooth transition
- modifying controllers speed
- choosing fallback options
- saving, loading, and visualizing data and configuration
- accessing and controlling all (three) cameras

4 Methods

To obtain a robust, flexible, and fast smart camera tracking system incorporating the requirements outlined in Section 3, we propose the workflow depicted in Fig. 1.

Here, the iterative loop starts with a new frame delivered by the PTZ camera. The frame is processed in the perception module subsequently, where the target object is detected, tracked, and identified. The objects information is then feed into the controller module. In combination with the user interface settings, the

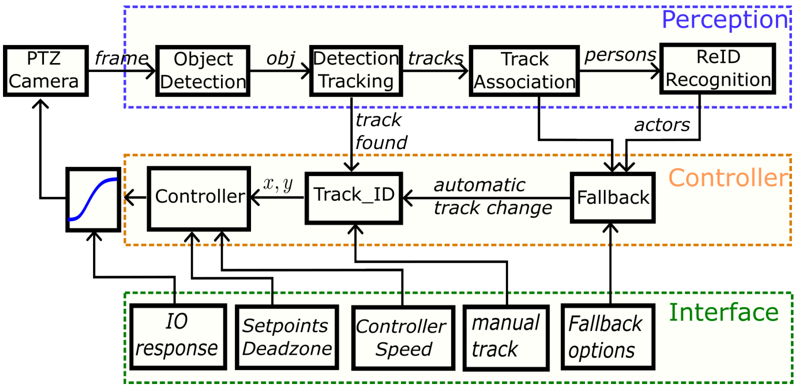


Figure 1: Key modules and the basic workflow for the proposed smart camera system.

controller computes the outputs for pan, tilt, and zoom velocities so as to close the feedback loop.

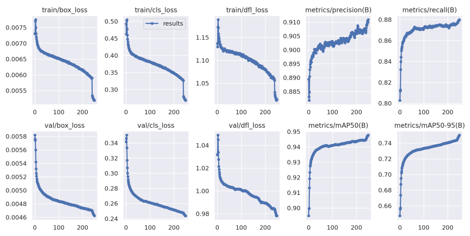
In the following, we describe the methods and approaches for the key modules seperately.

4.1 Perception Module

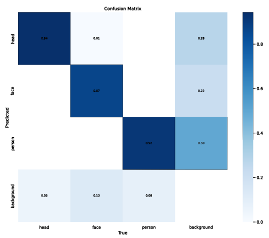
The perception module receives an image, detects all relevant objects in this image, tracks the objects from frame to frame, associates tracks with persons, and identifies them if possible.

4.1.1 Object Detection

To detect objects, namely persons, on the image, a custom model for YOLOv8 [1] has been developed. The model yields bounding boxes for three classes: head, face, and body. For training the model, several datasets such as Hollywood Heads [9], CrowdHuman [10], Facenet [11], and COCO [12] have been combined. Since the desired classes were not covered by all datasets, cross-inference was used to label the missing classes in each dataset.



(a) Training metrics for the custom detection model.



(b) Confusion matrix of the custom model.

Figure 2: Metrics of the custom Yolov8 Model.

The obtained dataset consisted of approx 120.000 labelled images. Using the open source tool FiftyOne [13], we excluded clones, crowds, and removed the most similar images. Furthermore, to decrease false positive detection rates, 10 % background images (no objects no labels) were added to the dataset. The training dataset finally consists of approx. 32.000 labelled images.

Training results and the confusion matrix are depicted in Fig. 2. Accuracy is very good for head (94 %), face (87 %), and body (92 %). As we could verify in various rehearsals, the model is robust as it copes very well with different light settings.

4.1.2 Object Tracking

For tracking the objects detected by our custom Yolov8 model, we used the Kalman-based tracking algorithm Bytetrack [16]. In comparison with other SOTA multi-object-trackers (see references in [16]), Bytetrack provided the best tradeoff between inference speed and accuracy for our purpose of tracking actors on stage.

4.1.3 Association and Recognition

To associate head, face, and body objects to persons, we utilize intersection over union (IoU). Thus, we compare the extend of overlap of all detected bounding boxes. The association works pairwise: If the smaller bounding box is covered by at least 90 %, we associate the two bounding boxes to the same person.

4.1.4 Fallbacks

On runtime, it happens that a track is lost, e.g. due to occlusion, turn arounds, or dancing moves where the head and face are temporarily not visible. Elaborating usefull fallback options thus is essential to ensure reliable tracking.

Basic idea of the fallback strategy is to use the associations obtained. If e.g. the head track is lost, the fallback consists of automatically switching the track to the persons face if available, otherwise to the persons body. If at some stage later a head is reassociated with currently tracked body or face, it atomatically switches back to head tracking. Note that it may be required to smoothly transit to the new setpoints.

4.2 Camera Control Module

The control module receives the location and size of the current object of interest in the image. Given the current setpoints, the controller computes the errors and a corrective feedback. To this end, we use separate PI controllers for pan, tilt, and zoom control.

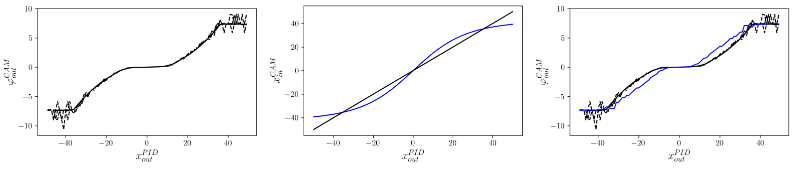
4.2.1 Output linearization

Controller output (x_{out}^{PID}) and cam output φ_{out}^{cam} are non-linearly correlated, see Fig. 3a.

The measured response Fig. 3a is corrected using a sigmoid function:

$$f_{cor}(x) = 100(\frac{1}{1+e^{-0.05x}} - 0.5)$$

(see Fig. 3b). The resulting (linearized) IO response is depicted in Fig. 3c. The response is suppressed in the range $0 \leq |x_{out}| \leq 9$, linear in the range



(a) Measure responses.   (b) IO correction.   (c) Corrected IO response.

Figure 3: IO Pan resonse curve and proposed correction.

$10 \leq |x_{out}| \leq 35$, and saturated for $|x_{out}| \geq 36$. The correction is applied to pan and tilt. The zoom IO response is already almost linear, so no correction is required (data not shown).

4.2.2 Distance estimation

We trained a small dense multilayer perceptron (MLP) from data acquired from the PTZ camera and laser rangefinder. The networks inputs are head width, head height, and the zoom level, the output is the distance estimate d $[m]$. The model provides distance predictions in real time, and validation showed acceptable performance. An analysis of it is out of scope of this paper.

4.2.3 Adaptive PID controllers

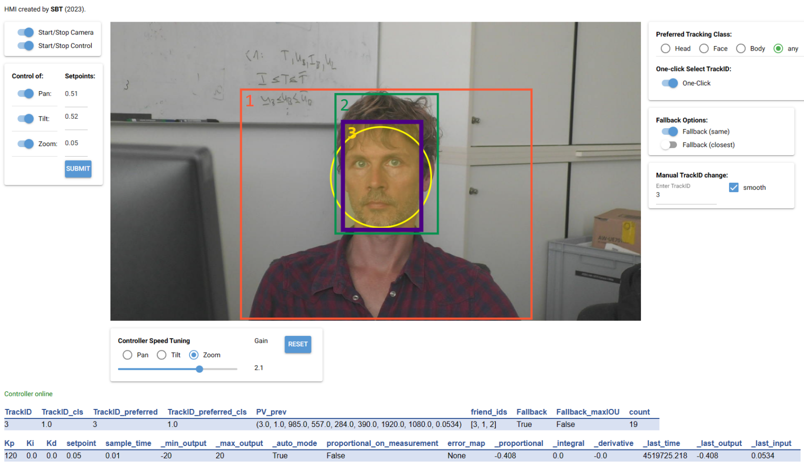
Pan, tilt, and zoom are controlled using PI controllers. The parameters have been tuned manually. The interface allows to adjust the gains online.

To compensate for a trigonometric non-linearity in pan and tilt, the camera-object distance is estimated using the trained MLP as described above. The pan and tilt controller gains are hence automatically scheduled to as:

$$K_p(d) = K_p^* \cdot \frac{3}{d},$$

where K_p^* is the presetted gain tuned for a object distance of 3 m. In the real-time setting, we filter the distance estimate of the head object averaging the last three distance estimates.

4.3 Interface module


HMI created by **SBT** (2023).

Start/Stop Camera
Start/Stop Control

Control of:	Setpoints:
Pan:	0.51
Tilt:	0.52
Zoom:	0.05

Preferred Tracking Class: Head Face Body [✓ any]

One-click Select TrackID: One-Click

Fallback Options: Fallback (same), Fallback (closest)

Manual TrackID change: Enter TrackID 3 [✓ smooth]

Controller Speed Tuning: Pan Tilt [✓ Zoom] Gain 2.1

Controller online

TrackID	TrackID_cls	TrackID_preferred	TrackID_preferred_cls	PV_prev	friend_ids	Fallback	Fallback_maxIOU	count
3	1.0	3	1.0	(3.0, 1.0, 985.0, 557.0, 284.0, 390.0, 1920.0, 1080.0, 0.0534)	[3, 1, 2]	True	False	19

Kp	Ki	Kd	setpoint	sample_time	_min_output	_max_output	_auto_mode	proportional_on_measurement	error_map	_proportional	_integral	_derivative	_last_time	_last_output	_last_input
120	0.0	0.0	0.05	0.01	-20	20	True	False	None	-0.408	0.0	-0.0	4519725.218	-0.408	0.0534

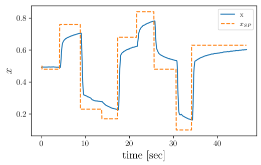
Figure 4: An overview of the interface.

The interface module is based on the python package nicegui [3]. The interface features the current image and detections, as well as interactive selection of tracks. The interface allows to turn on/off detection and the controllers, to select tracks and adjust setpoints interactively, as well as to set fallback options and tune the controller online.

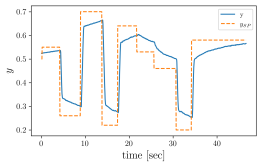
5 Implementation and results

The modules are programmed using Python 3.11. Communication between the modules is established by using UDP. Associations (persons) as well as the current tuning and settings are stored using an SQLite database.

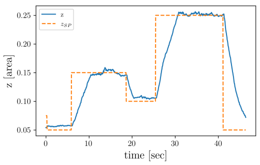
The system is currently subject to intensive tests. A current performance record is depicted in Fig. 5. Here, pan, tilt and zoom controllers are active, and control is induced by setpoint changes.



(a) Pan Control.



(b) Tilt control.



(c) Zoom control.

Figure 5: Control performance.

6 Discussion

This paper presents an intelligent camera system for stage performances based on single PTZ camera. The motivation behind this system is to capture close-ups and dynamic shots, allowing for projection on a screen on large stages, and accommodating hybrid formats. The system aims to provide maximum freedom of movement for actors, dancers, and musicians.

The camera system we developed here employs several perception methods based on recent advances in machine learning, first and foremost computer vision. The perception module delivers objects and its associated tracks in terms of coordinates on the current frame. Given the target location (setpoints), PI controllers stabilize the errors. Non-linearities resulting from the I/O response and from trigonometry are addressed.

Overall, the proposed system supports real-time, low-latency, and multi-camera setups. We handle approx. 40 frames per second, and the overall latency is approx. 80 ms. The system implementation utilizes APIs, deep learning frameworks, and interprocess communication for camera control, perception, and interface functionalities. Overall, this intelligent camera system offers an innovative solution for capturing stage performances with high precision, adaptability, and real-time capacity.

References

[1] G. Jocher et al. “Ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation”. Zenodo c7.0 doi:10.5281/zenodo.7347926 https://doi.org/10.5281/zenodo.7347926 2022.

[2] F. Z. Qureshi and D. Terzopoulos. “Surveillance in virtual reality: System design and multi-camera control.” IEEE Conference on Computer Vision and Pattern Recognition. IEEE 2007.

[3] F. Schindler and R. Trappe “NiceGUI: Web-based user interfaces with Python. The nice way.” https://github.com/zauberzeug/nicegui 2023.

[4] A. Hussein et al. “Autonomous off-road navigation using stereo-vision and laser-rangefinder fusion for outdoor obstacles detection.” in IEEE Intelligent Vehicles Symposium (IV). IEEE 2016.

[5] C. Ding et al. “Collaborative sensing in a distributed PTZ camera network.” in IEEE Transactions on Image Processing 21.7: 3282-3295 2012.

[6] J. Chen and P. Carr. “Autonomous camera systems: A survey.” in Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence 2014.

[7] AS Olagok and H. Ibrahim, and SS Teoh. “Literature survey on multi-camera system and its application.” in IEEE Access 8 (2020): 172892-172922 2020.

[8] M. Chapel and T. Bouwmans. “Moving objects detection with a moving camera: A comprehensive review.” in Computer science review 38 (2020): 100310.

[9] T. Vu and A. Osokin, and I. Laptev. “Context-aware CNNs for person head detection.” in Proceedings of the IEEE International Conference on Computer Vision. 2015.

[10] S. Shao et al. “Crowdhuman: A benchmark for detecting human in a crowd.” in arXiv preprint:1805.00123 (2018).

[11] F. Schroff, and D. Kalenichenko, and J. Philbin. “Facenet: A unified embedding for face recognition and clustering.” in Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[12] T. Lin et al. “Microsoft coco: Common objects in context.” in Proceedings of the ECCV: 13th European Conference 2014.

[13] BE Moore and JJ Corso “FiftyOne” in GitHub https://github.com/voxel51/fiftyone 2020.

[14] M. Cristani, M. Farenzena, D. Bloisi and V. Murino “Background subtraction for automated multisensor surveillance: A comprehensive review” in EURASIP Journal on Advances in Signal Processing 2010(24).

[15] E. Komagal, B. Yogameena “Foreground segmentation with PTZ camera: a survey” in Multimedia Tools and Applications 77 (17) (2018) 22489–22542.

[16] , Y. Zhang et al. “ByteTrack: Multi-Object Tracking by Associating Every Detection Box” in Proceedings of the European Conference on Computer Vision (ECCV), 2022.

Simulation Model Calibration for Condition Monitoring

Aleksandr Subbotin, Thomas Bartz-Beielstein

Technische Hochschule Köln, Institute for Data Science, Engineering, and Analytics
Steinmüllerallee 1, 51643 Gummersbach, Germany
E-Mail: {aleksandr.subbotin,thomas.bartz-beielstein}@th-koeln.de

Abstract

Condition monitoring is a key component of condition-based and predictive maintenance solutions and has applications in a wide range of industries. However, extracting long-term asset condition information from process data is not a trivial process. The objective of this paper is to present the first steps in developing a condition monitoring solution using a hybrid modeling approach. The paper provides an introduction to condition monitoring and hybrid modeling and focuses on the problem of calibration of first principles based simulation. Several possible approaches to model the calibration coefficients that vary during the process simulation were considered. Our results show that the developed piecewise constant approach, together with the tuned version of the Nelder-Mead optimization algorithm, allows to accelerate the calibration process without sacrificing the simulation error.

1 Introduction

The design life of equipment is often conservative because, in practice, actual operating and environmental conditions may differ significantly from those considered in the design. Therefore, during operation, a remaining life assessment is required to determine the actual remaining life of critical equipment, which may be shorter or longer than the design life [4]. However, extracting long-term asset condition information from process data is not a trivial process. One possible solution could be to use a condition monitoring solution based on a hybrid modelling approach—a combination of a first-principles-based simulation model with machine learning algorithms.

This article presents the results of an ongoing research project with an industry partner, and not all project details can be disclosed. The article is organized as follows. Section 2 provides an introduction to condition monitoring. The hybrid modeling approach and examples of its application to condition monitoring are presented in Section 3. Then, a problem of calibration of the first principles based simulation is introduced in Section 4. Section 5 presents a case study to demonstrate and test the developed model calibration approach. Finally, the conclusions are presented in Section 6.

2 Condition Monitoring

Condition Monitoring (CM) is the process of monitoring the condition of industrial assets (manufacturing equipment, machinery, parts, auxiliary systems and components, etc.) during operation. Condition monitoring is a main part of condition-based and predictive maintenance solutions and has applications in a broad range of industries [1]. In general, the development of a CM solution consists of three main parts: data collection, data exploration and processing, and the development of a CM algorithm [2]. Depending on the industry and field of application, all three parts can vary significantly from solution to solution. The data source for CM can be either specially designed and installed sensors [3] or the existing infrastructure used for process monitoring [**?**]. The basic idea behind data-driven condition monitoring is that it is possible to extract some patterns and trends—condition indicators—from a large amount of collected data and infer the deterioration status of equipment for which there are not available or do not exist condition monitoring sensors. Data-driven condition monitoring relies on various data sources and types of measurements acquired during equipment operation and uses various data mining techniques and algorithms [2].

3 Hybrid Modeling

Hybrid modelling is a combination of two paradigms: first principles-based and data driven models into a single architecture (Fig. 1). First-principles (physics-based) models are based on formalized expert knowledge of a problem, including design data, material properties, etc. In contrast, data-driven methods rely only on collected data. Hybrid modelling already has a portfolio

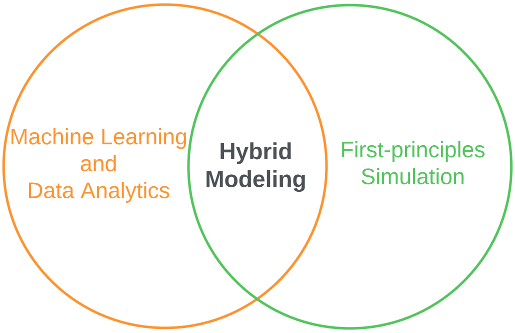


Figure 1: Hybrid modeling is the fusion of two worlds: machine learning and first-principles based simulation.

of applications in the process industry [6] and several examples of applications to condition monitoring. Leturiondo, et al. [8] use hybrid modeling to monitor the condition of rolling bearings. Gálvez, et al. [?] use hybrid modeling for condition monitoring of the heating, ventilation, and air conditioning (HVAC) system in passenger trains. In both works, the authors proceed from the hypothesis that, due to scheduled maintenance and service of equipment, the real measurements, collected by sensors located in the real system, contain very limited information about the degradation of elements, especially in the late stages of degradation. Therefore, in both studies, physics-based models are used to generate synthetic data for operation with known degradation levels and equipment failures. In our study, we would like to propose and test a different way of condition monitoring by using a hybrid modelling approach. We assume that in real measurements it is possible to track the degradation process of the equipment by using detailed and calibrated physics-based simulation of the process. The calibration coefficients of the physics-based model can possibly be used as condition indicators. For this purpose the process of physics-based model calibration should be relatively fast because we plan to trace the changes of physics-based model calibration coefficients during the whole lifetime of the equipment. If our hypothesis is successful, the next step would be to create a data-driven model that could extract condition indicators from process data without the use of simulation and optimization.

4 Simulation Calibration

One of the important steps in the development of a hybrid model is the calibration of the physics-based model (simulation) to better match the industrial data (real measurements). Model calibration is the manual or automated process of estimating and adjusting model parameters (calibration coefficients) by fitting the model output to real data.

The simulation calibration process consists of the following steps (Fig. 2). In the first step, the equipment design data and actual process data (measurements) are prepared and fed into the simulation. The second step is the execution of the process simulation. In the third step, the parameters calculated in the simulation are compared with the real measurements. The simulation error is calculated and passed to the optimization algorithm. The goal of the optimization algorithm in the fourth step is to minimize the simulation error by finding optimal calibration coefficients. The process is repeated until the desired accuracy is achieved or the number of simulations is exhausted. The choice of an effective optimization algorithm for model calibration is not obvious and requires considerable experience and some experimentation.

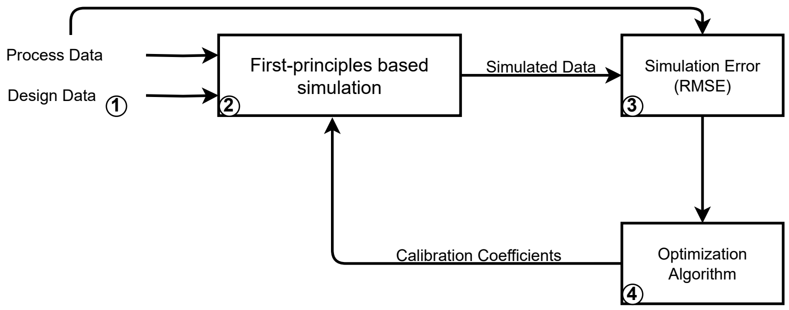


Figure 2: Flowchart of the Simulation Calibration Process

4.1 Calibration Coefficients

The process simulation has a set of calibration coefficients. From the available coefficients, we have selected two that could potentially be used as condition indicators. Since they are just coefficients without units, we can just label them as Calibration Coefficient 1 (CC1) and Calibration Coefficient 2 (CC2). The Calibration Coefficient 1 does not change during the one process simulation. The Calibration Coefficient 2, on the other hand, decreases during the process simulation (Fig. 3), but can be modeled as a constant for simplification. Modeling CC2 as a constant throughout the process simulation leads to oversimplification and is suitable for approximate estimation, but doesn't fit the purpose of what we are working on.

Due to the complex behavior of the CC2, two other calibration approaches were considered.

Approach 1 (exponentially decaying curve): In theory, the behavior of CC2 can be modeled with an exponentially decaying curve (Fig. 3) described by the equation $CC2 = A \cdot \exp\left(-\frac{t}{T}\right) + B$, where t is the simulation time; A, B and T are unknown parameters. Unfortunately, attempts to determine optimal values for the parameters A, B and T have not been successful because this approach requires too many time-consuming simulations.

Approach 2 (piecewise constant): Because of the difficulties we encountered in determining the coefficients of the exponential curve, we developed a different

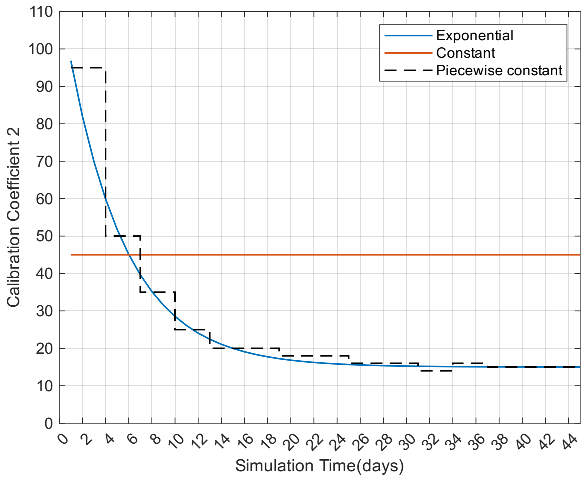


Figure 3: Three ways to model CC2: 1–Constant throughout the simulation; 2–Exponentially decaying curve; 3–Piecewise constant

approach. The simulation is divided into many small steps (overlapping windows) in which the CC2 is modeled as a constant. Then, using an optimization algorithm, the optimal value of the CC2 is calculated sequentially for each step. In this approach, the CC1 is optimized only on the first step.

4.2 Optimization Algorithm

The process of calibration of the simulation model can be considered as simulation-based optimization problem or Derivative-Free Optimization (DFO) problem. Simulation optimization is a very broad topic that involves the use of algorithms that come from many different fields, have connections to many different disciplines, and have been used in many practical applications [7]. DFO can be considered as a sub-field of simulation optimization. Most algorithms in DFO are specifically designed to consider that function evaluations or simulations are expensive.

The optimization objective is to minimize the difference between the real data and the simulated data. In our case, the Root Mean Square Error (RMSE) between the simulated parameter and the actual (observed) measurement used as a cost function. Since every objective function evaluation requires a time-consuming process simulation and gradient information is not available, DFO algorithms are best suited for our task. The list of DFO algorithms built into the MATLAB environment we use, including the Statistics and Machine Learning Toolbox, the Optimization Toolbox, and the Global Optimization Toolbox, is given in Table 1.

Table 1: List of DFO methods built into MATLAB and available in MATLAB Toolboxes

Algorithm	**Matlab Function**
Nelder-Mead Simplex Method	fminsearch
Golden Section Search	fminbnd
Pattern Search	patternsearch
Surrogate Optimization	surrogateopt
Genetic Algorithm	ga
Particle Swarm Solver	particleswarm
Simulated Annealing	simulannealbnd
Bayesian Optimization	bayesopt

5 Experiment

The goal of our work is not a detailed benchmarking of algorithms, but the selection of the most efficient optimization algorithm for our concrete problem. We formulated several requirements for the optimization algorithm:

- The calibration coefficients have physical bounds, e.g., from 0% to 100%, and the simulation cannot take values outside these limits. For this reason, the optimization algorithm must be constrained.
- Detailed simulation takes time, so the algorithm must use a limited number of function evaluations (as few as possible).
- Since we have several coefficients to optimize, the optimization algorithm should support multi-variable optimization.

The MATLAB software package contains several algorithms suitable for our problem. We compare three of them: a constrained version of the Nelder-Med simplex method, Pattern search and Bayesian optimization. Nelder-Mead and Pattern Search are widely used direct search methods [10]. In the MATLAB implementation, the Nelder-Mead algorithm does not support constraints. However, there is a popular community version of the algorithm with constraints. Bayesian optimization is a global optimization algorithm recently become extremely popular for tuning hyperparameters in machine learning models [11].

5.1 Experiment Setup

For the experiment, we used the industrial process simulation with a total duration of 85 days. According to the developed piecewise constant approach, the simulation is divided into 28 steps of 5 days each (window size is 5 days with an overlap of 2 days). All optimization algorithms are set to the same maximum number of iterations, 30 for the first step (window) and 15 for all subsequent steps. For the Nelder–Mead and Pattern Search algorithms, the starting point is the optimal solution from the previous step. The optimization constraints are the same for all algorithms.

For the Nelder-Mead algorithm, we carried out two experiments, one with the default settings and the other with a modified tolerance for both the objective function value and the variable.The tolerance settings of the Pattern Search algorithm are equivalent to the tuned version of the Nelder-Mead algorithm, and for Bayesian Optimization, there are no tolerance settings. The settings of the experiments are summarized in the Table 2.

5.2 Experiment Results

All three optimization algorithms solved the optimization problem, but with a significantly different number of objective function evaluations and a slightly different RMSE values after optimization. Since we have 28 steps in our

Table 2: Experiment Setup

Short Name	Algorithm	Tolerance
NM_1	Nelder-Mead	Default: 1e-4
NM_2	Nelder-Mead	TolFun: 1e-03 TolX: 0.1
BO	Bayesian Optimization	Does not support tolerance settings
PS	Pattern Search	TolFun: 1e-03 StepTolerance: 0.1 MeshTolerance: 0.1

proposed piecewise constant approach, the median RMSE value is used to compare the optimization algorithms. Figure 4 shows the number of evaluations of the objective function (simulations) that are used by each of the algorithms at each of the 28 steps. The Nelder-Mead algorithm with the default setting uses slightly more function evaluations than it was limited to use. However, by tuning the optimization tolerance, we were able to significantly reduce the number of function evaluations while keeping the median RMSE at the same level. This is illustrated in Figure 5, which shows the total number of function evaluations and the median RMSE value over 28 steps. The Pattern Search algorithm, with the same tolerance settings as the tuned Nelder-Mead algorithm, required more objective function evaluations and has a slightly worse RMSE values. As expected, Bayesian optimization without tolerance settings uses the entire limit of objective function evaluations, but does not show better RMSE values. The results of the experiment are summarized in Table 3. It should be noted that we also found that the constrained version of the Nelder-Mead algorithm can have convergence problems when the solution is very close to the boundary, which is not a problem for Bayesian optimization and Pattern search. The resulting values of CC2 after the optimization process are shown in Figure 6.

