Herausgeber

H. SCHULTE
F. HOFFMANN
R. MIKUT

Berlin | 25. – 26. November 2021

PROCEEDINGS **31. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

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

Proceedings. 31. Workshop Computational Intelligence

Berlin, 25. – 26. November 2021

PROCEEDINGS **31. WORKSHOP** COMPUTATIONAL INTELLIGENCE

Berlin, 25. – 26. November 2021

Herausgegeben von
H. Schulte
F. Hoffmann
R. Mikut

KIT Scientific Publishing

Impressum

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 2021 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-1131-1
DOI 10.5445/KSP/1000138532

Inhaltsverzeichnis

Interpretable Machine Learning: On the Problem of Explaining Model Change

Barbara Hammer[1], Eyke Hüllermeier[2]

[1]Faculty of Technology
Bielefeld University
E-Mail: bhammer@techfak.uni-bielefeld.de

[2]Institute of Informatics
LMU Munich
E-Mail: eyke@lmu.de

1 Introduction

Over the past couple of years, the idea of explainability and related notions such as transparency and interpretability have received increasing attention in artificial intelligence (AI) in general and machine learning (ML) in particular. This is mainly due to the ever growing number of real-world applications of AI technology and the increasing level of autonomy of algorithms taking decisions on behalf of people, and hence of the social responsibility of computer scientists developing these algorithms. Recent methods for improving the understandability and transparency of models produced by ML algorithms include both model-specific [6] as well model-agnostic approaches [12].

These approaches have largely focused on the explanation of *static* models, typically learned on a set of training data in a batch mode. Arguably more challenging is interpretability in the context of learning in non-stationary environments, where models are learned on a continuously evolving, potentially unbounded stream of temporally ordered data, and incrementally updated in the light of newly observed training examples [5, 4]. Corresponding algorithms must be able to react to changes in the underlying data-generating process,

which is referred to as *concept drift* [10, 8]. Concept drift may call for incremental adaptations and sometimes also more significant modifications of the model — in the extreme case of an abrupt change, the learner may even decide to abandon the current model completely and start learning from scratch.

Explaining model change, whether incremental or abrupt, is important in practical applications of online learning and can be seen as a key prerequisite for user acceptance. In particular, it is well known that humans prefer stability to change [2] — they tend to rely on what is predictable from the past and are cognitively challenged by deviations from an established solution. Hence, a good explanation is required to convince a user of any need for changing the current model.

Taking the stance that this explanation should focus on the change itself, that is, on the *differences* between the original and the updated model, we subsequently elaborate on the idea of *explaining model change* and identify a number of important problems to be addressed in this regard. In Section 3, we illustrate these problems for the specific example of instance-based learning on data streams.

2 Explaining Model Change

Consider a sequence of models $(h_t)_{t\in T}$ produced by an incremental learning algorithm A, where $T \subset [0,\infty)$ is a countable set of time indices, for example $T = \mathbb{N}$. The model $h_t : \mathscr{X} \longrightarrow \mathscr{Y}$ is produced on the basis of the data

$$\mathscr{D}_t = \big\{(x_i, y_i)\big\}_{i\in T\cap[0,t]} \subset \mathscr{X} \times \mathscr{Y}$$

observed by the learner till time t, where $\mathscr{X}$ and $\mathscr{Y}$ denote the underlying instance and outcome space, respectively (cf. Fig. 1 for an illustration). The data generating process is characterized by a corresponding sequence of probability distributions $(P_t)_{t\in T}$ on $\mathscr{X} \times \mathscr{Y}$, which may evolve over time (i.e., $P_s \neq P_t$ for $s \neq t$) and, of course, is not known to the learner; thus, we assume that each data point (x_t, y_t) is generated by P_t [16, 9].

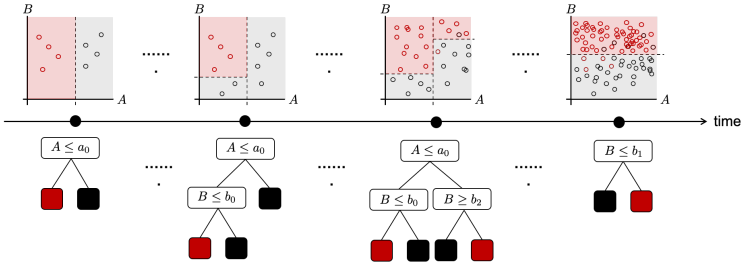


Figure 1: Illustration of model change (here for the case of decision trees) over the course of an incremental learning process.

If data and models evolve quickly, perhaps even in realtime, it will not be possible to explain every single model h_t to a user or human domain expert. Besides, individual explanations of that kind, isolated from each other, might be problematic for the user anyway, especially in the case of inconsistencies. Instead, the user might be more interested in how the model *changes* over the course of time, and in understanding the reasons for these changes. This gives rise to the idea of explaining model change in the sense of the "difference" between models, a task that appears to be more feasible, especially if changes are local, i.e., restricted to certain parts of a model or a local region of $\mathscr{X}$.

More concretely, consider a scenario in which, at every time point $t \in T$, the user has information about a previous model h_{t_0}, where $t_0 < t$. This *reference model* is not necessarily up to date, because the learning process has progressed since then and produced updated models $(h_s)_{s \in T \cap (t_0,t]}$. What we mean by explaining a model change is to inform the user about the "difference" $\Delta(h_{t_0}, h_t)$ between the reference and the current model and making h_t the new reference. Questions, problems, and challenges arising in this context include the following:

Q1 What are suitable representations of models and model change?

Q2 How to quantify model change, i.e., the difference $\Delta(h_{t_0}, h_t)$ between models h_t and h_{t_0} (distinguishing between *syntactic* difference referring to the representation of a model and *semantic* difference referring to the change of the functional dependence $\mathscr{X} \longrightarrow \mathscr{Y}$)?

Q3 How to compute $\Delta(h_{t_0}, h_t)$ efficiently, preferably in an incremental manner?

Q4 When and how often should a model change be explained, bearing in mind aspects of computational complexity, but perhaps more importantly the cognitive capacity of the human user (who is likely to prefer stability over change)?

Q5 How to complement the explanation of a change by convincing reasons for why it was needed?

Obviously, suitable answers to these questions will strongly depend on the learning task and the type of model produced by the learning algorithm. In the next section, we illustrate the problems for a specifically simple example, namely, the case of instance-based learning on data streams.

3 Instance-Based Learning on Data Streams

The notion of instance-based learning (IBL) refers to a family of machine learning algorithms, including memory-based learning, exemplar-based learning, and case-based learning [13, 7], which represent a predictive model in an indirect way via a set of stored data. Thus, in contrast to model-based machine learning methods which induce a general model (theory) from the data and use that model for further reasoning, IBL algorithms simply store the data itself and defer its processing until a prediction (or some other type of query) is actually requested — a property which qualifies them as a *lazy* learning method [1]. Predictions are then derived by combining the information provided by the stored examples, typically accomplished by means of the nearest neighbor (NN) estimation principle [3]. In this regard, examples are also referred to as *cases*, and the stored data as the *case base*.

More specifically, consider the simple example of binary classification with data of the form $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^N \subset \mathscr{X} \times \mathscr{Y}$, where $\mathscr{X} = \mathbb{R}^d$ and $\mathscr{Y} = \{0, 1\}$. The instance space $\mathscr{X}$ is equipped with a distance measure, for example the

Euclidean metric. Adopting the simple nearest neighbor rule, the model $h_{\mathscr{D}}$ induced by the data $\mathscr{D}$ is given by

$$h_{\mathscr{D}} : \mathscr{X} \to \mathscr{Y}, x \mapsto y_{\mathrm{NN}(x,\mathscr{D})},$$

where $\mathrm{NN}(x,\mathscr{D})$ denotes the (index of the) nearest neighbor[1] of x in $\mathscr{D}$, i.e.,

$$\mathrm{NN}(x,\mathscr{D}) = \arg\min_{1\leq i\leq N} \|x - x_i\|.$$

Obviously, using $h_{\mathscr{D}}$ to make predictions for new query instances x_q requires searching for the nearest neighbor of x_q in the data $\mathscr{D}$. Although the complexity of nearest neighbor search can be reduced by means of specific data structures [11], instance-based learning will typically remain more costly at prediction time than model-based learning.

On the other side, an instance-based approach naturally supports an incremental mode of learning. In fact, in the data stream scenario, where new cases are observed continuously over the course of time, the problem of learning essentially reduces to the problem of *case based editing* or *case based maintenance* [15]: every time a new example (x_{new}, y_{new}) arrives, one needs to decide whether or not this example should be added to $\mathscr{D}$, and if other cases should perhaps be removed. Disregarding computational complexity, the ideal case base $\mathscr{D}^* \subseteq \mathscr{D} \cup \{(x_{new}, y_{new})\}$ will maximize predictive performance (classification accuracy) of the induced classifier in the future. As this criterion cannot be used directly (future performance is difficult to anticipate, especially in the presence of concept drift), most methods fall back on suitable indicators of the usefulness of individual cases. For example, the IBLStreams approach [14] decides about the addition or removal of cases on the basis of the following criteria:

- Temporal relevance: Recent observations are deemed potentially more useful and are hence preferred to older ones.
- Spatial relevance: Examples can be redundant in the sense of not changing the nearest neighbor classification of any query. More generally (and less stringently), one might consider a set of examples redundant

[1] A tie breaking mechanism is needed in the case where the nearest neighbor is not unique.

if they are closely neighbored in the instance space and, hence, have a similar region of influence (Voronoi cell). In other words, a new example in a region of the instance space already occupied by many other examples is considered less relevant than a new example in a sparsely covered region.

- Consistency: An example should be removed if it seems to be inconsistent with the current concept, e.g., if its class label differs from most of the labels in its neighborhood. In this regard, however, it is important to distinguish between "noisy cases" and the possible beginning of a concept drift.

Bringing the aspect of explainability into play, we can imagine a learner adopting principles of this kind to edit its case base but delaying the update. In other words, the learner maintains a *candidate* case base $\mathscr{D}_t$ in parallel to the *reference* case base $\mathscr{D}_{t_0}$ that is used to make predictions. Thus, whenever $\mathscr{D}_t$ is modified, the learner has to decide whether to retain $\mathscr{D}_{t_0}$ or replace it by $\mathscr{D}_t$. Let us reconsider the questions Q1–Q5 for this particular scenario.

As for Q1, we already mentioned that models are represented indirectly in instance-based learning: a model $h_{\mathscr{D}_{t_0}}$ is represented by a set of cases $(x_i, y_i) \in \mathscr{D}_{t_0}$, which can be presented to a user as prototypical examples. Seen from this perspective, the case base should be kept as small as possible, because overly large case bases will compromise interpretability. Individual predictions $h_{\mathscr{D}_{t_0}}(x_q)$ are naturally "justified" by means of similarity-based or example-based explanations referring to local (nearest neighbor) information in the vicinity of the query x_q. In the simplest case, the nearest neighbor is retrieved and its class label is provided as a justification: "There is a case x_i that belongs to class y_i and resembles x_q, so x_q is likely to belong to y_i as well."

As for Q2, the syntactic difference between $h_{\mathscr{D}_{t_0}}$ and $h_{\mathscr{D}_t}$ is naturally defined in terms of the (cardinality of the) symmetric difference $(\mathscr{D}_{t_0} \cup \mathscr{D}_t) \setminus (\mathscr{D}_{t_0} \cap \mathscr{D}_t)$ between $\mathscr{D}_{t_0}$ and $\mathscr{D}_t$. Likewise, a natural definition of the semantic difference is the expected discrepancy

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) = \int_{\mathscr{X}} \| h_{\mathscr{D}_{t_0}}(x) - h_{\mathscr{D}_t}(x) \| \, p(x) dx,$$

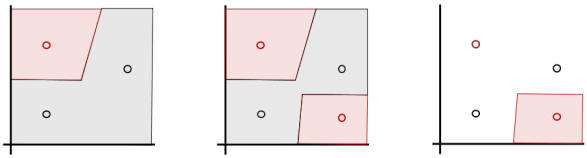


Figure 2: Illustration of a model change in the case of nearest neighbor classification. A model is characterized by a Voronoi tessellation. Adding a new example to the original model (left) leads to a change of the model (middle) and a corresponding difference (right) to be explained to the user.

where $p(x)$ is the probability (density) of observing x as a query. Because the latter is not known and difficult to estimate, especially in the presence of concept drift, one may think of

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) = \int_{\mathscr{X}} \|h_{\mathscr{D}_{t_0}}(x) - h_{\mathscr{D}_t}(x)\| \, dx \tag{1}$$

as an alternative, effectively assuming a uniform distribution on $\mathscr{X}$. Obviously, $\mathscr{X}$ must be bounded in this case, which can be guaranteed through normalization, for example by mapping $\mathscr{X}$ to $[0,1]^d$; a transformation of this kind is anyway advisable to assure commensurability between the different features (dimensions) and hence the meaningfulness of the Euclidean metric.

Turning to Q3, the computation of (1) is an algorithmically challenging problem, which comes down to identifying the (volume of) the *discrepancy region* in $\mathscr{X}$, *viz.* the set of points x for which the label of the nearest neighbor in $\mathscr{D}_{t_0}$ differs from the label of the nearest neighbor in $\mathscr{D}_t$ (cf. Fig. 2). While an efficient algorithmic solution to this problem is beyond the scope of this paper, we mention that a simple approximation can be obtained through Monte Carlo sampling:

$$\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t}) \approx \frac{1}{K} \sum_{k=1}^{K} \|h_{\mathscr{D}_{t_0}}(x'_k) - h_{\mathscr{D}_t}(x'_k)\| \,,$$

where $x'_1, \ldots, x'_K$ are sampled uniformaly at random from $\mathscr{X}$.

As for Q4 and Q5, the learner needs to take both $\Delta(h_{\mathscr{D}_{t_0}}, h_{\mathscr{D}_t})$ and the difference between $h_{\mathscr{D}_t}$ and $h_{\mathscr{D}_{t_0}}$ in terms of (estimated) usefulness into account. The

larger these quantities, the stronger the need for an update. The explanation of an update then essentially comes down to informing the user about the symmetric difference between the corresponding cases bases, i.e., explaining that some of the previous cases have become redundant or are no longer considered sufficiently prototypical, while other cases have been added as new prototypes. To convince the user of the need for a revision of the case base, one may present examples of queries that are classified correctly with the new model but incorrectly with the old one.

4 Conclusion

We motivated the task of explaining the change of models in the context of learning in dynamic environments, where data is coming in streams and continuously evolving over the course of time, possibly urging the learner might to react to concept drift. In this regard, we highlighted a number of problems and challenges to be addressed, and illustrated these problems for the specific case of instance-based learning on data streams. As a next step, we seek to realize these ideas on a more technical level, put them into practice, and evaluate them in the context of real applications. Besides, we shall study the problem of explaining model change also for other learning tasks and other model classes.

References

[1] D.W. Aha, editor. *Lazy Learning*. Kluwer Academic Publ., 1997.

[2] A. Clark. Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(3):181–204, 2013.

[3] B.V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[4] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015.

[5] G. Morales De Francisci and A. Bifet. SAMOA: scalable advanced massive online analysis. *Journal of Machine Learning Research*, 16(1):149–153, 2015.

[6] J.H. Friedman and B.E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[7] J.L. Kolodner. *Case-based Reasoning*. Morgan Kaufmann, San Mateo, 1993.

[8] V. Losing, B. Hammer, and H. Wersing. KNN classifier with self adjusting memory for heterogeneous concept drift. In *IEEE International Conference on Data Mining (ICDM)*, pages 291–300, 2016.

[9] V. Losing, B. Hammer, and H. Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[10] V. Losing, B. Hammer, and H. Wersing. Tackling heterogeneous concept drift with the Self-Adjusting Memory (SAM). *Knowledge and Information Systems*, 54(1):171–201, January 2018.

[11] Y. Malkov and D. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016.

[12] M.T. Ribeiro, S. Singh, and C. Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144. ACM Press, 2016.

[13] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[14] A. Shaker and E. Hüllermeier. IBLStreams: a system for instance-based classification and regression on data streams. *Evolving Systems*, 3(4):235–249, 2012.

[15] B. Smyth and E. McKenna. Competence models and the maintenance problem. *Computational Intelligence*, 17(2):235–249, 2001.

[16] G.I. Webb, L.K. Lee, F. Petitjean, and B. Goethals. Understanding concept drift. *arXiv preprint arXiv:1704.00362*, 2017.

Concepts for Automated Machine Learning in Smart Grid Applications

Stefan Meisenbacher, Janik Pinter, Tim Martin,
Veit Hagenmeyer, Ralf Mikut

Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen
E-Mail: stefan.meisenbacher@kit.edu

1 Introduction

Undoubtedly, the increase of available data and competitive machine learning algorithms has boosted the popularity of data-driven modeling in energy systems. Applications are forecasts for renewable energy generation [1, 2] and energy consumption [3]. Forecasts for load and generation, e. g., power, gas, and heat, on different temporal and spatial aggregation levels are elementary for sector coupling, where energy-consuming sectors are interconnected with the power-generating sector to address electricity storage challenges by adding flexibility to the power system [4]. However, the large-scale application of machine learning algorithms in energy systems is impaired by the need for expert knowledge, which covers machine learning expertise and a profound understanding of the application's process. The process knowledge is required for the problem formalization, as well as the model validation and application. The machine learning skills include the processing steps of i) data preprocessing, ii) feature engineering, iii) algorithm selection, iv) HyperParameter Optimization (HPO), and possibly v) post-processing of the model's output.

Tailoring a model for a particular application requires selecting the data, designing various candidate models and organizing the data flow between the processing steps, selecting the most suitable model, and monitoring the model during operation – an iterative and time-consuming procedure. Automated

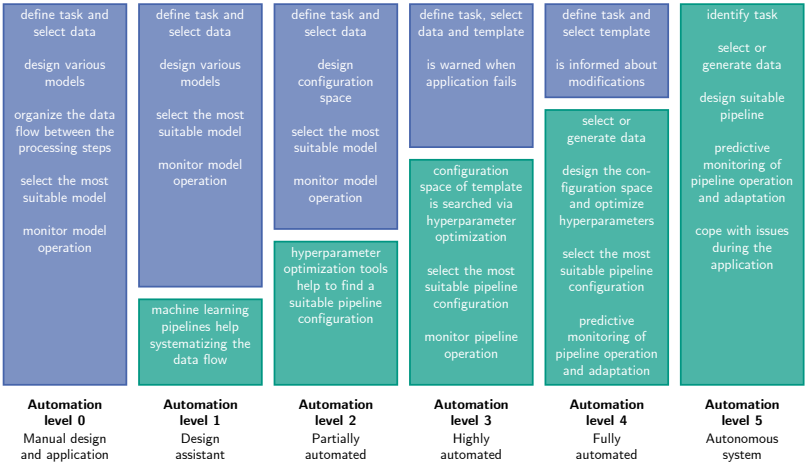


Figure 1: The five levels of automated forecasting, inspired by the SAE standard for autonomous driving of vehicles [5].

design and operation of machine learning aim to reduce the human effort to address the increasing demand for data-driven models. We define five levels of automation for forecasting where manual design and application reflect **Automation level 0**, see Figure 1. In **Automation level 1**, machine learning pipelines [6] assist the design process, systematizing the workflow by serially organizing the processing steps and managing the data flow through the steps' methods. Still, the pipeline requires manual tailoring by the data scientist to meet the specific requirements. Most published literature on energy forecasts range between Automation level 0 and 1, see reference [7]. Across the literature, standard procedures have emerged that can be used to create automated pipeline templates for specific tasks.

Partially automated forecasting is enabled in **Automation level 2**, where HPO tools[1] support the data scientist, automatically evaluating candidate models of a configuration space $\mathbf{\Lambda}$ defined by the data scientist. Still, the data scientist needs to analyze the optimization results, select the most suitable model, and monitor the model during operation. In the literature, few approaches exist for energy systems that we can classify as Automation level 2. A framework

[1] e. g., Hyperopt [8], SMAC [9], or NNI [10]

for automated HPO and forecasting algorithm selection is proposed by Rätz et al.[11]. Cui et al.[12], and Shahoud et al.[13] propose frameworks for the automated forecasting algorithm selection using meta information such as statistical properties of the time series and characteristics of the system. An approach for combining HPO and ensembling of forecasting algorithms is proposed by Wu et al.[14]. Maldonado et al.[15] and Valente and Maldonado [16] introduce embedded feature selection approaches for the Support Vector Regression (SVR), integrating exogenous weather information into electrical load forecasting.

Automation level 3 reaches highly automated forecasting by providing pipeline templates for specific tasks that include an associated configuration space $\mathbf{\Lambda}$ or a robust default configuration $\boldsymbol{\lambda}$. The data scientist needs to provide the data and select the template. Anomaly detection monitors operation and alerts the data scientist when suspicious model inputs or outputs are detected.[2] A highly automated framework for building energy management is proposed by Schachinger et al.[17], including a heuristic for the automated design of Artificial Neural Networks (ANNs), online assessment, and scheduled re-training.

In **Automation level 4**, the fully automated forecasting takes over the data selection. The data is either taken from a data storage assigned to the selected template or generated synthetically according to the template-specific task. During operation, the model predicts its performance and warns the data scientist before system borders are reached.

Finally, **Automation level 5** achieves a fully autonomous system that independently identifies the task, creates the model, and detects and resolves issues during operation.

The introduced automation levels are not rigid – interim levels are possible. To the best of our knowledge, there are yet no applications for smart grids in Automation levels 4 and 5. The remainder of this paper is organized as follows. First, we present a general approach to automate the design and operation of forecasting models in energy systems in Section 2. Then, we describe and

[2] Current open-source Automated Machine Learning (AutoML) tools, e. g., AutoSklearn [18], or TPOT [19], support automated design of regression and classification models. Monitoring of the model operation is not provided.

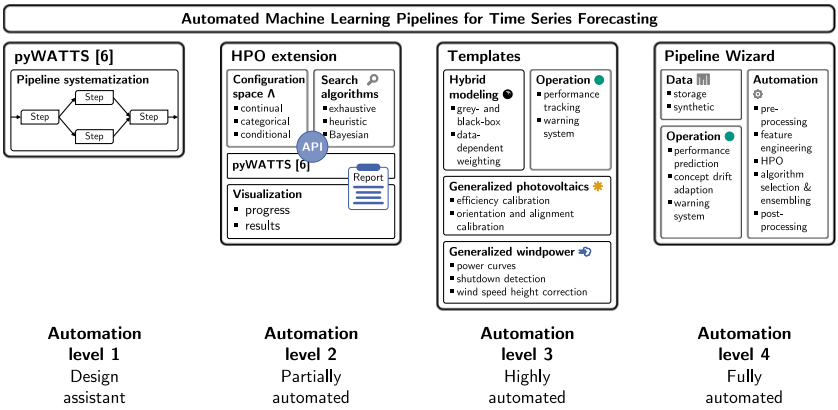


Figure 2: Identified Automated Machine Learning (AutoML) approaches for smart grid application. The HyperParameter Optimization (HPO) extension communicates via the Application Programming Interface (API) with pyWATTS [6].

evaluate automated design algorithms for a hybrid model (autonomous level 2.5) in Section 3. Finally, Section 4 concludes and provides an outlook on future research.

2 Approach

Although numerous methods for AutoML have been proposed in the literature, a toolkit tailored for forecasting in energy systems is lacking. Figure 2 shows a schematic overview of unexplored automation approaches and our long-term concept based on a taxonomy discussed in Subsection 2.1.

The open-source Python Workflow Automation Tool for Time Series (pyWATTS)[3] [6] assists researchers in the design process, systematizing the workflow through a pipeline with a uniform interface for various methods applied to the steps of the pipeline (**Automation level 1**).

For specialized tasks, the expertise of a data scientist and a process expert is still required, and we want to keep the human in the loop. To reduce the effort

[3] `https://github.com/KIT-IAI/pyWATTS`

of tailored pipeline design, an HPO extension for pyWATTS [6] is required (**Automation level 2**). The extension enables defining a configuration space $\mathbf{\Lambda}$ and selecting a search algorithm, and wraps around pyWATTS. Communication is established by an Application Communication Interface (API) of pyWATTS, allowing the optimization algorithm to configure pipeline parameters. The report interface of pyWATTS provides data for visualization of the optimization progress and results. The schematic process of HPO with pyWATTS is outlined in Subsection 2.2.

Recurring tasks with good generalizability, such as forecasting of PhotoVoltaic (PV) and Wind Power (WP) generation, can be handled with default templates for large-scale deployment (**Automation level 3**). A default template contains a forecasting pipeline with normalized output that needs little effort to calibrate for new operational environments. We introduce a template for PV forecasting in Subsection 2.3, and a template for WP forecasting in Subsection 2.4.

The hybrid modeling template couples two grey- or black-box models by a data-dependent weighting of the model outputs. In regions where the model input is well represented in the training data set, a sophisticated model is overweighted, whereas, in less representative regions, a robust model gains weight, as it is expected to have better extrapolation characteristics. We evaluate exhaustive and Bayesian HPO for the automated design of a black-box hybrid model without operation monitoring (**Automation level 2.5**) on ten benchmark data sets in Section 3.

The operation of the templates is supported by performance monitoring and a warning system, alerting the data scientist if any issue is detected during operation, such as unusually high forecasting errors.

The vision for fully automated pipeline design for energy systems requires a tool specific to energy systems to integrate domain knowledge – the *Pipeline Wizard* (**Automation level 4**). For tasks where comprehensive training data is missing, the data manager automatically selects appropriate training data from a related data storage or synthetically generates training data. The *Pipeline Wizard* automates the design of the forecasting pipeline and enables integrating specific methods for each pipeline section. The operation of the *Pipeline Wizard* is guided by predictive performance estimation to detect drifts

in the pipeline error at an early stage. This is required to trigger automated model adaption to cope with concept drifts and informs the data scientist about changes made. Detailed information on the planned realization of the *Pipeline Wizard* can be found in Subsection 2.5.

2.1 Literature Review

Review papers are fundamental for the evaluation of the state of science and the identification of research gaps. In the research area of AutoML, several literature review papers exist, e. g., [20, 21]. However, they are limited to regression and classification tasks. Further, AutoML methods focus on the problem of Combined Algorithm Selection and Hyperparameter optimization (CASH) [20]. For time series forecasting, pre-processing and feature engineering are vital sections of the machine learning pipeline and require specialized methods, considering the temporal sequence of data points. Consequently, a review on AutoML for time series forecasting must consider time series-specific methods and the complete pipeline – an unaddressed issue in the present review studies.

2.2 Hyperparameter Optimization Extension for pyWATTS

Systematizing the workflow with machine learning pipelines can be achieved with pyWATTS [6]. For enabling external HPO tools to access the pipeline configuration $\boldsymbol{\lambda}$ of pyWATTS, we target to define an API.

The schematic process for HPO is shown in Figure 3. The data scientist formalizes the problem, defines the structure of the machine learning pipeline, and selects the data. The data needs to be split into a set for tuning and a test set. The tuning set is used to find a suitable pipeline configuration $\boldsymbol{\lambda}$. We further split the tuning data set and use a portion to train the pipeline and evaluate the performance of $\boldsymbol{\lambda}$ on the validation set.[4] The test set is hold-out to evaluate the tuned pipeline afterward.

[4] To increase the robustness, Cross-Validation (CV) can be applied.

The data scientist parametrizes the pipeline sections to be optimized and defines the configuration space $\mathbf{\Lambda}$ accordingly. For the definition of $\mathbf{\Lambda}$, continuous, categorical, and conditional terms are available. While continuous terms are used to define the configuration space of hyperparameters, categorical terms are used for making decisions, such as choosing a polynomial or Radial Basis Function (RBF) kernel of an SVR or selecting an ANN or SVR as the forecasting algorithm. Depending on the choice, conditional terms enable the definition of corresponding sub-configuration spaces $\mathbf{\Lambda}_{\text{cond}} \subset \mathbf{\Lambda}$, e. g., the degree of the polynomial kernel if this kernel was selected.[5]

The HPO tool selects a hyperparameter configuration $\boldsymbol{\lambda} \in \mathbf{\Lambda}$, which is assigned to the pipeline. pyWATTS trains and validates the pipeline and returns the performance Q on the validation data split, which is usually the forecasting error. Depending on the selected search algorithm of the HPO tool, Q is used for the selection of the next $\boldsymbol{\lambda}$ to be evaluated or not. We target to implement the open-source HPO tool Neural Network Intelligence (NNI) [10], which allows the selection of various search algorithms, including exhaustive, heuristic, and Bayesian algorithms, while the definition of the configuration space $\mathbf{\Lambda}$ is standardized.
In HPO, parallel computing is crucial for feasible run times. We may parallelize the pipeline's training process, the computation of CV folds, the computation of configuration trials, or combinations of these. The best parallelization strategy depends on the actual problem and can be determined in a preceding experiment. The HPO extension for pyWATTS will include the above strategies.

The evolution of the pipeline performance during optimization and the optimization results need to be visualized to aid the data scientist in the design process. The evolution plot of Q may indicate the convergence of directed search algorithms and guide the data scientist in deciding on the termination. Visualization of the best performing hyperparameter configurations helps the developer to decide whether $\mathbf{\Lambda}$ was well defined.

[5] The applied definition of a configuration space $\mathbf{\Lambda}$ is shown in Subsection 3.2.

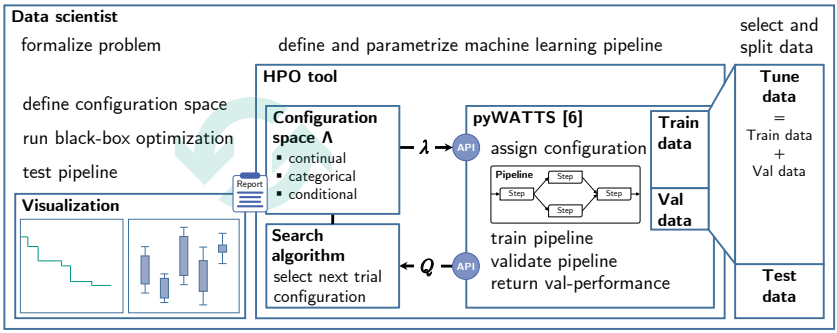
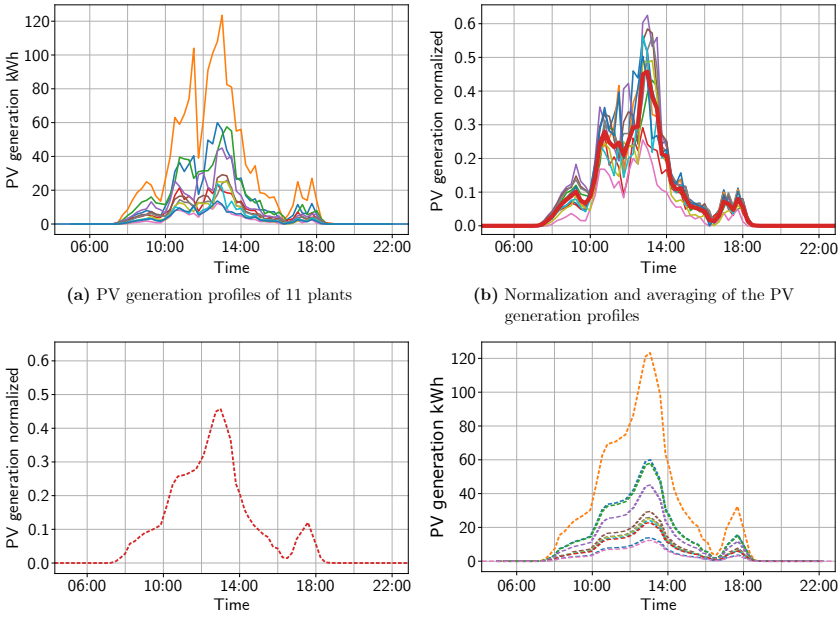


Figure 3: The schematic process of HyperParameter Optimization (HPO) with the pyWATTS extension: The data scientist is supported by the HPO tool in tailoring the pipeline to a specific problem. The HPO tool passes a hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ via the Application Programming Interface (API) to pyWATTS [6] and receives the pipeline performance Q on the validation (val) data.

2.3 Generalized Photovoltaics Template

The majority of published literature on PV forecasting is limited to individual plants, e. g., [22, 23, 24]. They differ in terms of input features, forecasting horizon, and forecasting algorithms. The increasing adoption of renewable energies and their integration into redispatch policies leads to a rapidly growing demand for PV forecasting models. Therefore, we expect that designing and training an individual model for each PV plant is infeasible due to the immense design effort and the need for a sufficient amount of training data for each plant, which are not present for new plants. Several commercial solutions exist for renewable energy forecasting in the context of redispatch actions, e. g., [25, 26, 27]. However, the applied methods are closed-source, making an evaluation in terms of forecasting performance and design efficiency impossible.

We propose a generalized PV forecasting template, which uses weather forecasts for the plant's location as input data – more precisely, global radiation and air temperature [28]. Thus, weather forecasting is an external module for which we may use a commercial weather forecasting service or an individual weather model. Figure 4 shows the process of the generalized PV generation forecasting template. We generalize the model using normalized training data of eleven PV plants, whose alignment and orientation are either unknown or



(a) PV generation profiles of 11 plants

(b) Normalization and averaging of the PV generation profiles

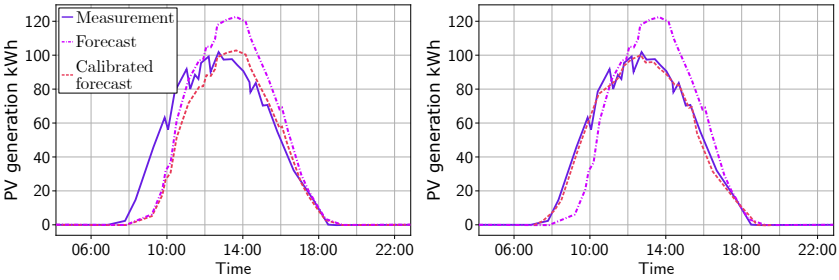
(c) Forecast of the normalized average PV generation profile

(d) Re-transforming the PV generation profile to PV plants

Figure 4: The process of the generalized PhotoVoltaic (PV) generation forecasting template.

ambiguous. After normalizing the generation profiles according to the peak power of the PV plants, we calculate the average generation profile. We train the generalized PV template to forecast the average normalized generation profile with the weather forecast as input data. After forecasting the average normalized profile, we re-transform the generation profile to individual plants in the post-processing.

We validate this approach out-of-sample and achieve a normalized Mean Absolute Error (nMAE) of 26.3 %. We may reduce the nMAE to 15.9 % if we would use a flawless weather forecast. To reduce the forecasting error of the template, it seems reasonable to introduce calibration factors. The factors allow the calibration of the template to individual plants to compensate for different efficiency levels, as well as inclinations and orientations, see Figure 5. In addition, the PV template will support various complex forecasting models



(a) Efficiency calibration

(b) Orientation and alignment calibration

Figure 5: The calibration of the generalized PhotoVoltaic (PV) generation forecasting template.

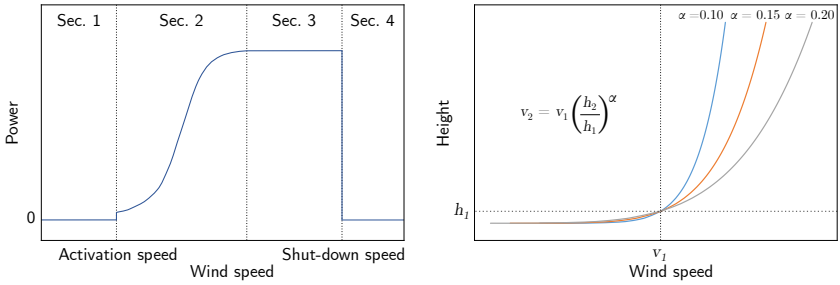
depending on the availability of data, i. e., depending on the availability and amount of site-specific historical data and weather forecasts.

The proposed generalized PV model was developed for the *Stadtwerke Karlsruhe Netzservice GmbH*. For the automated application, we target to implement online performance tracking and calibration. Thereby, the model can adapt to decreasing efficiencies due to aging or changing environmental conditions, e. g., shading from new buildings in the surrounding area. Once re-calibration is performed, the data scientist is informed about the modification.

2.4 Generalized Wind Power Template

Wind turbine manufacturers provide empirical power curves, which link the power output of the wind turbine to the wind speed at hub height. We propose to use these power curves to forecast WP generation, rather than designing and training individual data-driven models [28]. The input of a power curve is the wind speed of a weather forecast, which comes from a commercial service or an individual weather model. As the wind speed of the weather forecast is not at hub height, a correction is necessary. We use the wind profile power law

$$\frac{v_2}{v_1} = \left(\frac{h_2}{h_1}\right)^{\alpha}, \tag{1}$$



(a) Empirical wind power curve

(b) Height correction of the wind speed

Figure 6: The wind power curve and height correction with the wind profile power law for reference height h_1 and the wind speed at this height v_1 with different exponents α.

where v_1 and v_2 are the wind speeds at height h_1 and h_2 above the ground, and α is the empirically derived friction coefficient, depending on the topology of the landscape [29]. Figure 6 shows the four sections of a power curve and the height correction with the wind profile power law. Using the height correction, we are able to calibrate the power curve to the respective turbine. In most cases, the heights h_1 and h_2 are known. The velocity v_1 is the wind speed of the weather forecast. In this case, we calibrate the power curve with the exponent α. In the literature, α is given for different landscape topologies, which serve as a starting value for the calibration. The utilization of calibrated empirical power curves eliminates the need for extensive training data. If training data is available, re-calibration of α is possible. In the first application, we determined α using grid search and selected α such that the forecasting error on a validation data set becomes minimal.[6]

We validate the calibrated power curve out-of-sample and achieve an nMAE of 62.6 %. The model error seems high, but it is mainly related to the forecasting accuracy of the wind speed. A flawless forecast of the wind speed reduces the nMAE to 19.6 %. The wind speed forecast, in particular, has difficulty predicting single wind gusts, justifying the large discrepancy between the nMAE obtained with weather forecasts and flawless forecasts. During the compilation of the validation data set, we noticed anomalies due to the manual shutdown of the wind turbines. To clean the data set, we used a heuristic method that takes

[6] If apart from α, h_1 or h_2 is unknown, calibration is still possible but the effort increases.

advantage of the fact that there are two turbines in the immediate neighborhood and searches for deviating outputs between the turbines. In further work, we target to develop a universal anomaly detection method that also works for individual turbines.

As the PV template (Subsection 2.3), the WP template was developed for the *Stadtwerke Karlsruhe Netzservice GmbH*. We aim to implement online performance tracking and calibration, as well as data-dependent complexity of the forecasting model. In this way, the model may adapt to changing inflow characteristics, e. g., caused by transformations of the landscape topology. Re-calibration triggers the information of the data scientist about the model adaptation.

2.5 Pipeline Wizard

The objective of the *Pipeline Wizard* is the automated forecasting pipeline design and large-scale application for consumption data in energy systems, including electricity, gas, and heat. For these systems, sufficient training data is not always available. However, we may use data from related systems for training with similar environmental conditions, unit size, and utilization. The related data either can be taken from a data warehouse or generated synthetically. In order to select or generate suitable data, meta information is necessary that describe the behavior of the system. By eliminating the need for measurement data of the system, forecasting models can be applied widely.

Recent AutoML tools focus on the CASH problem for classification and regression tasks. Time series forecasting requires specialized methods for preprocessing and feature engineering that consider the temporal sequence of the data. We target to provide time series-specific methods for each section of the pipeline, including pre-processing, feature engineering, HPO, algorithm selection, and ensembling. Apart from these default methods, integrating specialized methods for particular system domains is possible, which are then taken into account in the automated design process, e. g., copy-paste imputation for energy time series to handle anomalies [30] or the engineering of energy-specific meta-features [12].

Concept drifts pose a major challenge in the application of forecasting models. A concept drift involves the change of the target variable's statistical properties [31]. Reasons for concept drifts are manifold: a change of utilization, changing exogenous influences affecting the system, or structural changes such as unit size or system boundaries. The changes may occur suddenly, incrementally, or gradually and may reoccur [32]. At the same time, the forecasting accuracy of a model decreases if the trained relationships between input and output variables no longer match the system's behavior. In this situation, adapting the forecasting model to the changed system is necessary [31]. Different adaptation strategies are possible. The most straightforward strategy is re-training the forecasting model with the data accumulated after the concept drift. An improvement can be achieved if not only the model parameters but also its structure and hyperparameters are tuned. Celik and Vanschoren [33] evaluated six adaptation and tuning strategies on evolving data for a classification task. We target to tailor these strategies to time series forecasting and evaluate their effectiveness.

3 Automated Hybrid Modeling

Robust models are necessary for the representation of participants in smart grids, e. g., the thermal dynamics of buildings or the characteristics of Electric Vehicle (EV) batteries [34]. Data-driven models can achieve high predictive accuracy if the input variables are in familiar range, thus, similar to the training data set (*interpolation)*. The prediction accuracy declines in *extrapolation* areas [35], i. e., if the model makes an inference about the system's behavior in a new range of variables [36]. Böhland et al.[37] propose a hybrid model for local adaption of the model complexity to interpolation and extrapolation. The hybrid model creates a hull around the interpolation areas using a fuzzy modeled One-Class Support Vector Machine (1C-SVM). The fuzzified hull serves as a weighting function for the extrapolation and the interpolation model.

3.1 Automated Design

The design algorithm for black-box modeling automatically determines a suitable combination of the interpolation and the extrapolation model (sub-models), as well as the 1C-SVM. The data set is split into training, validation, and test data. The algorithm creates candidate hybrid models using the training data and estimates their performance afterward with the validation data. After selecting the best performing hybrid model, it is retrained using the training and validation data (tuning data), and the performance is assessed with the test data.

Grid Search The most elementary algorithm for optimizing the configuration $\boldsymbol{\lambda}$ of a model is grid search, where a finite set of candidate configurations is defined and exhaustively evaluated. The configuration space $\boldsymbol{\Lambda}$ consists of sub-models of various prediction algorithms, incorporating the MultiLayer Perceptron (MLP), the SVR, Multivariate Adaptive Regression Splines (MARS) [38], and the LOcal LInear MOdel Tree (LOLIMOT) [39], and the 1C-SVM; each prediction and decision algorithm has a finite space of candidate hyperparameters. The algorithm of Böhland et al.[37] creates the candidate models and gathers all trained sub-models and 1C-SVMs in the model pool. Then, each possible combination of sub-models and 1C-SVMs of the pool is evaluated on the validation data, and the $\boldsymbol{\lambda}$ with the lowest MAE is selected.

Bayesian Optimization Rather than evaluating a finite search grid, Bayesian optimization explores and exploits the configuration space $\boldsymbol{\Lambda}$. The optimization scheme uses a probabilistic surrogate model to approximate the objective function $\mathscr{Q}$, mapping the model's performance Q over $\boldsymbol{\Lambda}$. In each iteration, the surrogate model is updated, and the optimization scheme uses an acquisition function to decide on the next hyperparameter configuration $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ to be observed [40]. To apply Bayesian optimization towards automated hybrid modeling, we need to define the configuration space $\boldsymbol{\Lambda}$.

3.2 Evaluation

We evaluate the automated design of the hybrid model as in reference [37] on ten data sets and compare grid search to Bayesian optimization.

Experimental Setup In the initial proposal of the automated hybrid model, Böhland et al.[37] showed that it performs significantly better than standard regression models on nine out of ten benchmark data sets. In this experiment, we compare HPO algorithms for automated model design, i. e., Bayesian optimization with exhaustive grid search. We evaluate which HPO algorithm achieves lower prediction errors and requires less computation time.

For grid search, we adopt the configuration space $\mathbf{\Lambda}$ of reference [37] with the 1C-SVM implementation of the Scikit-learn library[7] [41] (RBF kernel; $\sigma = 0.01, 0.025, 0.05, 0.1, 0.2, \ldots, 1, 1.5, 10$; $\varepsilon = 0.001$)[8]. Since the LOLIMOT model is not available in the Python programming language [42], we omit this model type but added Random Forest (RF), Gradient Boosting Machine (GBM), and Linear Regression (LR):

- MLP ($N_{\text{neurons}} = 2, 3, \ldots, 17, 30, 50$),
- SVR (RBF kernel; $\sigma = 0.1, 0.2, \ldots, 1, 1.2, 1.5, 2$; $C = 100$; $\varepsilon = 0.001$)[8],
- GBM ($N_{\text{estimators}} = 90, 100, \ldots, 150, 200, 300, \ldots, 1000$),
- RF ($N_{\text{estimators}} = 90, 100, \ldots, 150, 200, 300, \ldots, 1000$),
- MARS,
- LR

We implemented the automated design process in Python [42] and adapted the grid search from the implementation of the Scikit-learn library[7] [41]. SVR, RF, and LR are based on the Scikit-learn library as well. The GBM implementation is based on the XGBoost library[9] [43], and MARS relies on the Py-earth library[10] [44]. For Bayesian optimization, we apply the NNI toolkit[11] [10]

[7] `https://github.com/scikit-learn/scikit-learn`
[8] some references denote ε as ν
[9] `https://github.com/dmlc/xgboost`
[10] `https://github.com/scikit-learn-contrib/py-earth`
[11] `https://github.com/microsoft/nni`

with the Tree Parzen Estimator (TPE) optimizer [8]. The configuration space $\mathbf{\Lambda}$ consist of three categorical hyperparameters – the choice for the interpolation, extrapolation, and decision algorithm. For the interpolation and the extrapolation, the optimizer may choose the above-listed prediction algorithms. If an algorithm is chosen, the corresponding conditional configuration space $\mathbf{\Lambda}_{\text{cond}} \subset \mathbf{\Lambda}$ with continuous values and the limits corresponding to the respective minimum and maximum values of the grid search applies. For the decision algorithm, only the 1C-SVM algorithm with RBF kernel can be chosen with continuous hyperparameters and limits equivalent to the grid search.

We evaluate the automated design process on ten data sets and split the data randomly into training data (60 %), validation data (20 %), and test data (20 %). The test data is initially held out. With the remaining data, we perform a four-fold CV for each candidate configuration $\boldsymbol{\lambda}$ and calculate the mean MAE over the splits. Based on the mean MAE, the HPO algorithm determines the most suitable $\boldsymbol{\lambda}$. Afterward, the hybrid model is re-fitted with the chosen $\boldsymbol{\lambda}$ using the train and validation data (tuning data) and tested on the hold-out test data. We repeat this process five times for each data set with different random seeds for splitting the data.

Results We evaluate the performance of grid search and Bayesian optimization by comparing the MAE of the chosen configuration $\boldsymbol{\lambda}$ on the hold-out test data and the computation times. Table 1 shows the experimental results regarding the prediction error MAE and the computing time.

The comparison of the MAE on the hold-out test data shows that no HPO algorithm has a significant advantage in terms of prediction errors. The advantage of Bayesian optimization is that only the boundaries of the configuration space $\mathbf{\Lambda}$ have to be defined. Thus, configurations between the points of the grid search are considered, and a reasonable definition of $\mathbf{\Lambda}$ depends less on the skillful definition of candidates by the data scientist.

Grid search shows a clear advantage in terms of computing time. The advantage can be justified with the re-usability of already trained sub-models. For searching the configuration space $\mathbf{\Lambda}$ of the hybrid model, it is sufficient to fit the grid points of each prediction algorithm individually and cache the predictions

Table 1: Prediction error and computation time comparison of grid search and Bayesian optimization for the automated design of the hybrid model.

Data Set	**Prediction Error MAE** [10^{-3}]		**Computation Time** [s]	
	grid search	Bayesian optimization	grid search	Bayesian optimization
Abalone	52.98	**52.81**	**111**	3832
Airfoil	27.70	**26.72**	**51**	2112
Boston	**49.87**	50.22	**39**	1093
California	64.99	**64.78**	**607**	12637
Computer	16.85	**16.56**	**387**	7396
Concrete	37.79	**36.25**	**48**	1176
Ailerons	**26.28**	28.29	**173**	6049
Elevators	40.50	**40.29**	**261**	8040
Redwine	**68.71**	69.81	**144**	3541
Whitewine	**82.82**	84.15	**66**	2091

on the validation data. Then, we may calculate the prediction error for all possible $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}$ combinations of the interpolator, extrapolator, and decider by combining the cached results (similar to *dynamic programming*). Thereby, the number of configurations to be assessed does not increase exponentially with the points of the grid search but linearly, resulting in $N_{\text{MLP}} + N_{\text{SVR}} + N_{\text{MARS}} + N_{\text{GBM}} + N_{\text{RF}} + N_{\text{LR}} + N_{\text{1C-SVM}} = 79$ evaluations. In Bayesian optimization, in contrast, we define the number of candidate configurations (trials) to be evaluated $N_{\text{trials}} = 500$, and the optimization selects candidates based on the performance of previous trials (directed search).

Figure 7 shows the evolution of the mean MAE on the validation splits of the *computer* data set using Bayesian optimization and grid search. The 95 % confidence interval was determined over the five loops based on the Student's t-distribution. The progression of the Bayesian optimization converges well before the 500$^{\text{th}}$ trial. Thus, there is the potential of terminating the Bayesian optimization prematurely as soon as we are satisfied with the result. In contrast, the grid search cannot be terminated prematurely, as otherwise, areas of the configuration space $\boldsymbol{\Lambda}$ would not have been examined, and we would obtain an incomplete model. Therefore, it is reasonable to implement convergence regularization in future work that terminates the optimization automatically if no further improvement is expected (*early stopping*).

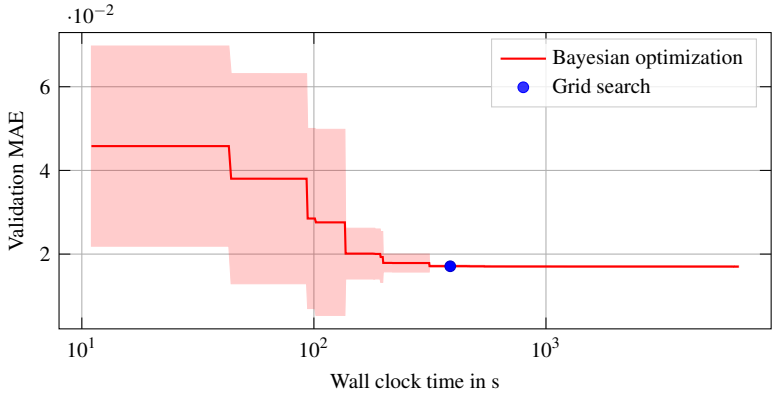


Figure 7: Development of the MAE on the validation splits of the *computer* data set using Bayesian optimization and grid search. The HPO was repeated five times with random splits. The red line reflects the mean progress over the independent runs, and the light red area the 95 % confidence interval, determined based on the Student's t-distribution.