Table 3: Experiment Result

Algorithm Short Name	Total Number of Function Evaluations	Optimal Value CC1	Median Simulation Error
NM_1	462	1.087	0.0025
NM_2	**218**	1.087	**0.0025**
BO	435	1.094	0.0028
PS	291	1.100	0.0031

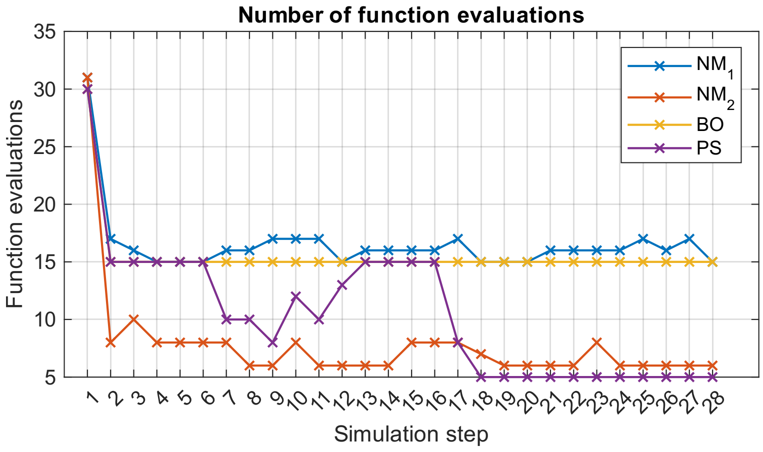
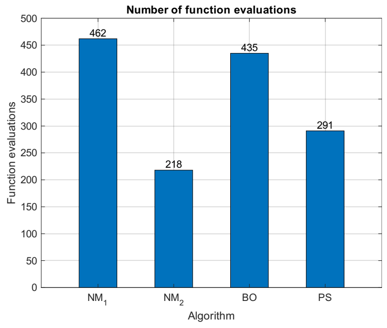


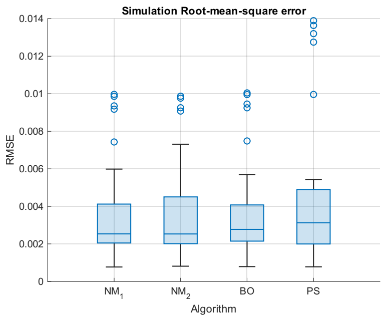
Figure 4: Number of function evaluations for each calibration step. Tuned version of the Nelder-Mead algorithm (NM_2) uses less objective function evaluations, especially in the first 17 steps, than other optimization methods.

6 Conclusion

This paper presents the first steps in the development of a condition monitoring solution using hybrid modeling. In this phase, we considered several possible ways to model calibration coefficients that vary during the process simulation. To address this issue, the piecewise constant approach was developed and then tested with three different optimization algorithms and different optimization tolerance settings. The experimental results show that the simulation calibration process can be significantly accelerated by tuning the tolerance parameter



(a) Number of Function Evaluations



(b) Simulation RMSE after optimization

Figure 5: Experiment Results. a) Total number of function evaluations performed by each algorithm; b) Simulation RMSE statistics over 28 calibrations steps for each algorithm

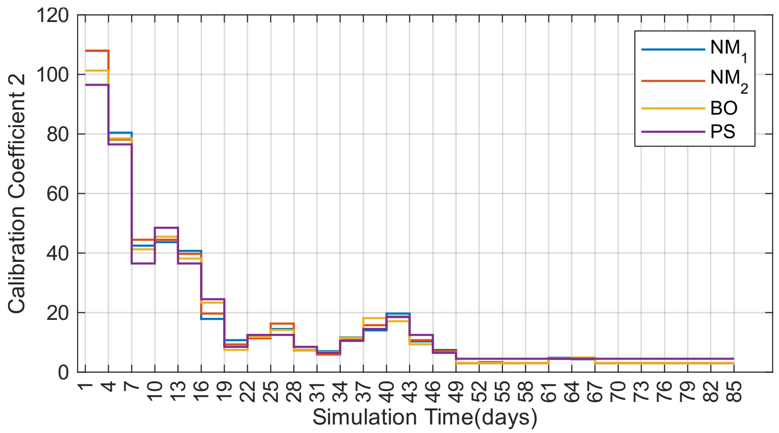


Figure 6: Resulting CC2 values after calibration procedure. The values are only slightly different from each other, but the number of function evaluations that are used is completely different, see Fig. 5

of the optimization algorithm without sacrificing the simulation error. The best results were obtained using the tuned version of the Nelder-Mead algorithm, but the optimal balance between optimization speed and simulation error needs to be further investigated. The next step is to use a developed simulation calibration approach to determine the potential of using the calibration coefficients as condition indicators in a condition monitoring solution.

References

[1] J. Naik, A. Acharya, J. Thaker "Revolutionizing condition monitoring techniques with integration of artificial intelligence and machine learning" In: *Materials Today: Proceedings* In Press https://doi.org/10.1016/j.matpr.2023.08.262 2023.

[2] R. Qi, J. Zhang, K. Spencer "A Review on Data-Driven Condition Monitoring of Industrial Equipment" In: *Algorithms* 16 Pages 1-9 https://doi.org/10.3390/a16010009 2023.

[3] P. Strauß, M. Schmitz, R. Wöstmann, J. Deuse "Enabling of Predictive Maintenance in the Brownfield through Low-Cost Sensors, an IIoT-Architecture and Machine Learning" In: *IEEE International Conference on Big Data (Big Data)* Pages 1474-1483 https://doi.org/10.1109/BigData.2018.8622076 2018.

[4] I. Animah, M. Shafiee "Condition assessment, remaining useful life prediction and life extension decision making for offshore oil and gas assets" In: *Journal of Loss Prevention in the Process Industries* 53 Pages 17-28 https://doi.org/10.1016/j.jlp.2017.04.030 2018.

[5] P. Zürcher, S. Badr, S. Knüppel, H.Sugiyama, "Data-driven equipment condition monitoring and reliability assessment for sterile drug product manufacturing: Method and application for an operating facility" In: *Chemical Engineering Research and Design* 188, Pages 301-314 https://doi.org/10.1016/j.cherd.2022.09.005 2022.

[6] J. Glassey, M. von Stosch "Hybrid modeling in process industries" CRC Press https://doi.org/10.1201/9781351184373 2018.

[7] A. Satyajith, S. Nikolaos V., S. Bikram, B. Scott J. "Simulation optimization: a review of algorithms and applications" In: *Annals of Operations Research* Pages 351–380 2015.

[8] U. Leturiondo, M. Mishra, D. Galar, O. Salgado "Synthetic data generation in hybrid modelling of rolling element bearings". In: *Insight*

- *Non-Destructive Testing and Condition Monitoring* Pages 395-400 2015.

[9] A. Gálvez, J. Rubio, D. Seneviratne, A. Gonzalez, A. Jimenez, U. Martinez-de-Estarrona, D. Galar, E Juuso “Hybrid Models and Digital Twins for Condition Monitoring: HVAC System for Railway”. In: *SNE Simulation Notes Europe* Pages 121–126 2021.

[10] J. Larson, M. Menickelly, S. M. Wild “Derivative-free optimization methods”. In: *Acta Numerica* 28 Pages 287-404 https://doi.org/10.1017/s0962492919000060 2019.

[11] P. I. Frazier “A Tutorial on Bayesian Optimization”. In: *arXiv 1807.02811* https://arxiv.org/pdf/1807.02811.pdf 2018.

MLOps for Scarce Image Data: A Use Case in Microscopic Image Analysis

Angelo Yamachui Sitcheu[1], Nils Friederich[1,2], Simon Baeuerle[1,3], Oliver Neumann[1], Markus Reischl[1], Ralf Mikut[1]

[1]Institute for Automation and Applied Informatics,
[2]Institute of Biological and Chemical Systems,
Karlsruhe Institute of Technology, Hermann-von-Helmholtz-Platz 1,
76344 Eggenstein-Leopoldshafen
E-Mail: angelo.sitcheu@kit.edu
[3]Robert Bosch GmbH,
Markwiesenstraße 46, 72770 Reutlingen

Abstract

Nowadays, Machine Learning (ML) is experiencing tremendous popularity that has never been seen before. The operationalization of ML models is governed by a set of concepts and methods referred to as Machine Learning Operations (MLOps). Nevertheless, researchers, as well as professionals, often focus more on the automation aspect and neglect the continuous deployment and monitoring aspects of MLOps. As a result, there is a lack of continuous learning through the flow of feedback from production to development, causing unexpected model deterioration over time due to concept drifts, particularly when dealing with scarce data. This work explores the complete application of MLOps in the context of scarce data analysis. The paper proposes a new holistic approach to enhance biomedical image analysis. Our method includes: a fingerprinting process that enables selecting the best models, datasets, and model development strategy relative to the image analysis task at hand; an automated model development stage; and a continuous deployment and monitoring process to ensure continuous learning. For preliminary results, we perform a proof of concept for fingerprinting in microscopic image datasets.

1 Introduction

In the field of image analysis, Machine Learning (ML), particularly its subfield Deep Learning (DL) is being explored to model complex problems such as image registration, classification, segmentation, object detection and tracking [1, 29]. In this context, the goal of ML is to build a model that generalizes across different images for the same analysis task, for example, image segmentation. Therefore, the research community focuses more on developing new methods that achieve better performances and are computationally more efficient [17]. While developing the ML model offline seems easy and cheap, operationalizing the model, which means, deploying the model and maintaining its performance over time, still faces numerous challenges [10]. Unlike standard non ML-based software whose operations include building, testing, deployment and monitoring, ML systems are more complex due to the two new components model and data, and their direct, and generally challenging relationship. These systems are often a source of high technical debt when not operationalized consequently [28]. Similarly to standard non ML-based software whose lifecycle follows the Development and Operations (DevOps) scheme [3], ML systems have their development and operation paradigm known as Machine Learning Operations (MLOps)(see Section 2.1).

Nevertheless, as shown by the multitude of approaches developed [17], it is very challenging to develop and operationalize one single model that generalizes across different images for the same task. Many experts develop new models for the same image analysis task when new input data is available. Due to the time and cost expensiveness of this process, they often focus more on developing the model and neglect the continuous monitoring and deployment aspect [16], resulting in a lack of continuous learning through the flow of feedback from production to development, and therefore to unexpected model deterioration over time [28]. Additionally, the initial training data in this field is often scarcity-prone in quantity and quality due to the tediousness of several tasks such as image acquisition and image annotation requiring skilled and expensive experts. This often leads to a dataset containing less relevant data from the experiments and more noise.

A potential solution is to create a production-oriented machine learning approach that harnesses the full range of available and well-established datasets and models to improve the efficiency of image analysis across multiple tasks. This will be the main research goal of our work. In our paper, MLOps for sparse image analysis will be explored. We propose a long-term vision holistic approach to enhance biomedical image analysis that includes: a fingerprinting process that enables selecting the best models, datasets, and model development strategy relative to the task at hand, therefore making use of available models and datasets; an automated model development stage; and a continuous deployment and monitoring process.

Our work is organized in the following manner. Section 2 explains the fundamental concepts related to our work and provides an overview of other related works, and Section 3 details the proposed approach. In Section 4, the preliminary experiments carried out during the investigations are described. Their results are presented and discussed in Section 5. Section 6 summarizes our investigation and outlines future research directions.

2 Background and Related Work

In this section, we provide fundamental notions around MLOps and other important fields related to our work. We also present state-of-the-art approaches related to ours in general and regarding the different building blocks in particular.

2.1 MLOps

MLOps is a discipline that combines ML with software engineering paradigms such as DevOps and data engineering to enable efficient deployment and operationalization of ML systems [16, 30]. It can be seen to a certain extent as DevOps for ML systems. The key differences and similarities between DevOps and MLOps can be seen in Figure 1. On the one hand, both entail two main concepts Continuous Integration and Continuous Deployment.

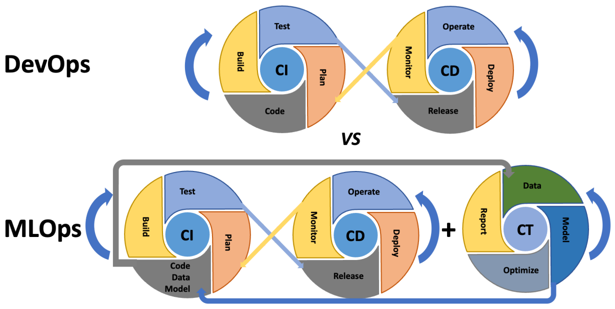


Figure 1: DevOps vs. MLOps [12]. While DevOps entails applying the main concepts of Continuous Integration (CI) and Continuous Deployment (CD) on code only, MLOps has the two new components data and model, which are added to the code component. Additionally, MLOps has a new concept named Continuous Training (CT).

- Continuous Integration (CI): consists in automatically building, testing and validating source code.
- Continuous Deployment (CD): enables frequent release cycles by automatically deploying the software in production. It distinguishes itself from continuous delivery by automatizing the deployment process.

On the other hand, in addition to normal source code, ML systems bring two new components: data and model, which are code-independent and therefore maintained separately from the code. This leads to the creation of pipelines to perform the complete processing. As a result, a novel concept specific to MLOps named Continuous Training (CT) emerges, which involves automatically retraining and serving the models. Furthermore, the CI and CD concepts in MLOps differ from those in DevOps in that CI does not only include integrating new code and components but also new data, models and pipelines, and CD does not deploy a single software package, but a complete ML training and/or serving pipeline. Additionally, continuous monitoring, i.e., automatically monitoring the IT system to detect potential problems such as compliance issues and security threads and address them, is extended in MLOps by monitoring production data and model performance metrics. This enables real-time understanding of model performances [32].

Although MLOps is a relatively new field, important work and progress have been made. While some researchers focus on properly defining MLOps and providing an overview of its concepts and best practices [16, 30, 32], others investigate the challenges faced in ML systems operationalization. Tamburri highlights in [31] the limited attention given to MLOps within academia and the lack of data engineering skills in academia as well as in industry. Renggli et al.[25] and Granlund et al.[7] extend on this and show that one massive problem in MLOps is data management. This is mostly due to the strong dependency between model performance and data quality. The cloud architecture center of Google proposes in [2] how to automate ML workflows. They classify MLOps into three different levels: MLOps Level 0 that refers to classical ML pipelines with no CI, no CD and manually operated workflows totally disconnected from the ML system, MLOps Level 1 in which data validation, model validation, continuous delivery and CT are introduced, and MLOps Level 2 where CI, CT and continuous delivery are fully explored. They apply semi-automated deployment in a pre-production environment and manual deployment in the production environment.

Several MLOps tools have been developed. A non-exhaustive list of tools and their features can be found in [16, 24, 30, 32]. There is no ideal tool, as each tool covers different MLOps aspects. In practice, tools are often combined to achieve maximal efficiency. However, the majority of tools focus on model versioning and tracking and ignore dataset versioning. This impedes the ability to reproduce results and renders it reliant on the coding practices of skilled experts [36]. In industry particularly, due to IT-Software's large and complex nature, the MLOps tools are often diverse and must match a specific established strategy.

Despite advancements in the MLOps domain, there are very few published real-world use cases in which MLOps is clearly designed, explained and applied. One use case found is Oravizio [8], a medical software for evaluating the risks associated with joint replacement surgeries. The researchers had four different risk models from which the best was selected and deployed. One issue they faced during development was related to data management, due to the multitude of data formats they had to process. A second use case is SemML [38] in which the researchers propose a ML-based system that leverages semantic technologies to enhance industrial condition monitoring for electric resistance welding. Their workflow enables them to reuse and enhance ML pipelines over time based on new input data. A third example is microbeSEG [26], a DL-based tool for instance segmentation of objects. The researchers automate several data management tasks and model building processes. They, however, do not focus on monitoring and deployment.

We found two ongoing works connected to ours. Firstly, Friederich et al.explore in [4] Artificial Intelligence (AI) to encode dynamic processes. They propose a MLOps-based pipeline, in which active learning will be used to predict and record potentially important events during experiments in light microscopy. Secondly, Zárate et al.present K2E [36], a new approach to governing data and models. They investigate MLOps environments for creating, versioning and tracking datasets as well as models. They are still in the conceptualization step and plan to build their platform by following the Infrastructure as Code (IaC) paradigm. To the best of our knowledge, there is no MLOps approach similar to ours, particularly in the field of biomedical image analysis.

2.2 AutoML and Meta-learning

Automated Machine Learning (AutoML) is the process of automating different stages of the ML development cycle. These stages often include data preprocessing, feature engineering, model training and model evaluation. AutoML addresses problems such as Dynamic Algorithm Configuration (DAC), Hyperparameter Optimization (HPO), Combined Algorithms Selection and Hyperparameter Optimization (CASH) and Neural Architecture Search (NAS) [13]. Several researchers work towards building fully automated systems for their applications. For example, Meisenbacher et al.explain in [18] how to achieve AutoML for forecasting applications. They define five levels of automation for designing and operating forecasting models.
Numerous tools have been developed, each addressing specific AutoML subproblems. The developed tools often log metadata such as hyperparameters tried, pipelines configurations set, model evaluation results, learned weights and network architectures. Based on these experiences and other dataset meta-

data, a model is trained with the aim to adapt faster to new tasks [11]. This is meta-learning, also known as learning to learn.

While it is clear that AutoML can be combined with MLOps to enable high level automation in the ML development lifecycle [2, 30], exploring the output of this combination for meta-learning seems not to be investigated. The information acquired during the continuous monitoring stages appears not to be widely researched in combination with AutoML or meta-learning.

2.3 Scarce Image Data

Scarce image data is a massive problem in ML. In the field of image analysis, particularly in biomedicine, the acquisition and annotation of images must be done by highly skilled experts, require a considerable amount of time and resources, and are often error-prone [27]. As a result of these constraints, the amount of image data present is either small in quantity, dominated by noise, or small in quality, very weakly or not labeled. A lot of approaches have been developed to address data scarcity. On the one hand, there are image processing techniques to augment image data such as scaling, rotation and cropping. On the other hand, DL methods such as Generative Adversarial Network (GAN) [6] and Variational Autoencoder (VAE) [14] are used to generate synthetic images. These DL approaches often use Self-Supervised Learning (SSL) to understand better how data points are sampled [22]. We notice however that these solutions often focus more on image classification tasks.

2.4 Image Fingerprinting

Fingerprinting is used in image processing to generate concise and distinct representations of images. It is useful for diverse objectives such as image retrieval, copyright protection, or image similarity analysis. We focus on image fingerprinting to measure the similarity between images and/or datasets.

Ranging from simple pixel distribution methods [21] to DL approaches [15], there are numerous methods to compute similarity between images. Godau and Maier-Hein present in [5] an image fingerprinting approach that consists of embedding images along with their labels in a fixed-length vector in order to capture semantic similarities in biomedical image datasets. Molina-Moreno et al.[22] build an autoencoder (AE), train this using SSL and obtain a two-dimensional latent space in which the disposal of image datasets displays their similarity. Such similarity measures enable researchers to apply transfer learning for their respective tasks. This is often achieved by selecting suitable pre-trained models and/or datasets for a new task based on the similarity measure obtained. Nevertheless, most state-of-the-art methods compute this similarity on a dataset level or on an image level and do not investigate the computation on an image patch level.

To fill the observed gaps in the context of image analysis, we propose a new approach, which is fully described in the following section.

3 Methodology

This section describes the proposed methodology to address the different problems identified and mentioned in Section 1 and Section 2. We first provide a global description of the system and subsequently delve into its individual building blocks.

3.1 Overview of the proposed approach

Figure 2 shows a conceptual architecture of our method. The main entering point is a scientist bringing a new image analysis task modeled as the triple(I, A, T_a), where I represents the image dataset, A the performance analysis metric, e.g., F1-Score, and T_a the task, e.g., image classification. At a time t, the performance metric A_t can be computed as defined in Equation (1), in which m_t represents the current model, I_t the current image dataset and $Y_{MD,t}$ the metadata relative to I_t. The following stages of our approach attempt to find the best model m^* defined through Equation (2).

$$A_t = m_t(T_a, I_t, Y_{MD,t})_{m \in \mathbb{M}} \tag{1}$$

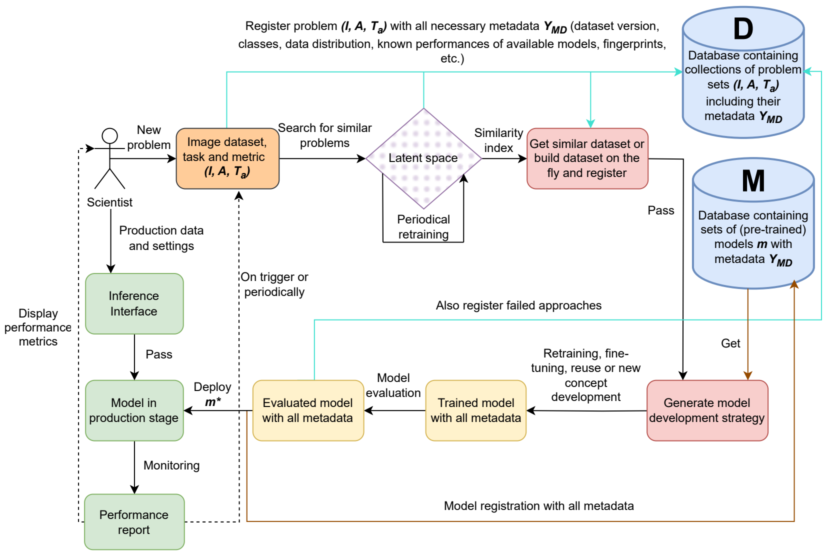

Register problem (I, A, T_a) with all necessary metadata Y_MD (dataset version, classes, data distribution, known performances of available models, fingerprints, etc.)
D
Database containing collections of problem sets (I, A, T_a) including their metadata Y_MD
New problem
Scientist
Image dataset, task and metric (I, A, T_a)
Search for similar problems
Latent space
Periodical retraining
Similarity index
Get similar dataset or build dataset on the fly and register
Pass
M
Database containing sets of (pre-trained) models m with metadata Y_MD
Production data and settings
Inference Interface
On trigger or periodically
Display performance metrics
Also register failed approaches
Get
Pass
Deploy m*
Model in production stage
Evaluated model with all metadata
Model evaluation
Trained model with all metadata
Retraining, fine-tuning, reuse or new concept development
Generate model development strategy
Monitoring
Performance report
Model registration with all metadata

Figure 2: Abstract architecture of the proposed MLOps-based image analysis approach. The turquoise arrows indicate exchanges with the data database $\mathbb{D}$ and the brown arrows exchanges with the model database $\mathbb{M}$. The black arrows model the information flow between different building boxes of the system and the dashed lines feedback from production to development and to the scientist. The orange box represents the input provided by the scientist. The red boxes are the meta-learning system that handles dataset and algorithm selection. The yellow boxes display the AutoML pipeline. The green boxes represent the continuous deployment and monitoring stage for production.

$$m^*(t) = \underset{m \in \mathbb{M}}{\arg\max}\,(A_t) \tag{2}$$

3.2 Image Similarity

At $t = 0$, the newly provided problem is initialized and registered in the image database $\mathbb{D}$. The initial image dataset I_0 along with the task T_a, the performance analysis metric A_0 set to $-\infty$ and other metadata $Y_{MD,0}$ such as dataset version, classes, data distribution, fingerprints, known performances of available models for the same task T_a, etc. build an entry in $\mathbb{D}$. In order to discover images similar to I_0, its fingerprints $f_i(I_0)$ are subsequently computed, saved and compared to those already present in $\mathbb{D}$ through a latent space embedding built in advance. This embedding will be periodically retrained to verify that the performance meets expectations continuously.

As emphasized in [5, 22], with such fingerprints, deploying new ML models for biomedical applications can be seed up by selecting appropriate pre-training models. Furthermore, this could solve scarcity issues by finding appropriate images for image augmentation on the one hand and for pre-training on the other hand.

3.3 Model Development Strategy

Inspired by early attempts in [35] and [37], the model development strategy aims to leverage the concept of meta-learning for scarce data. Given the tuple (I, A, T_a), the metadata Y_{MD} acquired during the registration phase and the computed fingerprints $f_i(I)$, the goal is to find the top $k (k \geq 1)$ cheapest and efficient model developing approaches. These approaches include both model and dataset and could range from model selection together with potential weights and development strategy, i.e., fine-tuning, retraining, to dataset selection or conception on the fly. Dataset conception in this context involves selecting the closest images to I in $\mathbb{D}$ and using these as training data.

The reason we envision using not only $f_i(I)$ is to minimize error propagation when fingerprinting is inaccurate. In this case, the meta-learner will solely rely on the metadata Y_{MD}.

3.4 Automatic Model Development

This stage is a direct application of the strategy established previously. It performs AutoML according to the strategy defined. Although AutoML is more computationally expensive than normal ML techniques, it is more efficient and faster in producing the best-trained model [2, 20]. It can be combined to MLOps to improve a project's automation level, therefore reducing the pipeline configurations overhead. For example, in a retraining process or new training, it could help solve the HPO problem faster at time t.

All development runs, including failing approaches, will be tracked and recorded in the model database $\mathbb{M}$. They would serve as input data for the meta-learner in the previous stage and may help identify flaws in the data or in the whole system. The best model m^* is sent in the last stage.

3.5 Continuous Deployment and Monitoring

The best model developed m^* is continuously deployed as a service and monitored during production. On the one side, we envision a deployment framework fulfilling fundamental requirements such as independency towards the ML Framework, rapid maintenance, accessibility, and parallel computing. Despite the existence of numerous deployment frameworks, we will focus on Representational State Transfer (REST) frameworks such as DEEPaaS [**?**] and EasyMLServe [23]. On the other side, the performance metric A_t as well as defined metrics by the scientist will be continuously monitored over time and reported. This monitoring will be particularly investigated, as it could help identify potential concept drift and decrease in performances, and act accordingly by triggering the complete framework from the start.

The high information reuse, which will be investigated in this work, will serve the purpose of exploiting available feedback and enabling transfer learning. Our approach will mostly be done using the Python programming language, due to its rich ecosystem of data science libraries and its expansive and highly engaged user community. We also plan to apply containerization with Docker [19] and its microservices, as it guarantees platform independency and enforces reproducibility.

4 Preliminary Experiments

This section describes the current state of the investigation. The experiments performed mainly focus on the first stage of our approach described in Section 3 which consists in building a latent space embedding in which image data can be represented along with their similarities. To this end, we build an autoencoder whose architecture will be depicted in Section 4.1, and evaluate

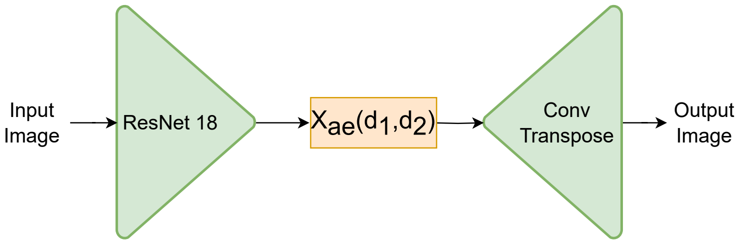


Figure 3: Autoencoder. The encoder is based on the ResNet18 architecture, the latent space representation is a two-dimensional space, and the decoder entails stack transpose convolutional layers.

it on biomedical image datasets presented in Section 4.2. Our implementation can be found in [33].

4.1 Autoencoder

An autoencoder is a special type of neural network that takes an input x, transforms it in a compressed and informative representation x_{ae} and reconstructs the initial input based on x_{ae}. The goal is to find an encoding function $f : x \mapsto x_{ae}$, and a decoding function, $g : x_{ae} \mapsto \hat{x}$ such that the difference between the reconstructed input $\hat{x}$ and the initial input x is minimal. This difference is measured by a loss $L : \min_{f,g} L(x, g(f(x)))$. The goal is to obtain a powerful representation x_{ae} that can be used for various tasks.

The architecture of our autoencoder can be seen in Figure 3. Our encoder is a vanilla (standard) ResNet18 [9] based neural network. The final fully connected layer is replaced by a linear layer mapping the 512 feature channels vector to a two-dimensional vector. Because we are more interested in x_{ae}, we build a simple decoder that entails five stacked convolutional transpose layers. We choose the Mean Squared Error (MSE) as loss function that computes the difference between the original input image and the reconstructed output image, and Adam as optimizer.

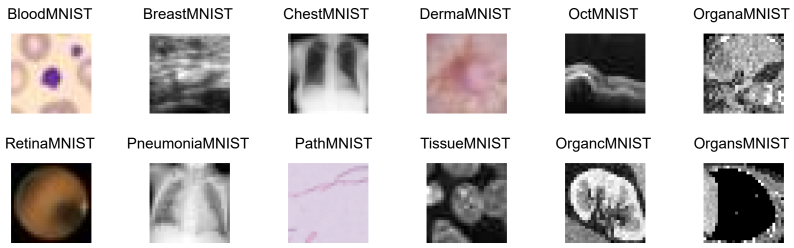


Figure 4: MedMNIST v2 2D datasets

4.2 Datasets

To evaluate our autoencoder, we select all the 2D image datasets of the MedMNIST v2 [34] dataset, a benchmark dataset for 2D and 3D biomedical image classification. The 12 selected datasets can be found in Figure 4. They consist of eight gray-scale image datasets (BreastMNIST, ChestMNIST, OctMNIST, OrganaMNIST, OrgancMNIST, OrgansMNIST, PneumoniaMNIST and TissueMNIST) and four color image datasets (BloodMNIST, DermaMNIST, RetinaMNIST and PathMNIST). The autoencoder is trained, validated and tested on the respective datasets splits. All the images are processed as $3 \times 32 \times 32$ images. For our architecture to be usable on all these datasets, the gray-scale images were loaded and processed as three-channel images.

5 Results and Discussions

We present in this section the results of the experiments performed in the previous section and discuss these.

Figure 5 shows the latent space representation of $N = 10000$ test samples collected from the 12 2D image datasets presented in the previous section. We notice that the color image datasets BloodMNIST, DermaMNIST and PathMNIST build a cluster at the top left. This is most likely due to the close distribution of the pixel values of the respective images. These color image datasets are, however dissimilar to the color image dataset RetinaMNIST, in

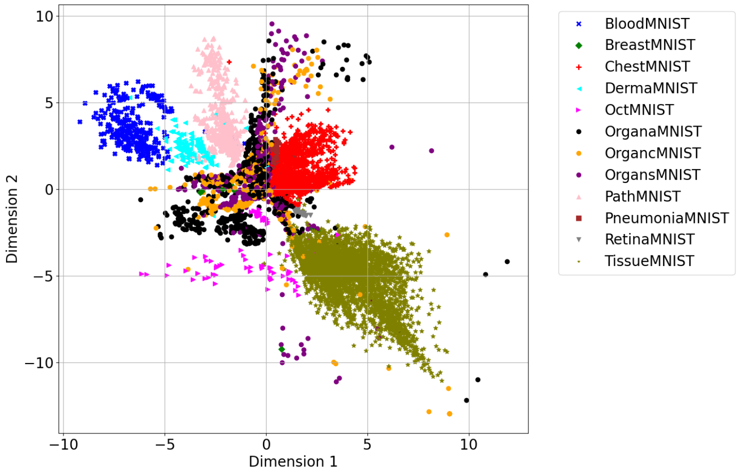


Figure 5: Latent space representation of the MedMNIST v2 2D datasets (test sets)

which the images of the retina all have a black contour. The OrganaMNIST, OrgancMNIST and OrgansMNIST datasets are mixed and almost not differentiable, leading to a non-identification of particular structures in the latent space. This is because all three datasets have images of the same organ taken on different views. In OrganaMNIST, the images are acquired on an axial view, in OrgancMNIST in a coronal, and in OrgansMNIST in a sagittal view. Nevertheless, all three datasets have multiple outliers that may hinder the model in positioning new incoming images.

A better view of the latent space is provided in Figure 6, in which the mean of all embedding vectors of the test images are computed for each dataset. We notice four main clusters. The first cluster entails the three-channel image datasets BloodMNIST, DermaMNIST and PathMNIST, the second cluster in which OrganaMNIST, OrgancMNIST and OrgansMNIST are very close, as well as PneumoniaMNIST and ChestMNIST. The third cluster consists of BreastMNIST and RetinaMNIST, and the final cluster of TissueMNIST and OctMNIST.

This autoencoder shows a promising ability to capture similarities between images. Despite being in the early stages of our investigation, we strongly believe

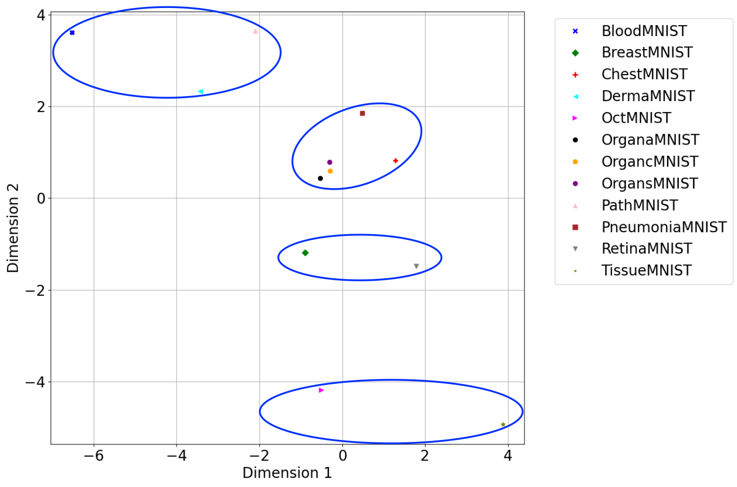


Figure 6: Mean value latent space representation of the MedMNIST v2 2D datasets (test sets)

that this approach to solving the image similarity question is encouraging and could be fine-tuned to identify specific structures in image crops.

6 Conclusion and Future Work

In this paper, we presented a new approach to improve biomedical image analysis. Our approach aimed to apply MLOps to image analysis tasks, particularly when the image dataset is scarce. To achieve this goal, we presented a multi-stage framework that leverages the existence of models and benchmark datasets to solve a given task. The first stage enables us to find image datasets similar to the images of the given task. The second stage applies meta-learning to select the best model development strategy for the given task. This strategy is executed via the third stage of AutoML. The final stage deploys the best-trained model and monitors the model's performance continuously to achieve optimal performance.

The preliminary experiments carried out in this paper mostly focused on the first stage, which consists in computing image fingerprints to identify similar datasets. We built a ResNet18-based autoencoder and showed that the resulting 2D latent space representation is interpretable enough to find similarities between images. We, however, faced some challenges when the images are all from the same object but taken from different angles and focused only on 2D image datasets.

Our future research will, therefore, focus on extending the image fingerprinting process to 3D image datasets and even to the level of image patches and improving the representation of different artifacts. Understanding the image at this granularity level would enable us to generate data or labels to solve the data scarcity problem. We also plan to investigate the effects of outliers, as they may highly impact the similarity measurements. Finally, we will continue our research on the other stages of the proposed approach, including meta-learning for biomedical image analysis, AutoML and efficient model deployment and monitoring.

Acknowledgements

This project is funded by the Helmholtz Association under the program "Natural, Artificial and Cognitive Information Processing (NACIP)". We would like to thank the Helmholtz Artificial Intelligence Cooperation Unit (Helmholtz AI) for providing the computing infrastructure via the Helmholtz AI COmpute REssources (HAICORE). We also extend our gratitude to all those who provided assistance, even if not explicitly mentioned here.

The authors have accepted responsibility for the entire content of this manuscript and approved its submission. We describe the individual contributions of A. Yamachui Sitcheu (AYS), N. Friederich (NF), S. Baeuerle (SB), O. Neumann (ON), M. Reischl (MR), R. Mikut (RM): Conceptualization: AYS, NF, RM; Methodology: AYS, NF; Software: AYS, SB; Investigation: AYS, NF, SB; Writing – Original Draft: AYS; Writing – Review & Editing: AYS, NF, SB, ON, MR, RM; Supervision: RM; Project administration: RM; Funding Acquisition: MR, RM.

References

[1] H.-P. Chan, R. K. Samala, L. M. Hadjiiski, and C. Zhou. Deep Learning in Medical Image Analysis. *Deep Learning in Medical Image Analysis: Challenges and Applications*, pages 3–21, 2020. Springer.

[2] G. Cloud Architecture Center. MLOps: Continuous delivery and automation pipelines in machine learning. *Google*. https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning Last Accessed: 10.08.2023.

[3] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano. DevOps. *IEEE Software*, 33:94–100, 05 2016.

[4] N. Friederich, A. Yamachui Sitcheu, O. Neumann, S. Eroğlu-Kayıkçı, R. Prizak, L. Hilbert, and R. Mikut. AI-based automated active learning for discovery of hidden dynamic processes: A use case in light microscopy. In *Proceedings of the 33st Workshop on Computational Intelligence, Berlin*, 2023. Accepted.

[5] P. Godau and L. Maier-Hein. Task Fingerprinting for Meta Learning in Biomedical Image Analysis. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part IV*, page 436–446, Berlin, Heidelberg, 2021. Springer-Verlag.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[7] T. Granlund, A. Kopponen, V. Stirbu, L. Myllyaho, and T. Mikkonen. MLOps Challenges in Multi-Organization Setup: Experiences from Two Real-World Cases. In *1st IEEE/ACM Workshop on AI Engineering - Software Engineering for AI, WAIN@ICSE 2021, Madrid, Spain, May 30-31, 2021*, pages 82–88. IEEE, 2021.

[8] T. Granlund, V. Stirbu, and T. Mikkonen. Towards regulatory-compliant MLOps: Oravizio's journey from a machine learning experiment to a deployed certified medical product. *SN Computer Science*, 2(5), 2021.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[10] E. Hechler, M. Oberhofer, and T. Schaeck. *The Operationalization of AI*, pages 115–140. Apress, Berkeley, CA, 2020.

[11] T. M. Hospedales, A. Antoniou, P. Micaelli, and A. J. Storkey. Meta-Learning in Neural Networks: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(9):5149–5169, 2022.

[12] S. Islam. MLOps vs. DevOps: What is the difference? https://www.phdata.io/blog/mlops-vs-devops-whats-the-difference/ Last Accessed: 18.09.2023.

[13] S. K. Karmaker ("Santu"), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni. AutoML to Date and Beyond: Challenges and Opportunities. *ACM Comput. Surv.*, 54(8), oct 2021.

[14] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. In Y. Bengio and Y. LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[15] M. A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, 1991.

[16] D. Kreuzberger, N. Kühl, and S. Hirschl. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access*, 11:31866–31879, 2023.

[17] M. Maška, V. Ulman, P. Delgado-Rodriguez, E. Gómez-de Mariscal, T. Nečasová, F. A. Guerrero Peña, T. I. Ren, E. M. Meyerowitz,

T. Scherr, K. Löffler, et al. The Cell Tracking Challenge: 10 years of objective benchmarking. *Nature Methods*, pages 1–11, 2023.

[18] S. Meisenbacher, J. Pinter, T. Martin, V. Hagenmeyer, and R. Mikut. Concepts for automated machine learning in smart grid applications. In *Proceedings of the 31st Workshop on Computational Intelligence, Berlin*, pages 25–26, 2021.

[19] D. Merkel. Docker: Lightweight Linux containers for consistent development and deployment. *Linux J.*, 2014(239), mar 2014.

[20] Microsoft. Machine Learning operations maturity model - Azure Architecture Center. *Microsoft Learn*, Last Accessed: 10.09.2023.

[21] H. B. Mitchell. *Image Similarity Measures*, pages 167–185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[22] M. Molina-Moreno, M. P. Schilling, M. Reischl, and R. Mikut. Automated Style-Aware Selection of Annotated Pre-Training Databases in Biomedical Imaging. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2023.

[23] O. Neumann, M. Schilling, M. Reischl, and R. Mikut. EasyMLServe: Easy deployment of REST machine learning services. In *Proceedings 32. Workshop Computational Intelligence*, volume 1, page 11, 2022.

[24] G. Recupito, F. Pecorelli, G. Catolino, S. Moreschini, D. D. Nucci, F. Palomba, and D. A. Tamburri. A Multivocal Literature Review of MLOps Tools and Features. In G. M. Callicó, R. Hebig, and A. Wortmann, editors, *48th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2022, Maspalomas, Gran Canaria, Spain, 31 August - 2 September 2022*, pages 84–91. IEEE, 2022.

[25] C. Renggli, L. Rimanic, N. M. Gürel, B. Karlas, W. Wu, and C. Zhang. A Data Quality-Driven View of MLOps. *IEEE Data Eng. Bull.*, 44(1):11–23, 2021.

[26] T. Scherr, J. Seiffarth, B. Wollenhaupt, O. Neumann, M. P. Schilling, D. Kohlheyer, H. Scharr, K. Nöh, and R. Mikut. microbeSEG: A deep

learning software tool with omero data management for efficient and accurate cell segmentation. *PLOS ONE*, 17(11):1–14, 11 2022.

[27] M. P. Schilling, S. Schmelzer, L. Klinger, and M. Reischl. KaIDA: a modular tool for assisting image annotation in deep learning. *J. Integr. Bioinform.*, 19(4), 2022.

[28] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, and D. Dennison. Hidden Technical Debt in Machine Learning Systems. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 2503–2511, 2015.

[29] S. Suganyadevi, V. Seethalakshmi, and K. Balasamy. A review on deep learning in medical image analysis. *Int. J. Multim. Inf. Retr.*, 11(1):19–38, 2022.

[30] G. Symeonidis, E. Nerantzis, A. Kazakis, and G. A. Papakostas. MLOps - Definitions, Tools and Challenges. In *12th IEEE Annual Computing and Communication Workshop and Conference, CCWC 2022, Las Vegas, NV, USA, January 26-29, 2022*, pages 453–460. IEEE, 2022.

[31] D. A. Tamburri. Sustainable MLOps: Trends and Challenges. In *22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2020, Timisoara, Romania, September 1-4, 2020*, pages 17–23. IEEE, 2020.

[32] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, and G. Vessio. MLOps: A taxonomy and a Methodology. *IEEE Access*, 10:63606–63618, 2022.

[33] A. J. Yamachui Sitcheu. Pomlia. GitLab Repository, https://gitlab.kit.edu/angelo.sitcheu/pomlia, 2023.

[34] J. Yang, R. Shi, D. Wei, Z. Liu, L. Zhao, B. Ke, H. Pfister, and B. Ni. MedMNIST v2: A Large-Scale Lightweight Benchmark for 2D and 3D Biomedical Image Classification. *CoRR*, abs/2110.14795, 2021.

[35] P. Yuan, A. Mobiny, J. Jahanipour, X. Li, P. A. Cicalese, B. Roysam, V. M. Patel, M. Dragan, and H. V. Nguyen. Few is enough: Task-augmented active meta-learning for brain cell classification. In A. L. Martel, P. Abolmaesumi, D. Stoyanov, D. Mateus, M. A. Zuluaga, S. K. Zhou, D. Racoceanu, and L. Joskowicz, editors, *Medical Image Computing and Computer Assisted Intervention - MICCAI 2020 - 23rd International Conference, Lima, Peru, October 4-8, 2020, Proceedings, Part I*, volume 12261 of *Lecture Notes in Computer Science*, pages 367–377. Springer, 2020.

[36] G. Zárate, R. Miñón, J. Díaz-de-Arcaya, and A. I. Torre-Bastida. K2E: Building MLOps Environments for Governing Data and Models Catalogues while tracking versions. In *IEEE 19th International Conference on Software Architecture Companion, ICSA Companion 2022, Honolulu, HI, USA, March 12-15, 2022*, pages 206–209. IEEE, 2022.

[37] L. Zhang, X. Wang, D. Yang, T. Sanford, S. A. Harmon, B. Turkbey, B. J. Wood, H. Roth, A. Myronenko, D. Xu, and Z. Xu. Generalizing deep learning for medical image segmentation to unseen domains via deep stacked transformation. *IEEE Trans. Medical Imaging*, 39(7):2531–2540, 2020.

[38] B. Zhou, Y. Svetashova, A. Gusmao, A. Soylu, G. Cheng, R. Mikut, A. Waaler, and E. Kharlamov. SemML: Facilitating development of ML models for condition monitoring with semantics. *J. Web Semant.*, 71:100664, 2021.

[39] Álvaro López García. DEEPaaS API: a REST API for Machine Learning and Deep Learning models. *Journal of Open Source Software*, 4(42):1517, 2019.

Machine Learning Forecasting of Daily Delivery Positions: A Modern Take on Industrial Workforce Planning

Lukas Hans[1], Patrick Eichenseer[2], and Matthias Groß[1]

[1]Institute for Data Science, Engineering, and Analytics, TH Köln
Steinmüllerallee 1, 51643 Gummersbach, Germany
E-Mails: lukas.hans@th-koeln.de, matthias.gross2@th-koeln.de

[2]Head of Operational Excellence, DEHN SE
Umelsdorfer Str. 8, 92280 Utzenhofen, Germany
E-Mail: patrick.eichenseer@dehn.de

1 Introduction

The logistics industry plays a pivotal role in global trade, and efficient warehouse operations are essential for the seamless movement of goods. In recent years, prevailing job market conditions have presented significant difficulties in recruiting skilled workers in warehouse operations [1]. This shortage of skilled logistics workers has challenged companies to meet dynamic market demands and hindered effective workforce planning. The labor shortage results in increased operating costs and decreased overall efficiency due to suboptimal resource utilization. The key challenge for warehouse managers is the fluctuating and unpredictable nature of customer demand. Short-term customer orders, seasonal fluctuations, and rapidly changing market demands make it difficult for companies to forecast and plan their logistics workforce accurately. These dynamic demands often result in overstaffing during off-peak periods and understaffing during peak periods. Understaffing leads to unmet customer demand and, in some cases, customer churn. It also runs counter to a common warehouse goal of maximizing service levels (i.e., the promise of fast and accurate delivery) as a measure of differentiation from competitors. Overstaffing leads to underutilization and inefficiencies. This results in financial losses, as customer order fulfillment through picking at the point of delivery is the most costly activity in the warehouse [2, 3, 4].

To tackle the pressing issues in workforce scheduling, it is essential to develop accurate forecasting frameworks that can efficiently predict delivery positions. Accurate forecasting enables companies to optimize their employee staffing in logistics, reducing the risks of both overstaffing and understaffing within the limited workforce. Currently, the forecast process is heavily reliant on the personal expertise and judgment of individual team members. These individuals draw on their years of experience and intuition to estimate future demand, utilizing the number of pre-orders already recorded in their software system as a key input. This approach, although common in small and medium-sized warehouses, is inherently subjective and susceptible to human error and bias.

In this context, integrating Machine Learning (ML) methods presents a promising solution to enhance workforce scheduling efficiency [5, 6]. ML algorithms offer the capability to process vast amounts of data, identify complex patterns, and make data-driven predictions. By reframing the workforce optimization problem as a forecasting challenge, ML models can be leveraged to provide accurate and reliable one-day-ahead delivery position predictions. This approach not only improves transparency and explainability in the estimation process but also enhances the overall scheduling efficiency.

2 Problem Statement and Data Description

In the complex domain of workforce planning within logistics, we aim to develop an accurate mathematical representation of the challenge. Instead of directly tackling workforce scheduling, we've transformed it into a prediction problem, using various ML models to forecast the next day's delivery positions. The problem is defined mathematically as minimizing the root-mean-square error between the actual and predicted number of delivery positions using the optimal model and its parameters.

$$i^* = \text{argmin}_i \quad \sqrt{\frac{1}{n}\sum_{t=1}^{n}(D_t^{act} - D_t^{pred,i})^2} \tag{1}$$

With

- D_t^{act}: The actual number of delivery positions for day t.
- $D_t^{pred,i}$: The predicted number of delivery positions for day t, using the i-th ML model.

Simultaneously, we aim for the predicted delivery positions, D_t^{pred,i^*}, from the best model to closely match the needed number of workers, W_t. The relationship between these is determined by past company data and workforce planning rules. This relationship is given by

$$W_t = h(D_t^{pred,i^*}) \tag{2}$$

where $h(\cdot)$ represents the historical link between delivery positions and required workforce. Nonetheless, our focus is on pinpointing the most effective ML technique, rather than deducing the exact form of $h(\cdot)$. The study relies on delivery position data from a collaboration between the Technical University of Cologne and a major electrical engineering firm, with adjustments made using a constant factor X to safeguard financial confidentiality. Table 1 provides a comprehensive statistical overview of the initial time series data.

The actual data entries amount to 1337 data points, given the non-operational days like weekends and winter breaks. Following an 80/20 split for partitioning, the training set contains 1069 entries, and the test set has 268. The initial data inspection highlighted significant downward fluctuations, leading to the removal of entries identified as outliers and those with fewer than 200 delivery positions. This refined dataset, now with 1321 entries, presents a more consistent distribution conducive to analysis. Cleaning paved the way for feature engineering, introducing variables like time details, lag intervals, and multiple rolling means to serve as inputs for the ML models.

Table 1: Feature summary

	Start Date	End Date	Time Span	
Lagerbewegung Zeitpunkt	2018-01-02	2023-05-08	1952 days	
	Mean	Std.	Min	Max
Lagerbewegung Pick ERP-Auftrag	2860.26	697.91	1	4525

3 Methods & Metrics

For forecasting, five primary models were utilized: Random Forest, XGBoost, LightGBM, Support Vector Regression (SVR), and Convolutional Neural Networks (CNN). Random Forest is an ensemble method known for its robustness and ability to manage non-linear relationships [7]. XGBoost and LightGBM are both gradient boosting frameworks, with the former being praised for its speed and flexibility [8], and the latter for efficiency and leaf-wise tree growth [9]. SVR excels in high-dimensional spaces and offers kernel function flexibility [10]. CNNs, while dominant in image classification, have shown prowess in time-series forecasting, capturing both local and global temporal dependencies [11].

Beyond the primary models, two ensemble strategies, Stacking and Averaging, were evaluated. Averaging involves computing the mean of individual model predictions, noted for its robustness [12]. Stacking harnesses multiple base models, in our case leveraging ridge regression as the meta-model [13].

Hyperparameter tuning for all models was automated using the Optuna package, streamlining optimal value discovery [14].

Model performance was gauged using Root Mean Square Error (RMSE) and Mean Absolute Error (MAE). While RMSE emphasizes large errors, offering a penalty, MAE offers a balanced view on error magnitude [15, 16]. Both metrics help in a holistic evaluation of model accuracy.

Table 2: Performance metric of the different modeling approaches

Algorithm	MAE	RMSE
LightGBM	335.56	440.63
CNN	382.83	518.00
SVR	356.51	470.66
XGBoost	331.86	437.15
Random Forest	323.78	429.55
Stacking Ensemble with CNN	683.47	796.16
Average Ensemble with CNN	318.91	416.48
Stacking Ensemble without CNN	365.08	476.35
Average Ensemble without CNN	313.98	413.96

4 Results

In evaluating the base models, tree-based algorithms - LightGBM, XGBoost, and Random Forest - showed close performance, with RMSEs between 429 to 441 and MAEs ranging from 323 to 336. The SVR demonstrated a conservative forecast, with MAE and RMSE values at 356.51 and 470.66, respectively. Conversely, the CNN trailed in performance, especially post-Christmas, resulting in an MAE of 382.83 and an RMSE of 518.

The ensemble strategies exhibited contrasting results. The stacking ensemble, despite capturing the time series' structure, tended to overestimate with MAEs and RMSEs at 683.47 and 796.16. Contrarily, the average ensemble outperformed all models, achieving an MAE of 318.91 and an RMSE of 416.48.

Considering the subpar performance of the CNN, ensembles were recomputed without CNN inputs. This led to an overall improvement. The average ensemble showed an MAE of 313.98 and an RMSE of 413.96, while the stacking ensemble registered 365.08 and 476.35 for MAE and RMSE, respectively. Even though the average ensemble's metrics were superior, the stacked method better captured individual peaks.

The performance metrics for all the discussed modeling approaches are summarized below:

5 Discussion & Outlook

Besides confirming a fundamental predictability in the used dataset, this work also showed that tree-based models like Random Forests, XGBoost and LightGBM are suitable algorithms for this task. Beyond that, stacking and averaging ensemble methods using these models proved to further increase the forecasting effectiveness.

While the SVM approach performed slightly worse but in the same approximate range as the tree-based models, CNNs clearly showed inferior results, especially when exposed to irregularities like the post-christmas dip in the used dataset. The small size of the dataset can be assumed as a possible reason for this lack in performance. The neural network might not have been exposed to a sufficient amount of data to pick up the more complex patterns. As the dataset grows over time, we could expect an improvement in CNNs' performance.

Going forward with these approaches, a possible next step is the incorporation of external variables that might influence the number of delivery positions. These variables can range from calendrical data to pre-order information and even weather data.

Furthermore, different model architectures should be examined, as this work mainly focused on tree-based models. Diversifying the model architectures could boost the forecasting effectiveness even further.

In summary, this research offers a promising start toward optimizing workforce scheduling in small and medium-sized warehouses in the logistics sector by leveraging ML methods.

Acknowledgments

The work was partly funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) as part of Project "IMProvT II" (FKZ: 03EN2086 A-E).

References

[1] "Skilled labour shortage: Freight, logistics sector needs 18,000 workers for jobs," June 2023. [Online; accessed 11. Sep. 2023].

[2] T. van Gils, K. Ramaekers, A. Caris, and R. B. M. de Koster, "Designing efficient order picking systems by combining planning problems: State-of-the-art classification and review," *Eur. J. Oper. Res.*, vol. 267, pp. 1–15, May 2018.

[3] Y.-C. Ho and J.-W. Lin, "Improving order-picking performance by converting a sequential zone-picking line into a zone-picking network," *Comput. Ind. Eng.*, vol. 113, pp. 241–255, Nov. 2017.

[4] S. Vanheusden, T. van Gils, A. Caris, K. Ramaekers, and K. Braekers, "Operational workload balancing in manual order picking," *Comput. Ind. Eng.*, vol. 141, p. 106269, Mar. 2020.

[5] M. Witthaut and C. Culotta, "Machine Learning in der Logistik – Eine empirische Studie über die Anwendung in deutschen Unternehmen," *Logistics Journal : nicht referierte Veröffentlichungen*, vol. 2021, June 2021.

[6] M. Akbari and T. N. A. Do, "A systematic review of machine learning in logistics and supply chain management: current trends and future directions," *Benchmarking: An International Journal*, vol. 28, pp. 2977–3005, Mar. 2021.

[7] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[8] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

[9] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[10] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems* (M. Mozer, M. Jordan, and T. Petsche, eds.), vol. 9, MIT Press, 1996.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] T. G. Dietterich, "Ensemble methods in machine learning," in *International workshop on multiple classifier systems*, pp. 1–15, Springer, 2000.

[13] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, no. 2, pp. 241–259, 1992.

[14] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2623–2631, 2019.

[15] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*. OTexts, 2018.

[16] T. Chai and R. R. Draxler, "Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature," *Geoscientific Model Development*, vol. 7, no. 3, pp. 1247–1250, 2014.