In future work, we plan to integrate the hybrid model into pyWATTS [6] with the grid search. The convergence regularization will be developed for the pyWATTS HPO extension (see Subsection 2.2), for HPO problems with exponential complexity.
In addition to the application as an EV battery model for representing the electrical behavior shown in reference [34], we target to model thermal building dynamics, using black-box models for interpolation and grey-box models (thermal-electrical analogy) for extrapolation. In this way, we target to design a robust default template for thermal building modeling, which can be used for demand side management, e. g., using model predictive control.

4 Conclusion and Outlook

The transformation of a fossil-based to a sustainable energy system requires the large-scale application of machine learning algorithms. For satisfying the rapidly growing demand for time series forecasts, we need to automate the design and application. We proposed five automation levels, where Automation level 0 is manual design and application and Automation level 5 is an autono-

mous system. For Automation levels 1, 2, 3, and 4, we introduced forecasting approaches for smart grid applications and described their concepts.

For one of the approaches – the hybrid model – we evaluated two HyperParameter Optimization (HPO) algorithms for the automated design (Automation level 2.5). The hybrid model weights the results of two models depending on whether the input values were represented in the training data set or not. In this way, a robust model is used for extrapolation and a sophisticated model for interpolation. The evaluation shows an advantage of grid search in terms of computation time if we re-use already trained models. Regarding the prediction error, there is no clear advantage of grid search or Bayesian optimization.

In future work, a performance tracking and warning system could monitor the templates' operation and alert the data scientist if degrading forecasting performance is detected. We target to improve the PhotoVoltaic (PV) and the Wind Power (WP) forecasting templates by online calibration (Automation level 3). More precisely, the PV template will be calibrated for individual plant efficiencies, orientations, and alignments, and the WP template will include HPO for the friction coefficient α. For Automation level 4, we develop the *Pipeline Wizard*, automating the design of the complete forecasting pipeline. The *Pipeline Wizard* includes automated data selection or generation, online performance prediction, and adaption strategies for concept drifts. In the long view, probabilistic interval forecasts will replace point forecasts, e. g., reference [45].

We plan to integrate the proposed automation approaches in the Energy Lab 2.0 [46], a real-world research environment for exploring intelligent coupling of various energy generation, storage, and supply capabilities. The approaches for each automation level help to solve forecasting tasks according to individual complexity and requirements.

Acknowledgements

This project is funded by the Helmholtz Association's Initiative and Networking Fund through *Helmholtz AI*, the Helmholtz Association under the Program *Energy System Design*. The authors would like to thank *Stadtwerke Karlsruhe*

Netzservice GmbH (Karlsruhe, Germany) for the provided PV and WP data, and Moritz Böhland for sharing the Matlab implementation of the Hybrid Model [37]. Conceptualization and methodology: S. M., R. M.; Experiments, validation, investigation, and visualization: J. P., T. M., S. M.; Writing – original draft preparation: S. M.; Writing – review and editing: S. M., J. P., T. M., V. H., R. M.; Supervision and funding acquisition: V. H., R. M.; All authors have read and agreed to the published version of the article.

References

[1] R. Ahmed, V. Sreeram, Y. Mishra, and M. D. Arif, "A review and evaluation of the state-of-the-art in PV solar power forecasting: Techniques and optimization," in Renewable and Sustainable Energy Reviews, vol. 124, pp. 109792, 2020.

[2] Z. Qian, Y. Pei, H. Zareipour, and N. Chen, "A review and discussion of decomposition-based hybrid models for wind energy forecasting applications," in Applied Energy, vol. 235, pp. 939–953, 2019.

[3] I. K. Nti, M. Teimeh, O. Nyarko-Boateng, and A. F. Adekoya, "Electricity load forecasting: A systematic review," in Journal of Electrical Systems and Information Technology, vol. 7, no. 13, 2020.

[4] G. Fridgen, R. Keller, M. F. Körner, and M. Schöpf, "A holistic view on sector coupling," in Energy Policy, vol. 147, pp. 111913, 2020.

[5] Society of Automotive Engineers, "SAE J3016 levels of driving automation," 2021, `https://www.sae.org/standards/content/j3016_202104/`.

[6] B. Heidrich, A. Bartschat, M. Turowski, O. Neumann, K. Phipps, S. Meisenbacher, K. Schmieder, N. Ludwig, R. Mikut, and V. Hagenmeyer, "pyWATTS: Python Workflow Automation Tool for Time Series," arXiv: 2106.10157, 2021.

[7] J. A. González Ordiano, S. Waczowicz, V. Hagenmeyer, and R. Mikut, "Energy forecasting tools and services," in WIREs Data Mining and Knowledge Discovery, vol. 8, no. 2, pp. e1235, 2018.

[8] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in Proceedings of the 30. International Conference on Machine Learning, pp. I-115–I-123, 2013, http://hyperopt.github.io/hyperopt/.

[9] M. Lindauer, K. Eggensperger, M. Feurer, S. Falkner, A. Biedenkapp, and F. Hutter, "SMAC v3: Algorithm configuration in Python,' 2017, https://automl.github.io/SMAC3/master/.

[10] Microsoft Corporation, "Neural Network Intelligence," 2017, https://nni.readthedocs.io/en/stable/index.html.

[11] M. Rätz, A. P. Javadi, M. Baranski, K. Finkbeiner, and D. Müller, "Automated data-driven modeling of building energy systems via machine learning algorithms," in Energy and Buildings, vol. 202, pp. 109384, 2019.

[12] C. Cui, T. Wu, M. Hu, J. D. Weir, and X. Li, "Short-term building energy model recommendation system: A meta-learning approach," in Applied Energy, vol. 172, pp. 251–263, 2016.

[13] S. Shahoud, H. Khalloof, M. Winter, C. Duepmeier, and V. Hagenmeyer, "A meta learning approach for automating model selection in big data environments using microservice and container virtualization technologies," in Proceedings of the 12. International Conference on Management of Digital EcoSystems, pp. 84–91, 2020.

[14] Z. Wu, X. Xia, L. Xiao, and Y. Liu, "Combined model with secondary decomposition-model selection and sample selection for multi-step wind power forecasting," in Applied Energy, vol. 261, pp. 114345, 2020.

[15] S. Maldonado, A. González, and S. Crone, "Automatic time series analysis for electric load forecasting via support vector regression," in Applied Soft Computing, vol. 83, pp. 105616, 2019.

[16] J. M. Valente, and S. Maldonado, "SVR-FFS: A novel forward feature selection approach for high-frequency time series forecasting using support vector regression," in Expert Systems with Applications, vol. 160, pp. 113729, 2020.

[17] D. Schachinger, J. Pannosch, and W. Kastner, "Adaptive learning-based time series prediction framework for building energy management," in Proceedings of the 2018 IEEE International Conference on Industrial Electronics for Sustainable Energy Systems (IESES), pp. 453–458, 2018.

[18] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in Advances in Neural Information Processing Systems, vol. 28, pp. 2962–2970, 2015, `https://automl.github.io/auto-sklearn/master/`.

[19] T. T. Le, W. Fu, and J. H. Moore, "Scaling tree-based automated machine learning to biomedical big data with a feature set selector," in Bioinformatics, vol. 36, no. 1, pp. 250–256, 2020, `http://epistasislab.github.io/tpot/`.

[20] M.-A. Zöller, and M. Huber, "Benchmark and survey of automated machine learning frameworks," in Journal Artificial Intelligence Research, vol. 70, pp. 409–472, 2021.

[21] S. K. K. Santu, M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, "A level-wise taxonomic perspective on automated machine learning to date and beyond: Challenges and opportunities," arXiv: 2010.10777, 2021.

[22] A. Chaouachi, R. M. Kamel, and K. Nagasaka, "Neural network ensemble-based solar power generation short-term forecasting," in Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 14, no. 1, pp. 69–75, 2010.

[23] A. Mellit, and A. M. Pavan, "A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a

grid-connected PV plant at Trieste, Italy," in Solar Energy, vol. 84, no. 5, pp. 807–821, 2010.

[24] S. Atique, S. Noureen, V. Roy, V. Subburaj, S. Bayne, and J. Macfie, "Forecasting of total daily solar energy generation using ARIMA: A case study," in Proceedings of the IEEE 9. Annual Computing and Communication Workshop and Conference (CCWC), pp. 0114–0119, 2019.

[25] Zentrum für Sonnenenergie- und Wasserstoff-Forschung Baden-Württemberg (ZSW), "GridSage," accessed: 27.08.2021, 2021, https://www.zsw-bw.de/leistung/netzintegration-und-mobilitaet/gridsage-prognosen-fuer-den-redispatch-20.html.

[26] energy & meteo systems GmbH, "FuturePowerFlow," accessed: 27.08.2021, https://www.emsysgrid.de/produkte/redispatch/futurepowerflow.php.

[27] KISTERS AG, "KISTERS Redispatching 2.0," accessed: 27.08.2021, https://www.redispatching.de.

[28] T. Martin, "Redispatch 2.0: Prognose der Einspeisemenge erneuerbarer Energien mithilfe von Machine Learning Ansätzen," Bachelor's thesis, Karlsruhe Institute of Technology, Institute for Automation and Applied Informatics, 2021.

[29] G. M. Masters, "Renewable and efficient electric power systems," 1. ed., Wiley, New York, 2004.

[30] M. Weber, M. Turowski, H. K. Çakmak, R. Mikut, U. Kühnapfel, and V. Hagenmeyer, "Data-driven copy-paste imputation for energy time series," in IEEE Transactions on Smart Grid, early access, 2021.

[31] I. Žliobaitė, M. Pechenizkiy, and J. Gama, "An overview of concept drift applications," in N. Japkowicz, J. Stefanowski (eds) Big Data Analysis: New Algorithms for a New Society, Studies in Big Data, vol. 16, pp. 91–114, Springer, Cham, 2016.

[32] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," in ACM Computing Surveys, vol. 46, n. 4, 2014.

[33] B. Celik, and J. Vanschoren, "Adaptation strategies for automated machine learning on evolving data," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 9, pp. 3067–3078, 2021.

[34] K. Schwenk, S. Meisenbacher, B. Briegel, T. Harr, V. Hagenmeyer, and R. Mikut, "Integrating battery aging in the optimization for bidirectional charging of electric vehicles," in IEEE Transactions on Smart Grid, early access, 2021.

[35] M. A. Babyak, "What you see may not be what you get: A brief, nontechnical introduction to overfitting in regression-type models," in Psychosomatic Medicine, vol. 66, no. 3, pp. 411–421, 2004.

[36] N. Matalas, and V. Bier, "B. Extremes, extrapolation, and surprise," in Risk Analysis, vol. 19, no. 1, pp. 49–54, 1999.

[37] M. Böhland, W. Doneit, L. Gröll, R. Mikut, and M. Reischl, "Automated design process for hybrid regression modeling with a one-class SVM," in at - Automatisierungstechnik, vol. 67, no. 10, pp. 843–852, 2019.

[38] J. H. Friedman, "Multivariate adaptive regression splines," in The Annals of Statistics, vol. 19, no. 1, pp. 1–67, 1991.

[39] B. Hartmann, T. Ebert, T. Fischer, J. Belz, G. Kampmann, and O. Nelles, "LMNTOOL–Toolbox zum Automatischen Trainieren Lokaler Modellnetze," in Proceedings of the 22. Workshop Computational Intelligence, pp. 341–355, 2014.

[40] M. Feurer, and F. Hutter, "Hyperparameter optimization," in F. Hutter, L. Kotthoff, J. Vanschoren (eds) Automated Machine Learning: Methods, Systems, Challenges, pp. 3–33, Springer International Publishing, Cham, 2019.

[41] 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, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” in Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011, https://scikit-learn.org/stable/index.html.

[42] Python Software Foundation, “Python language reference, version 3.8,” http://www.python.org.

[43] Chen, Tianqi, and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proceedings of the 22. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, 2016, https://xgboost.readthedocs.io/en/latest/index.html.

[44] J. Rudy, “Py-earth,” 2013, https://contrib.scikit-learn.org/py-earth/.

[45] K. Phipps, S. Lerch, M. Andersson, R. Mikut, V. Hagenmeyer, and N. Ludwig, “Evaluating ensemble post-processing for wind power forecasts,” arXiv: 2009.14127, 2021.

[46] V. Hagenmeyer, K. C. Hüseyin, C. Düpmeier, T. Faulwasser, J. Isele, H. B. Keller, P. Kohlhepp, U. Kühnapfel, U. Stucky, S. Waczowicz, and R. Mikut, “Information and communication technology in Energy Lab 2.0: Smart Energies System Simulation and Control Center with an open-street-map-based power flow simulation example,” in Energy Technology, vol. 4, no. 1, pp. 145–162, 2016.

Quantification Platform for Touch Response of Zebrafish Larvae using Machine Learning

Yanke Wang[1], Christian Pylatiuk[1], Ralf Mikut[1], Ravindra Peravali[2], Markus Reischl[1]

[1]Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen
E-Mail: yanke.wang@kit.edu

[2]Institute for Biological and Chemical Systems, Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen

Abstract

A touch-evoked response of zebrafish larvae provides information of the mechanism of the gene functional expressions. Recently, an automated system has been developed for precise and repeated touch-response experimentation with minor human intervention. The data collected by the system are analyzed with regard to an automated quantification pipeline for scientific conclusions, including five quantification criteria: latency time, C-Bend curvature maximum, C-Bend peak time, response time, and moving distance. To quantify these criteria, we propose a larva tracking based automatic quantification pipeline by using a U-Net for initialization of tracking, a particle filter as tracking strategy, and region growing for the segmentation of larvae. Experimental data with different treatments are analyzed by using the proposed quantification platform for demonstration, and the result proves that this platform can generate comparable touch-response behavior quantification readouts in an efficient and automatic way. This platform provides an alternative to automatically

screening the drugs for knowledge discovery according to the pattern of the touch-response behaviors of zebrafish larvae mutated by chemicals.

1 Introduction

Zebrafish larvae are commonly used animal models for the organism-based screenings due to small size, high fecundity and short reproductive cycle [8]. Their specific (repeatedly and obvious) behaviors indicate certain functional mechanisms of mutants by the treatments [1, 7], making it possible to do the large-scale high-throughput screening of chemicals or drugs. Automated experimental systems to acquire the data of these behaviors have been developed so far [4, 5, 6, 7, 20], so the automated high-throughput quantification of the data from the systems is also becoming in a higher demand, as manual quantification is time-consuming and not statistically comparable. In particular, the touch-response experimental data (videos) are in a high frame rate [2, 19], so the automated quantification is more essential in this case. During the touch-evoked response of zebrafish larvae, the larvae form into a series of C-Bends and swim away after touching, and it is important to quantify the time that the larvae take to respond as well as the strength of the response (such as the latency time, C-Bend curvature maximum, C-Bend peak time, response time, and moving distance). However, it is difficult to generate a precise number of C-Bend curvature and moving distance manually [20]. Furthermore, the operators cannot keep the same criteria all the time for each video, as the video has more than ten thousand frames in average. Thus, we proposed a touch-response quantification pipeline for single zebrafish larva in [2], but as for the multi-larvae case, we face more challenges: i) multiple larvae need to be tracked and segmented at the same time; ii) which larva is touched should be decided; iii) the quantification of multiple larvae has higher computational costs. To solve these problems, we optimized the pipeline to an automatically customized touch-response quantification platform in this work.

In this proposed quantification platform of touch-response experimental data, the tracking procedure plays the vital role, especially for the tracking of multiple larvae [19]. Recently, machine learning or deep learning based tracking

methods have emerged to promote the accuracy of the tracking procedure [3, 17], and many previous works focused on the tracking and segmentation of single or multiple adult zebrafish [9, 10, 11, 15]. To make the best of the deep learning methods, we used a U-Net based segmentation method for the initialization of tracking. However, those high-computational methods are difficult to be used in the tracking procedure of our high-frame-rate videos. In order to make the quantification pipeline much less complex, we proposed a optical flow based needle tracking procedure and particle filter based larvae tracking procedure. Besides, the segmentation for each larva is also of importance to the analysis of the movements. In [16], a Gaussian Mixture Model (GMM) based segmentation is used to detect the moving objects, and the noise is filtered according to the region size by using a global Otsu method. However, in our platform, considering global information makes the procedure more computational expensive. Therefore, a local region growing based segmentation method is used for each larva according to the result of tracking procedure. Based on the tracking and segmentation results, we proposed a pipeline to find the touched larvae and generate the behavior quantification according to the proposed experiment criteria. In order to test the performance of the proposed platform, we conducted six sets of experiments with different drugs and analyzed the experiment criteria and detected errors (failure cases). With the verification of the experiment results, this platform shows a high efficiency for analyzing the touch-response experimental data, and releases the operators. The methods used in this platform can make contributions to the application in the field of video analysis. As well, the platform can be also transformed to the quantification pipeline of other organisms (like medaka) and can be also added with more quantification criteria.

Organization of the article is as follows. Section 3 describes the tracking procedures, local segmentation for the larvae and the quantification pipeline of the proposed platform. Section 3 provides the setup of the experiments, the quantification criteria and results as well as the discussion. According to the above results, conclusions are drawn in Section 4.

2 Methodology

As the videos collected by the automated system are in a high frame rate (1000 frames per second), an efficient tracking procedure is required. The initial positions obtained from the first frame is vital to accuracy of the whole tracking procedure, so a U-Net based segmentation (Step 1 in Fig. 1) is used to generate the binarization of the larvae and the needle for the initialization of the tracking. However, we cannot use the U-Net for the tracking of the following frames, as deep learning inference is computationally expensive, causing the high temporal costs for one single video. Therefore, optical flow (Step 2 in Fig. 1) is used for the tracking of the needle, and particle filter (Step 2 $\sim$ 4 in Fig. 1) is used for the predictions of the positions of the larvae in the following frames. Based on the predictions, region growing (Step 5 in Fig. 1) is applied for the local segmentation of each larva. The output of the tracking procedure includes images patches of each larva as well as the positions of the needle and the larva in all frames.

2.1 U-Net for initialization

Ahead of the tracking procedure, the positions of the larvae need to be initialized, which is usually done manually, but in order to make this procedure fully automated, we used a U-Net [18] to do the segmentation of the first frame of the video for initialization. The U-Net mainly consists of down-sampling blocks - two convolutional layers (Conv) and one max pooling (Max-pool) - for feature extraction and up-sampling blocks - one deconvolutional layer (deConv) and two convolutional layers. As shown in Step 1 in Fig. 1, the U-Net based segmentation inputs the image within the well area cropped by Hough transform and generates two binary images with larvae and needle, followed by the needle tracking and larva tracking strategies respectively.

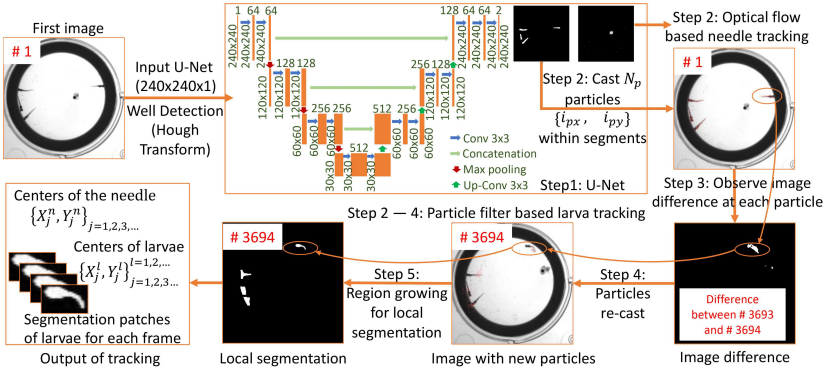


Figure 1: Overall architecture of the proposed tracking procedure. Step 1: Network architecture of a U-Net used to segment the larvae and needle for initialization. The outputs are two binary images for the larvae and needle respectively. Step 2: Two tracking strategies for the needle and larvae respectively. The optical flow tracking method is used for the positions of the needle as it moves slightly between two frames. For the particle filter based larva tracking method, particles are cast in this step within the segmented larvae areas. Step 3: According to the position of each particle, the image difference between two frames (with an example of the image difference between Frame #3693 and Frame #3694) is observed for the binary probability of the corresponding particle. Step 4: The particles with binary probability 0 are re-cast around the larvae center, details in Section 2.3. Step 5: For each larva, the segmentation is achieved by local region growing, discussed in Section 2.4. The outputs of the tracking contain the image patches of all larvae as well as the series of the centers of the larvae and the needle.

2.2 Optical flow based needle tracking

In the optical flow tracking procedure, the tracking target is assumed to move slightly between two frames [12, 13], and the movement of the needle meets this requirement. Thus, the optical flow based needle tracking strategy is used. Let $\{X_j^n, Y_j^n, t_j\}$ be the old needle (n) center at frame t_j, so the new needle center at frame t_{j+1}, $\{X_{j+1}^n, Y_{j+1}^n, t_{j+1}\}$, is estimated according to the gradients as described in [2].

2.3 Particle filter based larva tracking

For the tracking procedure of the larvae, optical flow does not meet the assumption as the larvae move significantly, so we used a particle filter based tracking

strategy. Particle filter, not like Kalman filter, has no constrained assumptions [3], and the tracking result is dependent on the score of each particle cast randomly according to the prior knowledge (the previous positions of the larvae in our case). As shown in Step 2 in Fig. 1, the particles (with number N_p) are cast randomly within the segments of each larva to do the following tracking procedure. Defined that ${}_iP_j^l = \{{}_ix_j^l, {}_iy_j^l, t_j\}$ be the particle i at position $\{{}_ix_j^l, {}_iy_j^l\}$ of frame t_j for larva l, the binary probability $b\{{}_ix_j^l, {}_iy_j^l, t_j\}$ indicates whether the l-th larva exists, shortened as ${}_ib_j^l$. The new center of the larva l at frame t_{j+1} $\left(\{X_{j+1}^l, Y_{j+1}^l, t_{j+1}\}\right)$ is estimated as follows

$$X_{j+1}^l = \frac{1}{N_p}\sum_{i=1}^{N_p}({}_ix_j^l \cdot {}_ib_{j+1}^l), \quad Y_{j+1}^l = \frac{1}{N_p}\sum_{i=1}^{N_p}({}_iy_j^l \cdot {}_ib_{j+1}^l) \tag{1}$$

where ${}_ib_{j+1}^l$ is the binary probability at $\{{}_ix_j^l, {}_iy_j^l\}$ of the lth larva in frame t_{j+1}. The binary probability is computed according to the image difference as follows

$${}_ib_{j+1}^l = \begin{cases} 1 & \text{if } {}_id_{j+1}^l > T_d \\ 0 & \text{else} \end{cases}, \quad {}_id_{j+1}^l = |f_{j+1}({}_ix_j^l, {}_iy_j^l) - f_j({}_ix_j^l, {}_iy_j^l)| \tag{2}$$

where ${}_id_{j+1}^l$ is the pixel difference at $\{{}_ix_j^l, {}_iy_j^l\}$ between frame t_{j+1} and frame t_j, $f_{j+1}({}_ix_j^l, {}_iy_j^l)$ is the pixel value at $\{{}_ix_j^l, {}_iy_j^l\}$ of frame t_{j+1}, $f_j({}_ix_j^l, {}_iy_j^l)$ of frame t_j, and T_d is the threshold for the image difference for the moving pixels. The particles ${}_iP_j^l$ with ${}_ib_{j+1}^l = 0$ are re-cast in a Gaussian distribution as follows

$$\begin{aligned} {}_i\hat{P}_{j+1}^l = \{{}_ix_{j+1}^l, {}_iy_{j+1}^l, t_{j+1}\}, \{{}_ix_{j+1}^l, {}_iy_{j+1}^l\} \sim \mathcal{N}(\boldsymbol{\mu}_{j+1}^l, \boldsymbol{\Sigma}) \\ \boldsymbol{\mu}_{j+1}^l = \begin{bmatrix} X_{j+1}^l \\ Y_{j+1}^l \end{bmatrix}, \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} \end{aligned} \tag{3}$$

where ${}_i\hat{P}_{j+1}^l$ is the updated (re-cast) particle at $\{{}_ix_{j+1}^l, {}_iy_{j+1}^l\}$ of the new frame t_{j+1}, $\boldsymbol{\mu}_{j+1}^l$ is the new center of the larva estimated by (1), and $\boldsymbol{\Sigma}$ is the heuristic variance for the range of re-casting the particles. The retained particles are used for storing the previous information of the positions of the larvae, and

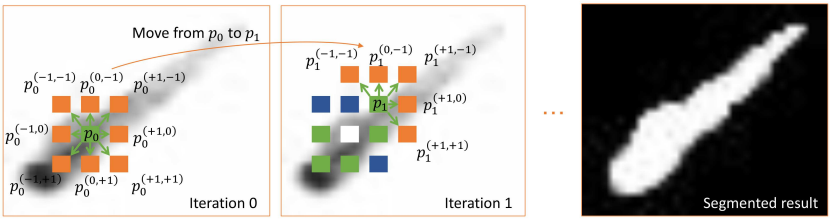


Figure 2: The principle of region growing based local segmentation of the larva. Region growing begins with an initialized point and grows according to the adjacent pixel values until no more new pixels meet the requirements, details in Section 2.4. The adjacent points of p_k are denoted as $p_k^{(*,*)}$, e.g. $p_k^{(-1,-1)}$ is the top left point of p_k. The pixels in green are the next centers for iterations with the adjacent points in orange, and the pixels that meet the requirements are labeled as white and otherwise as blue.

the re-cast particles are used for searching for new potential positions of the larvae.