Adaptive video game music as a multi-objective benchmark for conditional autoregressive models

Fabian Ostermann

Technische Universität Dortmund
Otto-Hahn-Str. 14, 44227 Dortmund, DE
E-Mail: fabian.ostermann@tu-dortmund.de

1 Introduction

The aim of this paper is to investigate and discuss the use of generative neural networks to reconstruct handcrafted adaptive music. We choose the dynamic compositions from the 1991 video game *Monkey Island 2* [1], which has consistently been recognized as a role model and masterpiece in this field [2, 3]. The music in this game is generated in real-time during gameplay, leading us to define our task as the successive prediction of note events using autoregressive models. Furthermore, the game's music adapts to player actions, so the generative task additionally is conditional.

In Section 2, we explain that music data contains very different types of information and show how similarly constructed statistical errors can have very different effects according to cognitive music perception, which brings additional complexity to this problem domain. In Section 3, we explain how the dataset was obtained and what attributes it has. The dataset generator itself is *openly available*[1] for further academic or educational use. We argue that, on the one hand, music generation is a *fun* problem domain without ethical issues, which has the potential to motivate people to engage with AI technology. On

[1] `https://github.com/fabianostermann/WoodtickWalkingSimulator`

the other hand, it really makes a challenging benchmark for complex multi-task learning concepts and multi-objective optimization strategies. Moreover, coping with the adaptive real-time aspect of video game music is exceptionally difficult and, to our knowledge, poses a unique challenge. For our practical investigation, Section 4, we define seven problem dimensions specific to this task. For each dimension, we conduct experiments and discuss the unique challenges associated with processing music data. In the process, we propose areas for further investigation, which will be summarized in Section 5.

2 Music as a problem domain

Music is a complex matter. Its description needs a lot of dimensions [4], which makes it vulnerable to combinatorial explosion. All attempts of symbolic representation, like modern western notation, need some form of severe simplification.[2] The example of MIDI is used to explain which minimum set of dimensions is required.

MIDI [5] is a widely used music data communication protocol that was also used for the music of *Monkey Island 2*. MIDI was designed as a standard to connect all kinds of electronic musical instruments. It defines atomic musical instructions called *MIDI events* that carry byte-sized information about pitch, timing and loudness[3]. The beginning and the end of a note are divided into two different events. The desired instrument sound is implicitly set. The MIDI stream consists of 16 different channels. A so-called *program change* event is sent to a channel to request an instrument change. A special case is channel 10, which is used for drums, and where the pitch information is remapped to specify which drum instrument to trigger (e.g., snare, bass drum, hihat).

MIDI and its cluttered specification appears a bit dated by today's standards. Its main advantage is its wide popularity. A lot of MIDI-encoded music exists and can be directly used to build machine learning datasets [6]. However, it is

[2] An exception may be the approach to process music at the audio signal level [7, 8].
[3] Be aware that the loudness of a note in the context of MIDI usually is referred to as *velocity*, because the velocity of pressing down a key on a piano keyboard determines its loudness. We stick to the term *loudness* here since velocity can easily be misinterpreted as a temporal property.

usually not practical to use MIDI events directly as input data for classification or generation tasks.[4] Therefore, it is necessary to convert a stream of MIDI events into a sequence of semantic features.

We choose to define every note as a 5-tuple

$$note_i = (\Delta\text{time, duration, loudness, pitch, instrument}) \tag{1}$$

and a music sequence of length n as $seq = \{note_1, note_2, \ldots, note_n\}$. The time difference Δtime is calculated as the distance in seconds to the previous note event. The duration is the distance of beginning and end of the note in seconds. The loudness is the MIDI velocity value normalized to [0,1]. The pitch is one of $\{0, 1, ..., 127\}$ which linearly maps to the keys of a piano, where 60 denotes the middle C. The instrument is an integer label determined by searching in the MIDI stream for the last *program change* event that occurred on the same MIDI channel.[5]

Note that this scheme is one possibility of many and that the concrete encoding is usually important [9, 10]. Also note that the different types of information are not equally important to the cognitive perception of the music, since, e.g., Δtimecan be allowed to vary by a few milliseconds without the perception of rhythmical flaws while missing a pitch by only a semitone[6] instantly results in severe harmonic dissonances. Changes in duration, loudness or instrument are rather perceived as variations than mistakes. Among all, Δtimeand pitch are (by far) the most important features to re-recognizing music.

That said, music makes a complex multi-objective learning problem that mixes regression (Δtime, duration, loudness) and classification (pitch, instrument) tasks. Therefore, it can be approached by multi-task learning, which was already applied to music with the most success in music transcription [11, 12].

[4] For example, we tested direct prediction of byte events on our training data. It resulted in accuracy of 90%+, which, however, is not at all useful for actual generation since the smallest mistake leads to fatal syntax corruption.

[5] The actual code implementation makes use of the python package *prettymidi* [13]

[6] A semitone is the smallest possible distance between two different tones in western tonal music.

3 Dataset of adaptive video game music

The 5-tuple encoding above can be applied to all music of information depth equal to simple MIDI. For the scope of our study, we decided to learn from data that consist of the adaptive game music that was originally composed and programmed by Michael Land and Peter McConnell for the all-time adventure game classic *Monkey Island 2* [1]. Despite the game's release in 1991, the significant effort invested in creating the adaptive compositions, including the manual crafting of musical transitions between numerous points in the music variations, remains extraordinary. It continues to serve as a role model in the eyes of experts and critics until today [2, 3].[7]

We selected a specific piece of music from the game: In the town known as Woodtick, which the main character visits at the beginning of the game, the music changes as the character enters different locations. But the music is not just replaced or blended over. Instead, variations of the town music smoothly appear in various manually-prepared dynamic transitions. We have developed a random walking simulator for the game, which is *openly available*[8]. It automatically walks the main character through Woodtick and records the stream of MIDI events using the *ScummVM* emulator [14]. We also track the location change events that trigger the musical transitions in the MIDI stream.

Since a note from a music sequence in close scope depends linearly on its previous notes, the process of creating the sequence can be modeled as an autoregressive task. However, in a wider scope, changing the location leads to distinct musical continuations. Therefore, the event of changing the location must be considered as a conditional input c_t for the autoregressive model

$$note_t = f(note_{t-1}, note_{t-2}, ..., note_{t-p}, c_t), \tag{2}$$

[7] The reason is interesting: The simple MIDI protocol was dropped when waveform sample playback became possible on home computers. This leap in acoustic complexity made the possibility of handcrafting complex adaptive music compositions impossible [2]. The game industry opted for the cinematic feel at the expense of musical immersion. And this trend continues to this day, with each increase in computing power primarily used for better and better game graphics. However, this circumstance is likely to change soon, as the power of home computers is increasing faster and faster due to current improvements in AI technology [15], which could reach a limit of necessary realism (as it was for 4k screen resolution).

[8] `https://github.com/fabianostermann/WoodtickWalkingSimulator`

where p is the context length, i.e. the number of considered previous events. We will use a neural network to map the function f.

The big difference to models usually applied to music generation is that we are not trying to create a general framework for diverse musical outcomes [16, 8]. Instead, we want to approximate a single composition. However, since this composition is adaptive, it differs every time it is played. But the number of possible variations is limited for a finite period of playback. That means, although the model is autoregressive as described, it does not need to generalize for all possible inputs.

4 Problem dimensions

In this section, we present seven different problem dimensions of our task: neural network architecture and type, context length, data input representation, multi-task learning, complexity reduction, multi-objective optimization and rare events. This list of seven is not exhaustive, as there are additional parameters, e.g., batch size, learning rate or the choice of the optimizer itself.[9] However, we chose to focus on discussing these seven dimensions because we found them to be less universal and, in some aspects, unique to the context of music data. We will present some practical experiments on how to optimize parameters for each of the dimensions. Given their interdependency, where changes in one parameter can consistently affect results in other problem dimensions, an all-at-once optimization approach would be ideal. However, since this is beyond the scope of complexity, we decided to propose a chain of optimizations knowing that changing their order may influence final results. Notice that it was also not possible to exhaustively optimize each dimension. We will present exemplary results, discuss how each problem dimension should be addressed, name applicable strategies and name aspects for future investigations.

To optimize the multi-task models, we used an averaging loss function as the overall mean of *cross entropy* loss for pitch and instrument and *mean*

[9] We choose a batch size of 4096, $l_r = 0.001$ and the Adam optimizer [17].

Table 1: Loss and accuracy score comparison of different neural architectures and types

type	recurrent layers	units	dense	loss	↑ accuracy
LSTM	2	256	–	**1.081±0.01**	**0.807±0.03**
LSTM	1	256	128	1.105±0.01	0.712±0.01
LSTM	1	256	–	1.107±0.01	0.706±0.02
GRU	2	256	–	1.147±0.00	0.693±0.02
GRU	1	256	128	1.117±0.01	0.657±0.01
GRU	1	256	–	1.178±0.01	0.565±0.02

absolute error loss for the regression tasks of Δtime, duration and loudness. We used the latter instead of *mean squared error* to mitigate averaging issues associated with large magnitude differences. For evaluation, we will mainly provide accuracy score. For Δtime, duration and loudness, we simply defined a deviation of less than 0.05 as sufficiently accurate.[10] Due to random weight initialization, we conducted 5 statistical repetitions on 5 different simulator runs, each consisting of 2 hours of music. Every loss and accuracy score below is reported as mean ± standard deviation of these 5 runs.

4.1 Architectures and neural network types

The first problem when coping with neural networks is always to choose a network type and to determine its structure and size. There are a few general rules to follow [18], but most parameters must be reconsidered with every new problem. Since we have time series data, the natural choice is to use recurrent networks. We chose the popular *long-short term memory* (LSTM) unit [19] and compare with its less complex but more efficient variant *gated reccurent unit* (GRU) [20].

Table 1 shows that an LSTM with 2 layers is performing best for accuracy and loss. A concatenated dense layer helped in case of only one layer. The same applies to GRU, which in comparison performed worse. Note that using only

[10] For Δtime and duration 0.05 equals to 50 ms.

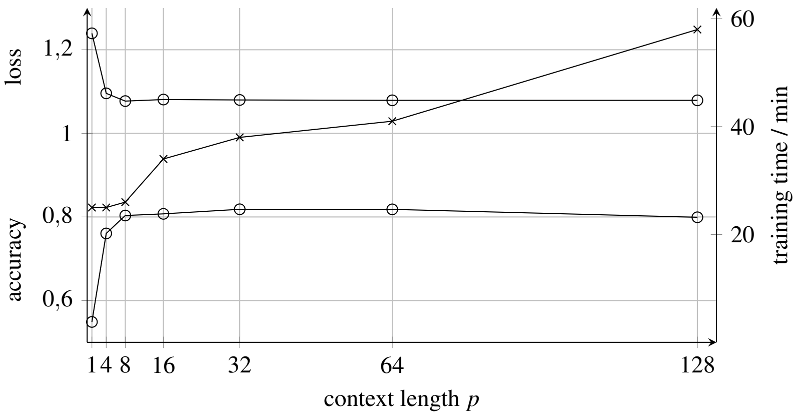


Figure 1: Accuracy score, loss and training time in relation to context length. Circle markers correspond to the left y-axis, cross markers to the right.

the last hidden state of the LSTM gave similar results to GRU, but using hidden state and last cell state improved the results significantly.

We can clearly see that the choice of model type and architecture is, as expected, of critical importance. Another approach could be to use 1D convolutional filters. State-of-the-art autoregressive modeling likely involves the use of transformer models [21, 9], the application of which remains future work.

4.2 Context length

The number of considered previous events is a crucial parameter for autoregressive tasks. It determines, how much context information the model has to predict the next event. For this problem dimension, we used the 2 layered LSTM, which performed best in the previous section. Figure 1 shows, that for already 8 events the accuracy reaches a plateau. After that, it only improves slightly. However, the time needed for calculation increases further since the number of weight parameters inside the LSTM increases non-linearly. With $p = 16$, the calculation needs about one third more time than with $p = 8$.[11] In

[11] 34 min to 26 min on a Nvidia A100 GPU.

order to have more capacity for diverse experiments on the following dimensions, we decided to use $p = 8$ from here on. Please note that a sequence length of $p = 8$ in music holds much information (cf. Eq.1). Just imagine, if someone sings to you 8 notes of a well-known melody, you will probably be able to recognize it.

4.3 Input representation

The internal representation of data can have great influence on the model performance [22]. The categorical information about pitch and instrument before was one-hot encoded with $n_c = 60$ unique classes for pitch and $n_c = 13$ different instruments. With a sequence length of $p = 8$ and the 3 other tasks there were already $8 \cdot (60 + 13 + 3) = 608$ input values. As the model complexity is relative to the input size, we aimed to decrease the number of inputs to the LSTM by adding an input embedding layer [18]. The embedding size was determined as $\lfloor \frac{n_c}{5} \rfloor + 1$. This definition can be considered another hyperparameter to be optimized.

By applying input embedding, we increased the accuracy from 0.807±0.03 to 0.867±0.02 with a decreasing loss from 1.081±0.01 to 0.867±0.02. A further approach to directly use the integer label as input[12], which reduces the input size to 1, performed worse with an accuracy of just 0.716±0.02.

4.4 Multi-task learning vs. ensemble learning

To learn all objectives separately and predict them with an ensemble of models improves evaluation results [23, 18] but comes at a higher computational cost. If not calculated in parallel, it is far slower in inference (and training, cf. Table 2), which might cause severe problems when used for real-time conditioned generation or on low performance hardware.[13]

[12] For pitches, this may be called an *ordinal encoding*, since it provides a natural order.

[13] Regarding the present context, both requirements are met for video games. In addition, computational resources are oftentimes reserved for rendering high-resolution 3D graphics.

Table 2: Loss and accuracy scores for different ensemble and multi-task models. Values belonging to one individual model share a frame.

	accuracy		
Δtime	0.999±0.00	0.955±0.01	0.945±0.01
duration	0.980±0.00	0.876±0.01	0.862±0.02
loudness	0.975±0.00	0.977±0.00	0.880±0.03
pitch	0.907±0.01	0.670±0.02	0.696±0.03
instr.	0.950±0.00	0.953±0.01	0.954±0.00
mean	0.962±0.00	0.886±0.00	0.867±0.02
∑ training time	106 min	43 min	29 min

Table 2 shows that learning single tasks (left column with numbers) is much easier than learning multiple tasks at once (right column). Learning the regression tasks (Δtime, duration, loudness) or the categorical task separately (middle column) only improves scores slightly (except for loudness). However, handling this interdependency is a major topic. E.g., the choice of the next pitch is highly dependent on which instrument is chosen. Before, they were predicted in parallel, but the accuracy may be improved by using an hierarchical learning approach to predict the objectives one after another.

4.5 Complexity reduction

This dimension is to be understood as the output equivalent of the input representation from Section 4.3. As Table 2 shows, predicting pitch is the most difficult objective. That probably is because it has the most classes and the largest dependencies to the previous notes. In addition, only a near 100% accurate prediction is satisfactory from a psychological point of view. When the timing is slightly off or the melody is played by the wrong instrument, the mistake is not perceived as serious as that of choosing the wrong pitch.[14] In this context, leveraging the semantic nature of music data can be used to further reduce complexity. The pitch information can be divided into two bits

[14] An exception is the drums, because when playing back a melody on the MIDI drum channel, it becomes completely unrecognizable. Hence, another approach to complexity reduction could involve segregating the prediction of drum instruments from melody and harmony instruments.

Table 3: Accuracy scores for predicting pitch divided into pitch class and octave.

pitch		Δtime	duration	loudness	instrument	mean
0.696±0.03		0.945±0.01	0.862±0.02	0.880±0.03	0.954±0.00	0.867±0.02
0.802±0.01	0.908±0.01	0.900±0.02	0.795±0.04	0.817±0.04	0.953±0.00	0.862±0.02
pitch class	octave					

of information: octave and pitch class. This introduces an additional task to the multi-task model but has the advantage that predicting one of 12 pitch classes can be learned with a higher accuracy. In addition, playing in the wrong octaves (of 5) is not perceived as nearly as disharmonic as any smaller semitone error.[15] Table 3 shows a boost in overall pitch accuracy by splitting up the task. But the multi-task difficulty increases and thus all the single regression tasks drop in accuracy. However, the average accuracy remains unchanged.

Another approach of complexity reduction is to transform each linear regression task into a classification task by binning or clustering. This might be advantageous since continuous regression is at times more difficult, especially when combined with a classification task in a multi-task learning problem (cf. Section 4.4).

A more complex approach may be to encode segments of the music with a (variational) autoencoder that is then controlled by another agent that is rewarded to recreate the composition in a reinforcement learning setting. Moreover, this approach would be able to come up with some novel variations of the music.

4.6 Multi-objective optimization

Since the present task is inherently multi-objective, we can also consider to improve the loss function itself. Up to here, the loss value was calculated as the equally-weighted mean of all single loss values of all the objectives. Since pitch was identified to be the hardest but most important objective, we try to

[15] The auditory phenomenon of *tonal fusion* [24] explains that two notes played in the interval of one octave are most likely to be perceived as a single tone among all possible intervals.

Table 4: Accuracy scores of different prioritization of pitch under weight w_p.

w_p	pitch	Δtime	duration	loudness	instrument	mean
0.0	0.017±0.00	0.859±0.03	0.735±0.04	0.778±0.04	0.950±0.00	0.668±0.02
0.2 (mean)	**0.681±0.02**	**0.949±0.01**	**0.869±0.01**	**0.885±0.01**	**0.955±0.00**	**0.868±0.01**
0.5	0.613±0.02	0.757±0.04	0.614±0.02	0.670±0.03	0.950±0.00	0.721±0.02
0.8	0.616±0.03	0.579±0.01	0.451±0.03	0.436±0.04	0.948±0.00	0.606±0.02
1.0	0.629±0.02	0.053±0.01	0.051±0.01	0.038±0.02	0.951±0.01	0.344±0.00
random	0.605±0.02	0.802±0.02	0.679±0.05	0.603±0.05	**0.955±0.00**	0.729±0.01

determine if it is possible to boost its accuracy by prioritizing it over the other objectives. The following formular will be used for prioritizing objective p with weight w_p over all the other n objectives:

$$loss = w_p \cdot loss_p + \sum_{i=1}^{n} (\frac{1 - w_p}{n}) \cdot loss_i \tag{3}$$

For $w_p = 0.2$, this formula corresponds to the average weighting used before.

Table 4 shows that the applied weighting procedure is not able to boost the accuracy of the pitch prediction. However, a w_p of 0.0 prevents any learning success for pitch, but surprisingly the other objectives do not benefit from it. When increasing w_p to 1.0, the other objective drop to an accuracy of nearly 0 as expected (except instrument). The pitch objective, however, does not benefit from this either, since its accuracy decreases in comparison to the equal weighting ($w_p = 0.2$). The baseline comparison to a random weighting, that randomly changes w_p on each call, also shows no improvement.

In total, weighting for prioritization was not successful, since equally weighting did indead perform better. This result may be explained by destabilizing the gradient descent. If so, parameters like batch size and learning rate must also be reconsidered here. In any case, this topic is complex and worth investigating in future work, e.g. by applying Chebyshev loss or other strategies from the domain of multi-objective optimization.

4.7 Rare events

Music compositions typically exhibit numerous redundancies, such as repetitions, reprises, and motivic variations. As a result, music data often is imbalanced. In the simulation, we observe that the main character spends roughly half of the time outside in the town scene (due to random walking), leading to a significantly higher presence of the corresponding music. Furthermore, location changes account for only a maximum of 5% of the data, resulting in a severe underrepresentation of individual musical transitions, which are about 5 to 10 per possible location transition. This may explain why all the training sessions could never reach 100% accuracy and why, e.g., instrument performs well even if zero weighted (cf. Table 4 where $w_p = 1.0$) To meet this circumstance, heavy dataset balancing [25] is needed. One approach without removing samples from the dataset is to use the loss values as an information of success and to prioritize samples of higher loss during training, either by loss weighting or by dynamic adjustment of selection probabilities. This topic also remains work for future investigations.

5 Conclusions

We have seen that music as a problem domain provides a rich ground for diverse research questions. Datasets with adaptive music from the video game *Monkey Island 2* can be easily generated using our openly available generator. We have also provided experimental setups and analyses covering the problem dimensions of neural network architecture and type, context length, data input representation, multi-task learning, complexity reduction, multi-objective optimization and rare events. These aspects are critically important and, in some aspects, unique to the task of adaptive music generation. We have presented numerous ideas for future work. Implementing more sophisticated multi-objective loss strategies and to cope with rare events by adaptive dataset balancing are both worth further investigations.

The concept of autoregressive music generation also holds the potential to be applied in the creation of original compositions for video games featuring adaptive music. Surprisingly, this powerful component [26] is still underutilized in the industry. The notion of using variational autoencoders controlled by a reinforcement learning agent is intriguing. A solution to this challenge surely is of interest not only to the video game industry but also to broader contexts, as the future trend is clearly heading towards “non-linear media” [27].

References

[1] R. Gilbert, T. Schafer, and D. Grossman. *Monkey Island 2 – LeChuck’s Revenge*. LucasArts/Lucasfilm Games, 1991. `https://archive.org/details/msdos_Monkey_Island_2_-_LeChucks_Revenge_1991`. Accessed 24 Sep 2023.

[2] M. Sweet. *Writing Interactive Music for Video Games: A Composer’s Guide (Game Design)*. Addison-Wesley Professional, Boston, USA, 2014.

[3] T. Summers. *Unterstanding Video Game Music*. Cambridge University Press, Cambridge, UK, 2016.

[4] M. Lepper. *de Linguis Musicam Notare – Beiträge zur Bestimmung von Semantik und Stilistik moderner Musiknotation durch mathematische Remodellierung* (german). In R. Großmann and H. Kinzler, editors, *Beiträge zur Medienästhetik der Musik*. epOs-Verlag, Osnabrück, Germany, 2021.

[5] The MIDI Association (TMA). *Musical Instrument Digital Interface – Official Specifications*. 1983. `https://midi.org/specifications`. Accessed 24 Sep 2023.

[6] C. Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. PhD thesis, Columbia University, New York, USA, 2016.

[7] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio. SampleRNN: An unconditional end-to-end neural audio generation model. In *International Conference on Learning Representations*, 2017.

[8] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank. MusicLM: Generating music from text. *arXiv preprint*, 2023.

[9] Y.-S. Huang and Y.-H. Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. MM '20, page 1180–1188, New York, USA, 2020. Association for Computing Machinery.

[10] D. P. Pazel. *Music Representation and Transformation in Software - Structure and Algorithms in Python*. Springer Cham, 2022.

[11] R. Kelz, S. Böck, and C. Widnaer. Multitask learning for polyphonic piano transcription, a case study. In *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, pages 85–91, 2019.

[12] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel. MT3: Multi-task multitrack music transcription. In *International Conference on Learning Representations*, 2022.

[13] C. Raffel and D. P. W. Ellis. Intuitive analysis, creation and manipulation of midi data with prettymidi. In *15th International Conference on Music Information Retrieval Late Breaking and Demo Papers*, 2014.

[14] P. Kołodziejski, E. Sandulenko, E. J. T. Sømåen, and L. S. Mari. *ScummVM: Script creation utility for maniac mansion – Virtual Machine*. ScummVM Team, 2020–2023. `https://docs.scummvm.org/en/v2.7.0`. Accessed 24 Sep 2023.

[15] H. C. Lin and A. Burnes. Nvidia DLSS 3: AI-powered performance multiplier boosts frame rates by up to 4x. *Nvidia GeForce News*, 2022. `https://www.nvidia.com/en-us/geforce/news/dlss3-ai-powered-neural-graphics-innovations/`. Accessed 24 Sep 2023.

[16] J.-P. Briot, G. Hadjeres, and F.-D. Pachet. *Deep Learning Techniques for Music Generation*. Springer, Berlin, Heidelberg, Germany, 2020.

[17] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, Massachusetts, USA, 2016.

[19] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[20] K. Cho, B. van Merrienboer, Çaglar Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 6000–6010, 2017.

[22] O. Neumann, N. Ludwig, M. Turowski, B. Heidrich, V. Hagenmeyer, and R. Mikut. Smart data representations: Impact on the accuracy of deep neural networks. In *Proceedings 31st Workshop Computational Intelligence*, pages 113–130, Berlin, Germany, 2021. KIT Scientific Publishing.

[23] P. Sollich and A. Krogh. Learning with ensembles: How overfitting can be useful. In *Proceedings of the 8th International Conference on Neural Information Processing Systems*, volume 8, page 190–196, 1995.

[24] M. Ebeling. Musical structures and their perception. In C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph, editors, *Music Data Analysis: Foundations and Applications*, Chapman & Hall/CRC Computer Science & Data Analysis. Taylor & Francis Group, New York, USA, 2019.

[25] V. Werner de Vargas, J. A. Schneider Aranda, R. dos Santos Costa, P. R. da Silva Pereira, and J. L. Victória Barbosa. Imbalanced data preprocessing techniques for machine learning: a systematic mapping study. *Knowledge and Information Systems*, 65(1):31–57, 2023.

[26] R. Stevens, D. Raybould, and D. McDermott. Extreme ninjas use windows, not doors: Addressing video game fidelity through ludo-narrative music in the stealth genre. In *Proceedings 56th International AES Conference: Audio for Games*, 2015.

[27] K. Tatar and P. Pasquier. Musical agents: A typology and state of the art towards musical metacreation. *Journal of New Music Research*, 48(1):56–105, 2019.

Holistic Modeling of Ultra-High Performance Concrete Production Process: Synergizing Mix Design, Fresh Concrete Properties, and Curing Conditions

Farzad Rezazadeh P.[1], Axel Dürrbaum[1], Gregor Zimmermann[2], Andreas Kroll[1]

[1]Department of Measurement and Control, University of Kassel
E-Mail: {farzad.rezazadeh, axel.duerrbaum, andreas.kroll}@mrt.uni-kassel.de

[2]German Technologies and Engineering Concepts, G.tecz Engineering GmbH
E-Mail: zimmermann@gtecz.com

Abstract

Concrete, the second most consumed resource worldwide after water [1], plays a fundamental role in construction. However, modeling the production process of concrete is challenging due to the incompletely understood physical and chemical relationships among its ingredients and various influencing factors, and the scarcity of data. Current models predominantly only rely on mix design (recipe) data, often overlooking the properties of fresh concrete, the interactions that result from curing conditions, and disturbances. This paper introduces a holistic view that integrates mix design, fresh concrete properties, and curing conditions to enhance predictive models for ultra-high performance concrete (UHPC) quality. This analysis highlights the significant effect of average power consumption, fresh concrete temperature, and curing storage conditions on the quality of concrete.

1 Introduction

Concrete is formulated from cement, aggregates (both fine and coarse), water, and occasionally, admixtures. Its production process begins with the combination of these raw materials, followed by a curing process to ensure the end product quality (Figure 1). The curing process typically requires maintaining specific moisture and temperature conditions for 28 days. This allows the cement to undergo hydration, the pivotal chemical reaction that imparts strength to concrete. The process's intricacy lies in the delicate balance of these components and stages, as well as its susceptibility to external environmental influences, resulting in potential variances in concrete quality [2]. The basic composition of conventional concrete is predominantly characterized by the amalgamation of primary constituents: Cement, fine and coarse aggregates, and water. However, advancements in concrete technology have underscored the integration of supplementary cementitious materials to optimize specific mechanical and rheological properties. Materials such as fly ash, silica fume, blast furnace slag, and superplasticizers, when judiciously incorporated into the mix, can enhance both the compressive strength (CS) and the workability of concrete. This yields, e.g., high-performance and ultra-high performance concrete (Table 1 [3]).

Table 1: Differences between conventional (CC), high-performance (HPC), and ultra-high performance concrete (UHPC) recipes and properties [3]. CS: Compressive strength.

Concrete type	Cement in kg/m^3	Water/binder in %	Workability in mm	CS in MPa
CC	260 – 380	0.45 – 0.65	-	20 – 50
HPC	400 – 700	< 0.4	455 – 810	50 – 100
UHPC	800 – 1000	0.2 – 0.3	260	> 100

Concrete production presents a multitude of challenges that influence the quality and consistency of the end product. The complexity of the process is influenced by the intrinsic properties of the raw materials, the mixing conditions and tools, the environmental factors, and the storage conditions (Figure 2).

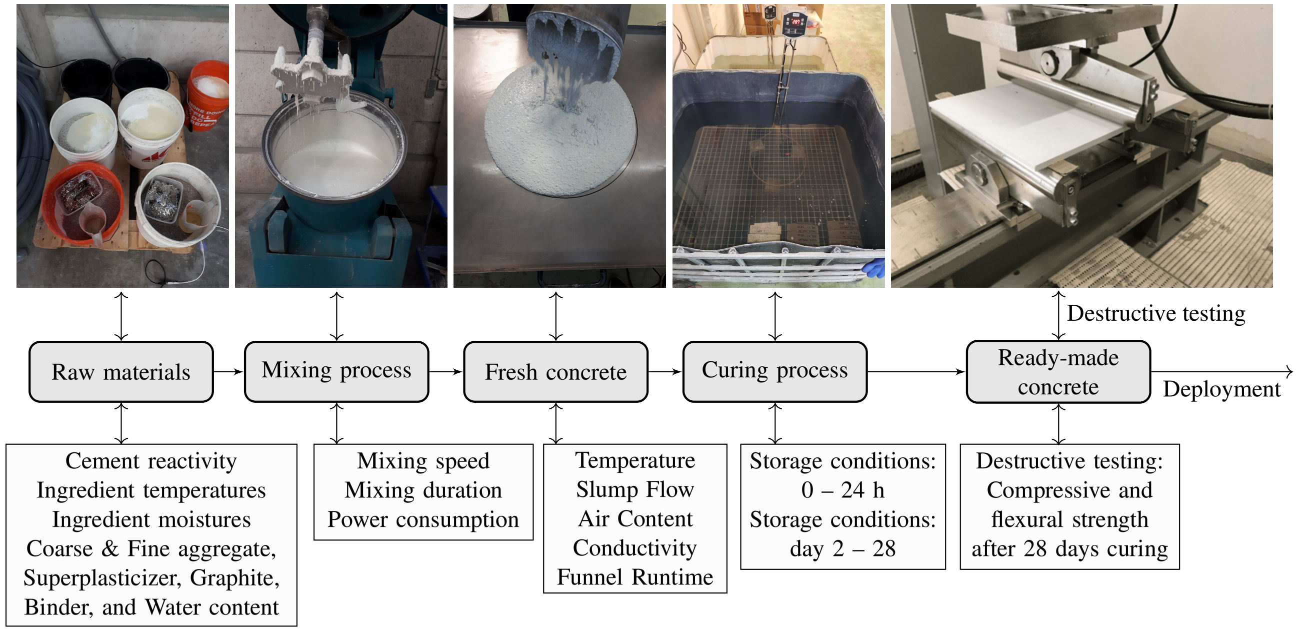


Figure 1: Concrete production and testing process: Mix design, fresh concrete properties, and curing conditions [4]

Traditional paradigms in concrete production modeling have predominantly gravitated towards mix designs, emphasizing input proportions and types [5, 6]. The mere act of mixing predetermined quantities does not invariably guarantee uniformity or the sought properties in the resultant concrete. Characteristics of fresh concrete, such as temperature and workability, are quintessential in predicting the final product quality [7]. Adding to this complexity, the curing process is inherently dynamic. Adjustments made herein, be it due to external environmental conditions or the targeted properties of the concrete, can substantially reshape its micro-structure, and by extension, its macro-behavior. Overlooking these complex nuances could culminate in a limited understanding of the production process, potentially manifesting as inconsistencies, inefficiencies, or even structural vulnerabilities [4].

The primary objective of this contribution, therefore, is to grasp the extent to which these multifaceted factors might shape the process and discern strategies to modulate or adapt them, ensuring reproducible outcomes. In light of these complexities and challenges, a comprehensive framework is proposed in this contribution to model the concrete production process. Designed to eclipse the constraints of traditional recipe-centric models, this framework assimilates insights from fresh concrete characteristics and delves deep into the intricacies of the curing process. Our contribution in this work can be summarized as:

- Determining the important influencing factors on the concrete process.
- Generating data based on the Taguchi orthogonal array L-50 [8] and the characteristics of fresh concrete.
- Adjusting different curing conditions to analyze their impact.
- Concrete process modeling based on four different approaches: Mix design, fresh concrete, curing conditions, and the entire production process, along with analysis of the results.

In our previous study [5], it was observed that two benchmark datasets, which neglected to consider environmental, mix process, and curing conditions in their content, exhibited distinctive behaviors when modeled using data-driven algorithms. In this paper, our primary focus is to analyze the exact impact of these omissions on modeling the concrete production process. Unlike in our

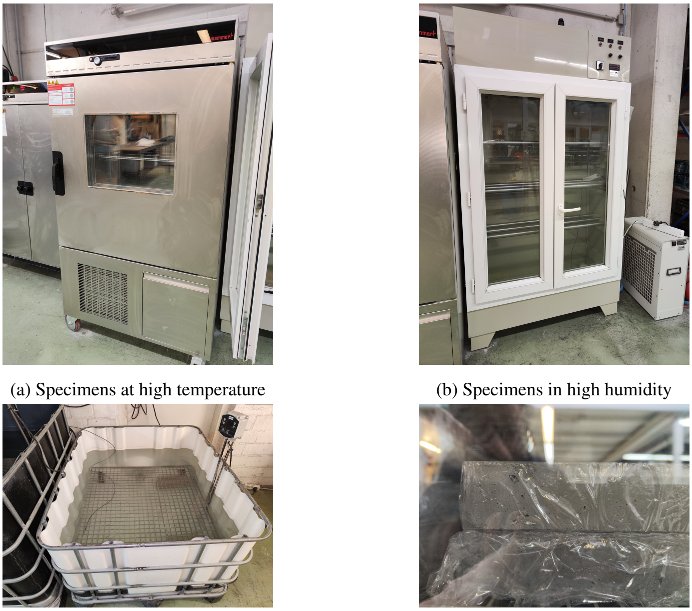

(a) Specimens at high temperature
(b) Specimens in high humidity
(c) Specimens under water
(d) Specimens in plastic foil

Figure 2: Illustration of different curing conditions in the concrete production process

previous work where multiple data-driven algorithms were compared, only the Gradient Boosting method will be used in this study to discern the effects of different modeling approaches.

2 Traditional Data-driven Concrete Modeling

Concrete quality estimation models largely fall into traditional models and machine learning approaches. The well-known Abram's law [9] relates the water-cement ratio (W/C) to the compressive strength (CS) after 28 days:

$$CS = \frac{b_1}{b_2^{W/C}}, \tag{1}$$

where b_1 and b_2 are empirical constants. Enhancing this, Zain et al. [10] introduced multiple linear regression, yielding

$$CS = b_0 + b_1 \frac{W}{C} + b_2 CA + b_3 FA + C. \tag{2}$$

Here, W denotes water volume, C represents cement, CA stands for coarse aggregate, and FA signifies fine aggregate. However, both methodologies neglect the ambient influences, mixing conditions, and the influence of the fresh concrete characteristics and curing conditions.

Modeling the concrete production process using traditional algorithms is challenging due to the many partially known effects on CS. Ling et al. [11] found that among Support Vector Machine (SVM), Artificial Neural Network (ANN), and Decision Tree, SVM was the superior method for studying the impact of environmental factors on CS. In contrast, Hoang et al. [12] determined that Gaussian Process Regression outperformed both ANN and SVM in estimating CS. Ensemble learning regression, however, provided the most accurate results, as indicated by [13]. Nevertheless, these studies overlook the properties of fresh concrete and curing conditions in their models.

Ozbay et al. [14] explored the mix proportions of high-strength self-compacting concrete using Taguchi's L-18 experimental design, focusing on six pivotal factors to achieve an optimal design. Notably, their work did not consider the potential influence of environmental factors and curing conditions on concrete production. Safranek [15] delved into the role of the mixing protocol, particularly examining the effects of mixing speed and time, in concrete production. Their findings suggest that UHPC necessitates an extended mixing period compared to its conventional counterpart to ensure uniformity. However, mixing at too high a speed could initiate thermal consequences, which might interfere with the chemical processes during blending. Cazacliu et al. [16] embarked on an investigation focusing on the importance of power usage patterns during the mixing process.

Assessing the workability of fresh concrete is vital, with the slump flow test being a key method [7, 17]. Kemer et al. [18] refined the correlation between yield stress and slump results. Hoang and Pham [19] employed LS-SVR for slump prediction. Farzampour [20] explored the relationship between environmental conditions during curing, and the impact of various cement types on concrete's compressive strength. Their findings highlighted that both severe weather conditions in the curing process and the water-to-cement ratio can significantly affect concrete quality.

While various subprocesses of concrete production have been investigated, modeling the entire process considering all major influences remains unexplored.

3 Holistic Concrete Production Modeling

In this contribution, a holistic modeling of the concrete production process is presented, integrating aspects of environmental factors, mix design, fresh concrete properties, and curing conditions. The Gradient Boosting (GB) algorithm [21, 22], in conjunction with recursive feature elimination (RFE) technique [23], is employed for this purpose. The selection of RFE was based on a comparative analysis with other standard methods, namely forward feature selection and backward feature elimination [24]. Among these techniques, RFE demonstrated superior performance, and as such, the outcomes of the other methods will not be discussed further. As for the choice of GB, the primary focus of this study is not to identify the optimal algorithm for modeling the concrete production process but rather to discern the influence of various factors on the final product's quality. Both Random Forest [25] and GB were considered in preliminary tests, with GB yielding better results. It's noteworthy that for techniques like RFE, only algorithms capable of inherently determining feature importance are viable, further justifying our choice.

The developed framework operates on a computer powered by an Intel(R) Core(TM) i9-10900X CPU with 64 GB RAM. Leave-One-Out Cross Validation (LOOCV) [26] learning processes with random initialization are conducted to validate result consistency. Subsequently, the average performance of Gradient Boosting is reported and analyzed, based on the test data garnered through the LOOCV process."

3.1 Modeling Approaches

In the context of mixing design, we refer to the specific recipes or raw material combination, along with their desired proportions (Figure 3). This modeling approach (MA 1) also encompasses the optimal mixing approach, including the appropriate speed and duration for mixing. The second modeling approach (MA 2) evaluates only the fresh concrete properties. Additionally, this modeling approach takes into account the average power consumption during the mixing process. The distinction between mixer adjustments (speed and duration) and average power consumption stems from two main factors. Firstly, mixer adjustments are controllable variables influenced by the type of raw materials, concrete type, and the desired attributes of the end product. Secondly, once water is introduced to the mixture, the subsequent mixing and the corresponding average power consumption after that offer insights into the rheological characteristics of fresh concrete. Because of that, mixer adjustments are analyzed in the first modeling approach (mix design), and average power consumption is examined in the second one. Unlike the second modeling approach that considers only the fresh concrete properties, in the third approach (MA 3), the effects of fresh concrete properties together with curing conditions are investigated (Figure 3).

During evaluation, the accuracy of the GB algorithm is assessed in the RFE process by selecting different numbers of features (3, 4, 5, 6, and 7) for three distinct modeling approaches. As fourth modeling approach (MA 4), a holistic approach integrates all modeling approaches to model the concrete production process (Figure 3). In this comprehensive attitude, the optimal number of features (3, 4, 5, 6, and 7) is re-evaluated using RFE to gauge the performance of the GB algorithm. The results for each phase are then analyzed to determine the most effective combination of factors from both modeling approaches for predicting concrete quality after a curing period of 28 days.

3.2 Recursive Feature Elimination

Recursive feature elimination is a method designed to address the issue of feature selection for machine learning algorithms. By training a model iteratively

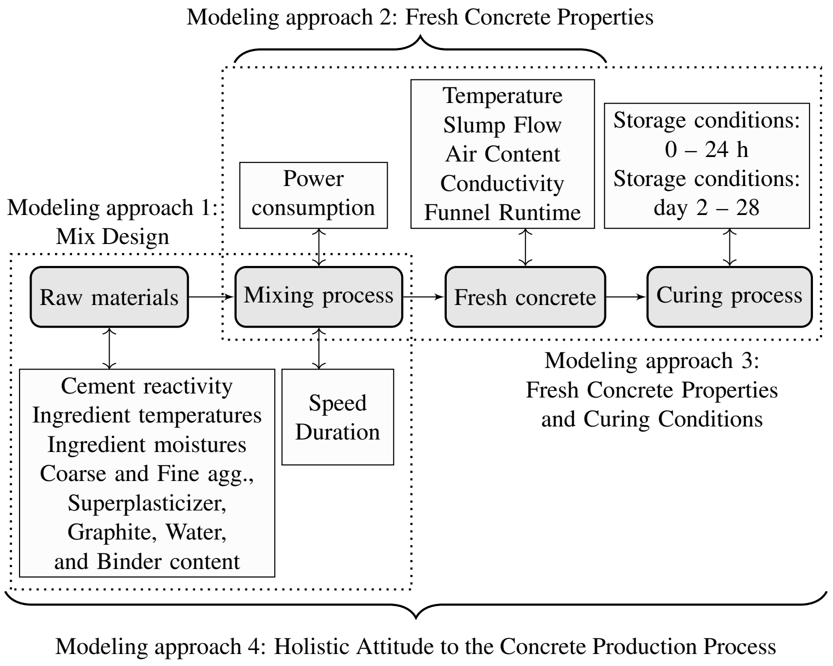

Modeling approach 2: Fresh Concrete Properties
Temperature
Slump Flow
Air Content
Conductivity
Funnel Runtime
Storage conditions: 0 – 24 h
Storage conditions: day 2 – 28
Power consumption
Modeling approach 1: Mix Design
Raw materials
Mixing process
Fresh concrete
Curing process
Cement reactivity
Ingredient temperatures
Ingredient moistures
Coarse and Fine agg.,
Superplasticizer,
Graphite, Water,
and Binder content
Speed
Duration
Modeling approach 3: Fresh Concrete Properties and Curing Conditions
Modeling approach 4: Holistic Attitude to the Concrete Production Process

Figure 3: Illustrating the holistic approach from raw material selection to the curing process, emphasizing mix design, fresh concrete properties, and curing conditions.

and eliminating systematically the least important features in each iteration, RFE ensures that only the most impactful features are retained. Gradient Boosting, by its nature, assigns feature importances based on how often a feature is employed to split the data across all trees. This ability makes GB an appropriate choice for RFE, as it can objectively rank features and provide a clear criterion for elimination. This recursive process continues until the desired number of features is retained (Algorithm 1).

3.3 Gradient Boosting Algorithm

Gradient Boosting is a machine learning algorithm that aims to construct a robust predictive model by iteratively building a series of weak learners. Typically, these learners are decision trees. The algorithm iterates by adjusting the

Algorithm 1.: Recursive Feature Elimination with Gradient Boosting

```
1: Input: Training data $X \in \mathbb{R}^{N \times D}$ with $N$ samples and $D$ features, targets $y \in \mathbb{R}^N$,
   Gradient Boosting model, desired number of features to select $k$
2: Output: Selected feature set $S$
3: Train the Gradient Boosting model on all features in $X$ to obtain feature importances
   $I$.
4: $S \leftarrow \{1, \ldots, D\}$                    ▷ Initialize feature set with all features
5: $n \leftarrow D$                                   ▷ Initialize with total number of features
6: while $n > k$ do
7:     Remove the feature with the lowest importance from $S$ and corresponding entry
       from $I$.
8:     Retrain the Gradient Boosting model on features in $S$ to obtain updated
       importances $I$.
9:     $n \leftarrow n - 1$
10: return Feature set $S$ with the $k$ top important features
```

weights of incorrectly predicted instances, ensuring that the following weak learner focuses more on these challenging instances. The entire process is governed by a predefined loss function, which the algorithm seeks to minimize (Algorithm 2).

4 Experiment Design, Data Collection, and Data Preprocessing

4.1 Controllable Influencing Factors

In order to achieve uniform quality and reproducibility in concrete production, identifying variables that affect consistency is crucial. This includes factors like mixing procedures, storage conditions, the presence of admixtures, and environmental influences, such as temperature and humidity. Tables 2 and 3 list the key factors that were chosen from an initial pool of 25 factors. In this study, cement is categorized as Cement-reactivity-class = 1 if it had been stored for long periods (more than one year), and as Cement-reactivity-class = 2 if it had been shortly stored (less than 3 months). Table 3 also detailed two curing scenarios: Storage-conditions-1T/C (first day storage conditions after mixing)

Algorithm 2.: Gradient Boosting Algorithm

```
1: Input: Training data $X \in \mathbb{R}^{N \times D}$ with $N$ samples and $D$ features, targets $y \in \mathbb{R}^N$,
   Number of boosting rounds $M$. $L(y_i, \gamma)$: Loss function measuring the discrepancy
   between the true target $y_i$ and prediction $\gamma$.
2: Output: Final boosted model $F_M(x)$
3: Initialize the model with a constant (mean value):
       $$F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, \gamma)$$
4: for $m = 1$ to $M$ do
5:     for each data point $i$ do
6:         Compute the negative gradient (pseudo-residuals):
               $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$
7:     Fit a weak learner $h_m(x)$ to pseudo-residual using $\{x_i, r_{im}\}$
8:     Compute multiplier:
           $$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i))$$
9:     Update the model:
           $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$$
10: return $F_M(x) = F_0(x) + \sum_{m=1}^{M} \gamma_m h_m(x)$
```

and Storage-conditions-28T/C (storage from day 2 to 28). During the first day, concrete was stored at 95 % humidity (Storage-conditions-1C = 1) or at 40 % humidity (Storage-conditions-1C = 2). From days 2 to 28, it was kept at 40 % humidity (Storage-conditions-28C = 1) or submerged in water (Storage-conditions-28C = 2). Given the costly and time-consuming nature of data collection in concrete production, 50 experiments were planned. Considering the factors detailed in Tables 2 and 3, and the constraints of the maximum number of experiments, the Taguchi Orthogonal Array L-50 was employed for data generation. The Taguchi Orthogonal Array ensures data robustness and an equal distribution of data points [8]. After curing for 28 days, the CS of the

Table 2: Factors of mix design: In this investigation, factor values span a range, represented by their designated levels (L: Level). For each category, two distinct aggregates are utilized: coarse and fine. These aggregates are labeled as type I and II within their categories.

Factor	Abb	Unit	L 1	L 2	L 3	L 4	L 5
Cement reactivity class	CRC	-	1	2	-	-	-
Ingredient moisture	IM	kg (%)	3.042 (4 %)	2.925 (0 %)	3.159 (8 %)	3.276 (12 %)	3.364 (15 %)
Ingredient temperature	IT	°C	10	20	25	30	40
Coarse aggregate I	CA-I	kg	6.900	6.000	5.400	6.300	5.100
Coarse aggregate II	CA-II	kg	8.925	10.500	11.550	9.975	12.075
Fine aggregate I	FA-I	kg	5.100	6.000	6.600	5.700	6.900
Fine aggregate II	FA-II	kg	0.863	0.750	0.675	0.788	0.638
Superplasticizer	SP	kg	0.290	0.323	0.306	0.355	0.339
Graphite	GP	kg	0.045	0.000	0.090	0.135	0.225
Mixing speed	MS	rad/s	200	350	500	350	350
Mixing duration	MD	s	300	300	300	210	480

specimens was determined using a destructive method. For each experiment, six specimens were tested, i.e. a total of 300 specimens were produced.

4.2 Fresh Concrete Properties

After each mixing process, the properties of fresh concrete are measured. A comprehensive overview of the general characteristics of each property can be found in Table 4. Fresh concrete temperature depends on the concrete mix condition [15], environmental factors, and raw material temperatures. Chemical reactions, notably cement hydration, can affect temperature too. High temperatures reduce workability, and low temperatures can extend setting times.

The air content test gauges the volume of air in fresh concrete as a percentage of its total volume, affecting durability and strength. While higher air content enhances workability and freeze-thaw resistance, it diminishes compressive strength.The average power consumption in concrete production indicates the

Table 3: Factors of the Curing Process: In this investigation, curing condition factors vary across a range, represented by their designated levels. The numbers before T and C denote the curing period in days. (T: Temperature; C: Class; L: Level)

Factor	Abb	Unit	L 1	L 2	L 3	L 4	L 5
Storage-conditions-1T	SC-1T	°C	20	20	10	30	40
Storage-conditions-1C	SC-1C	-	1	2	2	2	2
Storage-conditions-28T	SC-28T	°C	20	20	10	30	40
Storage-conditions-28C	SC-28C	-	1	2	2	2	2

mean power utilized for mixing raw materials and overcoming mixture resistance throughout the entire duration of the process. Environmental factors and mixer properties can influence the average power needs. Similarly, chemical reactions, notably between water and cement, can modify the average power demands. High average power consumption might hint at issues like insufficient water, while low average power may suggest a weak mix. Electrical conductivity in fresh concrete reflects largely the ionic content in the liquid phase. This property can indicate the water-to-cement ratio, vital for workability and durability.

The slump flow test evaluates the flowability of fresh concrete, particularly for fluid mixes like self-compacting concrete. Concrete is placed in a slump cone with an outlet diameter of 120 mm. When the cone is lifted, the concrete spreads, and after t = 30, 60, and 120 seconds, the diameter of the spread gives the slump flow test value [7]. High values suggest increased flowability, which can lead to issues like segregation, while low values might pose placement challenges. The funnel runtime assesses the flowability of fresh self-compacting concrete by timing its flow through a V-shaped funnel. Extended funnel times indicate workability concerns, while short times indicate risks like segregation or bleeding.

4.3 Data Preprocessing

In data preprocessing, steps were taken to ensure data integrity. Manual checks are conducted to verify the absence of outliers. The L-50 Taguchi Orthogonal

Table 4: Observed Quantities Related to Fresh Concrete

Factor	Abb	Unit	Min	Mean	Max	STD
Fresh Concrete Temp.	FCT	°C	17.60	25	31.90	3.94
Air Content	AC	%	0.40	1.93	7	1.53
Average Power Consumption	PC	kW	0.37	0.90	1.40	0.24
Slump Flow	SF	mm	120	327.33	395	53.36
Conductivity	CD	V	4.54	4.62	4.74	0.05
Funnel Runtime	FR	s	4	8.05	15	2.69

Array minimizes collinearity risks, and no issues were found. Six missing values in the fresh concrete characteristics led to the exclusion of related experiments. As a result of excluding the related experiments due to the six missing values in the fresh concrete characteristics, the analysis is based on the remaining 44 datapoints. In the project, min-max normalization was chosen due to the presence of varied scales and feature types, the absence of negative values, and the lack of outliers. Additionally, the use of the Taguchi Orthogonal Array inherently facilitated the application of min-max normalization to ensure consistent interpretation across all factors. If $X \in \mathbb{R}^{N\times D}$, each entry can be denoted as X_{ij}, where i ranges from 1 to N and j ranges from 1 to D:

$$X'_{ij} = \frac{X_{ij} - \min_j(X)}{\max_j(X) - \min_j(X)} \tag{3}$$

4.4 General setting for experiments

In the 50 experiments, the same mixing tool is employed (Figure 1). Environmental conditions for material storage and production are controlled, mitigating seasonal influences. All experiments utilized a single material batch for consistent properties. Both the old and new cement were of the same type and originated from the same factory, and production conditions. The mixer chamber temperature was measured before each experiment. Given that the laboratory's ambient temperature was consistently maintained at 20 °C, the

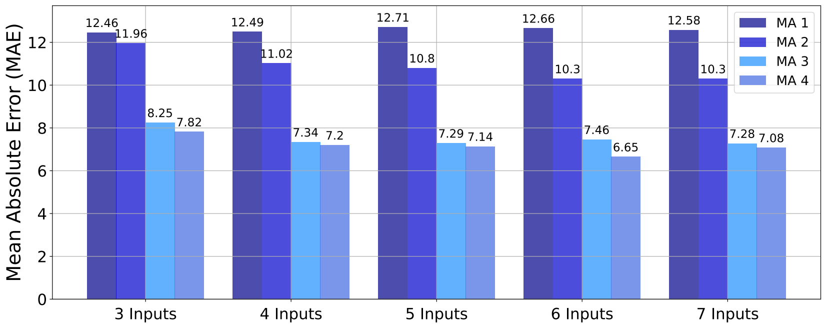


Figure 4: Comparing the prediction accuracy of Gradient Boosting across different modeling approaches (MAs) and also the number of features to be selected by REF. The barplot succinctly illustrates the average performance on the test data, derived from 44 LOOCV iterations. (MA 1: Mix Design, MA 2: Fresh Concrete Properties, MA 3: Fresh Concrete Properties & Curing Conditions, MA 4: Entire Concrete Production Process)

mixer chamber temperature was also close to 20 °C. As a result, this factor did not introduce any variability into the process.

5 Results and Discussion

In this study, the prediction accuracy of Gradient Boosting under four modeling approaches and the number of features selected (either three, four, five, six, or seven) by RFE are analyzed (Figure 4). The objective is to find the combinations of influencing factors from the entire concrete production process that would result in optimal model accuracy. When comparing the prediction accuracy of the models using the same number of selected features across the four modeling approaches (Figure 4), model training on the entire concrete production process consistently yielded the lowest MAE for all modeling approaches. Specifically, utilizing the complete concrete production process with six features yielded the most accurate results, achieving an MAE of 6.65. The mix design consistently exhibited the largest error, indicating that this subset of data might not be as informative for predictions compared to either the fresh concrete and curing conditions data or the comprehensive data from the entire process. In summary, adding more features doesn't always guarantee enhanced performance across all modeling approaches. The data underscores the need for careful feature selection. In the evaluation of feature contribution frequency across the considered modeling approaches, distinct patterns emerged (Figure 5). Within the mix design modeling approach, Ingredient-temperature (44 times) and Mixing-duration (44 times) distinctly stood out, highlighting their central role in modeling the recipe. Additionally, Superplasticizer (40 times), Mixing-speed (41 times), and Graphite (36 times) are of notable significance, reinforcing their essential roles in the mix design modeling approach. Conversely, Cement-reactivity-class (1 time) and Coarse-aggregate-II (2 times) showed minimal importance.

Although the modeling approach was based on fresh concrete data, it isn't elaborated upon in the discussion. This is due to the fact that only 6 fresh concrete features exist, which matches the number of inputs selected in the considered modeling method. In the fresh concrete properties and curing conditions modeling approach, average Power-consumption (44 times), Storage-conditions-28-T (44 times), and Storage-conditions-1-T (44 times) are consistently selected, emphasizing their significant roles in the modeling. Furthermore, Fresh-concrete-temperature (42 times) and Air-content (41 times) made significant appearances, underscoring their relevance. In contrast, electrical conductivity and Slump-flow are less influential.

In the entire concrete production process, the terms average Power consumption, Fresh-concrete-temperature, and Storage-conditions-28T each appeared 44 times, underscoring their critical roles. Storage-conditions-1T (43 times) and Superplasticizer (38 times) also held significant positions. However, features like electrical conductivity (1 time), Funnel-runtime (8 times), Air-content (12 times), and Slump-flow (2 times) are less prominent. To culminate, when examining combinations for a comprehensive representation of the concrete production process, the data from the entire process suggest that average Power-consumption, Fresh-concrete-production, Storage-conditions-28T, Storage-conditions-1T, Superplasticizer, and Graphite are the most vital. This combination promises a comprehensive and accurate modeling of the concrete production process.