2.4 Region growing for local segmentation

The particle filter might lose the larva as it only considers the moving pixels. Thus, the segment for each larva is required for a more precise center of the larvae as well as for the analysis of the behaviors. We used a region growing to do the local segmentation for each larva, as shown in Fig. 2. The region starts at the estimated center $\left(\{X_{j+1}^l, Y_{j+1}^l\}\right)$ of the larva (the initialized point) and label each pixel in a 3×3 adjacent area according to the pixel value and image gradient. Assumed that $p_k = \{p_{kx}, p_{ky}\}$ is the center of the adjacent area in each iteration (starting iteration: $p_0 = \{X_{j+1}, Y_{j+1}\}$), the label of each adjacent point $p_k^{(*,*)} = \{p_{kx}^{(*,*)}, p_{ky}^{(*,*)}, * = -1, 0, +1\}$ is calculated as

$$L(p_k^{(*,*)}) = \begin{cases} 1 & \text{if } g(p_{kx}^{(*,*)}, p_{ky}^{(*,*)}) < T_g \\ & \quad \text{and } T_l < f(p_{kx}^{(*,*)}, p_{ky}^{(*,*)}) < T_h \,, \\ 0 & \text{else} \end{cases} \tag{4}$$

$$g(p_{kx}^{(*,*)}, p_{ky}^{(*,*)}) = |f(p_{kx}^{(*,*)}, p_{ky}^{(*,*)}) - f(p_{kx}, p_{ky})|$$

where $L(p_k^{(*,*)})$ is the labelled (segmented) result for the position $p_k^{(*,*)}$ at k-th iteration, $g(p_{kx}^{(*,*)}, p_{ky}^{(*,*)})$ is the image gradient at $p_k^{(*,*)}$, $f(p_{kx}^{(*,*)}, p_{ky}^{(*,*)})$ is the pixel value at $p_k^{(*,*)}$, and T_g, T_l, T_h are the heuristic thresholds chosen for the binarization. The position $p_k^{(*,*)}$ labelled as 1 is the next center of the adjacent area at iteration $k+1$ for the growing of the region, until all new centers are labelled as 0. As the larva area is connected with other objects or noise, the iteration may not stop even if the area covers the larva in a larger scale. Thus, we set another size threshold (T_s) to end the iterations. As the growing of the regions only occurs in the local areas of the larvae, the computation is much faster than the global binarization methods or deep learning based methods [15, 16].

2.5 Quantification pipeline based on tracking procedure

When zebrafish larvae are touched, they exhibit characteristic (or specific) behaviors [1]. In this work, five typical quantification indices are considered, three of which were considered in a previous work (latency time t_l, response time t_r, and moving distance d_m) [2]. As for the quantification of the response strength, in this work, we consider to use the maximum of the C-Bend curvature that the larvae shaped (C-Bend curvature maximum, c_m), as the average cannot quantify the peak value of the response strength. Additionally, we propose to use another parameter, C-Bend peak time (t_{cp}), to describe the time that the larvae took to have the peak response strength.

The tracking procedure for the needle and larvae outputs: i) sets of image patches for each larva in each frame; ii) the centers of the larvae; iii) the centers of the needle in each frame, shown in Fig. 1 and Fig. 3. The touched larva is decided by comparing the final position (at t_f) of the needle $X_{t_f}^n$ and the initialized positions of the larvae $\{X_0^l\}^{l=1,2,\cdots}$. In order to compute t_1, t_2, t_3, and t_4, another two thresholds are defined: i) T_{nl} for the distance between the needle and the larva deciding the touch is successfully applied; ii) T_m for the movement of the larvae, with details in Fig. 3. According to the time points above, the quantification indices are computed as follows: i) the latency time is computed as $t_l = t_2 - t_1$; ii) the C-Bend peak time is computed as

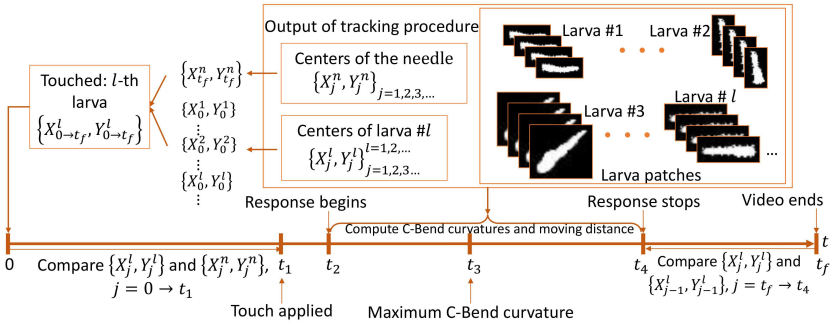


Figure 3: The pipeline for behavior quantification according to the results of tracking procedure. According to the output of the tracking procedure, important time points (t_1: touch applied, t_2: response begins, t_3: the time point of maximum C-Bend curvature, t_4: response stops) are searched: i) the distance between lth larva and the needle is compared from $t = 0$ until the time point with the distance less than T_{nl}, as t_1; ii) from t_1, the distance of the lth larva between two frames (X_j^l and X_{j+1}^l) is computed until the time point over T_m, as t_2; iii) the time point with maximum C-Bend curvature is t_3; iv) from t_f back to the previous time points, the positions of the lth larva between two frames (X_j^l and X_{j-1}^l) are compared until the time point over T_m, as t_4. More details for each index are described in Section 2.5

$t_{cp} = t_3 - t_2$; iii) the response time is computed as $t_r = t_4 - t_2$; iv) from t_2 to t_4, the moving distance d_m and C-Bend curvatures c_m of the l-th larva in each frame are computed according to the corresponding image patches (methods described in [2]).

3 Experiments and results

3.1 Experiment setup

In order to test the performance of the proposed platform on the experimental data from the automated touch-response system [2], we quantified six sets of experimental data (as Table 1 shows): videos of *wild* type (without treatment), larvae with Dimethyl sulfoxid (*DMSO*)[1], as well as larvae treated by Diazepam (*Dia*) to reduce the movements [14], Isoproterenol hydrochloride (*Iso*) with

[1] As each treatment is prepared with *DMSO*, the experiments on the larvae with only *DMSO* are also conducted as controls.

Table 1: The experimental data (number of videos) to be quantified.

Type	Treatment	Concentration	Age	Number of videos
Wild	Fish water	-	73 hpf	24
DMSO	Dimethyl sulfoxide	1%	73 hpf	27
Dia	Diazepam	100 μmol/mL	73 hpf	38
Iso	Isoprenaline hydrochloride	100 μmol/mL	73 hpf	30
Caffi	Caffeine	100 μmol/mL	73 hpf	24
Saha	Suberoylanilide hydroxamic acid	100 μmol/mL	73 hpf	30

unknown effects, Caffeine (*Caffi*) for also reduction of movements [14], and Suberoylanilide hydroxamic acid (*Saha*) with unknown effects, respectively. Each treatment is in a concentration of 100 μmol/mL for the demonstration. The larvae were dechorionated and treated at 27 hpf, and the experiments were conducted at 73 hpf, as visualized in Table 1.

The parameters used for the quantification platform are outlined in Table 2. The average size of the larvae is 162.66 pixels, computed by 320 images (4 larvae in each), so the threshold (T_s) is set as 200 pixels for safety. The other parameters are selected heuristically.

3.2 Experiment criteria

The experiment criteria (quantification indices) are discussed in Section 2.5 to verify whether the proposed quantification platform can generate corresponding results to the assumptions of the effects of the treatments, as described in Section 3.1. Besides, the detected errors of the quantification pipeline should also be analyzed, e.g. the inaccuracy of the segmentation method and missing objects by the tracking procedure. As well, the videos collected by the automated system contain some unquantifiable ones, such as the larvae were not touched, and the larvae or needle might not be detected. Thus, we also aim

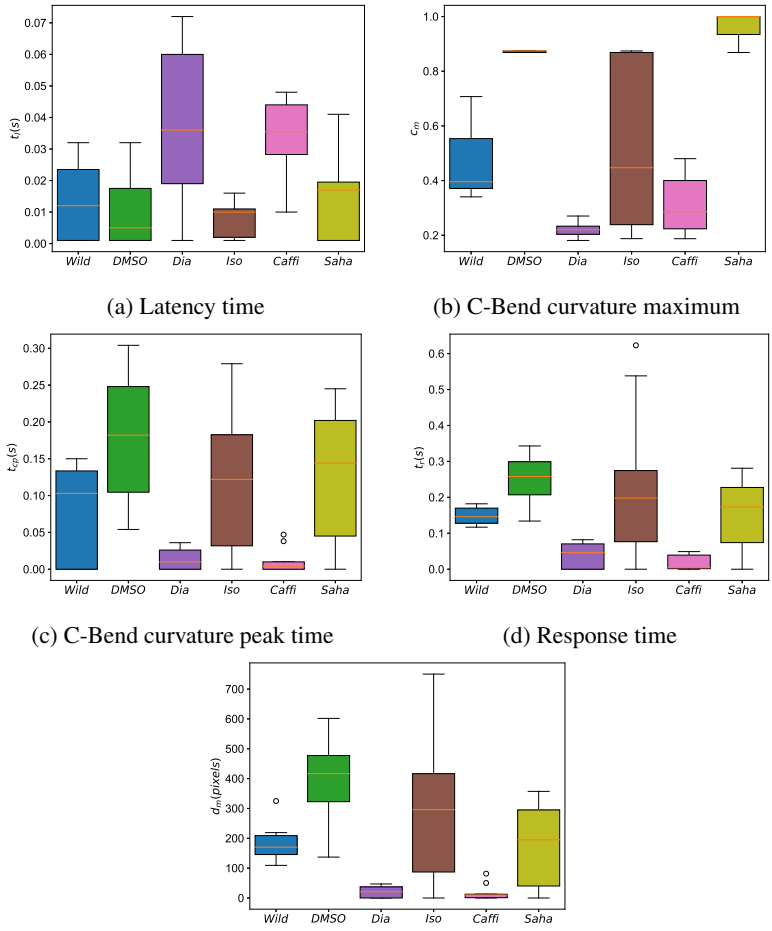
Table 2: The parameters used for the proposed quantification platform.

Symbol	Quantity	Value
N_p	the number of the particles	50
T_d	the threshold for the image difference (pixels)	10
σ_x, σ_y	the standard deviation for the range of re-casting the particles (pixels)	7
T_g	the threshold for the image gradient in local segmentation (pixels)	100
T_l	the lower threshold for the binarization in local segmentation (pixels)	50
T_h	the higher threshold for the binarization in local segmentation (pixels)	220
T_s	the size threshold for the larvae (pixels)	200
T_{nl}	the threshold for the distance between the larva and needle (pixels)	10
T_m	the threshold for the movement of the larvae	50%

to give the analysis of detected errors by using the number of videos with no larvae touched ($\#NT$) as well as those with failure of quantification ($\#QF$), with details in Section 5.

3.3 Results

We applied our quantification pipeline (described in Section 2.5) to the experimental data outlined in Table 1 and visualized the quantification results for the touched larvae in Fig. 4, including latency time (t_l), C-Bend curvature maximum (c_m), C-Bend curvature peak time (t_{cp}), response time (t_r), and moving distance (d_m). The five quantification indices give a consistent output: The larvae with longer latency time have lower response strength (lower c_m), shorter time to shape the C-Bend peak (lower t_{cp}), and less response duration (lower t_r and d_m), examples seen from the cases of Dia and $Caffi$. This



(a) Latency time

(b) C-Bend curvature maximum

(c) C-Bend curvature peak time

(d) Response time

(e) Moving distance

Figure 4: Five quantification indices of six experiment cases (*wild*, *DMSO*, *Dia*, *Iso*, $Caffi$, and *Saha*) generated by the quantification pipeline in Section 2.5, including latency time, C-Bend curvature maximum, C-Bend curvature peak time, response time, and moving distance.

result can also prove that the larvae under the treatments of *Dia* and $Caffi$ respond less compared with the *Wild* and *DMSO*, verifying the assumptions in Section 3.1. Additionally, the treatment *Iso* can not change the touch-

response behaviors of the larvae significantly as the five indices show similar results to the *Wild* type. As for the treatment *Saha*, the result is similar to the case of *DMSO*, verifying that this treatment cannot change the touch response of zebrafish larvae too much. The results above verified that our proposed platform can generate comparable quantification according to the experimental data acquired by the automated touch-response system and is also potentially useful for drug screening.

Despite the useful results in Fig. 4, some problems still exist apparently, like the detected errors of the proposed platform. As mentioned in Section 3.2, among the videos collected ($\#T$), we first compared the manually screened $\#NT_g$ (the ground-truth number of the videos with no larvae touched) with the numbers output from the proposed quantification platform ($\#NT_p$), shown in Table 3, with the percentage ($E_{NT} = |\#NT_g - \#NT_p|/\#T$). As well, we also give the number of failure of quantification ($\#QF$) with the percentage ($E_{QF} = \#QF/|\#T - \#NT_g|$). Our proposed platform can with more than 90% in average find the larvae not touched. Besides, no results were generated from around 10% of valid videos ($\#T - \#NT_g$) by our system. In addition, we assume that the larvae under the treatment of *Dia* scarcely have response, so the output of latency time is expected to be infinite, and the other indices are expected to be 0. However, the system cannot generate infinite number, but from Fig. 4a, the latency time is the highest which is still useful to be compared with the other cases. Furthermore, the results in Fig. 4b-4e are not exactly zero, caused by following reasons: i) Some larvae still have slight response; ii) The movements of the needle might push the larvae away (fake response); ii) The tracking procedure generates the movements of the larvae because of the slight environment changes or other inaccuracy. Nonetheless, the results of Fig. 4b-4e are still comparable to the other cases, and in other words, our proposed system verified our assumption on treatment *Dia* even with slight variance. Finally, the proposed platform can achieve the quantification in a higher efficiency with in average 63 ms per frame on CPU, compared with using U-Net for the tracking procedure (in average 2.60 s per frame on CPU).

Table 3: The analysis of the detected errors (failure cases) of the proposed platform.

Type	$\#T$	$\#NT_g$	$\#NT_p$	E_{NT}	$\#QF$	E_{QF}
Wild	24	4	7	12.5%	1	5%
DMSO	27	3	8	18.5%	0	0%
Dia	38	4	1	7.9%	8	23.5%
Iso	30	6	6	0%	2	8.3%
Caffi	24	7	5	8.3%	3	17.6%
Saha	30	5	6	3.3%	2	8%
Average	-	-	-	8.4%	-	10.4%

3.4 Discussion

The results in Section 5 verified that our proposed platform can work as an automated quantification tool for the multi-larvae touch-response experimental data in a high frame rate. This platform has following advantageous strategies: i) The decision of t_1 as well as the touched larva is decided by the last point of the needle and the initialized point of the larvae, as the local segmentation of the larva during tracking procedure is not as accurate as the initialized segmentation by the U-Net; ii) The movement of the larvae is decided by the change of each particle instead of the change of the larva center, as the centers of the larvae might change slightly but constantly during the tracking procedure, even when the larvae do not actually move; iii) The decision of t_4 is done from the last frame to the previous, since the larva might move slowly (no significant changes of pixels) for a moment and start moving strongly again; iv) The design of the quantification pipeline makes it possible to consider the global information for a more reasonable quantification result, as the quantification is achieved after the tracking task of all frames.

However, some drawbacks are still needed to consider carefully when the users apply this platform or pipeline to the customized data. The tracking procedure and local segmentation of the larvae are the keys for this quantification platform, but they cannot be guaranteed for a good result in following cases: i) the larvae overlap with each other when moving; ii) the well edge area has similar brightness with the larvae; iii) the needle overlaps with the larvae.

Nevertheless, these problems can be solved by statistical analysis of a large set of data, so our proposed platform is vital in such case.

4 Conclusion

In this work, we introduced a machine learning based quantification platform for touch response of zebrafish larvae, which can generate five quantification indices (latency time, C-Bend curvature maximum, C-Bend curvature peak time, response time, and moving distance) automatically without human intervention. This platform uses an automated quantification pipeline based on a multi-larvae tracking procedure, with a U-Net for initialization of tracking procedure, a particle filter for tracking, and region growing for local segmentation of larvae. To test the performance of the proposed quantification pipeline, six sets of experiments (2 controls and 4 treatments) were conducted and the results generated from this platform as well as the analysis of the detected errors verified the effectiveness of the platform. A high efficiency is also guaranteed with in average 63 ms per frame for the quantification pipeline on CPU. Our future work will be to apply our proposed platform on more data from other drug screening of touch response of zebrafish larvae.

Acknowledgements

This work was financially funded by China Scholarship Council (CSC). The authors would also like to thank the program of Natural, Artificial and Cognitive Information Processing (NACIP) and the BioInterfaces International Graduate School (BIF-IGS) at the Karlsruhe Institute of Technology (KIT).

References

[1] L. Saint-Amant and P. Drapeau. “Time course of the development of motor behaviors in the zebrafish embryo”. In: *Journal of Neurobiology* 37.4. S. 622-632. 1998.

[2] Y. Wang, D. Marcato, V. Tirumalasetty, N.K. Kanagaraj, C. Pylatiuk, R. Mikut, R. Peravali and M. Reischl. “An automated experimentation system for the touch-Response quantification of zebrafish larvae”. In: *IEEE Transactions on Automation Science and Engineering* S. 1-13. 2021.

[3] Q. Zhu, Y. Wang, Y. He and X. Hong. “Object tracking with particles weighted by region proposal network”. In: *Multimedia Tools and Applications* 78.9. S. 12083–12101. 2019.

[4] D. Marcato: “An automated and high-throughput photomotor response platform for chemical screens”. In: *37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (Alshut, R.; Breitwieser, H.; Mikut, R.; Strähle, U.; Pylatiuk, C.; Peravali, R.), S. 7728-7731. 2015.

[5] G. Audira, B.P. Sampurna, S. Juniardi, S.T. Liang, Y.H. Lai and C.D. Hsiao. “A simple setup to perform 3D locomotion tracking in zebrafish by using a single camera”. In: *Inventions* 3.1. S. 11. 2018.

[6] M. Schutera, T. Dickmeis, M. Mione, R. Peravali, D. Marcato, M. Reischl, R. Mikut and C. Pylatiuk. “Automated phenotype pattern recognition of zebrafish for high-throughput screening”. In: *Bioengineered* 7.4. S. 261-265. 2016.

[7] R.M. Basnet, D. Zizioli, S. Taweedet, D. Finazzi and M. Memo. “Zebrafish larvae as a behavioral model in neuropharmacology”. In: *Biomedicines* 7.1. S. 23. 2019.

[8] A.A. Popova, D. Marcato, R. Peravali, I. Wehl, U. Schepers and P.A. Levkin. “Fish-microarray: a miniaturized platform for single-embryo high-throughput screenings”. In: *Advanced Functional Materials* 28.3. S. 1703486. 2018.

[9] X. Wang: “Crowdsourced generation of annotated video datasets: a zebrafish larvae dataset for video segmentation and tracking evaluation”. In: *IEEE Life Sciences Conference (LSC)* (Cheng, E.; Burnett, I.S.; Huang, Y.; Wlodkowic, D.), S. 274-277. 2017.

[10] X. Wang: "Automatic tracking of multiple zebrafish larvae with resilience against segmentation errors". In: *IEEE 15th International Symposium on Biomedical Imaging (ISBI)* (Cheng, E.; Burnett, I.S.; Wilkinson, R.; Lech, M.), S. 1157-1160. 2018.

[11] Y.X. Bai, S.H. Zhang, Z. Fan, X.Y. Liu, X. Zhao, X.Z. Feng and M.Z. Sun. "Automatic multiple zebrafish tracking based on improved HOG features". In: *Scientific Reports* 8.1. S. 1-14. 2018.

[12] B.K. Horn and B.G. Schunck. "Determining optical flow". In: *Artificial Intelligence* 17.(1-3). S. 185–203. 1981.

[13] T. Senst, V. Eiselein and T. Sikora. "Robust local optical flow for feature tracking". In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.9. S. 1377-1387. 2012.

[14] H. Richendrfer, S.D. Pelkowski, R.M. Colwill and R. Creton. "On the edge: pharmacological evidence for anxiety-related behavior in zebrafish larvae". In: *Behavioural Brain Research* 228.1. S. 99-106. 2012.

[15] F. Romero-Ferrero, M.G. Bergomi, R.C. Hinz, F.J. Heras and G.G. de Polavieja. "Idtracker. ai: tracking all individuals in small or large collectives of unmarked animals". In: *Nature Methods* 16.2. S. 179-182. 2019.

[16] X. Wang, E. Cheng, I.S. Burnett, Y. Huang and D. Wlodkowic. "Automatic multiple zebrafish larvae tracking in unconstrained microscopic video conditions". In: *Scientific Reports* 7.1. S. 1-8. 2017.

[17] S. Ren, K. He, R. Girshick and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems* 28. S. 91-99. 2015.

[18] O. Ronneberger: "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention* (Fischer, P.; Brox, T.), S. 234-241. 2015.

[19] Y. Wang, N.K. Kanagaraj, C. Pylatiuk, R. Mikut, R. Peravali, M. Reischl. “High-throughput data acquisition platform for multi-larvae touch-response behavior screening of zebrafish”. In: *IEEE Robotics and Automation Letters*. submitted in 2021.

[20] V. Bedell, E. Buglo, D. Marcato, C. Pylatiuk, R. Mikut, J. Stegmaier, W. Scudder, M. Wray, S. Züchner, U. Strähle and R. Peravali. “Zebrafish: a pharmacogenetic model for anesthesia”. In: *Methods in Enzymology*, 602. S. 189-209. 2018.

VectorRL: Interpretable Graph-based Reinforcement Learning for Automated Driving

Christopher Diehl, Tamino Waldeyer, Frank Hoffmann, Torsten Bertram

Institute of Control Theory and Systems Engineering, TU Dortmund
44227 Dortmund, Germany
E-Mail: forename.surname@tu-dortmund.de

1 Introduction

Safe and efficient motion planning and control are crucial components for automated driving. The modeling and prediction of multi-agent interactions in traffic provides a challenge for current decision-making in driving tasks. Often driving policies are designed manually for specific scenarios, which is time-consuming both in development and maintenance. On the other hand, reinforcement learning (RL) learns and improves driving policies in a trial-and-error fashion, with little design and engineering effort. Current RL approaches for automated driving utilize a variety of state-space representations. Hoel et. al [1] propose a feature vector composed of position, speed, and lane information. This representation requires a fixed size input. Huegle et. al [3] employ deep sets [4] to process perceptions of variable dimensionality. However, they do not encode detailed context information. Fixed-size multi-layer grid maps (MLG) [2] easily represent semantic context information in the vehicle's environment. However, they impose a trade-off between computational efficiency, memory consumption, and performance. Recent work [5] in the area of trajectory prediction proposes to encode object and context information as vectors. This comes with the advantage of low discretization errors and computational workload while achieving better performance. To

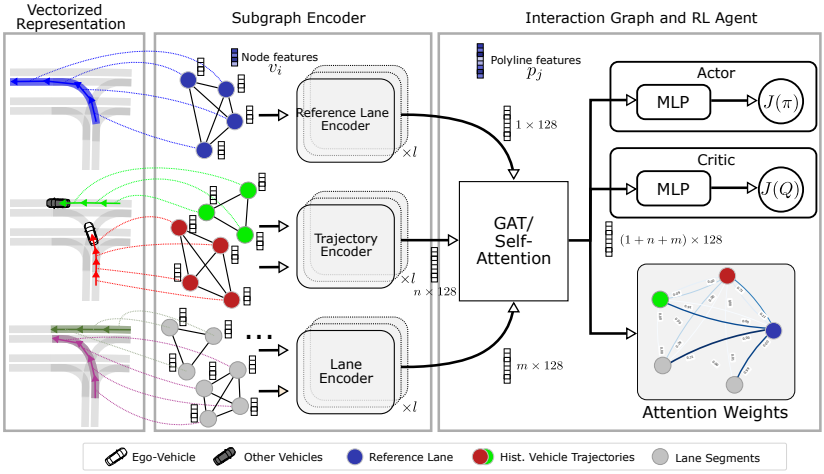


Figure 1: VectorRL system architecture.

overcome the previously described representation complexity, this work proposes a novel graph-based RL approach that relies on a vectorized environment representation. Different attention mechanisms provide insight into the regions and objects relevant to the agent's decision-making. Visualization of the attention states contributes to the interpretability of the learned policy. The graph-based RL approach is evaluated in an urban scenario in a realistic simulation environment. It is compared to several state-of-the-art baselines, which rely on grid-based environment representations. The analysis shows that the graph-based approach outperforms the baselines on all metrics.

2 Vector-based Reinforcement Learning

This section introduces the RL problem formulation and the proposed architecture.

Problem Formulation. Let us model the RL task as a *Markov Decision Process* (MDP), defined by the tuple $\mathscr{M} = (\mathscr{S}, \mathscr{A}, p, r, \gamma)$. $r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ denotes the *reward*. The *policy* $\pi : \mathscr{S} \rightarrow \mathscr{A}$ maps from *states* $\mathbf{s} \in \mathscr{S}$ to a probability distribution over actions $\mathbf{a} \in \mathscr{A}$. The goal is to estimate an *optimal*

policy function $\pi^* = \arg\max_{a \in \mathscr{A}} \sum_{t=1}^{H} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$ that maximizes the finite-horizon cumulative reward over the horizon H. $\gamma \in [0 \dots 1]$ denotes a discount factor.