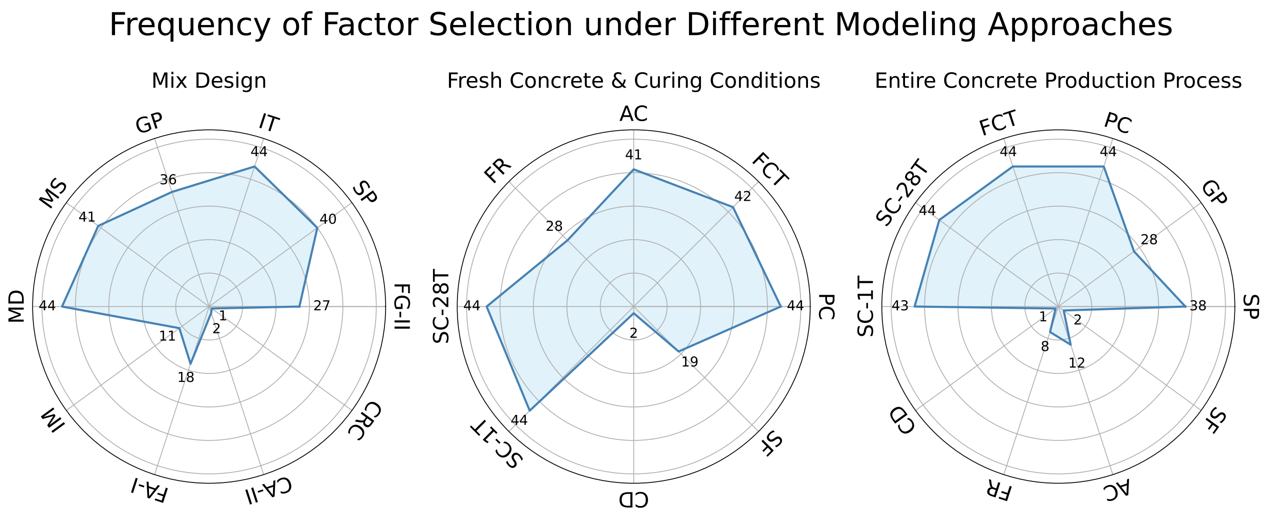


Figure 5: Illustration of how often each of the six influencing factors is selected during the Gradient Boosting training process based on LOOCV by RFE in different modeling approaches. There is a total of 44 iterations. Results from the modeling approach based solely on fresh concrete data are not shown. This is because there are only six features for fresh concrete, and the number of inputs considered for selection in the modeling approaches is also six. This implies that all features were selected in every iteration. **SC-28T:** Storage-conditions-28T, **PC:** Average Power Consumption, **SC-1T:** Storage-conditions-1T, **FCT:** Fresh Concrete Temperature, **SP:** Superplasticizer, **GP:** Graphite, **AC:** Air Content, **MD:** Mixing duration, **IT:** Ingredient temperature, **MS:** Mixing speed, **FR:** Funnel Runtime, **FG-II:** Fine-aggregate-II, **SF:** Slump Flow, **FA-I:** Fine-aggregate-I, **IM:** Ingredient moisture, **CD:** Conductivity, **CA-II:** Coarse-aggregate-II, **CRC:** Cement-reactivity-class, **SC-28C:** Storage-conditions-28C, **CA-I:** Coarse-aggregate-I, **SC-1C:** Storage-conditions-1C

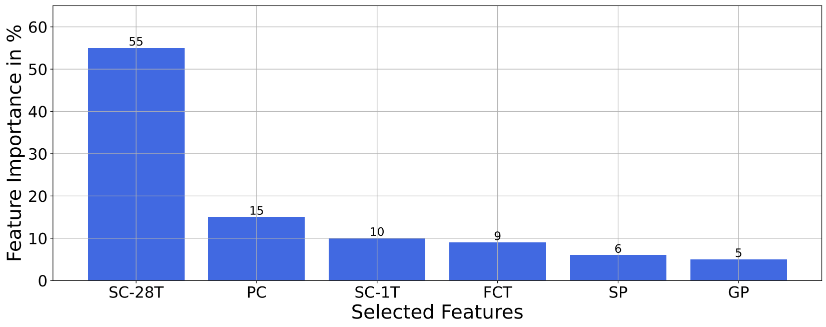


Figure 6: Illustrating the percentage importance of features selected for predicting the compressive strength of concrete, as determined by the chosen GB algorithm, using six inputs and the entire concrete production process as the modeling approach. The SC-28T feature exhibits the highest importance. **SC-28T:** Storage-conditions-28T, **PC:** Average Power Consumption, **SC-1T:** Storage-conditions-1T, **FCT:** Fresh Concrete Temperature, **SP:** Superplasticizer, **GP:** Graphite

In Figure 6, a detailed breakdown of the feature importance is presented. This breakdown was determined from a model that was identified from a series of models trained using various approaches based on LOOCV. Among all these modeling approaches, the one that delivered the best accuracy performance was selected. Within this chosen approach, several models were generated due to the nature of LOOCV. From these models, the one exhibiting a performance closest to the average performance over LOOCV was selected. The feature importances displayed in Figure 6 are derived from this specific model. The chart illustrates that the feature Storage-conditions-28T is of the highest importance, contributing 55 % to the decision-making process of the model. This is followed by average Power-consumption at 15 %, with the remaining features each contributing less than 11 %. In general, that means during the monitoring of the concrete production process, from mix design to the final fresh concrete state, one can predict the eventual quality of the end product. If this predicted quality falls short or is not up to the desired standard, modifications can be made to the curing conditions. By implementing these suitable adjustments, it becomes feasible to achieve the desired quality for the final product, ensuring that the concrete aligns with or surpasses the established benchmarks.

6 Conclusion and Future work

In our previous study [5], it was observed that two benchmark datasets, which neglected to consider environmental, mix process, and curing conditions in their content, exhibited distinctive behaviors when modeled using data-driven algorithms. The presented research underscores the intricacies inherent in the concrete production process and the significance of incorporating mix design, fresh concrete properties, and curing conditions to enhance predictive models for UHPC quality. With this perspective in mind, modifications can be made to the curing conditions. By implementing these suitable adjustments, it becomes feasible to achieve the desired quality for the final product, ensuring that the concrete aligns with or surpasses the established benchmarks.

This contribution also emphasizes that it is not necessary for modeling to measure all factors in the concrete production process. This insight is particularly valuable for concrete plants, considering the costs associated with sensors and the monitoring process. This investigation identified the crucial factors pivotal in enhancing the predictive model's precision, namely: average Power-consumption, Fresh-concrete-temperature, Storage-conditions-28T, Storage-conditions-1T, Superplasticizer, and Graphite. However, it's worth noting that this study was conducted under laboratory conditions. In a real concrete plant, the situation might differ. For instance, controlling the curing process is tough. Wear of the mixing tools and outdoor storage of raw materials, especially before mixing in harsh weather, can impact product quality.

For our subsequent steps, we aim to generate more data and delve deeper into modeling the concrete production process.

Acknowledgment

This project (HA project no. 1011/21-13) is financed with funds of LOEWE – Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz, Förderlinie 3: KMU-Verbundvorhaben (State Offensive for the Development of Scientific and Economic Excellence).

LOEWE

Exzellente Forschung für Hessens Zukunft

HESSEN

Hessisches Ministerium für Wissenschaft und Kunst

We thank *Tino Sablotny*, *Michelle Apholz*, and *Amin Abrishambaf* from *G.tecz Engineering GmbH* for carrying out the experiments, and joint work on the analysis of the results.

References

[1] T. Czigler, S. Reiter, P. Schulze and K. Somers. “Laying the foundation for zero-carbon cement”. In: *McKinsey and Company Chemicals Insights* 9. 2020.

[2] Z. Li, J. Yoon, R. Zhang, F. Rajabipour, W. V. Srubar III, I. Dabo and A. Radlińska. “Machine learning in concrete science: applications, challenges, and best practices”. In: *Npj Computational Materials* 8.1. p. 127. 2022.

[3] M. T. Marvila, A. R. G. de Azevedo, P. R. de Matos, S. N. Monteiro and C. M. F. Vieira. “Materials for production of high and ultra-high performance concrete: review and perspective of possible novel materials”. In: *Materials* 14.15. p. 4304. 2021.

[4] F. Rezazadeh P., A. Dürrbaum, G. Zimmermann and A. Kroll. “Leveraging ensemble structures to elucidate the impact of factors that influence the quality of ultra–high performance concrete”. In: *IEEE Symposium Series on Computational Intelligence (SSCI)* Accepted. 2023.

[5] F. Rezazadeh and A. Kroll. “Predicting the compressive strength of concrete up to 28 days-ahead: comparison of 16 machine learning algorithms on benchmark datasets”. In: *Workshop Computational Intelligence* 1, p. 52. 2022.

[6] I. C. Yeh. “Modeling of strength of high-performance concrete using artificial neural networks”. In: *Cement and Concrete Research* 28, pp. 1797–1808. 1998.

[7] A. Dürrbaum, F. Rezazadeh P. and A. Kroll. “Automatic camera-based advanced slump flow testing for improved reliability” In: *IEEE Sensors* Accepted. 2023.

[8] G. Taguchi. “System of experimental design: engineering methods to optimize quality and minimize costs”. In: *UNIPUB/Kraus International Publications, American Supplier Institute* 1987.

[9] S. Propovics. “Analysis of concrete strength versus water-cement ratio relationship”. In: *Materials Journal* 87.5. pp. 517–529. 1990.

[10] M. F. M. Zain, S. M. Abd, K. Sopian, M. Jamil and A. I. Che-Ani. “Mathematical regression model for the prediction of concrete strength”. In: *WSEAS International Conference. Mathematics and Computers in Science and Engineering* 10. 2008.

[11] H. Ling, C. Qian, W. Kang, C. Liang and H. Chen. “Combination of support vector machine and k-fold cross validation to predict compressive strength of concrete in marine environment”. In: *Construction and Building Materials* 206. pp. 355–363. 2019.

[12] N. D. Hoang, A. D. Pham, Q. L. Nguyen and Q. N. Pham. “Estimating compressive strength of high performance concrete with Gaussian process regression model”. In: *Advances in Civil Engineering*. 2016.

[13] J. Yu, R. Pan, and Y. Zhao. “High-dimensional, small-sample product quality prediction method based on MIC-stacking ensemble learning”. In: *Applied Sciences* 12.1. p. 23. 2021.

[14] E. Ozbay, A. Oztas, A. Baykasoglu, and H. Ozbebek. “Investigating mix proportions of high strength self compacting concrete by using Taguchi method”. In: *Construction and Building Materials* 23. p. 694. 2009.

[15] K. Safranek. “Influence of different mixing processes on the strength of ultra-high strength concretes”. Available at:

https://repositum.tuwien.at/handle/20.500.12708/10881 Diploma thesis, Vienna University of Technology, reposiTUm. 2007.

[16] B. Cazacliu, and N. Roquet. “Concrete mixing kinetics by means of power measurement”. In: *Cement and Concrete Research* 39. p. 182. 2009.

[17] H. Y. Aydogmus et al. “A comparative assessment of bagging ensemble models for modeling concrete slump flow”. In: *Computers and Concrete* 16. pp. 741–757. 2015.

[18] H. Kemer, R. Bouras, M. Sonebi, N. Mesboua, and A. Benmounah. “Slump test: a new empirical model for high yield stress materials”. In: *Journal of Applied Engineering Sciences* 11. pp. 107–112. 2021.

[19] N. D. Hoang, and A. D. Pham. “Estimating concrete workability based on slump test with Least squares support vector regression”. In: *Journal of Construction Engineering* 2016.

[20] A. Farzampour. “Compressive behavior of concrete under environmental effects”. In: *Compressive Strength of Concrete* pp. 92–104. 2019.

[21] J. H. Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of Statistics* pp. 1189–1232. 2001.

[22] F. Pedregosa, et al. “Scikit-learn: Machine learning in Python”. In: *Journal of Machine Learning Research* 12. p. 2825. 2011.

[23] X. Chen and J. C. Jeong. “Enhanced recursive feature elimination”. In: *International Conference on Machine Learning and Applications (ICMLA)* pp. 429–435. 2007.

[24] F. J. Ferri, P. Pudil, M. Hatef, and J. Kittler. “Comparative study of techniques for large-scale feature selection”. In: *Machine Intelligence and Pattern Recognition* 16. pp. 403–413. 1994.

[25] L. Breiman. “Random forests”. In: *Machine Learning* 45. pp. 5–32. 2001.

[26] P. Refaeilzadeh, L. Tang and H. Liu. “Cross-validation”. In: *Encyclopedia of Database Systems* 5. pp. 532–538. 2009.

Modellierung der Raumluftfeuchtigkeit in historischen Gebäuden: Ein Vergleich datengetriebener Methoden in der Praxis

Marcel Zehner, Alessio Cavaterra, Steven Lambeck

FB Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123, 36037 Fulda
E-Mail: {marcel.zehner,alessio.cavaterra,steven.lambeck}@et.hs-fulda.de

1 Einführung

Bei der Lagerung und Ausstellung von Kulturgütern in historischen Gebäuden spielen die Temperatur und relative Luftfeuchtigkeit eine bedeutende Rolle für deren langfristigen Erhalt. Damit ein beschleunigter Zerfall der Kulturgüter verhindert werden kann, müssen die Raumklimakomponenten gemäß dem Fachgebiet der Präventiven Konservierung (PK) innerhalb vorgegebener konstanter und dynamischer Grenzwertbereiche liegen. Insbesondere der relativen Raumluftfeuchtigkeit und deren Änderungsrate wird eine hohe Relevanz beigemessen [1]. Durch die Implementierung eines modellprädiktiven Regelungsansatzes (MPC, engl.: model predictive control) ist es möglich, die konservatorischen Anforderungen in Form von Regelgrößenbeschränkungen im Regelgesetz zu berücksichtigen. Im Hrabanus-Maurus-Saal (HRMS) innerhalb der Bibliothek des Bischöflichen Priesterseminars in Fulda soll ein solcher modellprädikitver Regelungsansatz realisiert werden. Dort lagern historisch wertvolle Schriften, welche den oben genannten konservatorischen Anforderungen genügen müssen. Bevor der MPC praktisch in der empfindlichen Lagerumgebung umgesetzt werden kann, muss eine simulative Erprobung und Evaluierung des Reglers stattfinden, um einen zuverlässigen Regelbetrieb gewährleisten zu können. Hierfür wird ein Simulationsmodell des HRMS benötigt, welches das hygrothermische Verhalten des Saals mit hoher Genauigkeit approximiert. Verschiedene Gebäudesimulationsumgebungen, wie beispielsweise EnergyPlus [2] oder TRNSYS [3] erlauben es, das hygrothermische Verhalten eines Gebäudes mitsamt der darin enthaltenen Räume detailgetreu nachzubilden. Für die Nutzung einer solchen Gebäudesimulationsumgebung sind jedoch eine Vielzahl an bauphysikalischen Informationen notwendig. Dazu zählen u. A. die exakten Gebäudeabmessungen, der Wandaufbau oder auch die verwendeten Baustoffe. Da für den HRMS nur sehr wenige bauphysikalische Informationen vorliegen, muss an dieser Stelle ein datengetriebener Modellierungsansatz gewählt werden. Durch langjährige Messungen innerhalb und außerhalb des HRMS steht eine große Datenbasis zur Verfügung. Hiermit ist eine Identifikation eines Simulationsmodells des HRMS möglich, welches im Anschluss für die Reglerevaluierung genutzt werden kann. Der Sachverhalt, dass zusätzlich zum internen Prozessmodell des MPC ein weiteres weitaus detaillierteres Modell des zu regelnden Prozesses notwendig ist, wird in [4] beschrieben. Der vorliegende Beitrag beinhaltet die dynamische Modellierung des hygrischen Verhaltens des HRMS. Auf eine hygrothermische Modellierung des Raums, welche auch das Wärmeübertragungsverhalten einschließt, wird an dieser Stelle verzichtet. In den folgenden Abschnitten werden verschiedene Modellansätze zur datengetriebenen Systemidentifikation vorgestellt. Nach dem Training der Modelle mit Hilfe der Messdaten des HRMS werden diese anschließend miteinander verglichen. Der Beitrag schließt mit einer Diskussion der Ergebnisse ab.

2 Datengetriebene Modellbildung

2.1 Datenbasis

Die Datenbasis erstreckt sich über einen Zeitraum von 140 Tagen vom 20. Dezember 2022 bis zum 8. Mai 2023, wobei die Messdaten mit einer Abtastzeit von 10 Minuten erfasst wurden. Die Daten sind teilweise im geschlossenen Regelkreis und teilweise im offenen Regelkreis aufgezeichnet worden. Im geschlossenen Regelkreis sind Zweipunktregler mit Schwellwerten eingesetzt worden, deren Reglerparameter mehrmals geändert worden sind. Zwischenzeitlich ist der Regelkreis geöffnet worden, um Testsignale in Form von APRB-

Tabelle 1: Übersicht der gemessenen Größen für die Identifikation des Simulationsmodells

	Formelzeichen	Beschreibung
1	ϑ_R	Raumtemperatur
2	ϑ_A	Außentemperatur
3	$\dot{Q}_y$	Heizeingriff im Raum
4	$\dot{Q}_g$	Globalstrahlung
5	$\dot{m}_h$	Be- und Entfeuchtungseingriff im Raum
6	φ_A	rel. Außenluftfeuchtigkeit
7	φ_R	rel. Raumluftfeuchtigkeit

und Chirpsignalen auf den Prozess zu schalten. Die Aufzeichnungen werden in Trainings-, Validierungs- und Testdaten aufgeteilt, wobei die Trainingsdaten 90 % der Gesamtdatenmenge repräsentieren, während jeweils 5 % der Daten für Validierungs- und Testdaten vorgesehen sind. Die Aufteilung der Datensätze hängt von vielen Faktoren ab: beispielsweise müssen die jahreszeitlichen Einflüsse auf das Prozessverhalten sowie Abschnitte mit deutlichen Stellgrößeneingriffen im Trainingsdatensatz enthalten sein, um eine repräsentative Datenbasis zu erreichen. Alle drei Teildatensätze stellen jeweils eine aufeinanderfolgende Zeitreihe dar, um den temporalen Zusammenhang der Daten zu erhalten. Der Testdatensatz beinhaltet den Anfang der Gesamtdatenmenge, während der Validierungsdatensatz am Ende des Gesamtdatensatzes verortet ist. Die übrigen Daten sind Bestandteil des Trainingsdatensatzes.
Im Rahmen der Datenerhebung sind die in der Tabelle 1 aufgelisteten Messgrößen erfasst worden. Im Zuge der Identifikation werden die Messgrößen eins bis sechs als Systemeingänge genutzt, wohingegen die rel. Raumluftfeuchtigkeit φ_R als Systemausgang definiert ist und die zu identifizierende Größe darstellt. Dadurch ergibt sich ein Mehrgrößensystem mit mehreren Eingängen und einem Ausgang (engl. Multi-Input-Single-Output-System, MISO-System). Für das Training der Modelle werden alle Daten auf einen einheitlichen Wertebereich skaliert.

2.2 Takagi-Sugeno-NARX-Modelle

Die gesammelten Daten stellen allesamt Zeitreihen dar, welche mit Hilfe von ARX-Zeitreihenmodellen (engl. autoregressive-model-with-exogenous-input) angenähert können. Im Folgenden wird beispielhaft ein linear-affines SISO-ARX-Modell veranschaulicht.

$$\hat{y}_{\mathrm{ARX}}(k) = \underline{\boldsymbol{\theta} x} = \left(b_1, \ldots, b_{n_b}, -a_1, \ldots, -a_{n_a}, 1\right) \begin{pmatrix} u(k-1) \\ \vdots \\ u(k-n_b) \\ y(k-1) \\ \vdots \\ y(k-n_a) \\ o \end{pmatrix} \tag{1}$$

Aufgrund der zugrundeliegenden Nichtlinearitäten in Wärme- bzw. Feuchteübertragungsvorgängen eignen sich lineare ARX-Modelle gemäß Gleichung 1 nur bedingt, wenn mit dem geschätzten ARX-Modell eine Simulation (unendlicher Vorhersagehorizont) des Prozesses angestrebt wird. Zur Approximation nichtlinearer Prozesse werden deshalb in der einschlägigen Literatur verschiedene Erweiterungen vorgeschlagen, die unter dem Begriff der NARX-Modelle (N für engl. nonlinear) eingruppiert werden. NARX-Modelle erweitern die ARX-Modellstruktur in Gleichung 1 um eine nichtlineare Funktion $f(\cdot)$, wie in der folgenden Gleichung 2 dargestellt wird.

$$\hat{y}_{\mathrm{NARX}}(k) = f(\underline{\boldsymbol{\theta} x}) \ . \tag{2}$$

Takagi-Sugeno (TS) Fuzzy-Modelle beschreiben eine Klasse von nichtlinearen Modellen mit einem unscharfen (engl. fuzzy) Regelwerk, welche in der Konklusion eine oder mehrere scharfe Ausgangsgrößen, z. B. über ARX-Modelle, bilden. Derartige TS Fuzzy-Modelle werden häufig kurz als TS-NARX-Modelle bezeichnet. In diesem Beitrag wird im weiteren Verlauf TSNX als Abkürzung genutzt.

Zur Schätzung eines passenden TSNX-Modells wurde die LMN-Toolbox [5] mitsamt des darin implementierten LOLIMOT-Konstruktionsalgorithmus verwendet.

$$\underline{\hat{y}}_{\mathrm{TSNX}}(k) = \sum_{i=1}^{n_m} \Phi_i(\underline{z})\underline{\theta}_{\mathrm{TSNX},i}\underline{x}\,. \tag{3}$$

Die Fuzzy-Basisfunktionen $\Phi_i(\underline{z})$ haben ihren Wertebereich zwischen null und eins. Berechnet werden sie aus dem Mittel der Zugehörigkeitsgrade

$$\Phi_i(\underline{z}) = \frac{\mu_i(\underline{z})}{\sum\limits_{i=1}^{n_m} \mu_i(\underline{z})} \quad , \quad \sum_{i=1}^{n_m} \Phi_i(\underline{z}) = 1\,. \tag{4}$$

Alle $\mu_i(\underline{z})$ Zugehörigkeitsfunktionen (ZF) werden als Gaußglocken definiert. Mit dem Zentrum ν und der Standardabweichung σ gilt für eine Gauss'sche ZF:

$$\mu_{\mathrm{Gauss}}(x) = \exp\left(\frac{-(x-\nu)^2}{2\sigma^2}\right)\,. \tag{5}$$

Die Schedulingvariable $\underline{z}$ ist im vorliegenden Fall ein Vektor, der die Außentemperatur, die Innentemperatur, die relative Luftfeuchtigkeit im Außenbereich und die relative Luftfeuchtigkeit im Innenraum enthält. Diese Größen werden allesamt aus dem letzten Abtastschritt herangezogen.

$$\underline{z} = [\varphi_R(k-1),\ \vartheta_R(k-1),\ \varphi_A(k-1),\ \vartheta_A(k-1)]^T \tag{6}$$

Die Festlegung der Hyperparameter wird durch eine systematische Suche realisiert. Die Anzahl n_a und $n_{b,1},\ldots,n_{b,6}$ der zurückliegenden Regressoren wird variiert, sodass vergangene Einflussgrößen aus einer, zwei, vier, acht bzw. zwölf Stunden in die Modelle einfließen. Bei einer 10-minütigen Abtastung des Datensatzes entspricht dies 6, 12, 24, 48 und 72 Abtastschritte je Einflussgröße. Als Abbruchkriterium für den LOLIMOT-Algorithmus wird die maximale Teilmodellanzahl ausgewählt und im Bereich von 20, 40, 80 und 160 variiert.

Das resultierende TS-Modell besteht aus $n_m = 16$ Teilmodellen mit insgesamt $n_{\mathrm{par}} = 8080$ Parametern und greift auf die Einflussgrößen der letzten zwölf Stunden zurück. Im Zuge des Vergleichs mit den anderen Modellen wird das

TSNX-Modell simuliert, wobei der Modellausgang zurückgeführt wird und eine Output-Error-Struktur (OE) entsteht.

2.3 Local Model State Space Networks

Local Model State Space Networks (LMSSN) repräsentieren eine neue Klasse nichtlinearer Zustandsraummodelle, deren Struktur auf lokalen Modellnetzen (LMN, engl. local model networks) aufbaut [6]. Zur Approximation der Nichtlinearitäten eines Prozesses wird eine ähnliche Herangehensweise basierend auf einem Regelwerk wie bei den zuvor besprochenen TSNX-Modellen gewählt: mehrere Teilmodelle, welche für unterschiedliche Arbeitspunkte, externe Planungsparameter usw. gültig sind, können einen nichtlinearen Prozess besser abbilden, als ein vergleichbares lineares Zustandsraummodell. Die Verknüpfung dieser Teilmodelle zu einem nichtlinearen Gesamtmodell ermöglicht hierdurch große Gültigkeitsbereiche. Die Besonderheit bei LMSSN ist, dass die nichtlinearen Funktionen $f(\cdot)$ und $g(\cdot)$ in

$$\underline{x}(k+1) = f(\underline{x}(k), u(k)) \tag{7}$$

$$y(k) = g(\underline{x}(k), u(k)) \tag{8}$$

auf unterschiedliche Art und Weise angenähert werden können. Einerseits besteht die Möglichkeit jede i-te Zeile der Zustandsgleichung (7) als einzelnes LMN zu betrachten und anschließend für jedes einzelne LMN einen Konstruktionsalgorithmus wie LOLIMOT oder HILOMOT separat zu nutzen. Andererseits kann die gesamte Zustandsgleichung in einem lokalen Modellnetz in Form eines MIMO-LMSSN berücksichtigt und anschließend mit den genannten Algorithmen konstruiert werden. Gleiches gilt für die Ausgangsgleichung der Zustandsraumdarstellung (8). Die LMSSN-Toolbox bietet dem Nutzer zudem die Möglichkeit eine individuelle Auswahl der mit nichtlinearen Phänomenen behafteten Eingangsgrößen und Zustandsgrößen. Somit kann vom Nutzer gezielt apriorisches Wissen in den Algorithmus eingebracht und möglicherweise der Rechenaufwand verringert werden. Das in diesem Beitrag vorgestellte LMSSN zur Modellierung der Feuchteübertragungsmechanismen stellt ein nichtlineares, zeitdiskretes Zustandsraummodell zweiter Ordnung ($n_x = 2$)

mit $n_p = 6$ Eingängen und einem Ausgang $n_q = 1$ dar.

$$\hat{x}_i(k+1) = \sum_{j=1}^{n_{m,i}} \left[o_{i,j} + \underline{a}_{i,j}^T \underline{\hat{x}}(k) + \underline{b}_{i,j}^T \underline{u}(k)\right] \Phi_{i,j}^{[s]}(k) \; , \tag{9}$$

$$\hat{y}_{\text{LMSSN}}(k) = \sum_{m=1}^{n_o} \left[p_m + \underline{c}_m^T \underline{\hat{x}}(k) + \underline{d}_m^T \underline{u}(k)\right] \Phi_m^{[o]}(k) \; . \tag{10}$$

Jede Zeile in dieser Zustandsraumdarstellung ist als eigenständiges LMN definiert, sodass das LMSSN insgesamt aus drei einzelnen LMN besteht. Die erste Zustandsgleichung bzw. das erste LMN wird vom LOLIMOT-Algorithmus in zwei Teilmodelle aufgeteilt: $n_{m,1} = 2$. Die zweite Zustandsgleichung bzw. das zweite LMN besteht nur aus einem Teilmodell, $n_{m,2} = 1$. Die Ausgangsgleichung (drittes LMN) wird mit zwei Teilmodellen abgebildet, $n_o = 2$. Obwohl in der LMSSN-Toolbox alle Einflussgrößen als Kandidaten zur weiteren Aufteilung deklariert werden, hat der Algorithmus für das erste und das dritte LMN nur bezüglich der ersten beiden Zustände aufgeteilt. Das resultierende LMSSN beinhaltet insgesamt 45 Parameter.