Approach. The RL problem relies on a graph-based state representation. Figure 1 visualizes the architecture of the proposed approach. The planned route, lane information, and all object trajectories are represented as polylines of length $d \in \mathbb{R}$. Each polyline $P_j \in \mathscr{P}$ with index $j \in \mathbb{N}^+$ is mapped onto $n-1$ equidistant vectors $\mathbf{v}_i \in P_j$ with $\mathbf{v}_i = [\mathbf{d}_i^s, \mathbf{a}_i, j]$. $\mathbf{d}_i^s, \mathbf{d}_i^e \in \mathbb{R}^2$ are the 2-D start and end positions w.r.t. the self-driving vehicles coordinate system with vector index $i \in \mathbb{N}^+$. Further, $\mathbf{a}_i$ is a set of attributes. The route and lane polylines contain width, velocity limit, and intersection information. The attributes of the vehicle polylines characterize its width and length, and orientation. Furthermore, polylines contain a node indicator. Following the work of [5], fully connected sub-graphs encode the corresponding information. Global graph models capture the higher-order interactions between sub-graphs. Whereas the original approach relies on a self-attention (SA) [6] mechanism, our approach in addition investigates graph-attention (GAT) mechanisms [7]. The Soft Actor-Critic (SAC) [8] agent employs the resulting embedding as state-space representation. The action $\mathbf{a}_t$ at time t consists of a normalized continuous acceleration $a \in [0 \dots 1]$, braking signal $b \in [0 \dots 1]$, and steering angle $\delta \in [-1 \dots 1]$. The reward function is similar to the work of [2]:

$$r(s_t, a_t) = \lambda_1 r_{\mathrm{v}} + \lambda_2 r_{\mathrm{lat}} + \lambda_3 r_{\mathrm{col}} + \lambda_4 r_{\mathrm{lane}} - 0.1 \tag{1}$$

r_{col}, r_{lat} and r_{v} penalize collisions as well as deviations from the reference lane and the reference velocity, respectively. r_{lane} and the constant term impose high negative reward, in case the vehicle leaves its lane or stops. The main advantage of the proposed scheme is the ability to visualize the individual attention weights as illustrated in Figure 2. A high color saturation indicates a strong attention of the agent to these polylines. The attention visualization provides insights into the decision-making progress of the SDV, which is crucial for the acceptance of learning-based driving policies. While the agent pays close attention to nearby vehicles during merging (Left image of Figure 2), the attention remote vehicles (Right image of Figure 2) remains low, as these do not compromise the immediate safety of the SDV.

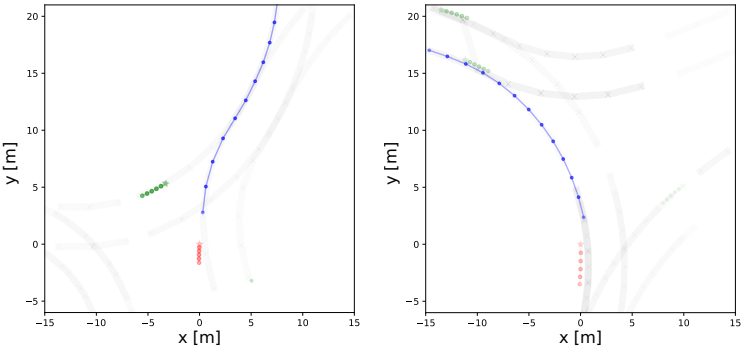


Figure 2: Visualization of the attention weights. The current positions of the SDV (red) and other agents (green) are marked by a star. The history is denoted with circles. Grey indicates the graph of the lane centers. Blue denotes they waypoints of the global route. A high saturation visualizes a high attention weight.

3 Evaluation

This section evaluates the approach in a challenging urban roundabout scenario in the CARLA [9] simulator (Version 0.9.11). The RL algorithm is implemented within Open AI Gym [11]. We compare against multiple baselines considering different metrics. A roundabout scenario in Town 1 based on the work of [2] is constructed for the purpose of policy evaluation.

Baselines. *BEV-OFF*: The approach of [2] first trains an autoencoder (AE) offline. The AE maps a Bird's-eye view (BEV) image to a latent space representation. Then the SAC algorithm is trained based on the latent representation *BEV-ON*: The work of [10] trains the autoencoder together with the RL agent by minimizing a multi-task objective. *VectorRL-SA, VectorRL-GAT*: The proposed approach either employs self-attention or graph-attention for the global graph interaction.

Metrics. *Success Rate* (SR): The proportion of collision-free episodes in which the SDV reaches its final destination. *Progress* (P): The average distance the vehicle travels. *Velocity Tracking Precision* (VTP): Average normalized tracking error of the reference velocity. One indicates the optimal tracking

Table 1: Performance in the roundabout scenario.

Approach	SR [%]	P [m]	VTP	LTE [m]
BEV-OFF	64	83.80 ± 0.81	0.64 ± 0.29	0.40 ± 0.29
BEV-ON	68	91.30 ± 0.79	0.59 ± 0.25	$\mathbf{0.29} \pm 0.28$
VectorRL-GAT	96	108.10 ± 0.80	0.71 ± 0.29	0.41 ± 0.33
VectorRL-SA	**98**	$\mathbf{110.00} \pm 0.80$	$\mathbf{0.73} \pm 0.3$	$\mathbf{0.29} \pm 0.39$

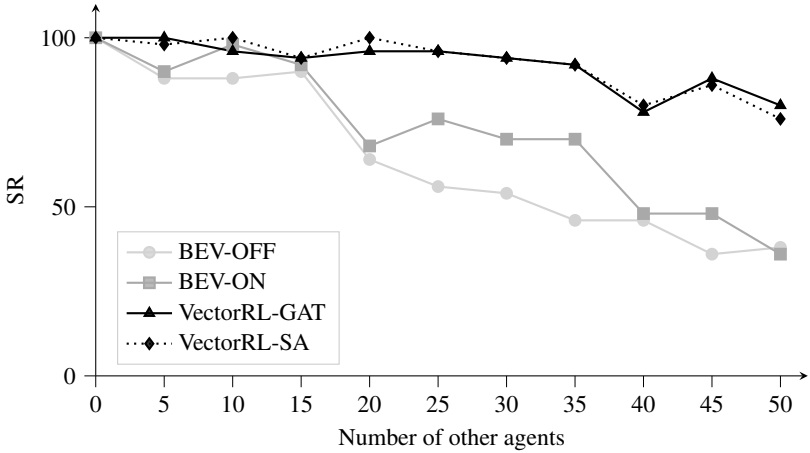


Figure 3: Generalization capabilities using an increasing number of obstacle objects and changing exits.

performance. *Lateral Tracking Error* (LTE) The mean lateral deviation to the reference lane.

Performance. In the first experiment, the SDV is supposed to follow the global route. This route always navigates the agent towards the second exit of the roundabout. During training and testing, 20 other vehicles are spawned at random locations in the vicinity of the roundabout. Testing is performed on 50 randomly generated scenarios, and the results are reported in Table 1. Notice, that the graph-based approaches outperforms the BEV image-based approaches consistently across all metrics.

Generalization. An additional experiment, in which the agent is trained to take the second exit in a scenario with 100 obstacles *spawned over the whole map* evaluates the generalization capability of the state-representations. Note, that this scenario exhibits sparser traffic compared to the original setup. During testing the nominal exit is chosen randomly and moreover the number of agents spawned *in the roundabout* varies as illustrated in Figure 3. The graph-based approaches generalize better to a higher number of agents and achieve a more consistent SR.

4 Conclusion

This work presented a graph-based RL approach for automated driving. The method encodes different semantic information in a vector-based environment representation. The evaluation shows that the proposed approach outperforms other baselines with a grid-based state representation. Future work evaluates graph-based approaches in the offline RL setting, in which the agent learns a policy merely from a static dataset without interactions with the environment.

References

[1] C. Hoel, K. Wolff and L. Laine “Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning”. In: *Intelligent Transportation Systems Conference (ITSC)*. 2018.

[2] J. Chen, B. Yuan and M. Tomizuka “Model-free Deep Reinforcement Learning for Urban Autonomous Driving”. In: *Intelligent Transportation Systems Conference (ITSC)*. 2019.

[3] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling and J. Boedecker “Dynamic Input for Deep Reinforcement Learning in Autonomous Driving”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019.

[4] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola “Deep sets”. In: *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2017.

[5] J. Gao et al. “VectorNet: Encoding hd maps and agent dynamics from vectorized representation”. In: *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[6] A. Vaswani et al. “Attention is All you Need”. In *Proc. of the Advances in Neural Information Processing Systems (NIPS)*. 2017.

[7] P. Veličković et al. “Graph attention networks”. In *Proc. International Conference on Learning Representations (ICLR)*. 2017.

[8] T. Haarnoja, A. Zhou, P. Abbeel and S. Levine. “Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor”. In *Proc. of the International Conference on Machine Learning (ICML)*, 2018.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. “CARLA: An open urban driving simulator”. In *Proc. of the Conference on Robot Learning (CORL)*. 2017.

[10] Yarats, D., A. Zhang, I. Kostrikov, B. Amos, J. Pineau und R. Fergus: “Improving Sample Efficiency in Model-Free Reinforcement Learning from Images”. In*Conference on Artificial Intelligence (AAAI)*. 2021.

[11] G. Brockman et al.“Openai gym”. arxiv:1606.01540. 2016.

Ensemble-based Uncertainty Quantification: Bayesian versus Credal Inference

Mohammad Hossein Shaker[1], Eyke Hüllermeier[2]

[1]Department of Computer Science
Paderborn University
E-Mail: mhshaker@mail.upb.de

[2]Institute of Informatics
LMU Munich
E-Mail: eyke@lmu.de

1 Introduction

A distinction between two different types of uncertainty, *aleatoric* and *epistemic* [6], has received increasing attention in the recent machine learning literature [8, 14]. While the former refers to statistical uncertainty in the sense of inherent randomness, the latter captures systematic uncertainty caused by a lack of knowledge.

In this paper, we consider ensemble-based approaches to uncertainty quantification, i.e., to derive meaningful measures of aleatoric and epistemic uncertainty in a prediction. In this regard, we propose a distinction between three types of uncertainty-aware learning algorithms: probabilistic agents, Bayesian agents, and Levi agents (Section 2). We address the question of how to quantify aleatoric and epistemic uncertainty in a formal way (Section 3), both for Bayesian and Levi agents, and how to approximate such quantities empirically using ensemble techniques (Section 4). Moreover, we analyze the effectiveness of corresponding measures in an empirical study on classification with a reject option (Section 5).

2 Representing Predictive Uncertainty

We consider a standard setting of supervised learning, in which a learner is given access to a set of (i.i.d.) training data $\mathscr{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N \subset \mathscr{X} \times \mathscr{Y}$, where $\mathscr{X}$ is an instance space and $\mathscr{Y}$ the set of outcomes that can be associated with an instance. In particular, we focus on the classification scenario, where $\mathscr{Y} = \{y_1, \ldots, y_K\}$ consists of a finite set of class labels, with binary classification ($\mathscr{Y} = \{0,1\}$) as an important special case.

Suppose a *hypothesis space* $\mathscr{H}$ to be given, where a hypothesis $h \in \mathscr{H}$ is a mapping $\mathscr{X} \longrightarrow \mathbb{P}(\mathscr{Y})$, with $\mathbb{P}(\mathscr{Y})$ the class of probability measures on $\mathscr{Y}$. Thus, a hypothesis maps instances $\boldsymbol{x} \in \mathscr{X}$ to probability distributions on outcomes. The goal of the learner is to induce a hypothesis $h^* \in \mathscr{H}$ with low risk (expected loss)

$$R(h) = \int_{\mathscr{X} \times \mathscr{Y}} \ell(h(\boldsymbol{x}), y) \, dP(\boldsymbol{x}, y) \ , \tag{1}$$

where P is the (unknown) data-generating process (a probability measure on $\mathscr{X} \times \mathscr{Y}$), and $\ell : \mathbb{P}(\mathscr{Y}) \times \mathscr{Y} \longrightarrow \mathbb{R}$ a loss function.

Eventually, one is often interested in the *predictive uncertainty*, i.e., the uncertainty related to the prediction $\hat{y}_q$ for a concrete query instance $\boldsymbol{x}_q \in \mathscr{X}$. Given such a query, different learning methods proceed on the basis of different types of information. Depending on how the uncertainty is represented as a basis for prediction and decision making, we propose to distinguish three types of learning methods, which we call, respectively, probabilistic, Bayesian, and Levi agents.

2.1 Probabilistic Agents

A common practice in machine learning is to consider learners that fully commit to a single hypothesis $\hat{h} \in \mathscr{H}$ and use this hypothesis to make predictions. Such a learner will predict a single probability distribution

$$q = \hat{h}(\boldsymbol{x}_q) = (q_1, \ldots, q_K) \in \mathbb{P}(\mathscr{Y}) \, , \tag{2}$$

where q_k is the probability of the k^{th} class y_k. This prediction is considered as an estimation of the (true) conditional probability $p(y \mid \boldsymbol{x}_q)$. We call a learner of that kind a *probabilistic agent*. Such an agent's uncertainty about the outcome y is purely aleatoric. At the level of the hypothesis space, the agent pretends full certainty, and hence the absence of any epistemic uncertainty about the best hypothesis h.

2.2 Bayesian Agents

Adhering to the principle of (strict) Bayesianism as advocated by statisticians such as De Finetti [4], a *Bayesian agent* will represent its belief about the best hypothesis in terms of a probability distribution on $\mathscr{H}$. Thus, instead of committing to a single hypothesis $\hat{h}$, the agent will assign a probability (density) $p(h)$ to each candidate $h \in \mathscr{H}$. Moreover, belief revision in the light of observed data $\mathscr{D}$ is accomplished by replacing this distribution with the posterior $p(h \mid \mathscr{D})$.

Since every $h \in \mathscr{H}$ gives rise to a probabilistic prediction (2), a Bayesian agent's belief about the outcome y_q is represented by a second-order probability: a probability distribution of probability distributions. If needed, p can be "collapsed" into a single distribution q on $\mathscr{Y}$. This is typically accomplished by inducing q from p (or, more generally, a corresponding measure P) via Bayesian model averaging (BMA):

$$q = \mathrm{bma}(p) = \int_{\mathscr{H}} h(\boldsymbol{x}_q)\, dP(h) \tag{3}$$

2.3 Levi Agents

As a further generalization, instead of committing to a single probability distribution $p \in \mathbb{P}(\mathscr{H})$ on the hypothesis space, the learner may work with a *set* $Q' \subseteq \mathbb{P}(\mathscr{H})$ of such distributions, all of which are deemed plausible candidates. Each distribution $p \in Q'$ again gives rise to a probability distribution

according to (3). Eventually, the relevant representation of the learner is a set of probability distributions

$$Q = \left\{ \mathrm{bma}(p) \,|\, p \in Q' \right\} \subseteq \mathbb{P}(\mathscr{Y}) \,. \tag{4}$$

The reasonableness of taking decisions on the basis of sets of probability distributions (and thus deviating from strict Bayesianism) has been advocated by decision theorists like Levi [11, 12]. Correspondingly, we call a learner of this kind a *Levi agent*. The set Q' (and thereby the set Q) can be produced in different ways, for example as a *credal set* in the context of imprecise probability theory [16].

3 Uncertainty Quantification

According to our discussion so far, different types of learners represent their information or "belief" about the outcome y_q for a query instance $\boldsymbol{x}_q$ in different ways. What we are mainly interested in is a quantification of these learner's epistemic and aleatoric uncertainty, i.e., we are seeking a measure of epistemic uncertainty, EU, and a measure of aleatoric uncertainty, AU.

For ease of notation, we subsequently omit the conditioning on the query instance $\boldsymbol{x}_q$, i.e., all probabilities of outcomes should be understood as conditional probabilities given $\boldsymbol{x}_q$ (for example, we write $p(y)$ instead of $p(y \,|\, \boldsymbol{x}_q)$ and $p(y \,|\, h)$ instead of $p(y \,|\, h, \boldsymbol{x}_q)$). We denote the set of all probability distributions (probability vectors) $q = (q_1, \ldots, q_K) \in [0,1]^K$ by Δ_K.

3.1 Probabilistic Agents: Entropy

The most well-known measure of uncertainty of a single probability distribution is the (Shannon) entropy, which, in the case of discrete $\mathscr{Y}$, is given as

$$S(q) = -\sum_{y \in \mathscr{Y}} q(y) \log_2 q(y) \,, \tag{5}$$

where $0 \log 0 = 0$ by definition. This measure can be justified axiomatically, and different axiomatic systems have been proposed in the literature [3]. It is the most obvious candidate to quantify the aleatoric uncertainty of a probabilistic agent, i.e., $\mathrm{AU}(q) = S(q)$. As such an agent pretends to have precise knowledge of the predictive distribution, the epistemic uncertainty is 0.

3.2 Bayesian Agents: Entropy and Mutual Information

A principled approach to measuring and separating aleatoric and epistemic uncertainty on the basis of classical information-theoretic measures of entropy is proposed by [5]. This approach is developed in the context of neural networks for regression, but the idea as such is more general and can also be applied to other settings. A similar approach was recently adopted by [13].

More specifically, the idea is to exploit the following information-theoretic separation of the total uncertainty in a prediction, measured in terms of the (Shannon) entropy of the predictive posterior distribution (in the case of discrete $\mathscr{Y}$ given by (5)): Considering the outcome as a random variable Y and the hypothesis as a random variable H, we have

$$S\big(Y\big) = I\big(Y,H\big) + S\big(Y \,|\, H\big)\,,$$

where $I(Y,H)$ is the mutual information between hypotheses and outcomes (i.e., the Kullback-Leibler divergence between the joint distribution of outcomes and hypotheses and the product of their marginals):

$$I(Y,H) = \mathbf{E}_{p(y,h)} \left\{ \log_2 \left(\frac{p(y,h)}{p(y)p(h)} \right) \right\}. \tag{6}$$

This term qualifies as a measure of epistemic uncertainty, as it captures the dependency between the probability distribution on $\mathscr{Y}$ and the (uncertain) hypothesis h.

Finally, the conditional entropy is given by

$$S\big(Y\,|\,H\big) = \mathbf{E}_{p(h\,|\,\mathscr{D})}\left\{S\big(p(y\,|\,h)\big)\right\} = \tag{7}$$
$$= -\int_{\mathscr{H}} p(h\,|\,\mathscr{D}) \left(\sum_{y\in\mathscr{Y}} p(y\,|\,h)\log_2 p(y\,|\,h)\right) dh$$

This measure qualifies as a measure of aleatoric uncertainty: By fixing a hypothesis $h \in \mathscr{H}$, the epistemic uncertainty is essentially removed. Thus, the entropy $S(p(y\,|\,h))$, i.e., the entropy of the conditional distribution on $\mathscr{Y}$ predicted by h (for the query $\boldsymbol{x}_q$) is a natural measure of the aleatoric uncertainty. However, since h is not precisely known, aleatoric uncertainty is measured in terms of the expectation of this entropy with regard to the posterior probability $p(h\,|\,\mathscr{D})$.

3.3 Levi Agents: Uncertainty Measures for Credal Sets

In the case of a Levi agent, uncertainty degrees ought to be specified for a set of probability distributions $Q \subseteq \Delta_K$. In the literature, such sets are also referred to as *credal sets* [16]. There is quite some work on defining uncertainty measures for credal sets and related representation, such as Dempster-Shafer evidence theory [15], asking for a generalized representation

$$\mathrm{U}(Q) = \mathrm{AU}(Q) + \mathrm{EU}(Q)\,, \tag{8}$$

where U is a measure of total (aggregate) uncertainty, AU a measure of aleatoric uncertainty (a generalization of the Shannon entropy), and EU a measure of epistemic uncertainty.

As for the latter, the following generalization of the Hartley measure, a well-established measure of uncertainty for sets, has been proposed by various authors [2]:

$$\mathrm{GH}(Q) = \sum_{A\subseteq\mathscr{Y}} \mathrm{m}_Q(A)\,\log(|A|)\,, \tag{9}$$

where $\mathrm{m}_Q : 2^{\mathscr{Y}} \longrightarrow [0,1]$ is the Möbius inverse of the capacity function $\nu : 2^{\mathscr{Y}} \longrightarrow [0,1]$ defined by

$$\nu_Q(A) = \inf_{q \in Q} q(A) \tag{10}$$

for all $A \subseteq \mathscr{Y}$, that is,

$$\mathrm{m}_Q(A) = \sum_{B \subseteq A} (-1)^{|A \setminus B|} \nu_Q(B)\,.$$

This measure is "well-justified" in the sense of possessing a sound axiomatic basis and obeying a number of desirable properties [9].

Regarding $\mathrm{AU}(Q)$, an extension of Shannon entropy, "well-justified" in the same sense, has not been found so far. As a possible way out, it was suggested to define a meaningful measure of total or aggregate uncertainty $\mathrm{U}(Q)$, and to *derive* a generalized measure of aleatoric uncertainty via *disaggregation*, i.e., in terms of the difference between this measure and the measure of epistemic uncertainty (Hartley), or vice versa, to derive a measure of epistemic uncertainty as the difference between total uncertainty and a meaningful measure of aleatoric uncertainty.

The upper and lower Shannon entropy play an important role in this regard:

$$S^*(Q) = \max_{q \in Q} S(q)\,, \quad S_*(Q) = \min_{q \in Q} S(q) \tag{11}$$

Based on these measures, the following disaggregations of total uncertainty (8) have been proposed [1]:

$$S^*(Q) = \big(S^*(Q) - \mathrm{GH}(Q)\big) + \mathrm{GH}(Q) \tag{12}$$

$$S^*(Q) = S_*(Q) + \big(S^*(Q) - S_*(Q)\big) \tag{13}$$

In both cases, upper entropy serves as a measure of total uncertainty $U(Q)$, which is again justified on an axiomatic basis. In the first case, the generalized Hartley measure is used for quantifying epistemic uncertainty, and aleatoric uncertainty is obtained as the difference between total and epistemic uncertainty. In the second case, lower entropy is used as a (well-justified) measure

of aleatoric uncertainty, and epistemic uncertainty is derived in terms of the difference between upper and lower entropy.

4 Ensemble-Based Uncertainty Quantification

Ensemble-based approaches to uncertainty quantification have recently been advocated by several authors [10]. Adopting a Bayesian perspective, the variance of the predictions produced by an ensemble is inversely related to the "peakedness" of a posterior distribution $p(h \mid \mathscr{D})$. Thus, an ensemble can be considered as an approximate representation of a second-order distribution $p(h \mid \mathscr{D})$ in a Bayesian setting.

Given this motivation, we address the question of how the measures of uncertainty introduced above can be realized by means of ensemble techniques, i.e., how they can be computed (approximately) on the basis of a finite ensemble of hypotheses $H = \{h_1, \ldots, h_M\}$, which can be thought of as a sample from the posterior distribution $p(h \mid \mathscr{D})$. More specifically, we consider this question for the case of a Bayesian and a Levi agent. The following notation will be used:

- $p_{k,m} = p(y_k \mid h_m, \boldsymbol{x}_q)$ is the probability predicted for class y_k by hypothesis h_m for query $\boldsymbol{x}_q$, i.e., $(p_{1,m}, \ldots, p_{K,m}) = p(\cdot \mid h_m, \boldsymbol{x}_q)$;
- $l_m = p(\mathscr{D} \mid h_m)$ denotes the likelihood of h_m;
- $q_k = \sum_{m=1}^{M} p(h_m \mid \mathscr{D})\, p_{k,m}$ is the posterior probability estimate for class y_k produced by the ensemble through weighted averaging.

4.1 Bayesian Agents

Recalling the approach presented in Section 3.2, it is obvious that (6) and (7) cannot be computed efficiently, because they involve an integration over the

hypothesis space $\mathscr{H}$. Based on an ensemble $H = \{h_1, \ldots, h_M\}$, an approximation of (7) can be obtained by

$$\mathrm{AU}(\boldsymbol{x}_q) = -\sum_{m=1}^{M} p(h_m \,|\, \mathscr{D}) \sum_{k=1}^{K} p_{k,m} \log_2 p_{k,m}\,, \tag{14}$$

an approximation of total uncertainty, i.e., Shannon entropy (5), by

$$\mathrm{U}(\boldsymbol{x}_q) = -\sum_{k=1}^{K} q_k \log_2 q_k\,, \tag{15}$$

and finally an approximation of (6) by $\mathrm{EU}(\boldsymbol{x}_q) = \mathrm{U}(\boldsymbol{x}_q) - \mathrm{AU}(\boldsymbol{x}_q)$. Assuming a uniform prior, which is quite natural in the case of ensembles, the posterior probability of hypotheses can be obtained from $p(h_m \,|\, \mathscr{D}) \propto l_m$.

4.2 Levi Agents

How could the idea of a Levi agent be implemented on the basis of an ensemble approach? As explained above, credal inference yields a set of probability estimates, each of which is obtained by Bayesian model averaging according to a different prior. Thus, instead of assuming a uniform prior $p(h_m) \equiv 1/M$, we should now proceed from a set of priors. A simple example is the family

$$S_\delta = \left\{ \boldsymbol{s} = (s_1, \ldots, s_M) \,|\, \frac{1}{\delta M} \leq s_m \leq \frac{\delta}{M}, \sum_{m=1}^{M} s_m = 1 \right\} \tag{16}$$

of distributions δ-close to uniform, where $\delta \geq 1$ is a (hyper-)parameter. Thus, compared to the uniform prior, the probability of a single hypothesis can now be decreased or increased by a factor of at most δ. The set of posterior probabilities is then given by

$$\left\{ p(h_m \,|\, \mathscr{D}) = \frac{s_m\, l_m}{\sum_{i=1}^{M} s_i\, l_i} \,|\, \boldsymbol{s} \in S_\delta \right\},$$

and hence the credal set on $\mathscr{Y}$ by

$$Q = \left\{ q = \sum_{m=1}^{M} s_m l_m h_m(\boldsymbol{x}_q) / \sum_{m=1}^{M} s_m l_m \,|\, \boldsymbol{s} \in S_\delta \right\}$$

To compute the decompositions (12) and (13) for Q, we need to compute the measures S^*, S_*, GH. According to (9), the computation of the measure GH requires the capacity (10), i.e., the lower probability $\nu_Q(A)$ of each subset of classes $A \subseteq \mathscr{Y}$. For $A = \{y_j\}_{j \in J}$ identified by an index set $J \subseteq [K]$, the latter is given by

$$\nu_Q(A) = \min_{q \in Q} q(A) = \min_{\boldsymbol{s} \in S_\delta} \frac{\sum_{j \in J} \sum_{m=1}^{M} s_m l_m p_{j,m}}{\sum_{m=1}^{M} s_m l_m} .$$

Thus, finding $\nu_Q(A)$ comes down to solving a linear-fractional programming problem (for which standard solvers can be used). Moreover, finding S^* comes down to solving

$$\max_{\boldsymbol{s} \in S_\delta} \sum_{k=1}^{K} \frac{\sum_{m=1}^{M} s_m l_m p_{k,m}}{\sum_{m=1}^{M} s_m l_m} \log \frac{\sum_{m=1}^{M} s_m l_m p_{k,m}}{\sum_{m=1}^{M} s_m l_m} ,$$

and similarly for S_* (with max replaced by min).

5 Experiments

Predicted uncertainties are often evaluated indirectly, for example by assessing their usefulness for improved prediction and decision making, because the data does normally not contain information about any sort of "ground truth" uncertainties. Here, we conducted such an evaluation by producing *accuracy-rejection curves*, which depict the accuracy of a predictor as a function of the percentage of rejections [7]: a learner, which is allowed to abstain on a certain percentage p of predictions, will predict on those $(1-p)$ % on which it feels most certain. Being able to quantify its own uncertainty well, it should improve its accuracy with increasing p, hence the accuracy-rejection curve should be monotone increasing (unlike a flat curve obtained for random abstention).

5.1 Data Sets and Experimental Setting

We compare the Bayesian agent with different variants of the Levi agent in terms of their ability to quantify aleatoric and epistemic uncertainty. The Bayesian agent quantifies these uncertainties according to (14) and (15). The Levi agent is implemented as described in Section 4.2. Uncertainty is quantified based on the generalized Hartley measure (Levi-GH) according to (12), or based on upper and lower entropy (Levi-Ent) according to (13). In this experiment, we set the hyper-parameter $\delta = 2$.

We performed experiments on various well-known data sets from the UCI repository[1]. The data sets are randomly split into 70% for training and 30% for testing, and accuracy-rejection curves are produced on the latter. Each experiment is repeated and averaged over 100 runs. We create ensembles using the Random Forest Classifier from SKlearn. The number of trees within the ensemble is set to 10. Each tree can grow to a maximum of 10 splits. Probabilities are estimated by (Laplace-corrected) relative frequencies in the leaf nodes of a tree.

5.2 Results

Fig. 1 shows the accuracy-rejection curves for the different learners, separated into epistemic uncertainty (EU) in the left, aleatoric uncertainty (AU) in the middle, and total uncertainty (TU) on the right column. Due to space restrictions, we only show the results for five data sets, noting that the results for other data sets are very similar. The following observations can be made.

- As suggested by the shape of the accuracy-rejection curves, both the Bayesian and the Levi agent perform quite well in general. On total uncertainty, they are basically indistinguishable, which is almost a bit surprising, given that these uncertainties are quantified on the basis of different principles.

[1] `http://archive.ics.uci.edu/ml/index.php`

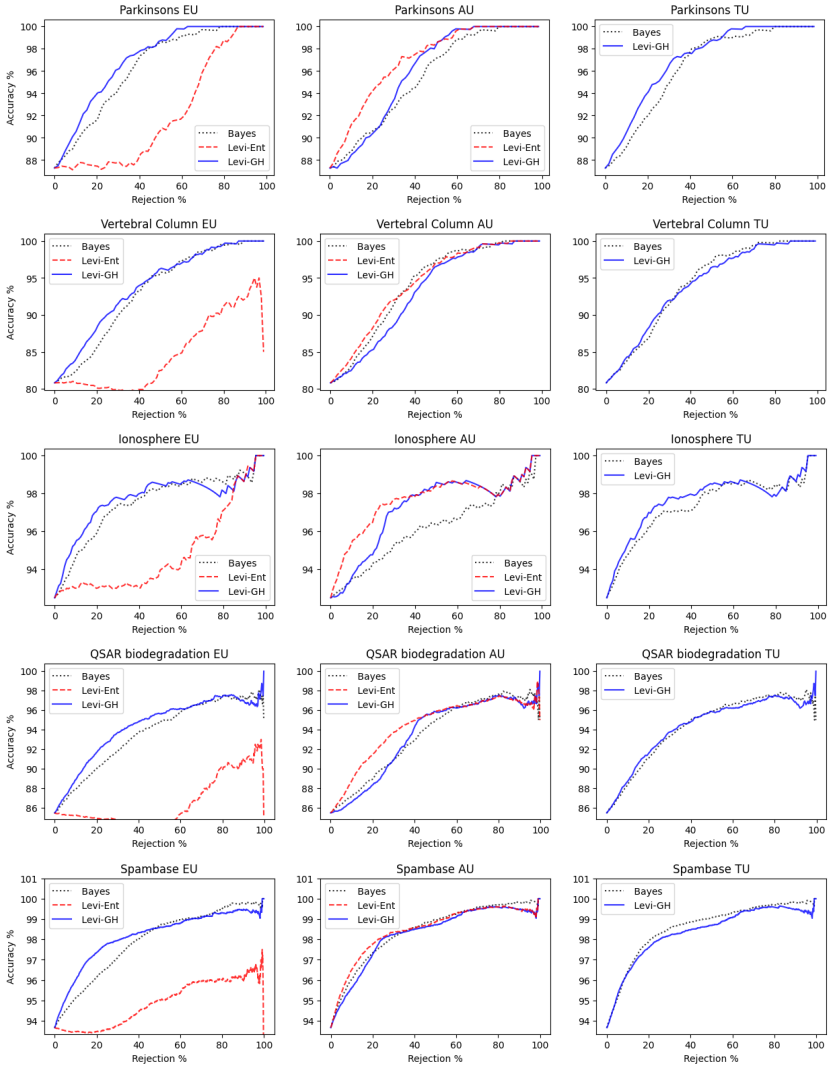


Figure 1: Accuracy-rejection curves for the Bayesian and the Levi agent.

- Levi-GH seems to have an advantage over the Bayesian agent on epistemic uncertainty, providing evidence for the generalized Hartley measure as a reasonable measure of epistemic uncertainty.
- Levi-Ent seems to have an advantage over the Bayesian agent on aleatoric uncertainty, providing evidence for the lower entropy as a reasonable measure of aleatoric uncertainty.
- The "derived" measures, $S^*(Q) - \mathrm{GH}(Q)$ for aleatoric and $S^*(Q) - S_*(Q)$ for epistemic uncertainty, both perform quite poorly.

6 Conclusion

We proposed a distinction between different types of uncertainty-aware learning algorithms, discussed measures of total, aleatoric and epistemic uncertainty of such learners, and developed ensemble-methods for approximating these measures. In particular, we compared the classical Bayesian approach with what we call a Levi agent, which makes predictions in terms of credal sets.

In an experimental study on uncertainty-based abstention, both methods show strong performance. While the Bayesian and the Levi agent are on a par for total uncertainty, improvements of the Bayesian approach can be achieved for the two types of uncertainty separately: The generalized Hartley measure appears to be superior for epistemic and the lower entropy for aleatoric uncertainty quantification. On the other side, the alternative measures of aleatoric and epistemic uncertainty obtained through disaggregation perform quite poorly. These results can be seen as an interesting empirical complement to the theoretical (axiomatic) research on uncertainty measures for credal sets.

In future work, we seek to further deepen our understanding of ensemble-based uncertainty quantification and elaborate on the approach presented in this paper. An interesting problem, for example, is the tuning of the (hyper-)parameter δ in (16), for which we simply took a default value in the experiments. Obviously, this parameter has an important influence on the uncertainty of the Levi agent. Besides, we also plan to develop alternative approaches for constructing

ensembles. Last but not least, going beyond abstention and accuracy-rejection curves, we plan to apply and analyze corresponding methods in the context of other types of uncertainty-aware decision problems.

References

[1] J. Abellan, J. Klir, and S. Moral. Disaggregated total uncertainty measure for credal sets. *International Journal of General Systems*, 35(1), 2006.

[2] J. Abellan and S. Moral. A non-specificity measure for convex sets of probability distributions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 8:357–367, 2000.

[3] I. Csiszár. Axiomatic characterizations of information measures. *Entropy*, 10:261–273, 2008.

[4] B. De Finetti. Foresight: It's logical laws, it's subjective sources. In H.E. Kyburg and H.E. Smokler, editors, *Studies in Subjective Probability*. R.E. Krieger, New York, 1980.

[5] S. Depeweg, J.M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *Proc. ICML*, Stockholm, Sweden, 2018.

[6] S.C. Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering and System Safety*, 54(2–3):217–223, 1996.

[7] J. Hühn and E. Hüllermeier. FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems*, 17(1):138–149, 2009.

[8] A. Kendall and Y. Gal. What uncertainties do we need in Bayesian deep learning for computer vision? In *Proc. NIPS*, pages 5574–5584, 2017.

[9] G.J. Klir and M. Mariano. On the uniqueness of possibilistic measure of uncertainty and information. *Fuzzy Sets and Systems*, 24(2):197–219, 1987.

[10] B. Lakshminarayanan, A. Pritzel, C. and Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proc. NeurIPS*, 2017.

[11] I. Levi. On indeterminate probabilities. *Journal of Philosophy*, 71:391–418, 1974.

[12] I. Levi. *The Enterprise of Knowledge*. MIT Press, Cambridge, 1980.

[13] A. Mobiny, H.V. Nguyen, S. Moulik, N. Garg, and C.C. Wu. DropConnect is effective in modeling uncertainty of Bayesian networks. *CoRR*, abs/1906.04569, 2017.

[14] R. Senge, S. Bösner, K. Dembczynski, J. Haasenritter, O. Hirsch, N. Donner-Banzhoff, and E. Hüllermeier. Reliable classification: Learning classifiers that distinguish aleatoric and epistemic uncertainty. *Inf. Sciences*, 255:16–29, 2014.

[15] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[16] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, 1991.

Approximation der zulässigen Parametermenge bei der Bounded-Error-Schätzung durch ein Ray-Shooting-Verfahren

Felix Wittich[1], Andreas Kroll[2]

[1,2]Fachgebiet Mess- und Regelungstechnik, Universität Kassel
Mönchebergstr. 7, 34125 Kassel
E-Mail: {felix.wittich; andreas.kroll}@mrt.uni-kassel.de

1 Einführung

Sollen bei der datengetriebenen Modellbildung Unsicherheiten der zu schätzenden Parameter quantifiziert werden, wird typischerweise eine probabilistische Beschreibung der Unsicherheit herangezogen. Dabei wird eine bekannte Wahrscheinlichkeitsdichtefunktion (WDF) vorausgesetzt, bzw. es werden viele Daten für die Schätzung der WDF benötigt, was bei praktischen Anwendungsfällen meist nicht gegeben ist.

Ein alternativer Ansatz ist hier die mengenbasierte Bounded-Error-(BE-)-Parameterschätzung [1]. Dabei wird die Annahme getroffen, dass der Prädiktionsfehler in einem Intervall mit garantierten Schranken liegt und gegeben der Annahmen wird die zulässige Parametermenge (Feasible Parameter Set, FPS) bestimmt [2]. Für einen Anwendungsfall der Randschichtprädiktion beim Hartdrehen mit Takagi-Sugeno-Multimodellen konnten BE-Verfahren in [3, 4] erfolgreich eingesetzt werden. Für einen Einsatz in der datengetriebenen Modellbildung mit einer großen Parameteranzahl n ergibt sich allerdings ein hoher Rechenaufwand für die exakte Bestimmung der zulässigen Parametermenge. Um diesem Problem bei der BE-Parameterschätzung zu begegnen, können Approximationsverfahren eingesetzt werden. Verbreitet werden dabei Hyperellipsoide als Kompromiss zwischen kompakter Beschreibung und Flexibilität

eingesetzt [5]. Allerdings kommt es hierbei zu starken Überschätzungen des FPS, da die Approximationen zwar in jedem Schritt optimal sind, global aber nicht. Somit kann nicht garantiert werden, dass die Fehlerschranken durch das approximierte Modell eingehalten werden.

Aus den oben genannten Problemen ist die Idee für ein neuartiges Verfahren zur approximativen Bestimmung der zulässigen Parametermenge entstanden. Dabei wird das aus der Computergrafik bekannte Ray-Tracing bzw. im n-Dimensionalen in der algorithmischen Geometrie als Ray-Shooting bekannte Verfahren der Strahlenverfolgung eingesetzt. Die Schnittpunkte des Ray-Shootings in konvexen Polytopen ergeben sich dabei als Lösung eines linearen Optimierungsproblems [6]. Durch die Strahlenverfolgung ergibt sich ein samplingbasierter Ansatz, der die Geometrie des konvexen Polytops des FPS ausnutzt. Das Ray-Shooting identifiziert garantiert eine Untermenge der wahren Parametermenge, wodurch die festgelegten Fehlerschranken immer eingehalten werden. In dieser Arbeit wird die Idee des Verfahren konzeptionell vorgestellt und anhand einer Fallstudie demonstriert.

2 Methoden

2.1 Bounded-Error-Fehlerbeschreibung

In diesem Beitrag wird die Parameterschätzung aus einer mengentheoretischen Sichtweise betrachtet. Die Idee der Bounded-Error-Schätzung kann dabei wie folgt beschrieben werden. Es soll ein parametrisches Model $\hat{y} = f(\boldsymbol{\theta}, \boldsymbol{x})$, mit dem Parametervektor $\boldsymbol{\theta} \in \mathbb{R}^n$ bestimmt werden, um den funktionalen Zusammenhang zwischen einer Ausgangsgröße $y \in \mathbb{R}$ und Eingangsgrößen $\boldsymbol{x} \in \mathbb{R}^{n_p}$ herzustellen. Das Model soll dabei aus einem Datensatz $Z^N = \{\boldsymbol{x}(k), y(k)\}, k = 1, \ldots, N\}$ gelernt werden. Das Ziel bei der BE-Schätzung ist dabei die Bestimmung einer zulässigen Untermenge des Parameterraums $\mathbb{S}_{\text{FPS}} \subseteq \mathbb{P}$, die zu einer Modellausgabe führt, die die Annahme einer spezifizierten Fehlerschrankenmenge $\mathbb{E}$ erfüllt:

$$\mathbb{S}_{\text{FPS}} = \{\boldsymbol{\theta} \in \mathbb{R}^n | \boldsymbol{e}(\boldsymbol{\theta}) \in \mathbb{E}\} \tag{1}$$

Diese Menge wird als zulässige Parametermenge bezeichnet. Der Prädiktionsfehler wird als $e(k,\boldsymbol{\theta}) = y(k) - \hat{y}(k,\boldsymbol{\theta})$ definiert. Die Fehlerschranken werden als Intervalle beschrieben:

$$e(k,\boldsymbol{\theta}) \in [e_{\min}(k), e_{\max}(k)], k = 1,\ldots,N \tag{2}$$

und können individuell für jeden Punkt festgelegt werden. Gewöhnlich werden dabei die selben symmetrischen Fehlerschranken $\delta = (e_{\max} - e_{\min})/2$ für jedes k angenommen. Somit folgt für (1):

$$\mathbb{S}_{\text{FPS}} = \{\boldsymbol{\theta} \in \mathbb{R}^n | y(k) - \delta \leq \hat{y}(k,\boldsymbol{\theta}) \leq y(k) + \delta \; \forall k\} \tag{3}$$

2.2 Bounded-Error-Schätzung für LiP-Modelle

Für Modelle, die linear-in-den-Parametern (LiP) sind, ist $\mathbb{S}_{\text{FPS}}$ ein Polytop, wenn $N \geq d_{\text{ind}} > n$ unabhängige lineare Ungleichungen existieren. Ist ein LiP-Modell $\hat{y} = \boldsymbol{\varphi}^T\boldsymbol{\theta}$ mit dem Regressionsvektor $\boldsymbol{\varphi} \in \mathbb{R}^n$ und sind N Beobachtungen gegeben, schränkt jedes der N Intervalle

$$y(k) - \delta \leq \boldsymbol{\varphi}(k)^T\boldsymbol{\theta} \leq y(k) + \delta \tag{4}$$

das FPS durch zwei Ungleichungen im $\mathbb{R}^n$ ein. Insgesamt existieren also $2N$ Ungleichungen. Ein Polytop $\mathscr{P}$ kann durch die eingrenzenden Halbräume beschrieben werden:

$$\mathscr{P} = \{\boldsymbol{\theta} \in \mathbb{R}^{n_\theta} | \boldsymbol{\Phi}\boldsymbol{\theta} \leq \boldsymbol{Y}\} \tag{5}$$

mit

$$\boldsymbol{Y} = \begin{bmatrix} -y(1)+\delta \\ y(1)+\delta \\ \vdots \\ -y(N)+\delta \\ y(N)+\delta \end{bmatrix}, \boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}(1)^T \\ \boldsymbol{\varphi}(1)^T \\ \vdots \\ \boldsymbol{\varphi}(N)^T \\ \boldsymbol{\varphi}(N)^T \end{bmatrix}, \boldsymbol{\theta} = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{n_\theta} \end{bmatrix}, \tag{6}$$

somit folgt: $\boldsymbol{\Phi} \in \mathbb{R}^{2N \times n_\theta}$, $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$, und $\boldsymbol{Y} \in \mathbb{R}^{2N}$.

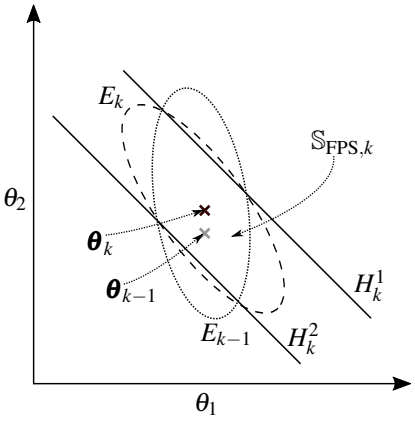


Bild 1: REOB-Verfahren (Dargestellt ist die k-te Iteration)

2.3 Rekursive Approximation des FPS mit Ellipsoiden

In der Literatur werden verschieden Methoden zur approximativen Schätzung des FPS vorgeschlagen. Dazu wird das exakte FPS durch Geometrien wie Parallelotope, Zonotope oder Ellipsoide angenähert. Ein verbreiteter Ansatz ist dabei die rekursive Schätzung mit begrenzenden Ellipsoiden (Recursive Ellipsoidal Outer Bounding, REOB). Dabei wird eine äußere Approximation $\hat{\mathbb{S}}_{\mathrm{FPS}} \supset \mathbb{S}_{\mathrm{FPS}}$ durch die rekursive Berechnung der Schnittmengen von volumenminimalen Ellipsoiden E_k in der k-ten Iteration mit dem k-ten Paar der Halbebenen H_k^1 und H_k^2 berechnet, wie in Bild 1 illustriert. Eine ausführliche Beschreibung ds Verfahrens ist in [5] zu finden. Neben den Verfahren zur äußeren Einschränkung des FPS existieren auch rekursive Verfahren zur inneren Einschränkung, die allerdings zu einer starken Unterschätzung tendieren [2].

2.4 Lokal-affine Multi-Modell

Als parametrischer Modellansatz werden im Folgenden lokal-affine Multimodelle betrachtet. Diese zeichnen sich durch eine hohe Modellflexibilität bei

kompakter Modellstruktur aus und können für die Identifikation nichtlinearer Systeme verwendet werden.

Dabei werden $c \in \mathbb{N}_+$ Teilmodelle $\hat{y}_j = f(\boldsymbol{\theta}_{j,\mathrm{LM}}, \tilde{\boldsymbol{\varphi}}) : \mathbb{R}^{n_p} \to \mathbb{R}$, durch Fuzzy-Basisfunktionen $\phi_j(\boldsymbol{z}) : \mathbb{R}^{n_z} \to [0,1]$ überlagert, welche von Schedulingvariablen $\boldsymbol{z} = [z_1 \dots z_{n_z}]^T \in \mathbb{R}^{n_z}$ abhängen:

$$\hat{y}(\boldsymbol{z}, \boldsymbol{\theta}, \tilde{\boldsymbol{\varphi}}) = \sum_{j=1}^{c} \phi_j(\boldsymbol{\theta}_{\mathrm{MF}}, \boldsymbol{z}) \hat{y}_j(\boldsymbol{\theta}_{j,\mathrm{LM}}, \tilde{\boldsymbol{\varphi}}). \tag{7}$$

Dabei werden affine Teilmodelle verwendet:

$$\hat{y}_j(\boldsymbol{\theta}_{j,\mathrm{LM}}, \tilde{\boldsymbol{\varphi}}) = \sum_{r=0}^{n} \theta_{j,r,\mathrm{LM}} \cdot \tilde{\varphi}_r = \boldsymbol{\theta}_{j,\mathrm{LM}} \cdot \tilde{\boldsymbol{\varphi}}, \tag{8}$$

mit dem r-ten Element $\tilde{\varphi}_r$ des Regressionsvektors

$$\tilde{\boldsymbol{\varphi}} = [1\, x_1 \dots x_{n_p}]^T \tag{9}$$

und dem r-ten Element $\theta_{j,r,\mathrm{LM}}$ des entsprechenden lokalen Parametervektor $\boldsymbol{\theta}_{j,\mathrm{LM}} \in \mathbb{R}^n$. Das Multimodell (7) kann dann wie folgt geschrieben werden:

$$\hat{y} = \bar{\boldsymbol{\varphi}}^T \boldsymbol{\theta}_{\mathrm{LM}} \tag{10}$$

mit dem erweiterten Regressionsvektor

$$\bar{\boldsymbol{\varphi}} = \left[\phi_1 \;\; \phi_1 x_1 \;\; \dots \;\; \phi_1 x_{n_p} | \dots | \phi_c \;\; \phi_c x_1 \;\; \dots \;\; \phi_c x_{n_p} \right]^T \tag{11}$$

und dem Vektor der lokalen Modellparameter

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{LM}}^T &= \left[a_{0,1} \;\; a_{1,1} \;\; \dots \;\; a_{n_p,1} | \dots | a_{0,c} \;\; a_{1,c} \;\; \dots \;\; a_{n_p,c} \right] \\ &= \left[\boldsymbol{\theta}_{1,\mathrm{LM}}^T \dots \boldsymbol{\theta}_{c,\mathrm{LM}}^T \right] \in \mathbb{R}^{n \times c}. \end{aligned} \tag{12}$$

Dabei sind

$$\phi_j(\boldsymbol{z}) = \frac{\mu_j(\boldsymbol{z})}{\sum_{m=1}^{c} \mu_m(\boldsymbol{z})}, \tag{13}$$

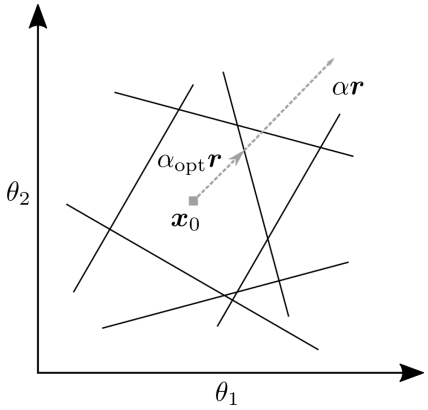


Bild 2: Ray-Shooting im konvexen Polytop

die Fuzzy-Basisfunktionen mit der Fuzzy-C-Means-Zugehörigkeitsfunktion

$$\mu_j(\boldsymbol{z}) = \left[\sum_{i=1}^{c} \left(\frac{\|\boldsymbol{z} - \boldsymbol{v}_j\|_2}{\|\boldsymbol{z} - \boldsymbol{v}_i\|_2} \right)^{\frac{2}{v-1}} \right]^{-1}. \tag{14}$$

Die Partitionierungsparamter $\boldsymbol{v}_j, \boldsymbol{v}_i \in \mathbb{R}^{n_z}$ werden im Parametervektor $\boldsymbol{\theta}_{\text{MF}} = [\boldsymbol{v}_1^T, \ldots, \boldsymbol{v}_c^T]^T$ aggregiert. Im Folgenden wird für die Schedulingvariable $\boldsymbol{z} = \boldsymbol{x}$ angenommen.

2.5 Ray-Shooting in konvexen Polytopen

Das Prinzip des Ray-Shooting wird in Bild 2 dargestellt. Ein Strahl wird ausgehend von einem initialen Punkt $\boldsymbol{x}_0 \in \mathbb{R}_n$ im Inneren eines durch M Halbebenen $H_k,\ k = 1, \ldots, M$ beschriebenen konvexen Polytops gesendet. Das Ziel ist es, den Schnittpunkt $\alpha_{\text{opt}} \boldsymbol{r} + \boldsymbol{x}_0$ zwischen dem Strahl und der ersten Halbebene, die getroffen wird, zu bestimmen. Die Richtung des Strahls wird dabei durch den Vektor $\boldsymbol{r} \in \mathbb{R}^n$ und die Länge des Strahls durch $\alpha \in \mathbb{R}^+$ festgelegt.