2.4 NARX-Netze

NARX-Netze (NXN) basieren auf Gleichung 2. Die nichtlineare Funktion $f(\cdot)$ wird dabei mit Hilfe eines künstlichen neuronalen Netzes (KNN) angenähert. Bei den Neuronen der Zwischenschicht des KNN findet eine Zündung mit der $f_{\text{tanh}}(\cdot)$ - Aktivierungsfunktion statt. Die Parametermatrix $\underline{\theta}_{\text{NXN}}$ gewichtet die Regressoren $\underline{x}$. $\underline{W}_{\text{out}}$ sorgt für eine weitere Gewichtung und stellt die eigentlichen Gewichte des KNN dar. Zusätlich dazu wird ein Offsetvektor $\underline{\beta}_{\text{out}}$ addiert. Dadurch ergibt sich für die erste Zwischenschicht des NARX-Netzes folgende Gleichung:

$$\hat{y}_{\text{NXN}}(k) = \underline{W}_{\text{out}} f_{\text{tanh}} \left(\underline{\theta}_{\text{NXN}} \underline{x} + \underline{\beta}_{\text{out}}\right) \; . \tag{11}$$

Im Verlauf des Trainingsprozesses werden neben der Anzahl der Regressoren, beispielsweise auch die Anzahl der Neuronen variiert. Das daraus folgende Modell hat eine Zwischenschicht mit 10 Neuronen und die Regressoranzahl sowohl für die Eingänge als auch den Ausgang liegt bei 24. Bei einer Abtastzeit von 10 Minuten entspricht dies einem Zeitraum von 4 Stunden. Für das

Training wird der Levenberg-Marquardt-Algorithmus genutzt. Bei genannter Konfiguration liegt eine Parameteranzahl von $n_{\text{par}} = 1701$ vor. Für den späteren Vergleich liegt das Modell im Gegensatz zum Trainingsprozess nicht mehr in der vorgestellten NARX-Modellstruktur vor. Dies liegt daran, dass für die Simulation der Modellausgang zurückgekoppelt werden muss. Dadurch ergibt sich eine OE-Struktur. Die Implementierung und das Training der NARX-Netze findet mit der Deep Learning Toolbox von MATLAB statt.

2.5 Neural State Space Models

Die grundlegende Formulierung von Neural State Space Models (NSSM) ist in den Gleichungen (12) und (13) zu erkennen. Hierbei sind f_{nss} und g_{nss} KNNs und $x(k)$ ist der Zustand des Systems zum Zeitpunkt k. Die Eingangsgrößen des Systems zum Zeitpunkt k werden durch $u(k)$ repräsentiert. Bei p handelt es sich um die Gewichte des KNN, die im Laufe des Trainingsprozesses identifiziert werden müssen.

$$\underline{x}(k+1) = f_{\text{nss}}\left(\underline{x}(k), \underline{u}(k), \underline{p}\right) \tag{12}$$

$$y(k) = g_{\text{nss}}\left(\underline{x}(k), \underline{p}\right) \tag{13}$$

Bei einer beispielhaften Betrachtung der ersten Zwischenschicht eines NSSM ergeben sich die Gleichungen 14 und 15. Dabei stellen $\underline{V}_{\text{A}}$, $\underline{V}_{\text{B}}$ und $\underline{V}_{\text{C}}$ die Gewichtung der Zustände und Eingänge dar. Zusätzlich dazu und ähnlich wie in Gleichung 11 gibt es die zusätzlichen Gewichtungen $\underline{W}_{\text{AB}}$ und $\underline{W}_{\text{C}}$. Weitere Parameter, die während des Trainingsprozesses identifiziert werden müssen, sind die zusätzlichen Offsetvektoren β_{AB} sowie β_{C} [7].

$$\hat{x}(k+1) = \underline{W}_{\text{AB}} f_{\text{tanh}}\left(\underline{V}_{\text{A}}\underline{\hat{x}}(k) + \underline{V}_{\text{B}}\underline{u}(k) + \underline{\beta}_{\text{AB}}\right) \tag{14}$$

$$\hat{y}_{\text{NSSM}}(k) = \underline{W}_{\text{C}} f_{\text{tanh}}\left(\underline{V}_{\text{C}}\underline{\hat{x}}(k) + \underline{\beta}_{\text{C}}\right) \tag{15}$$

Im Rahmen einer Hyperparameterstudie des NSSM werden beispielsweise die Anzahl der Zwischenschichten, die Anzahl der Neuronen pro Schicht oder auch die Größe der Batches variiert. Auch die Anzahl der Zustände wird im

Laufe der Studie mitberücksichtigt. Das aus der Hyperparameterstudie resultierende Modell erster Ordnung besitzt 2 Zwischenschichten mit je 16 Neuronen und wird während des Trainings mit einer Batchgröße von 1024 Datenpunkten gespeist. Für das Training wird der Adam-Optimierer genutzt. Das resultierende Modell weist zudem eine Parameteranzahl $n_{\text{par}} = 417$ auf, was den Gewichten im Zustandsnetzwerk entspricht. Auf das Training eines Ausgangsnetzwerkes wird an dieser Stelle verzichtet, wodurch g_{nss} der Einheitsmatrix I entspricht. Die Umsetzung und Implementierung der NSSM erfolgt mit Hilfe der Deep Learning Toolbox und der System Identification Toolbox von MATLAB.

3 Vergleich der Ergebnisse

Die vorgestellten Modelle werden auf den Testdaten des HRMS evaluiert und anschließend miteinander verglichen. Diese erstrecken sich über einen Zeitraum von 9 Tagen im Dezember 2022. Für den Vergleich der Modelle werden unterschiedliche Fehler- und Gütemaße herangezogen. Neben dem root mean squared error (RMSE) und dem sum of squared errors (SSE) wird auch der variance accounting for (VAF) für die Beurteilung des Modells genutzt. Bei dem RMSE und dem SSE handelt es sich um Fehlermaße. Im Gegensatz dazu ist der VAF ein Gütemaß. Mit dem RMSE und dem SSE kann grundlegend die Abweichung zwischen Messdaten und Modellausgang evaluiert werden. Mit dem VAF kann untersucht werden, mit welcher Genauigkeit das Modell vor Allem die Dynamik der Messdaten abbildet, weil Offsetfehler keinerlei Einfluss auf das VAF-Gütemaß haben. Für eine detaillierte Erläuterung der Fehler- und Gütemaße wird auf [8] verwiesen. In Tabelle 2 sind die Fehler- und Gütemaße für alle Modelle aufgelistet. Außerdem zeigt Tabelle 2 die Anzahl der Modellparameter zur Beurteilung der Modellkomplexität. Es ist ersichtlich, dass sowohl der RMSE als auch der SSE für das NXN-Modell, LMSSN-Modell und TSNX-Modell nahe beieinander liegen. Im Gegensatz dazu kann das NSSM-Modell den geringsten RMSE und SSE aufweisen. Das TSNX-Modell besitzt den höchsten VAF und bildet die Dynamik der Daten mit hoher Genauigkeit ab. Eine graphische Gegenüberstellung der gemessenen Daten und der Ausgänge der unterschiedlichen Modellstrukturen auf den Testdaten ist in

Tabelle 2: Übersicht der Fehler- und Gütemaße sowie die Anzahl der Parameter aller Modelle

	RMSE [%rH]	SSE $\left[\%\mathrm{rH}^2\right]$	VAF [%]	n_{par}
NSSM	0.68	585.5	87.07	417
NXN	1.05	1370.1	82.57	1701
LMSSN	1.02	1318.8	83.21	45
TSNX	1.06	1413.1	94.65	8080

Bild 1 zu finden. Hier sind neben den Modellausgängen und den dazugehörigen Fehlerzeitreihen, auch die skalierten Eingänge des Modells sowie jeweils ein Geigenplot aller Modelle zur Beurteilung der Fehlerverteilung erkennbar. Der im Geigenplot mit einer etwas dunkleren Farbgebung dargestellte Bereich repräsentiert den Interquartilsabstand, welcher 50 % aller Daten beinhaltet. Grundsätzlich weisen alle Modelle gute Approximationseigenschaften auf und können die Messdaten unter Berücksichtigung der Dynamik annähern. Bei Betrachtung des zeitlichen Verlaufs des Fehlers oder auch des Geigenplots fällt auf, dass sich der Fehler ungefähr im Bereich $\pm$ 2 %rH aufhält, was den Toleranzen typischer Feuchtesensoren entspricht. Das NSSM-Modell weist die geringste Abweichung auf. Dies ist nicht nur im Fehlerverlauf erkennbar, sondern auch im Geigenplot des NSSM-Modell. Hier liegt der Median (weißer Punkt im Geigenplot) der Fehlerverteilung bei annähernd null. Bei Betrachtung des NXN-Modells und LMSSN-Modells ist eine Verschiebung der Fehlerverteilung in den positiven Wertebereich ersichtlich. Dies bedeutet, dass die Modelle die Testdaten zu einem Großteil der Zeit unterschätzen. Das TSNX-Modell bringt eine noch größere Abweichung mit sich, ist jedoch mit der kleinsten Varianz gekennzeichnet. Das TSNX-Modell weist zudem die höchste Modellkomplexität mit $n_{\mathrm{par}} = 8080$ Parametern auf. Dazwischen liegen das NXN-Modell mit 1701 Parametern und das NSSM-Modell mit einer Parameteranzahl von 417. Die geringste Modellkomplexität besitzt das LMSSN-Modell mit 45 zu schätzenden Parametern.

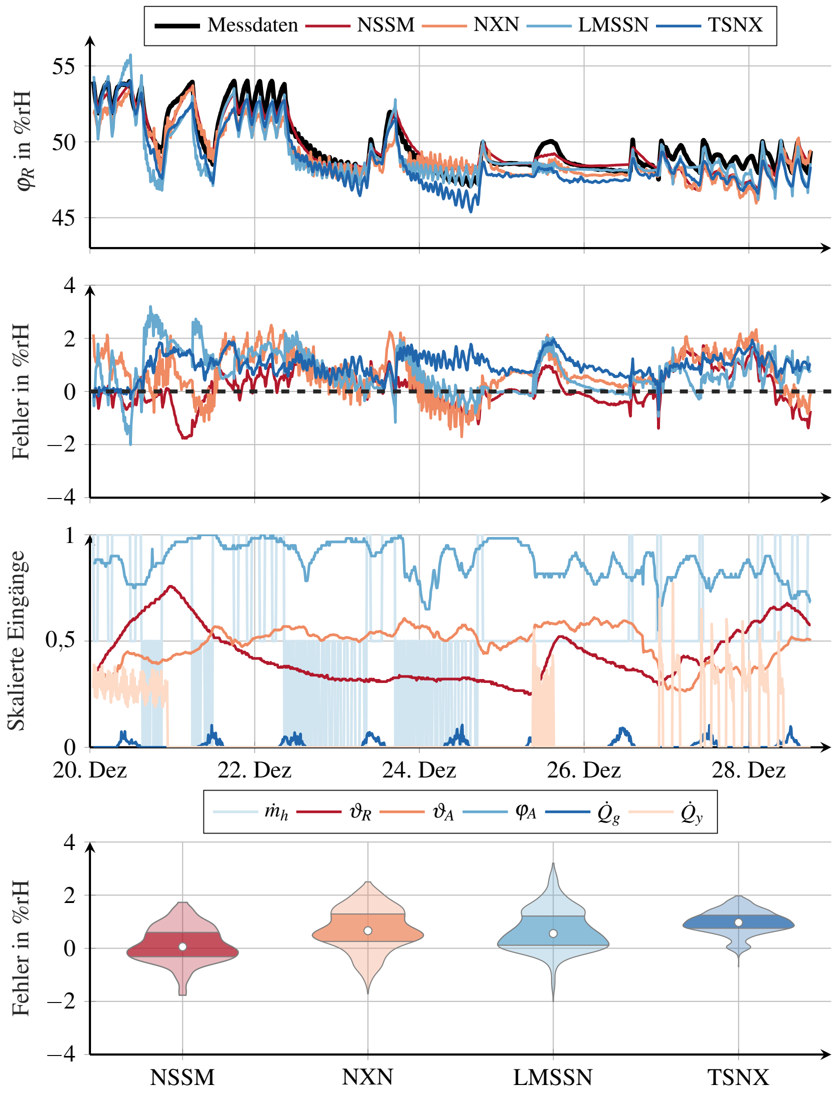


Bild 1: Überblick über die gemessenen Feuchtedaten, die Modellausgänge, die Fehlerzeitreihen, die skalierten Eingangsgrößen der Modellstrukturen, wobei für $\dot{m}_h$ ein Wert > 0.5 eine Befeuchtung und ein Wert kleiner < 0.5 eine Entfeuchtung darstellt, sowie einen Geigenplot zur Beurteilung der Fehlerverteilung zwischen gemessenen Daten und Modell

4 Diskussion und Ausblick

Die Erhebung von aussagekräftigen Messdaten gestaltet sich aufgrund der Notwendigkeit der Einhaltung der konservatorischen Anforderungen an die Kulturgüter schwierig: schnelle Änderungen der Raumklimaparameter sind zu vermeiden, sodass die Anwendung von Testsignalen im Prozess mit äußerster Vorsicht und unter Berücksichtigung weiterer Kriterien, wie z. B. den Außenbedingungen, vorzunehmen sind. Unter Umständen können so wichtige Arbeitspunkte bzw. Arbeitsbereiche des Prozesses nur unzureichend abgedeckt werden. Hinzu kommt, dass ein Teil der Messdaten im geschlossenen Regelkreis aufgezeichnet worden sind, wodurch eine Korrelation der Messgrößen zustande kommt, welche sich negativ auf eine erfolgreiche Systemidentifikation auswirken können. Der interessierte Leser wird auf die Identifikationsbedingungen von [9] verwiesen. Im Laufe des Beitrags wird deutlich, dass die untersuchten Modellstrukturen trotz der genannten Nachteile im Messdatensatz grundsätzlich gute Approximationsgüten aufweisen. Jedes Modell hat unterschiedliche Vorteile und Nachteile, wobei drei von vier Modellen einen Offsetfehler aufzeigen. Dieser Fehler resultiert unter Umständen aufgrund der Korrelation zwischen Eingangs- und Ausgangsgrößen, wodurch die verschiedenen Schätzmethoden keine konsistenten Ergebnisse liefern. Das TSNX-Modell besticht mit einer sehr geringen Varianz in der Fehlerverteilung, bringt aber mit 8080 Parametern die größte Modellkomplexität mit sich. Im Gegensatz dazu, hat das LMSSN-Modell die geringste Modellkomplexität aller Modelle, zeigt jedoch die ausgeprägteste Fehlerstreuung unter den untersuchten Modellen. Ein weiterer Nachteil bei dem LMSSN-Modell ist die lange Trainingsdauer (48 Stunden) im Vergleich zu den anderen Modellen. Mit Blick auf die Nutzung als Simulationsmodell für die Evaluierung modellprädiktiver Regelungsansätze zeigt das NSSM-Modell Potential, da es nicht nur die geringste Abweichung zu den Messdaten zeigt, sondern auch in Kombination mit der Model Predictive Control Toolbox von MATLAB ohne große Portierungsmaßnahmen genutzt werden kann. In künftigen Arbeiten stehen weitere datengetriebene Methoden, wie z. B. physikgeführte neuronale Netze im Fokus.
Wir möchten dem Team von Prof. Dr.-Ing. Oliver Nelles für die Bereitstellung der LMSSN-Toolbox sehr herzlich danken.

Literatur

[1] A. Burmester. „Die Beteiligung des Nutzers bei Museumsneubau und -sanierung: Oder welche Klimawerte sind die Richtigen?“ In: *Raumklima in Museen und historischen Gebäuden*. S. 9–24. 2000

[2] D. Crawley, L. Lawrie, O. Pederseb und F.C. Winkelmann. „EnergyPlus: Energy simulation program“. *Ashrae Journal*. 42, S. 49–56. 2000

[3] S.A. Klein. „TRNSYS-A Transient System Simulation Program“. *University of Wisconsin-Madison*. 1988

[4] G. Serale, M. Fiorentini, A. Capozzoli, D. Bernardini und A. Bemporad. „Model Predictive Control (MPC) for Enhancing Building and HVAC System Energy Efficiency: Problem Formulation, Applications and Oppurtunities“. *Energies*. 11. 2018

[5] B. Hartmann, T. Ebert, T. Fischer, T.Belz, J. Kampmann und O. Nelles. „LMNtool – Toolbox zum automatischen Trainieren lokaler Modellnetze“. *22. Workshop Computational Intelligence, KIT Scientific Publishing*. 45, S. 341–355. 2012.

[6] M. Schüssler. „Machine Learning with nonlinear state space models“. Dissertation. *Universität Siegen. Institut für Mechanik und Regelungstechnik – Mechatronik* 2022

[7] J. A. K. Suykens, B. L. R. De Moor und J. Vandewalle „Nonlinear system identification using neural state space models, applicable to robust control design“. *International Journal of Control*. Vol. 62., S. 129–152. 1995

[8] A. Kroll und H. Schulte. „Benchmark problems for nonlinear system identification and control using Soft Computing methods: Need and Overview“. *Applied Soft Computing*. 22, S. 496–513. 2014

[9] R. Isermann und M. Münchhof. „Parameter Estimation in Closed Loop“. In: *Identification of Dynamic Systems: An Introduction with Applications Springer*. 2011

Ein Jahr auf ländlichen Straßen: Aufnahme eines multimodalen Datensatzes für das automatisierte Fahren

Franz Albers, Torsten Bertram

Lehrstuhl für Regelungssystemtechnik
Technische Universität Dortmund
Otto-Hahn-Straße 8, 44227 Dortmund
E-Mail: franz.albers@tu-dortmund.de

1 Einführung

In den vergangenen Jahren wurden vor allem durch die Entwicklung leistungsfähiger datengetriebener Methoden erhebliche Fortschritte im Bereich des automatisierten Fahrens erzielt. Ein zentraler Erfolgsfaktor ist die Verfügbarkeit von großen Datensätzen, die das Training und das Testen von datenbasierten Modellen in realistischen Szenarien ermöglichen. Aktuell verfügbare Datensätze fokussieren sich dabei überwiegend auf Autobahnen und urbane Gebiete mit einer gut ausgebauten Infrastruktur. Ländliche Regionen wurden bisher hingegen nur wenig berücksichtigt, obwohl sie einige besondere Herausforderungen aufweisen: Spärlich ausgebaute Infrastruktur, teilweise fehlende bzw. schlecht sichtbare Fahrbahnmarkierungen, ungeräumte Fahrbahnen bei Schnee, morgendlicher Nebel, Wildwechsel oder wechselnde Lichtverhältnisse bei Fahrten durch Waldgebiete stellen unter anderem aufgrund der Domain-Gaps zu bestehenden Datensätzen mit definierten Operational Design Domains (ODD) besondere Herausforderungen an die Sensoren und Modelle.

Daten aus der ländlichen Domäne sind für die Entwicklung und Evaluierung von domänenübergreifenden lernbasierten Algorithmen von großer Bedeutung. Sun et al. konstatieren bei der Evaluierung von Objektdetektionsalgorithmen auf dem Waymo Open Dataset [1] einen deutlichen Leistungsabfall, falls die Modelle auf einem Datensatz aus einer städtischen bzw. einer vorstädtischen Region trainiert werden und auf dem jeweils anderen Datensatz validiert werden [1]. Lang et al untersuchen dabei die 3D Detektionen von Fußgängern und Fahrzeugen mit dem lidar-basierten Pointpillars Modell [2]. Diese Erkenntnis unterstreicht die Bedeutung domänenübergreifender Datensätze und insbesondere den Bedarf an aufgezeichneten Fahrten im ländlichen Raum.

Der vorliegende Beitrag stellt die Vorgehensweise und Herausforderungen bei der Aufnahme des neuen DEMANDAR Datensatzes, den Aufbau des verwendeten Messfahrzeugs (siehe Bild 1), beispielhafte Daten sowie die Maßnahmen zur Annotierung der Daten vor. Um verschiedenste Witterungsverhältnisse abzubilden, werden über einen Zeitraum von einem kompletten Jahr Daten auf einer definierten Messstrecke in der Region Südwestfalen erhoben. Der Datensatz bildet somit die gleiche Strecke zu verschiedenen Tages- und Jahreszeiten bei unterschiedlichen Witterungsverhältnissen ab.

Nach bestem Wissen der Autoren ist bisher kein multimodaler und multisaisonaler Datensatz öffentlich verfügbar, der umfangreiche Fahrten aus ländlichen Regionen enthält. Des Weiteren wurden bisher nur wenige Datensätze publiziert, die neben Lidar- und Kameradaten auch Daten von Automotive-Radarsensoren enthalten. Ein weiteres Alleinstellungsmerkmal des Datensatzes ist die zentimetergenaue Referenzlokalisierung durch ein kinematisches Echtzeit-Inertial-Navigationssystem (RTK-DGNSS/INS).

2 Stand der Technik

Es gibt bereits einige Datensätze für verschiedene Aufgaben im Bereich des automatisiertes Fahren, wie zum Beispiel die Objektdetektion oder Lokalisierung. Oftmals umfassen Datensätze eigene Benchmarks zum Vergleich verschiedener Ansätze. In diesem Abschnitt werden einige in der Forschung zum automatisierten Fahren häufig genutzte Datensätze vorgestellt.

Der KITTI-Datensatz [3] war einer der ersten öffentlich verfügbaren Datensätze. Es werden Lidar-, Kamera- und RTK-GPS/IMU-Daten synchronisiert bereitgestellt. Der Datensatz enthält allerdings keine Radardaten. Der KITTI-Datensatz enthält verschiedene Szenarien, die auf zumeist städtischen Straßen in Karlsruhe aufgenommen wurden. Im Laufe der Zeit wurde der KITTI-Datensatz mit umfassenden Annotationen und Benchmarks für verschiedene Anwendungen erweitert (z.B. Straßenerkennung [4] oder semantische Punktwolkenannotationen [5]).

Das Waymo Open Dataset [1] wurde mit einer Flotte selbstfahrender Fahrzeuge aufgezeichnet, die jeweils mit mehreren Lidarsensoren, Kameras und einem RTK-GPS/IMU-System ausgestattet waren. Radarsensoren wurden ebenfalls nicht zur Umfelddetektion verwendet. Der Datensatz wurde in verschiedenen städtischen und vorstädtischen Regionen in San Francisco, Phoenix und Mountain View, USA sowohl bei Tag als auch bei Nacht aufgezeichnet. Damit ist der Waymo-Datensatz einer der umfangreichsten, öffentlich verfügbaren Datensätze für das automatisierte Fahren.

Der nuScenes-Datensatz [6] stellt als einer der ersten großen Datensätze für automatisiertes Fahren neben Lidar-, Kamera- und GNSS/IMU-Daten auch die Daten von Automotive-Radarsensoren bereit. nuScenes liefert hochauflösende Daten für über 1.000 Szenarien. Unterschiedliche Wetter- und Lichtbedingungen sind ebenfalls enthalten. Der Datensatz deckt ca. 1.000 km von hauptsächlich städtischen Straßennetzen in Boston und Singapur ab. Der nuScenes-Datensatz wurde später um punktuelle semantische Lidar-Annotationen und einen panoptischen Benchmark erweitert [7].

Im Oxford RobotCar-Datensatz [8] sind Lidar- sowie GPS/IMU-Daten und Stereokamerabilder enthalten. Die Sensorik wurde später um einen rotierenden 360°-NavTech-Radar ergänzt [10]. Die RobotCar-Datensätze wurden durch wiederholte Fahrten auf der gleichen städtischen Route in Oxford, Großbritannien aufgezeichnet. Zu unterschiedlichen Tageszeiten wurden die Messfahrten für den ursprünglichen Datensatz über ein Jahr und für die Radarerweiterung über einen Monat durchgeführt. Nach bestem Wissen der Autoren sind der nuScenes- und der Oxford Radar RobotCar-Datensatz die bisher einzigen veröffentlichten Datensätze, die das komplette 360°-Sichtfeld um das Fahrzeug mit Lidar-, Radar- und Kameramodalitäten abdecken.

Der Boreas-Datensatz [9] wurde ebenfalls durch das wiederholte Befahren einer städtischen Route über ein ganzes Jahr hinweg in Toronto, Kanada aufgezeichnet. Der Datensatz umfasst verschiedene herausfordernde Wetter- und Lichtbedingungen (z. B. Regen, Schnee, Nacht usw.). Die Aufnahmeplattform verfügt über einen rotierenden 360°-Lidar auf dem Dach, eine nach vorne gerichtete Kamera und einen GNSS/IMU-Sensor, der eine Lokalisierungsgenauigkeit von bis zu 2-4 cm erreicht. Ähnlich wie beim Oxford Radar RobotCar-Datensatz [10] wird auch hier ein rotierender 360° NavTech Radar verwendet. Die Doppler-Messungen des Radars sind als relevante Zustandsgröße jedoch nicht enthalten. Somit wird in beiden Datensätzen einer der wesentlichen Vorteile von Radar-Sensoren gegenüber Lidar-Sensoren nicht genutzt [11].

Der View-of-Delft-Datensatz [12] umfasst Daten von einem rotierenden 360°-Lidar, einer Stereokamera und einem RTK-GPS/IMU Navigationssystem. Zusätzlich wurde ein nach vorne gerichteter 3+1D Automotive Radarsensor verwendet. Neben Entfernungs-, Azimuth- und Doppler-Messungen liefert dieser Sensor auch eine Messung in der Elevation. Die Daten wurden auf den städtischen Straßen von Delft in den Niederlanden aufgezeichnet.

Tabelle 1 zeigt einen kurzen Überblick über die erwähnten Datensätze.

Tabelle 1: Überblick der Datensätze im Bereich des automatisierten Fahrens

Datensatz	**Kamera**	**Lidar**	**Radar**	**GNSS**	**Ort**
KITTI [3]	✓	✓	✗	✓	Karlsruhe
Waymo [1]	✓	✓	✗	✓	Phoenix, San Francisco
nuScenes [6]	✓	✓	✓	✓	Boston, Singapur
Radar RobotCar [10]	✓	✓	✓	✓	Oxford
Boreas [9]	✓	✓	✓	✓	Toronto
View-of-Delft [12]	✓	✓	✓	✓	Delft
Demandar	✓	✓	✓	✓	Südwestfalen

3 Messfahrzeug

Als Messfahrzeug diente ein Nissan Leaf ZE0 mit einer umfangreichen Sensorik (siehe Bild 1). Die folgenden Abschnitten beschreiben den technischen

Aufbau sowie die verwendeten Koordinatensysteme und die Vorgehensweise bei der Kalibrierung der Sensoren.

Bild 1: TU Dortmund Forschungsfahrzeug

3.1 Technischer Aufbau des Messfahrzeugs

Das Messfahrzeug verfügt über einen Mid-Range- (Ouster OS1), einen Long-Range-Lidar (Ouster OS2) und sechs RGB-Kameras (FLIR Chameleon 3), die auf einem Dachgepäckträger montiert sind. Hinter der Windschutzscheibe befindet sich eine Serien-Frontkamera von Mobileye. An den Ecken des Fahrzeugs sind vier prototypische 77 GHz Mid-Range Automotive Radar-Sensoren hinter den Stoßfängern verbaut. Ein RTK-DGNSS/INS (GeneSys ADMA-G Eco+) ermöglicht eine zentimetergenaue Referenzlokalisierung.