Der Strahl geht von einem initialen Punkt $\boldsymbol{x}_0 \in \mathbb{R}_n$ im inneren des Polytops aus. Um α_{opt} zu bestimmen wird hierzu das lineare Optimierungsproblem:

$$\begin{aligned} \alpha_{\text{opt}} = \quad & \max \quad \alpha \\ & \text{u.d.N.} \quad \boldsymbol{A}(\alpha \boldsymbol{r} + \boldsymbol{x}_0) \leq \boldsymbol{b} \end{aligned} \tag{15}$$

gelöst, das sich in polynomialer Zeit berechnen lässt.

2.6 Ray-Shooting für die Approximation des FPS

Die Idee besteht darin, das Ray-Shooting in konvexen Polytopen für eine innere Approximation des FPS bei LiP-Modellen einzusetzen. Für die Bounded-Error-Schätzung wird das Ray-Shooting im Raum der Parameter $\boldsymbol{\theta} \in \mathbb{R}_n$ durchgeführt, um Randpunkte des durch (5) gegebenen Polytops zu finden. Als Suchgebiet wird die achsparallele begrenzende Box gewählt, also das (Hyper-)Rechteck, das durch $[\theta_{\text{min},1}, \theta_{\text{max},1}] \times [\theta_{\text{min},2}, \theta_{\text{max},2}] \times \cdots \times [\theta_{\text{min},n}, \theta_{\text{max},n}]$ gegeben ist. Dieses Rechteck lässt sich durch das Lösen von $2n$ linearen Programmen:

$$\theta_{\text{min,k}} = \min \quad \theta_{\text{k}} \tag{16}$$

$$\theta_{\text{max,k}} = \max \quad \theta_{\text{k}} \tag{17}$$

$$\text{u.d.N.} \quad \boldsymbol{A}\boldsymbol{\theta} \leq \boldsymbol{b} \tag{18}$$

für $k = 1, \ldots, n$ finden [1]. Es werden Parametervektoren zufällig gleichverteilt innerhalb der achsparallelen Box erzeugt und mit (4) auf Zulässigkeit überprüft. Die zulässigen Punkte werden dann als initiale Punkte $\boldsymbol{\theta}_0$ verwendet. Um eine gute Abdeckung im Parameterraum zu erreichen, müssen bei hoher Parameteranzahl n entsprechend viele Punkte initialisiert werden, da deren Dichte exponentiell mit zunehmendem n sinkt. Die Richtungsvektoren $\boldsymbol{r}$, die die Suchrichtungen vorgeben, werden ebenfalls zufällig gleichverteilt gewählt. Dazu werden diese als Vektoren vom Ursprung zur Oberfläche einer n-dimensionalen Einheitskugel festgelegt. Die mit dem Ray-Shooting aus Abschnitt 2.5 bestimmten Randpunkte bilden dann die konvexe Hülle der approximierten zulässigen Parametermenge $\text{FPS}_{\text{RS-BE}}$.

3 Fallstudie

Die RS-BE-Methode soll anhand einer akademischen Fallstudie demonstriert werden. Hierzu wurden $N = 50$ Trainingsdaten mit einer einfachen nichtlinearen Funktion $y = f(x) = x^2$ gleichverteilt im Intervall $[-1;1]$ erzeugt. Der Ausgang $\tilde{y} = y + d$ wurde mit gleichverteiltem Rauschen $d \sim \mathscr{U}(-0,05;0,05)$ beaufschlagt. Für die Modellierung wurden TS-Multi-Modelle mit $\nu = 1,4$ und $c = 4$ Teilmodellen herangezogen, wobei die Clusterzentren äquidistant im Intervall der Eingangsgröße verteilt wurden. Im Fallbeispiel wird die Unsicherheit durch die lokalen Modellparametern $\boldsymbol{\theta}_{\mathrm{LM}}$ ausgedrückt und die Partitionierungsparameter $\boldsymbol{\theta}_{\mathrm{MF}}$ werden als fixiert angenommen, wodurch ein LiP-Model resultiert. Als zulässige Fehlerschranke wurde $\delta = 0,15$ gewählt. Weitere Informationen zu Modellansatz und Identifikation sind in [3] zu finden. Die zulässige Parametermenge $\mathrm{FPS}_{\text{RS-BE}}$ wurde mit dem RS-BE-Verfahren aus Abschnitt 2.6 bestimmt. Es wurden 10^6 initiale Punkte $\boldsymbol{\theta}_0$ erzeugt und für jeden der 221 zulässigen Punkte wurden 100 zufällige Richtungsvektoren $\boldsymbol{r}$ erzeugt. Somit wurden insgesamt 22100 Randpunkte gefunden. Die Rechenzeit für das Sampling der zulässigen initialen Punkte betrug 1,8 s und für die Ray-Shooting-Prozedur 21,4 s. Außerdem wurde eine Approximation mit dem REOB-Verfahren aus Abschnitt 2.3 bestimmt. Die Rechenzeit beträgt hierbei 2,6 s.

In Bild 3 sind die geschätzten zulässigen Parametermengen für die einzelnen Teilmodelle dargestellt. Dabei ist die deutliche Überschätzung mit dem REOB-Verfahren ist erkennbar. In Bild 4 sind die Trainingsdaten mit Fehlerschranken und die Prädiktionsfehlerschranken für beide Ansätze dargestellt. Es ist zu erkennen, dass RS-BE die vorgegebenen Fehlerschranken einhält, während REOB die Schranken deutlich reißt. Alle Berechnungen wurden auf einer Workstation mit Intel i5-6500 3,2 GHz CPU und 16 GB RAM in MATLAB durchgeführt. Dabei wurde die Implementierung nicht auf Parallelisierbarkeit hin optimiert. Dies ist ein potentieller Ansatzpunkt, die Geschwindigkeit des Verfahrens zu erhöhen.

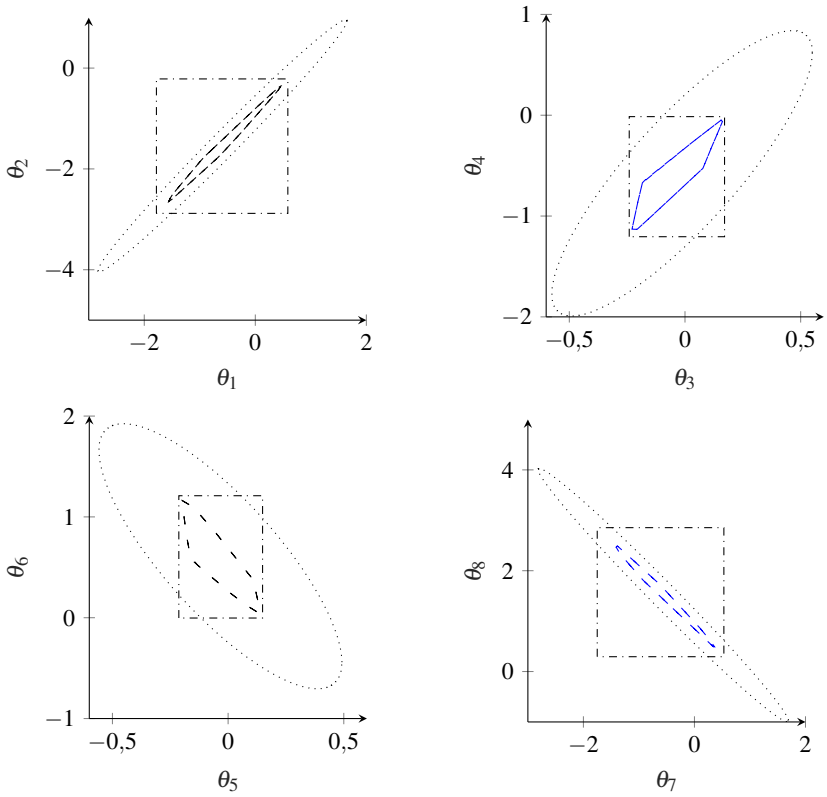


Bild 3: Projektionen für die Ergebnisse der Parameterschätzung mit REOB (gepunktet) und RS-BE (gestrichelt), sowie die achsparallelen begrenzenden Boxen (gestrichpunktet)

4 Zusammenfassung und Ausblick

In diesem Beitrag wurden die Idee und erste Ergebnisse für ein neuartiges Verfahren für die Bounded-Error-Schätzung bei LiP-Modellen vorgestellt. Es wurde gezeigt, dass eine Approximation der zulässigen Parametermenge gefunden wird, die eine Modellprädiktion innerhalb der vorgegebenen Fehlerschranken gewährleistet. Das Verfahren lässt sich über die Anzahl der initialen Punkte und der ausgesendeten Strahlen skalieren. Somit erlaubt das Verfahren

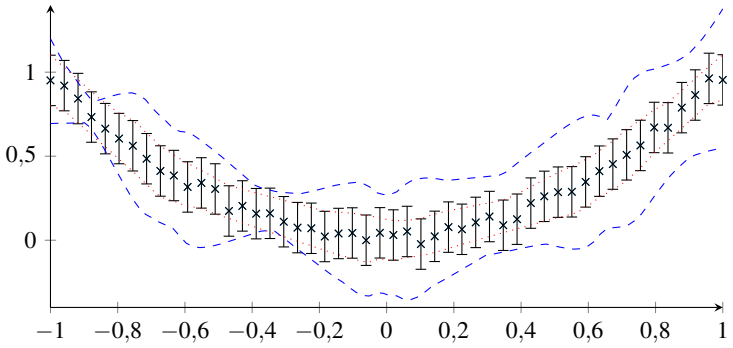


Bild 4: Trainingsdaten (x) mit Fehlerschranke (Fehlerbalken) und Prädiktionsschranken für FPS_{REOB} (gestrichelt) und $FPS_{RS\text{-}BE}$ (gepunktet)

durch Festlegen von Abbruchkriterien den Trade-Off zwischen Genauigkeit und Rechenzeit einzustellen.

Als samplingbasiertes Verfahren unterliegt es allerdings dem "Curse of Dimensionality", d.h. bei steigender Parameterdimension werden exponentiell mehr Samplingpunkte benötigt. Hierzu soll untersucht werden, wie ein effizienteres Sampling der initialen Punkte möglich ist. Zudem soll in Zukunft untersucht werden, inwiefern sich der Ansatz der Strahlenverfolgung auch für nicht-LiP BE-Schätzungen eignet. Dabei treten aber weitere Probleme wie nicht-konvexe und nicht-zusammenhängende Parametermengen auf. Zudem lässt sich das Ray-Shooting dann nicht mehr als Lösung eines linearen Programms (15) formulieren.

5 Danksagung

Dieser Beitrag wurde von der Deutschen Forschungsgemeinschaft DFG im Rahmen des Schwerpunktprogramms SPP 2086 (KR 3795/8-1) gefördert. Die Autoren danken der DFG für die finanzielle und technische Unterstützung.

Literatur

[1] M. Milanese, J. Norton, H. Piet-Lahanier, É. Walter. „Bounding approaches to system identification“. New York: Springer. 2013.

[2] E. Walter, H. Piet-Lahanier. „Estimation of parameter bounds from bounded-error data: a survey“. In: *Mathematics and Computers in Simulation* Bd. 32. S. 449-–468. 1990.

[3] F. Wittich, M. Kahl, A. Kroll. „Zur Schätzung zulässiger Parametermengen nichtlinearer Takagi-Sugeno-Multi-Modelle mit garantierten Fehlerschranken“. In: *Proc., 29. Workshop Computational Intelligence* (Mikut, R.; Reischl, M., Hg.), S. 247–254. KIT Scientific Publishing. 2019.

[4] F. Wittich, L. Kistner,, A. Kroll, C. Schott, T. Niendorf. „On data-driven nonlinear uncertainty modeling: Methods and application for control-oriented surface condition prediction in hard turning“. In: *tm - Technisches Messen*, Bd. 87, Nr. 11, S. 732–741. 2020.

[5] G. Favier, L. Arruda. „Review and comparison of ellipsoidal bounding algorithms“. In: *Bounding approaches to system identification* S. 42-68. New York: Springer. 1996.

[6] J. Matoušek, O. Schwarzkopf, „On ray shooting in convex polytopes“. In: *Discrete & Computational Geometry* Bd. 10. S. 215–232. 1993.

Interval-based Interpretable Decision Tree for Time Series Classification

Malte Schmidt, Volker Lohweg

inIT – Institute Industrial IT, Technische Hochschule Ostwestfalen-Lippe
Campusallee 6, 32657 Lemgo, Germany
E-Mail: {malte.schmidt, volker.lohweg}@th-owl.de

Abstract

In this paper we present the first iteration of a novel time series classification algorithm which is *globally* and *inherently* interpretable. The need for model interpretability or explainability is commonly agreed upon in industry [1]. Model interpretability is an important characteristic of a classifier to build trust in the decisions of the classifier and makes it possible to iteratively improve a model with domain knowledge.

The proposed algorithm first performs an unsupervised clustering of random segments of random length of a time series to find the most discriminating patterns. After finding segments with discriminating patterns, a decision tree is trained using the cluster labels as features. Therefore, the decision tree is restricted to learn a mapping from discriminating clusters to given class labels.

The performance of our algorithm is compared to state-of-the-art algorithms with a computational feasible subset of the University of California, Riverside, time series archive [2]. The first iteration of our algorithm is computationally expensive and does not achieve state-of-the-art accuracy. We point out shortcomings of the current iteration and discuss planned improvements to our algorithm to tackle these shortcomings. We find that our algorithm creates shallow decision trees which boosts interpretability. In contrast, not all state-of-the-art approaches provide interpretable models.

1 Introduction

During the last decades research on time series classification (TSC) has made considerable progress and the University of California, Riverside, time series archive (UCR TSA) [2] is often used to benchmark novel TSC algorithms on one dimensional time series. Often the term time series refers to any ordered series and is not limited to value-index pairs ordered by time. For example, the UCR TSA also includes series generated by spectrographs and object outlines mapped to one-dimensional series.

In industrial and medical applications interpretability of a model is regarded as an important characteristic for a wide adoption of machine learning techniques in these fields [1, 3]. Furthermore, the type of interpretability a model provides is of interest. Here, we differentiate between types of interpretability regarding two different viewpoints.

First, it is important to know how an explanation of a decision is produced. We adopt the differentiation from Rudin [4] and differentiate between the following types:

- *Post hoc explanation of models*. A model is explained post hoc by a second model. An example of a post hoc explanation method often applied to neural networks is LIME [5].
- *Inherently interpretable models*. The model itself provides a faithful explanation of its decisions. An example for an inherently interpretable model is a (small) decision tree with interpretable features.

Second, we are interested in what type of explanation is provided by the model. Here, we adopt the differentiation from Hong [1]:

- *Locally interpretable models*. The model explanation is given on a per instance basis. An example for this type of explanations are saliency maps.
- *Globally interpretable models*. The logical structure of the model itself explains how it works globally. An example for globally interpretable models are, once again, decision trees with interpretable features.

In this paper we propose an algorithm which is globally and inherently interpretable. The features the algorithm utilises are regions of interest in the time series based on their visual appearance (shape). These regions of interest or intervals are phase-dependent which makes our algorithm appropriate for applications which require phase-dependency.

The rest of the paper is organised as follows. In Section 2 we give an overview of state-of-the-art TSC algorithms which are related to our work. Next, we present the design of our algorithm in Section 3. In Section 4 we evaluate the performance of our algorithms and discuss advantages and shortcomings of it before finishing the paper with a conclusion and outlook in Section 6.

2 Related Work

One of the most basic approaches to TSC is a k-nearest-neighbours classifier using an elastic distance metric as similarity measure. An often used elastic distance metric is dynamic time warping (DTW) [6] or variations of it [7, 8]. While this approach is not competitive to current state of the art in terms of accuracy, it still provides a reasonable baseline.

As in other fields, there exist a growing number of approaches to TSC which use neural networks [9]. Neural networks, especially neural networks including convolutional layers, are found to be competitive to other state-of-the-art approaches in terms of accuracy. Some of the recent approaches rely on fully convolutional networks [10] or are inspired by successful architectures in computer vision like the Inception architecture [11]. Wang et al. [10] and Fawaz et al. [9] also explored explaining models with CAM, a post hoc explanation method for CNN-based models [12].

In 2016 Bagnall et al. [13] published an extensive review of the current state of the art in TSC. The best performing algorithm was an ensemble of classifiers, named COTE [14]. COTE combines state-of-the-art classifiers which work in different transformation domains. It was later extended and called HIVE-COTE [15]. This ensemble still achieves state-of-the-art accuracy on the UCR

TSA benchmark due to continuous updating of the ensemble with the latest developments in TSC [16].

A class of features which provides inherent explanations when combined with suitable classifiers are shapelets. Shapelets are phase-independent discriminative time series sub-sequences [17]. Classification is done based on the presence or absence or the count of these discriminative subsequences. One successful approach transforms the time series with a shapelet transformation and then a standard classifiers is trained on the transformed time series [18]. Learning of the k best shapelets through a heuristic gradient descent with a k-means clustering as shapelet initialization is presented by Grabocka et al. [19]. Brunello et al. [20] use a decision tree to build a classifier after finding phase-independent shapelets with evolutionary algorithms.

Decision trees or forests are common classifiers for TSC problems due to their fast training time and interpretability. Deng et al. propose a time series forest (TSF) which uses statistics calculated from random interval as features [21]. They also propose a post hoc explanation through importance curves. More recently multiple ensembles of decision trees for TSC, which achieve state-of-the-art accuracy, are proposed [22, 23].

Another algorithm which achieves state-of-the-art accuracy and is inherently interpretable is the algorithm proposed by Nguyen et al. [24]. A symbolic representation of time series is combined with a sequence learner originally developed for biological sequence classification to search for the most discriminating sub-sequences in the symbolic representations. This approach provides an inherently and locally interpretable model through saliency maps. Nguyen et al. recently compared the explanations provided by CAM, LIME, and the inherent explanations of their sequence learner [25].

Although many decision tree approaches for TSC exist, we think there is still room for further exploration of this approach. We focus on designing an algorithm which creates inherently and globally interpretable models by relying on (shape-based) clustering results of intervals as features. Our hypothesis is that this gives us a distinctive separation of phase-dependent shapes of the times series which improves interpretability. A high level of interpretability enables verification and improvement of the model by an expert.

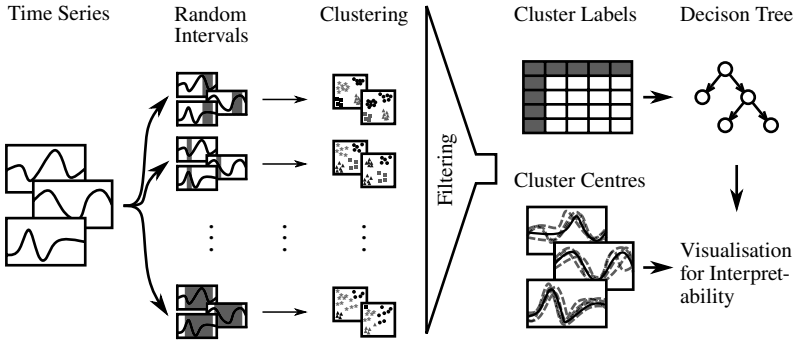


Figure 1: Concept of the proposed classification algorithm.

3 Algorithm Design

The overall concept of our algorithm can be seen in Fig. 1. First, intervals with random start index and random length are chosen. Next, for each interval multiple k-means clusterings with different configurations are computed. After a preliminary filtering of the clusters by the silhouette score [26], the remaining clustering results provide the features for a decision tree. Finally, after training the decision tree, the cluster centres of the selected clusterings visualise the decision process of the decision tree.

In the following required definitions and notations are introduced.

Time Series. A time series is a sequence $\mathbf{t} = (t_1, \ldots, t_L)$ of L values (observations) ordered by some criterium (e.g. time, frequency or wavelength). The length of time series $\mathbf{t}$ is L.

Discrete Interval. A discrete interval $\mathscr{I} = [s..e]$ is a set of integers, i.e. $\{s, s+1, \ldots, e\}$. We express the indices of a sub-sequence of a time series with an interval. For example, $\mathbf{t}(\mathscr{I}) = (t_s, \ldots, t_e)$ is the sub-sequence of time series $\mathbf{t}$ over the interval $\mathscr{I}$. The length of interval $\mathscr{I}$ is given by its cardinality $|\mathscr{I}|$. We assume all intervals are valid, i.e. $1 \leq s \leq e \leq L$ holds.

3.1 Interval Selection

Instead of choosing all possible intervals, we limit the number of intervals for a time series with length L to $O(L)$ to reduce the time complexity of our algorithm. For the selection of the intervals, we follow the approach of Deng et al. [21]:

1. Select $\sqrt{L}$ window lengths from the set of possible window lengths $W_p = \{1,\ldots,L\}$ by random sampling without replacement.
2. For each window length w, select $\sqrt{L-w+1}$ start indices from the set of possible start indices $\mathscr{S} = \{1,\ldots,L-w+1\}$ by random sampling without replacement.

Each pair of selected window length w and start index i forms an interval $\mathscr{I} = [i..i+w-1]$. We use these intervals to extract sub-sequences from the time series for further processing.

By selecting sub-sequences from time series, we follow an interval-based approach for our algorithm and introduce phase-dependency. The idea is to select regions of interest which possibly contain distinctive shapes. Ideally, they should have a causal relation to the class labels.

3.2 Clustering

If we are interested in inherently and globally interpretable models, we require meaningful features for our model. In TSC one type of meaningful features are distinctive shapes. After choosing interval candidates which represent possible regions of interest containing such shapes, one way to find meaningful features is to cluster the sub-sequences resulting from the intervals.

We follow this rationale and apply k-means clustering with DTW as dissimilarity measure to find clusters of sub-sequences which intuitively match in shape. Each cluster is then represented by an average series calculated with DTW barycentre averaging (DBA) [27]. We used the k-means clustering implementation `TimeSeriesKMeans` from the `tslearn` [28] library. k-means++ [29] is applied as cluster initialization method.

We are only interested in cluster results which give a good separation between clusters. Therefore, before using the clustering results as training input for a decision tree, we pre-filter the results to exclude clustering results with high overlap between different clusters. For this we calculate the mean silhouette score [26] of all samples for each clustering result with DTW as the dissimilarity measure.

The (mean) silhouette score can take values between -1 and 1. A value below zero indicates overlapping of clusters while a value above 0 indicates a non-overlapping separation of clusters. We accept all cluster results with an overall silhouette score greater than zero for further processing.

3.3 Decision Tree Induction

In the last step, a decision tree is trained on the cluster labels of the remaining clustering results. As the decision tree induction algorithm we use an implementation of the ID3-algorithm by Quinlan [30] with two domain specific modifications:

- Induce bias towards a preferred interval length.
- Restrict the allowed overlap for intervals in the same tree branch.

For attribute selection gain ratio [30] is applied. In early experiments we noticed that intervals with a high overlap often have the same (maximal) gain ratio. To break the tie, we introduce a weighting function which weights the gain ratio depending on the interval length. With this weighting function, we induce bias towards a preferred interval length. We prefer a shorter interval length over a longer one because the shape present in shorter intervals is usually less complex and easier to interpret. However, if the interval length gets too small (a single value in the extreme case), the shape may not be meaningful and dominated by noise.

We propose to use a parametrisable unimodal weighting function

$$f(x) : [0,1] \rightarrow [0,1]$$

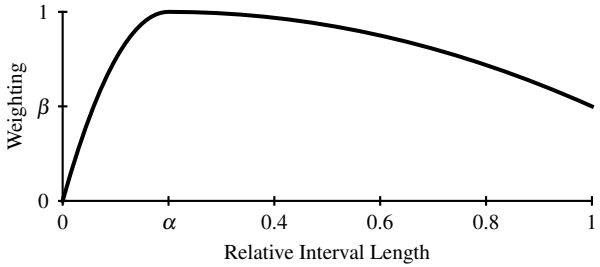


Figure 2: Proposed weighting function.

which maps the relative interval length to a weight for the gain ratio. The parameters are α, the preferred interval length relative to the length of the time series, and β, the weighting value for the whole series length. We set $f(\alpha) = 1$ and $f(1) = \beta$. The weighting function is given by

$$f(x) = \begin{cases} -\frac{1}{\alpha^2}(x-\alpha)^2 + 1 & \text{for } x \leq \alpha, \\ \frac{\beta-1}{(1-\alpha)^2}(x-\alpha)^2 + 1 & \text{for } x > \alpha \end{cases} \tag{1}$$

with $0 < \alpha < 1$ and $0 \leq \beta \leq 1$. Fig. 2 shows an example for $f(x)$ with $\alpha = 0.2$ and $\beta = 0.5$.

A quadratic function is chosen for the weighting function because of its simplicity (in terms of parameters) while still having a modest slope around its maximum in contrast to e.g. a triangle function. However, other unimodal function types are also valid candidates.

It can also happen that clustering results for the same interval but with a different number of cluster centres have the same (maximal) gain ratio. In this case we select the clustering result with the highest silhouette score because we want the features to be as interpretable as possible. However, other preferences, i.e. selecting the result with the fewest cluster centres, are also valid options.

The consecutive selection of multiple highly overlapping intervals in one tree branch may lead to overfitting. Suppose one distinctive sub-sequence is covered by overlapping intervals multiple times. Then this sub-sequence is im-

plicitly selected as classification criterium multiple times. To prevent this, we restrict the allowed overlap for the intervals used consecutively in a tree branch.

Let $\mathscr{I}_1, \ldots, \mathscr{I}_{N-1}$ be the intervals used consecutively in one tree branch and let $\mathscr{I}_N$ be the interval which we want to use for splitting at the next node. Then the maximum relative overlap o_{max} for any of these intervals is given by

$$o_{\max}\left(\mathscr{I}_1, \ldots, \mathscr{I}_N\right) = \max_x \frac{\left| \mathscr{I}_x \cap \bigcup_{i=1, i \neq x}^{N} \mathscr{I}_i \right|}{|\mathscr{I}_x|} \,. \tag{2}$$

o_{max} can have values between 0 (no overlap) and 1 (at least one interval fully overlaps with others). At each new node in a tree branch o_{max} is calculated including the new interval we want to use. The new interval is only accepted if o_{max} does not exceed a threshold θ.