Bild 2 zeigt einen Überblick über den technischen Aufbau des Versuchsfahrzeugs. Die Daten der verschiedenen Sensoren werden auf einem Zentralrechner mit AMD Ryzen 7 7700X Prozessor, 64 GB RAM, 8 TB SSD Speicher, einer Nvidia GTX 1050 Ti GPU und einem 24 V Netzteil zusammengeführt. Der Rechner verfügt über zwei PCI CAN FD Interface Karten von PEAK Systems, über die mithilfe der SocketCAN Treiber mit verschiedenen CAN Geräten kommuniziert werden kann. Über diese Schnittstelle werden unter anderem die Daten vom fahrzeugeigenen CAN-Bus empfangen, dekodiert und geloggt. Einige Signale vom Fahrzeug-CAN-Bus, wie beispielsweise die Fahrzeugeschwindigkeit, dienen wiederum als Eingangssignale für die Radarsensoren und für die Mobileye Frontkamera. Die weiteren sechs Kameras sind über USB 3.1 mit dem Zentralrechner verbunden.

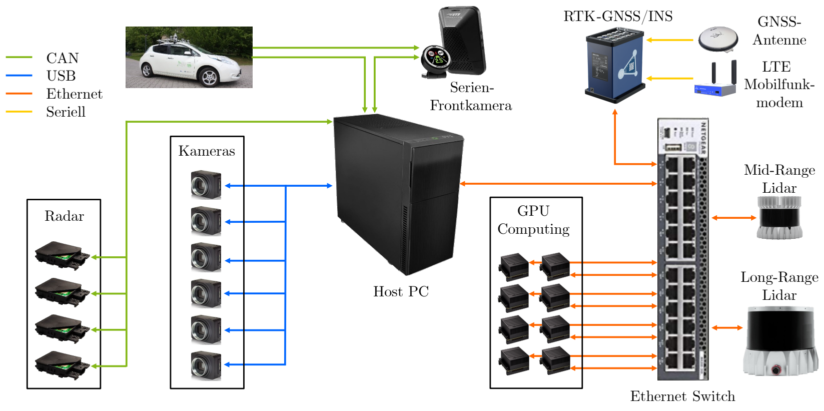


Bild 2: Technischer Aufbau des Messfahrzeugs

Die beiden Lidar-Sensoren und das RTK-DGNSS/INS sind über einen 10 Gigabit Ethernet Switch mit dem Zentralrechner verbunden. Neben den Sensoren sind zudem acht Nvidia Jetson AGX Xavier Developer Kits für besonders rechenintensive Anwendungen im Kofferraum des Forschungsfahrzeugs verbaut.

Beide Lidare verfügen über eine Auflösung von 1.024x64 Punkten bei 20 Hz. Der Ouster OS2 Long-Range Lidar hat eine Reichweite von bis zu 210 m bei einem Sichtfeld von 360°×22.5°. Pro Sekunde werden somit 1.310.720 Punkte gemessen. Der Ouster OS1 Mid-Range Lidar hat hingegen eine etwas geringere Reichweite von bis zu 170 m bei einem etwas größeren Öffnungswinkel von 360°×45°. Der Mid-Range Lidar verfügt zudem über eine Dual Return Funktionalität, bei dem nicht nur der erste, sondern auch der zweite Reflex eines Laserpulses gemessen wird. Dadurch können auch Objekte erfasst werden, die sich hinter anderen Objekten (z.B. Glasscheiben oder Nebel) befinden. Der Mid-Range Lidar kann somit 2.621.440 Messpunkte pro Sekunde erfassen.

Neben einer Punktwolke liefert der Mid-Range Lidar zudem eine Intensitätsmessung, welche die Reflektivität der Objekte erfasst. Beide Lidare verfügen zudem über eine integrierte IMU mit 6 Freiheitsgraden.

Bei den Radarsensoren handelt es sich um prototypische Frequency Modulated Continuous Wave (FMCW) Automotive Sensoren. Die Sensoren verwenden ein Frequenzspektrum im 77 GHz Band und besitzen ebenfalls eine Aufnahmerate von 20 Hz. Aufgezeichnet werden sowohl die von den Sensoren detektierten Objekte als auch die Radar-Targets.

Die FLIR Chameleon3 RGB Kameras verwenden einen Global Shutter, um unerwünschte Bildartefakte bei bewegten Objekten zu vermeiden. Die Auflösung der Kameras beträgt 2.048$\times$1.152 Pixel (ca. 3,2 Megapixel) bei 20 Hz. Um die Datenrate möglichst gering zu halten, werden Kamerabilder nicht als Rohbilder sondern als komprimierte Einzelbilder aufgezeichnet. Die Mobileye Frontkamera verfügt über eine integrierte Objekterkennung sowie -klassifizierung und liefert statt einem Kamerabild direkt eine Objektliste mit detektierten Hindernissen und Verkehrszeichen.

Das RTK-DGNSS/INS besteht aus einer GNSS Antenne auf dem Dach des Fahrzeugs, einem LTE Mobilfunkmodem, über das NTRIP Korrekturdaten von Bodenstationen empfangen werden können, und der Automotive Dynamic Motion Analyzer (ADMA) Einheit, die in einer IMU mit 9 Freiheitsgraden hochgenaue Beschleunigungssensoren und Gyroskope (Faserkreisel) enthält. Die Genauigkeit der Positionsbestimmung liegt bei 0,01 m, die Genauigkeit des erfassten Roll- und Nick-Winkels liegt bei 0,01° und des Gier-Winkels bei 0,025°. Im Falle einer Verschlechterung oder eines Ausfalls des GNSS Empfangs, beispielsweise in Tunneln oder in bewaldeten Gebieten, wird die Position durch diese Sensoren für einen gewissen Zeitraum sehr genau geschätzt und bereitgestellt.

Alle umfelderfassenden Sensormodalitäten (Lidar, Radar und Kamera) decken die komplette 360° Rundumsicht ab. Durch die vergleichsweise hohen Aufnahmeraten und die Vielzahl von Sensoren ergibt sich eine entsprechend hohe Datenrate von insgesamt ca. 350 MB/Sekunde. Die Aufnahme einer einminütigen Fahrt erzeugt somit bereits eine Datenmenge von 20 GB.

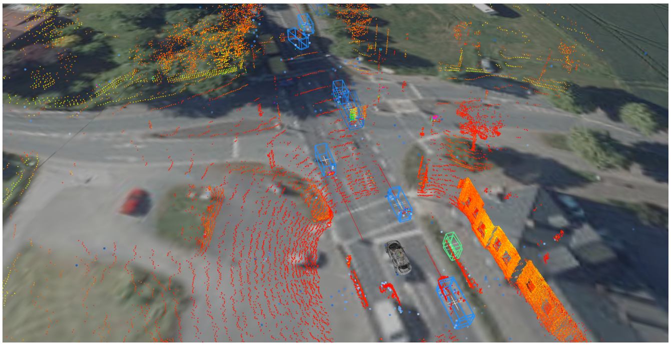


Bild 3: Beispielhafte Darstellung der Sensordaten.

Beispielhaft dargestellt sind die Sensordaten in Bild 3. Die Lidar-Punktwolke ist rötlich dargestellt. Die Radar-Targets sind als bläuliche Punktwolken abgebildet. Die dargestellten Bounding Boxen veranschaulichen die durch die Radarsensoren detektierten Objekte, die dazugehörigen Geschwindigkeitsvektoren (rote Linie an der Spitze der Bounding Boxen) und die Klassifizierung der Objekte (z.B. entsprechen blaue Bounding Boxen Fahrzeugen, grüne Boxen Fahrrädern). Das statische Orthofoto im Hintergrund zeigt die zentimetergenaue Lokalisierung des Ego-Fahrzeugs auf der unteren Linksabbiegerspur.

Als grundlegendes Framework wird im Forschungsfahrzeug das Robot Operating System 2 (ROS 2) [13] verwendet. ROS 2 ist ein Open-Source Publisher/Subscriber Framework, das ursprünglich aus der Robotik stammt. Unter anderem auch aufgrund seiner Modularität wird ROS 2 inzwischen in vielen Forschungsfahrzeugen eingesetzt. Ein großer Vorteil ist zudem die weite Verbreitung von ROS 2, wodurch auf eine Vielzahl von bereits verfügbaren ROS 2 Paketen zur Datenverarbeitung zurückgegriffen werden kann. Basierend auf ROS 2 wurde Autoware [14] entwickelt. Autoware ist ein Open-Source Software Stack für das automatisierte Fahren. Teile von Autoware, wie beispielsweise die Komponenten Sensing, Perception und Localization, werden im hier vorgestellten Forschungsfahrzeug verwendet.

Die Spannungsversorgung der Messtechnik ist komplett vom Bordnetz getrennt und erfolgt über zwei 12 Volt Batterien mit 120 Ah.

3.2 Koordinatensysteme

Dieser Abschnitt beschreibt die Koordinatensysteme des Messfahrzeugs. Alle Koordinatensysteme sind rechtshändig. Für jeden Sensor ist mindestens ein eigenes Sensorkoordinatensystem definiert. Zudem existieren mehrere fahrzeugfeste Koordinatensysteme, bei denen die x-Richtung jeweils in Fahrtrichtung nach vorne zeigt: *base_link* (Mitte der Hinterachse auf den Boden projiziert), *front_axle_center_link* (Mitte der Vorderachse), *rear_axle_center_link* (Mitte der Hinterachse) sowie *front_bumper* (Vorderster mittiger Punkt des Fahrzeugs auf den Boden projiziert).

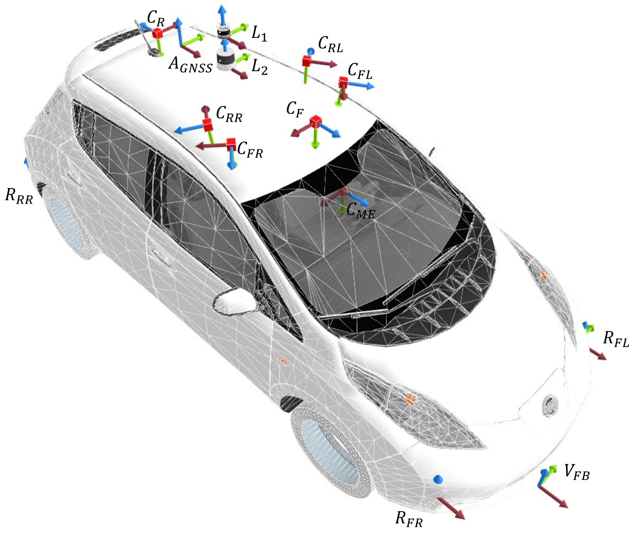


Bild 4: Koordinatensysteme des Messfahrzeugs. Die x-Richtung ist jeweils rot, die y-Richtung grün und die z-Richtung blau dargestellt.

Bild 4 zeigt die verschiedenen Sensorkoordinatensysteme des Messfahrzeugs. Von den fahrzeugeigenen Koordinatensystemen ist in der Abbildung lediglich das *front_bumper* Koordinatensystem (V_{FB}) dargestellt, da die anderen Koordinatensysteme innerhalb des Fahrzeugs liegen.

Die Koordinatensysteme der beiden Lidar-Sensoren werden in der Abbildung durch die Lidar-Modelle markiert. Die beiden Koordinatensysteme *lidar_os1_*-

base_link (in Bild 4: L_1) und *lidar_os2_base_link* (L_2) liegen gemäß dem Sensoraufbau in der Mitte des Fahrzeugdachs übereinander. Für beide Lidare sind weitere Sub-Koordinatensysteme für die jeweilige IMU sowie für den Laser definiert.

Die Einbaupositionen der Kameras sind in Bild 4 durch die roten Boxen und die Koordinatensysteme C markiert. Dargestellt sind jeweils die Bildkoordinatensysteme (x-Achse zeigt nach rechts, y-Achse nach unten, z-Achse ins Bild). Die Mobileye Frontkamera C_{ME} ist hinter der Windschutzscheibe verbaut.

Die Koordinatensysteme R der Radarsensoren liegen an den vier Ecken des Fahrzeugs. Die x-Achse zeigt dabei nicht in die Einbaurichtung, die jeweils +/-45° zur Fahrtrichtung rotiert ist, sondern ist entsprechend der fahrzeugeigenen Koordinatensysteme in Fahrtrichtung nach vorne ausgerichtet. Die Berücksichtung der Einbauwinkel erfolgt intern in den Radarsensoren.

Zwischen den Koordinatensystemen der Lidare und der Rückkamera liegt ein weiteres Koordinatensystem A_{GNSS} für die GNSS-Antenne. Das Koordinatensystem des RTK-DGNSS/INS ist nicht im Bild verzeichnet, da der ADMA im Kofferraum des Fahrzeugs installiert ist und es somit ebenfalls dort liegt.

3.3 Kalibrierung

Die intrinsische Kalibrierung der Kameras erfolgte mit dem ROS 2 Paket *intrinsic_camera_calibrator* aus dem Tier4 CalibrationTools Repository[1], das intern die Funktionalitäten der OpenCV Bibliothek [15] zur Kalibrierung [16] verwendet. Als Kalibrierungsmuster wurde ein Schachbrettmuster mit bekannten Abmessungen verwendet.

Die extrinsische Kalibrierung zwischen den einzelnen Kameras und einem Lidar wurde mit den Paketen *extrinsic_interactive_calibrator* und *extrinsic_-tag_based_calibrator* aus demselben Repository durchgeführt. Die genaue Position des Lidars relativ zum *base_link* Koordinatensystem des Fahrzeugs sowie die Position der GNSS-Antenne relativ zum RTK-DGNSS/INS relativ bzw.

[1] Tier4 CalibrationTools GitHub Repository
`https://github.com/tier4/CalibrationTools`

zum *base_link* Koordinatensystem wurden manuell vermessen. Die Transformationen zwischen den verschiedenen fahrzeugfesten Koordinatensystemen (Mitte der Vorderachse, Mitte der Hinterachse, Mitte des vorderen Stoßfängers, *base_link*) wurden aus technischen Datenblättern bzw. einem CAD-Modell des Nissan Leaf ZE0 abgeleitet.

Die intrinsische und extrinsische Kalibrierung der vier Radarsensoren in Bezug auf die Mitte der Vorderachse des Fahrzeugs und die Kalibrierung der Mobileye Frontkamera in Bezug auf den auf den Boden projizierten vordersten Punkt des Fahrzeugs (*front_bumper*) wurden von dem jeweiligen Hersteller bzw. Lieferanten durchgeführt.

4 Datensatz

Seit August 2022 fährt das Messfahrzeug ca. einmal wöchentlich eine definierte Messstrecke in der Region Südwestfalen zur Datenaufnahme ab. Die Daten werden aktuell aufbereitet und annotiert. Geplant ist die Veröffentlichung des Datensatzes im Laufe des Jahres 2024. In den folgenden Abschnitten werden die Messtrecke und die für den Datensatz geplanten Annotierungen vorgestellt.

4.1 Messstrecke

Die Messstrecke umfasst mit Wohn-, Industrie- und Waldgebieten sowie Feldwegen und Bundesstraßen verschiedene für den ländlichen Raum charakteristische Abschnitte. Bei der Auswahl der Messstrecke wurde darauf geachtet, dass die Strecke eine Vielzahl unterschiedlicher Szenarien abbildet. Ein weiteres wichtiges Kriterium für die Messstrecke war ein durchgehend guter Mobilfunkempfang entlang der Route, um die NTRIP Korrekturdaten für das RTK-DGNSS/INS zu empfangen.

Die Länge der Messstrecke beträgt insgesamt ca. 23 km. Bei üblichem Verkehrsaufkommen liegt die Fahrzeit bei ca. 35 Minuten. Lokalisiert ist die Messstrecke zwischen den Mendener Ortsteilen Bösperde und Halingen sowie dem

Iserlohner Ortsteil Sümmern. Bild 5 zeigt den Verlauf der Messstrecke anhand einer RTK-DGNSS/INS Messung. Im dargestellten Streckenverlauf farblich kodiert ist die vom RTK-DGNSS/INS geschätzte Standardabweichung der Position in Metern. Kleinere Abweichungen kommen vor allem durch die eingeschränkte Sicht zum Himmel im Bereich eines Waldes vor. Auf den übrigen Streckenabschnitten liegt die Standardabweichung der Lokalisierung im Bereich weniger Zentimeter.

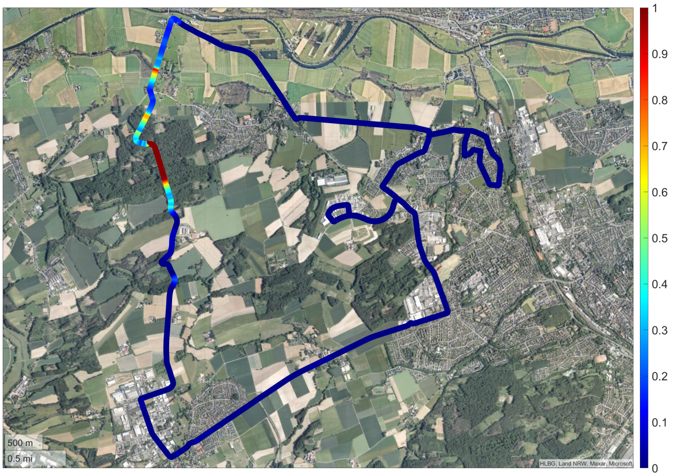


Bild 5: Route der Messstrecke. Die Farbskala gibt die Standardabweichung der vom RTK-DGNSS/INS bestimmten Position in Metern an.

4.2 Annotierungen

Der eigentliche Wert von Datensätzen für das automatisierte Fahren beruht auf der Verfügbarkeit von annotierten Ground-Truth-Daten. Für den Datensatz geplant sind Referenzdaten für die Ego-Lokalisierung, Objekte und Wetterdaten. Die meist zentimetergenaue Referenzlokalisierung wird dabei inklusive Standardabweichung direkt durch das RTK-DGNSS/INS aufgezeichnet.

Die Objektannotationen erfolgen mithilfe eines Offline-Auto-Labelling-Tools auf Basis der Lidar-Punktwolke mit anschließender manueller Validierung. Die Offline-Annotation besitzt den Vorteil, dass zukünftige Sensordaten bekannt sind und in die Entscheidungsfindungen mit einbezogen werden. Durch mehrfaches alternierendes zeitliches Forward- und Backward-Tracking können somit auch Objekte verlässlich annotiert werden, die in einigen Zeitschritten nur wenige oder keine Messpunkte aufweisen. Neben der Detektionsgüte können auch die Abmessungen der Bounding Boxen von detektierten Objekten auf diese Weise verbessert werden. In Bild 6 werden beispielhaft die Ergebnisse der automatisierten Annotierung dargestellt. Das Ego-Fahrzeug ist blau dargestellt, andere Fahrzeuge sind grün, Fußgänger sind gelb und unbekannte statische Objekte sind orange.

Im Bild erkennbar sind auch teilweise Fehlklassifikationen und zu kleine Bounding Boxen für Fahrzeuge. Zum Zeitpunkt der Beitragsverfassung wird noch eine Version des Auto-Labelling-Tools verwendet, die für das Einsatzgebiet Autobahn entwickelt wurde. In Kürze wird eine neue Version des Tools verfügbar sein, die universeller einsetzbar ist und somit auch im ländlichen Raum bessere Ergebnisse liefert. Kleinere Fehler werden durch eine manuelle Validierung der Annotierungen korrigiert.

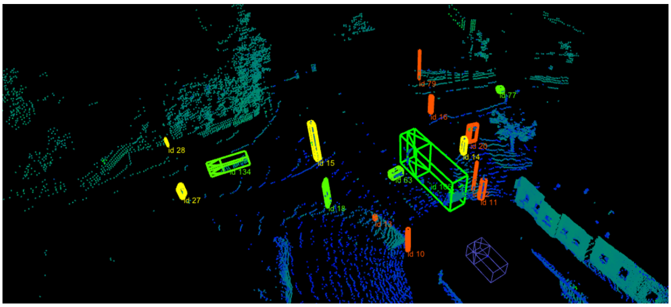


Bild 6: Ergebnisse der automatisierten Annotierung.

Die Wetterdaten werden über öffentlich verfügbare Wetterdatenbanken[2] bezogen.

[2] z.B. `https://www.timeanddate.de/wetter/deutschland/menden/rueckblick`

4.3 Aktueller Stand der Datenaufnahme

Seit August 2022 wurde die Messstrecke befahren. Auf diese Weise wurden bisher über 1.100 km an Daten auf der Messstrecke in ca. 30 Stunden aufgenommen. Auf der Messstrecke wurde bisher eine Datenmenge von insgesamt über 20 TB aufgezeichnet.

Neben den Aufnahmen der Messstrecke wurden auch die Hin- und Rückfahrten von bzw. zur TU Dortmund aufgezeichnet (städtische Umgebung, Autobahn und ländliche Umgebung), wodurch pro Fahrt weitere 70 km an Daten aufgenommen wurden. Die gesamte aufgezeichnete Datenmenge liegt somit bei über 60 TB bei einer Gesamtstrecke von über 4.500 km.

Im Laufe der Messfahrten veränderte sich die Sensorkonfiguration leicht. Die vier Radarsensoren wurden im Herbst 2022 installiert. Die vier Seitenkameras wurden erst zu Beginn des Jahres 2023 in Betrieb genommen. Zudem fiel einer der beiden Lidare für mehrere Monate aufgrund eines Wasserschadens bei einer Regenfahrt aus und wurde erst kürzlich durch ein neueres Modell ersetzt. Dementsprechend liegen die Datenraten bei den letzten Messfahrten deutlich höher als bei den ersten Messfahrten. Unter anderem aufgrund dieser Umstände werden die Messfahrten noch bis Ende März 2024 fortgeführt, um ein komplettes Jahr mit der gleichen Sensorkonfiguration abzudecken.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde das Forschungsfahrzeug der TU Dortmund sowie die Vorgehensweise bei der Aufnahme und Aufbereitung eines neuen multimodalen und multisaisonalen Datensatzes aus dem ländlichen Raum und die dafür verwendete Messstrecke vorgestellt. Der Datensatz umfasst Lidar-, Kamera- und (Automotive-)Radardaten sowie eine hochgenaue Referenzlokalisierung durch ein RTK-DGNSS/INS.

Aufgrund der langfristigen Datenaufnahme stehen für die jeweiligen Streckenabschnitte jeweils eine Vielzahl verschiedener Aufnahmen zu wechselnden

Tages- und Jahreszeiten bei unterschiedlichen Wetterbedingungen zur Verfügung. Auf diese Weise kann der Einfluss der Witterungsverhältnisse auf die Messungen unterschiedlicher Sensoren und auf Algorithmen der Umfelderfassung und Ego-Lokalisierung systematisch untersucht werden.

Der Datensatz wird aktuell für die Veröffentlichung aufbereitet und im Laufe des Jahres 2024 veröffentlicht.

Danksagung

Dieser Beitrag entstand im Zuge des Forschungsprojekts „DEmandäR“, das im Rahmen der Förderrichtlinie „Förderung der Vernetzten Mobilität und des Mobilitätsmanagements“ durch das Ministerium für Umwelt, Naturschutz und Verkehr des Landes Nordrhein-Westfalen und die Bezirksregierung Arnsberg gefördert wurde.

Literatur

[1] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen und D. Anguelov. „Scalability in Perception for Autonomous Driving: Waymo Open Dataset“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[2] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang und O. Beijbom. „Pointpillars: Fast encoders for object detection from point clouds“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] A. Geiger, P. Lenz, C. Stiller und R. Urtasun. „Vision meets robotics: The KITTI dataset“. In: *The International Journal of Robotics Research*, 32.11, S. 1231–1237, 2013.

[4] J. Fritsch, T. Kuhnl und A. Geiger. „A new performance measure and evaluation benchmark for road detection algorithms“. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 2013.

[5] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, J. Gall und C. Stachniss. „Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset“. In: *The International Journal on Robotics Research*, 40.8-9, S. 959–967, 2021.

[6] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan und O. Beijbom. „nuScenes: A Multimodal Dataset for Autonomous Driving“. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, S. 11618–11628. 2020.

[7] W. K. Fong, R. Mohan, J. V. Hurtado, L. Zhou, H. Caesar, O. Beijbom und A. Valada. „Panoptic Nuscenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking“. In: *IEEE Robotics and Automation Letters*, 7.2, S. 3795–3802, 2022.

[8] W. Maddern, G. Pascoe, C. Linegar und P. Newman, „1 year, 1000 km: The Oxford RobotCar dataset“. In: *The International Journal of Robotics Research*, 36.1, S. 3–15, 2017.

[9] K. Burnett, D. Yoon, Y. Wu, A. Li, H. Zhang, S. Lu, J. Qian, W.-K. Tseng, A. Lambert, K. Leung, A. Schoellig, T. Barfoot. „Boreas: A multi-season autonomous driving dataset“. In: *The International Journal of Robotics Research*, 42.1-2, S. 33–42, 2023.

[10] D. Barnes, M. Gadd, P. Murcutt, P. Newman und I. Posner, „The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset“. In: *IEEE International Conference on Robotics and Automation (ICRA)*, S. 6433–6438, 2020.

[11] O. Schumann, M. Hahn, N. Scheiner, F. Weishaupt, J. F. Tilly, J, Dickmann und C. Wöhler. „RadarScenes: A Real-World Radar Point Cloud Data Set for Automotive Applications“. In: *Proceedings of 24th*

International Conference on Information Fusion (FUSION), S. 1-8, 2021.

[12] A. Palffy, E. Pool, S. Baratam, J. Kooij und D. Gavrila. „Multi-class Road User Detection with 3+1D Radar in the View-of-Delft Dataset“. In: *IEEE Robotics and Automation Letters*, 7.2, S. 1, 2022.

[13] S. Macenski, T. Foote, B. Gerkey, C. Lalancette und W. Woodall. „Robot Operating System 2: Design, architecture, and uses in the wild“. In: *Science Robotics*, 7.66, 2022.

[14] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa,A. Monrroy, T. Ando, Y. Fujii und T. Azum. „Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems“. In: *9th ACM/IEEE International Conference on Cyber-Physical Systems*, S. 287–296, 2018.

[15] G. Bradski und A. Kaehler. „Learning OpenCV: Computer vision with the OpenCV library“. *O'Reilly Media, Inc.*, 2008.

[16] Z. Zhang. „A flexible new technique for camera calibration“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22.11, S.1330–1334, 2000.

[17] J.-K. Huang, S. Wang, M. Ghaffari und J. W. Grizzle. „LiDARTag: A Real-Time Fiducial Tag System for Point Clouds“. In: *IEEE Robotics and Automation Letters*, 6.3, S. 4875–4882, 2021.

Dieser Tagungsband enthält die Beiträge des 33. Workshops „Computational Intelligence“ der vom 23.11. – 24.11.2023 in Berlin stattfindet.

Die Schwerpunkte sind Methoden, Anwendungen und Tools für
- Fuzzy-Systeme,
- Künstliche Neuronale Netze,
- Evolutionäre Algorithmen und
- Data-Mining-Verfahren

sowie der Methodenvergleich anhand von industriellen und Benchmark-Problemen.

Die Ergebnisse werden von Teilnehmern aus Hochschulen, Forschungseinrichtungen und der Industrie in einer offenen Atmosphäre intensiv diskutiert. Dabei ist es gute Tradition, auch neue Ansätze und Ideen bereits in einem frühen Entwicklungsstadium vorzustellen, in dem sie noch nicht vollständig ausgereift sind.

ISBN 978-3-7315-1324-7

9 783731 513247

Gedruckt auf FSC-zertifiziertem Papier