4 Evaluation

4.1 UCR TSA Subset Selection

The current iteration of our algorithm is computationally expensive due to the clustering and silhouette score computation. Therefore, for this early evaluation of the algorithm, a subset of the UCR TSA is selected. To be as objective as possible, we define a computational complexity score with which we rank the datasets and pick the first 25 datasets for our evaluation. We limit our selection to datasets of the 2015 version of the UCR TSA [31] because accuracies for the provided train-test-splits are available for these datasets on the UCR TSA website [32].

We define the complexity score S of a dataset as

$$S = L \cdot \left(k^2 \cdot N \cdot L^2 + k \cdot I_{\mathrm{K}} \cdot N \cdot L^2 + I_{\mathrm{K}} \cdot I_{\mathrm{B}} \cdot N \cdot L^2\right) \tag{3}$$

Table 1: The UCR TSA dataset subset selected for evaluation.

No.	Dataset	No.	Dataset
1	ItalyPowerDemand	14	MiddlePhalanxTW
2	SonyAIBORobotSurface1	15	ProximalPhalanxTW
3	SonyAIBORobotSurface2	16	DistalPhalanxTW
4	MoteStrain	17	MiddlePhalanx-
5	TwoLeadECG		OutlineCorrect
6	ECGFiveDays	18	DistalPhalanx-
7	CBF		OutlineCorrect
8	SyntheticControl	19	ProximalPhalanx-
9	ECG200		OutlineCorrect
10	GunPoint	20	Plane
11	ProximalPhalanx-	21	ArrowHead
	OutlineAgeGroup	22	MedicalImages
12	MiddlePhalanx-	23	Coffee
	OutlineAgeGroup	24	Wine
13	DistalPhalanx-	25	ToeSegmentation1
	OutlineAgeGroup		

with the number of classes n_c in the dataset, the maximal number of clusters $k = \max\{10,\ 2 \cdot n_c\}$ to compute, the number of samples N, the time series length L, the number of iterations of the k-means algorithm I_K, and the number of iterations for barycentre calculation I_B of this dataset. The score is composed of the time complexity of the k-means++ [29] cluster centre initialization (first summand), the time complexity of the distance calculation to the cluster centres across all iterations (second summand), and the time complexity of the DBA across all iterations [27] (third summand).

The 25 datasets with the lowest score S are listed in Tab. 1. It is important to note that selecting datasets by complexity ranking necessarily introduces a bias towards datasets with shorter time series and fewer training samples. However, for an evaluation of this early iteration of our algorithm, the selected datasets are sufficient to draw preliminary conclusions and point out future research directions.

Table 2: Important parameter settings of the `TimeSeriesKMeans` algorithm.

Parameter	Values	Explanation
n_clusters	$\{2, \ldots, \max\{10, 2 \cdot n_c\}\}$	Number of clusters. n_c: Number of classes.
max_iter	50	Iterations for k-means.
metric	dtw	Metric to be used.
max_iter_barycenter	10	Iterations for DBA.
init	k-means++	Cluster initialization method.

4.2 Experiment Setup

Important parameters of the `TimeSeriesKMeans` algorithm are listed in Tab. 2 algonside the values we used. The limits for `n_clusters`, `max_iter`, and `max_iter_barycenter` were chosen to limit the computation time required by the algorithm. The parameters of the interval weighting function are set to $\alpha = 0.2$ and $\beta = 0.6$. The threshold for the maximal allowed overlap $o_{\max}$ is set to $\theta = 0.4$ and a minimal gain ratio of 0.05 is required for a node split to be considered. For the current evaluation no hyperparameter optimization is considered and the allowed warping path for the DTW calculation has no additional restrictions.

4.3 Results

First, the performance of our interval-based decision tree (IBIT) and a 1-nearest-neighbour classifier with DTW as distance metric (1NN-DTW) is compared in Fig. 3a. Each point represents accuracies for one dataset. Points above the dashed line indicate a better performance of IBIT. We expect an IBIT model to perform better than a simple 1NN-DTW model because it is based on the same underlying distance metric while having a more sophisticated decision process. However, for 16 out of 25 datasets the 1NN-DTW performance is better.

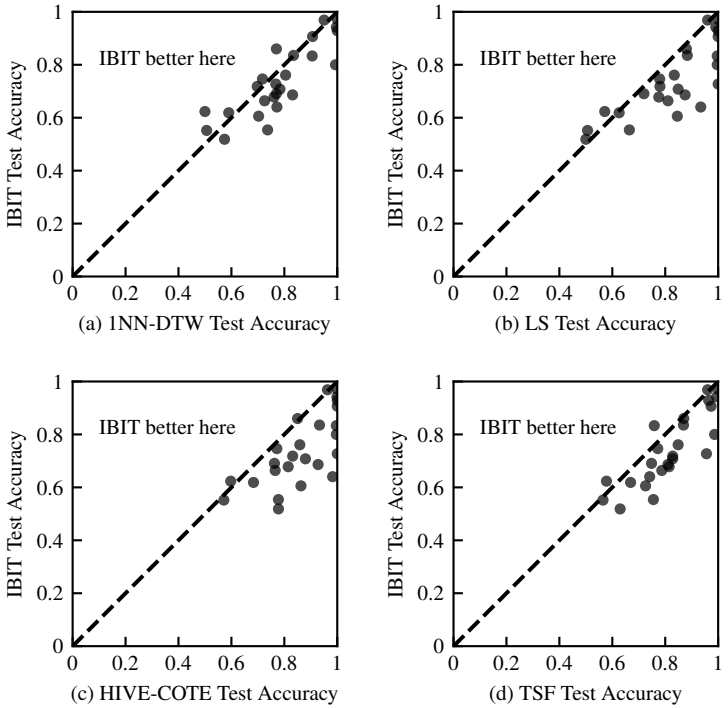


Figure 3: Test accuracies of IBIT model compared to test accuracies of selected algorithms as reported on the UCR TSA website [32] on 25 UCR TSA datasets.

This observation needs further investigating in the future. One possible reason for the lower performance of IBIT is the unsupervised clustering using k-means. For example, k-means clustering does not cope well with points which would be best clustered together but which are spread across a line in the feature space.

In addition, by looking at the cluster results, we observe that the hard cut-off of the time series at the interval limits may lead to a clustering dominated by shapes close to the interval limits. These shapes may be present inside the interval or outside of it depending on the stretch of the time series. Fig. 4 shows an example of this phenomenon for the interval $\mathscr{I} = [33..92]$ for the `ECG200` dataset. All time series posses steep slopes near the interval limits. However,

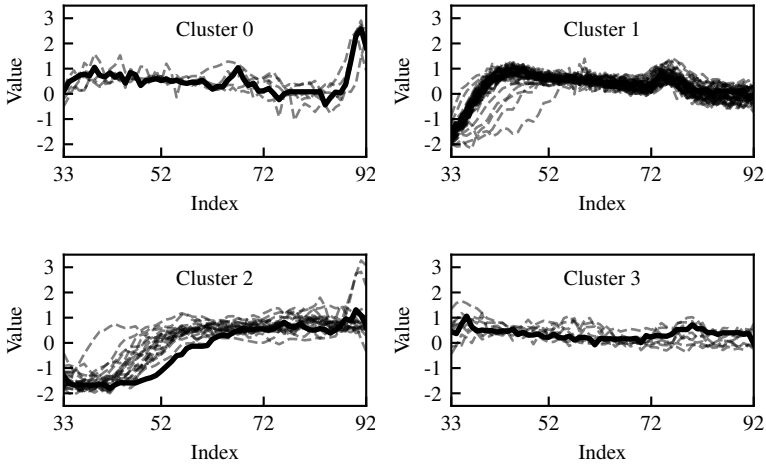


Figure 4: An example of a bad clustering due to the hard cut-off at the interval limits. The steep slopes at the limits of the interval do not always lie inside the interval. The presence or absence of these slopes inside the interval dominates the clustering result. Barycentres are displayed as solid lines.

these sections of steep slope are not always captured inside interval $\mathscr{I}$ because some time series are more stretched than others. The presence or absence of these slopes inside the interval dominates the clustering result and leads to a limited meaningfulness of the clustering result. One possible solution to limit the influence of the values close to the interval limits is to apply a weighted DTW penalizing these values. This should be investigated in the future.

Fig. 3b and 3c compare the IBIT performance to a shapelet-based approach (LS) [19] and an ensemble of classifiers including shapelet-based classifiers (HIVE-COTE). For 7 datasets LS and HIVE-COTE both achieve a classification accuracy close to 100% and the IBIT performance is not competitive for at least 3 of these datasets (`TwoLeadECG`, `ECGFiveDays`, `SyntheticControl`). All these datasets include approximately phase-aligned samples. Therefore, the low performance of IBIT on these datasets contradicts our expectations. A possible reason for the low performance of IBIT on these datasets is a selection of intervals which misses the important regions of interest in these time series.

This highlights the importance of interval selection. A further detailed analysis is required to come to a conclusive result in this case.

A performance comparison to TSF, a random forest with simple statistics of intervals as features, is shown in Fig. 3d. Although TSF only uses simple features (mean, standard deviation, slope) it outperforms the IBIT accuracy on most datasets. TSF has two main differences to our decision tree classifier. First, at each node in a decision tree of the TSF ensemble a new selection of intervals is considered. Therefore, the algorithm evaluates more intervals than IBIT does. Second, an ensemble of decision trees is used increasing the evaluation of different features further. While considering more than $O(L)$ intervals can be a suitable improvement to our algorithm, using an ensemble of decision tress cannot. This would lead to loss of interpretability of our models.

Although the unmodified IBIT models do not achieve state-of-the-art performance, they have the advantage of being interpretable. This does not only mean that the models can be verified but it also means that the IBIT models can be improved iteratively. An expert can investigate the intervals and cluster results an IBIT model uses for its decision process and iteratively refine the intervals or can add new clustering results with modified configurations. For instance, an expert can identify the inappropriate clustering results shown in Fig. 4 and make suitable adjustments to the intervals.

An evaluation of the tree complexities of 250 IBIT models trained on 10 different subsets of the training data shows that most decision trees are not overly complex and can easily be interpreted and modified by an expert. Each model is trained on 80% of the available training data. Fig. 5 shows the tree depths of all 250 models investigated. Interestingly, none of these 250 models has a tree depth greater than six. Possibly this is due to the fact that we put a restriction on the maximum overlap of intervals and at some deeper nodes of the tree no new interval candidates are available. For this evaluation, we did not prune the decision trees and the same underlying intervals were selected. Variation of intervals and evaluation of pruning techniques is planned in the future.

Fig. 6 shows the number of decision tree leafs across all 250 models. 50% of models have fewer than 11 leaves and 90% of models have fewer than 40

Figure 5: Decision tree depth counts of 250 trees from a 10-fold cross-validation for each of the 25 datasets.

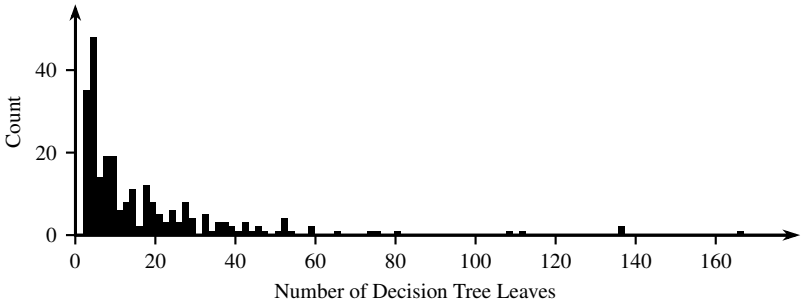


Figure 6: Decision tree leaf counts of 250 trees from a 10-fold cross-validation for each of the 25 datasets.

leaves. Only 10% of models have more than 40 leaves making them hard or at least tedious to interpret. This supports the hypothesis that IBIT models can be iteratively improved by an expert. This also shows that the learned decision trees are shallow but wide decision trees.

5 Conclusion and Outlook

In this paper we presented an algorithm to train interval-based interpretable decision trees. The algorithm is designed to create easy to interpret models

which can be iteratively improved by an expert. Modifications to improve the models can be identified and applied by an expert because of the inherent and global interpretability of the models. The simplicity of the resulting decision trees and the intuitive features help achieve this goal.

Although the algorithm does not achieve state of the art in terms of accuracy, it is important to note that accuracy is not the single most important criterium in all circumstances. Interpretable models can be analysed and verified by experts easily and spurious correlations in the data learned by the model can be identified and prohibited. Interpretable model are easy to improve iteratively to achieve certain goals and optimisation is not restricted to a single metric, e.g. the accuracy score. To investigate this hypothesis, cases studies where IBIT models are improved iteratively are a future field of research.

The evaluation presented in this paper shows preliminary results and a more comprehensive study is planned in the future. Once a more comprehensive study is done, we also plan to publish the code of our algorithm to make the results as reproducible as possible for the research community.

Further improvements to the algorithm we plan to investigated are

- extending the features by interpretable shapelet-based features to include phase-independent features,
- improving the scalability of the clustering through pruning strategies [33] or using a clustering strategy based on autocorrelation [34],
- applying pruning strategies to the decision trees,
- using weighted DTW to penalize values near interval limits,
- and optimising hyperparameters of the algorithm.

Acknowledgments

We would like to thank all UCR TSA data contributors and Bagnall et al. [32] for providing the datasets and accuracy results. Furthermore, we would like

to thank all contributors of open source software we used to implement our algorithm with, especially all contributors of the `tslearn` [28] library.

This work was partly funded by the German Federal Ministry of Education and Research (BMBF) within the project ITS.ML (grant no. 01IS18041D) and the Ministry of Economic Affairs, Innovation, Digitalisation and Energy of the State of North Rhine-Westphalia (MWIDE) within the project ML4Pro^2 (grant no. 005-1807-0090).

References

[1] S. R. Hong, J. Hullman and E. Bertini. “Human factors in model interpretability: Industry practices, challenges, and needs”. In: *Proceedings of the ACM on Human-Computer Interaction 4.CSCW1*, pp. 1–26. 2020.

[2] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana and E. Keogh. “The UCR time series archive”. In: *IEEE/CAA Journal of Automatica Sinica 6.6*, pp. 1293–1305. 2019.

[3] A. Vellido. “The importance of interpretability and visualization in machine learning for applications in medicine and health care”. In: *Neural Computing and Applications 32.24*, pp. 18069–18083. 2019.

[4] C. Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence 1.5*, pp. 206–215. 2019.

[5] M. T. Ribeiro, S. Singh and C. Guestrin. “"Why should i trust you?" Explaining the predictions of any classifier”. In: phProceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144. 2016.

[6] H. Sakoe and S. Chiba. “Dynamic programming algorithm optimization for spoken word recognition”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing 26.1*, pp. 43–49. 1978.

[7] Y.-S. Jeong, M. K. Jeong and O. A. Omitaomu. “Weighted dynamic time warping for time series classification”. In: *Pattern Recognition 44.9*, pp. 2231–2240. 2011.

[8] T. Górecki and M. Łuczak. “Non-isometric transforms in time series classification using DTW”. In: *Knowledge-Based Systems 61*, pp. 98–108. 2014.

[9] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar and P.-A. Muller. “Deep learning for time series classification: A review”. In: *Data Mining and Knowledge Discovery 33.4*, pp. 917–963. 2019.

[10] Z. Wang, W. Yan and T. Oates. “Time series classification from scratch with deep neural networks: A strong baseline”. In: ph2017 International Joint Conference on Neural Networks (IJCNN), pp. 1578-1585. 2017.

[11] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P.-A. Muller and F. Petitjean. “InceptionTime: Finding AlexNet for time series classification”. In: *Data Mining and Knowledge Discovery 34.6*, pp. 1936–1962. 2020.

[12] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra. “Grad-CAM: Visual explanations from deep networks via gradient-based localization”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626. 2017.

[13] A. Bagnall, J. Lines, A. Bostrom, J. Large and E. Keogh. “The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances”. In: *Data Mining and Knowledge Discovery* 31.3, pp. 606–660. 2016.

[14] A. Bagnall, J. Lines, J. Hills and A. Bostrom. “Time-series classification with COTE: The collective of transformation-based ensembles”. In: *IEEE Transactions on Knowledge and Data Engineering 27.9*, pp. 2522–2535. 2015.

[15] J. Lines, S. Taylor and A. Bagnall. “Time series classification with HIVE-COTE”. In: *ACM Transactions on Knowledge Discovery from Data 12.5*, pp. 1–35. 2018.

[16] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom and A. Bagnall. "Hive-cote 2.0: a new meta ensemble for time series classification". In: *arXiv preprint arXiv:2104.07551*. 2021.

[17] L. Ye and E. Keogh. "Time series shapelets: : A new primitive for data mining". In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 947–956. 2009.

[18] J. Hills, J. Lines, E. Baranauskas, J. Mapp and A. Bagnall. "Classification of time series by shapelet transformation". In: *Data Mining and Knowledge Discovery 28.4*, pp. 851–881. 2013.

[19] J. Grabocka, N. Schilling and L. Schmidt-Thieme. "Learning time-series shapelets". In: *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 392–401. 2014.

[20] A. Brunello, E. Marzano, A. Montanari and G. Sciavicco. "J48SS: A novel decision tree approach for the handling of sequential and time series data". In: *Computers 8.1*, pp. 21. 2019.

[21] H. Deng, G. Runger, E. Tuv and M. Vladimir. "A time series forest for classification and feature extraction". In: *Information Sciences 239*, pp. 142–153. 2013.

[22] M. Middlehurst, J. Large and A. Bagnall. "The canonical interval forest (CIF) classifier for time series classification". In: *2020 IEEE International Conference on Big Data (Big Data)*, pp. 188–195. 2020.

[23] A. Shifaz, C. Pelletier, F. Petitjean and G. I. Webb. "TS-CHIEF: A scalable and accurate forest algorithm for time series classification". In: *Data Mining and Knowledge Discovery 34.3*, pp. 742–775. 2020.

[24] T. L. Nguyen, S. Gsponer, I. Ilie, M. O'Reilly and G. Ifrim. "Interpretable time series classification using linear models and multi-resolution multi-domain symbolic representations". In: *Data Mining and Knowledge Discovery 33.4*, pp. 1183–1222. 2019.

[25] T. T. Nguyen, T. L. Nguyen and G. Ifrim. “A model-agnostic approach to quantifying the informativeness of explanation methods for time series classification”. In: *International Workshop on Advanced Analytics and Learning on Temporal Data*, pp. 77–94. 2020.

[26] P. J. Rousseeuw. “Silhouettes: A graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of Computational and Applied Mathematics 20*, pp. 53–65. 1987.

[27] F. Petitjean, A. Ketterlin and P. Gançarski. “A global averaging method for dynamic time warping, with applications to clustering”. In: *Pattern Recognition 44.3*, pp. 678–693. 2001.

[28] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar and E. Woods. “Tslearn, a machine learning toolkit for time series data”. In: *Journal of Machine Learning Research 21.118*, pp. 1–6. 2020.

[29] D. Arthur and S. Vassilvitskii. “K-means++: The advantages of careful seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035. 2007.

[30] J. R. Quinlan. “Induction of decision trees”. In: *Machine Learning 1.1*, pp. 81–106. 1986.

[31] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen and G. Batista. “The UCR time series classification archive”. `www.cs.ucr.edu/~eamonn/time_series_data/` (Online, 23.09.2021). 2015.

[32] A. Bagnall, E. Keogh, J. Lines, A. Bostrom, J. Large and M. Middlehurst. “UEA/UCR time series classification repository”. `www.timeseriesclassification.com` (Online, 23.09.2021).

[33] N. Begum, L. Ulanova, J. Wang and E. Keogh. “Accelerating dynamic time warping clustering with a novel admissible pruning strategy”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.*, pp. 49–58. 2015.

[34] J. Paparrizos and L. Gravano. “k-shape: Efficient and accurate clustering of time series”. In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data.*, pp. 1855–1870. 2015.

Smart Data Representations: Impact on the Accuracy of Deep Neural Networks

Oliver Neumann[1], Nicole Ludwig[2], Marian Turowski[1], Benedikt Heidrich[1], Veit Hagenmeyer[1], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology
Hermann-von-Helmholtz-Platz 1, 76344 Eggenstein-Leopoldshafen
[2] Cluster of Excellence Machine Learning,
University of Tübingen
Maria-von-Linden Str. 6, 72076 Tübingen
E-Mail: oliver.neumann@kit.edu

Abstract

Deep Neural Networks (DNNs) are able to solve many complex tasks with less engineering effort and better performance. However, these networks often use data for training and evaluation without investigating its representation, i.e. the form of the used data. In the present paper, we analyze the impact of data representations on the performance of DNNs using energy time series forecasting. Based on an overview of exemplary data representations, we select four exemplary data representations and evaluate them using two different DNN architectures and three forecasting horizons on real-world energy time series. The results show that, depending on the forecast horizon, the same data representations can have a positive or negative impact on the accuracy of DNNs.

1 Introduction

Deep Neural Networks (DNNs) can better solve complex tasks such as image classification [1, 2], object detection [3], or instance segmentation [4, 5] with less effort than traditional approaches. Nevertheless, DNNs require data for training and evaluation. However, data is often used by DNNs without further investigation of different data representations.

Since data representations influence what DNNs learn and which architectures can be used, data representations should be investigated further. The representation of the data can be changed through transformations such as reshaping, aggregation, or selection. Although recent literature, including work on feature engineering [6, 7, 8], introduces new data representations and compares them [9, 10, 11, 12, 13], it does not systematically investigate the influence of data representations on the performance of DNNs.

In this paper, we analyze the impact of data representations on the performance of DNNs at the commonly investigated example of energy time series forecasting (see e.g. [14, 15, 16, 13]). For this purpose, we investigate the time series in its original form and the derivative of the time series, and both reshaped as an image. For the analysis, we use two different architectures, namely a Fully Connected Network (FCN) and a Convolutional Neural Network (CNN).

The remainder of the paper is structured as follows. In the second chapter, we introduce different transformations for data representations. In the third chapter, we present an energy forecasting use case to demonstrate the impact of different data representations. In the fourth chapter, we discuss the findings of this paper, before we finally give a conclusion.

2 Transformations for Data Representations

In this chapter, we introduce different transformations for changing data representations based on the data types vectors or matrices. Transformations allow to convert data from one data representation to another. Unlike data

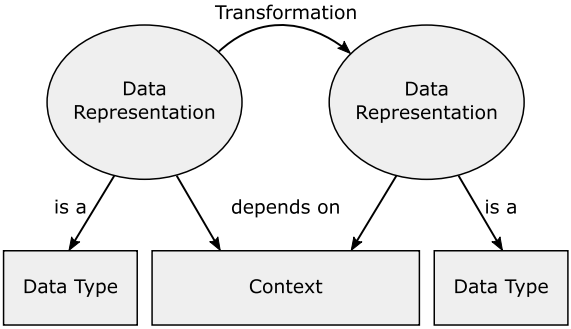


Figure 1: Relationship between data representation, data type, transformation, and context. A transformation converts a data representation into another, while a data representation is a data type and depends on a context.

types, data representations are context dependent and thus are typically difficult to characterize. For comprehensibility, we thus present transformations, which are per se context independent (see Figure 1). For the present paper, we consider reshaping, selection, aggregation, differences, convolution, rescaling, clustering, and latent space transformation as exemplary transformations for converting data from one representation into another. For each transformation, we briefly describe its key idea and underlying concept.

Reshaping Vectors and matrices can be transformed by changing their shape. This reshaping allows the use of different model architectures. Reshaping is defined by a function that maps each element of a vector or matrix to a resulting vector or matrix with a different dimensionality.

Selection Data representations can be transformed by selecting specific elements. To select certain elements, one can specify the related indices, which then define a subset of the considered vector or matrix. Choosing specific elements of a vector or matrix can be beneficial because a data representation can contain unnecessary information.

Aggregation The aggregation of data can be used as a transformation for data representations. Aggregations can help a DNN to achieve higher performances because aggregations can, for example, reduce the dimensionality and noise in the data. They can be applied on single vectors or matrices along one or more axes, leading to a matrix or vector depending on the dimensionality of the input.

Differences Calculating the differences is similar to the discrete derivation and can be applied to vectors and matrices. The data is transformed by subtracting the values of certain points or axes, vectors, or matrices depending on the input data representation. For vector inputs, the difference between certain points is calculated. For matrix inputs, the differences are calculated for certain subvectors or submatrices depending on the dimensionality of the matrix and along which axis the differences should be calculated.

Convolution A convolution is a mathematical operation that combines two functions. The convolution, e.g. a frequency filter, can be described by a kernel that is multiplied iteratively over the input data and summed afterward. Both the kernel and the input data can be vectors or matrices.

Rescaling The representation of data can be transformed by fitting a function on the data and resampling from that function. Thereby, data can be upscaled or downscaled, where upscaling increases and downscaling decreases the amount of data. Exemplary methods to approximate the underlying function are linear, cubic, or spline interpolation.

Clustering Data can also be clustered such that it is represented by cluster representatives. The cluster representatives are determined by similarity measures based on, for example, density or distances. Common clustering approaches are k-Means [17], fuzzy c-means [18], BIRCH [19], OPTICS [20], or DBSCAN [21].

Latent Space Transformation Latent space transformations learn a latent data representation of a dataset. This latent data representation has a lower dimensionality as the original dataset and could be a vector or matrix data representation. There are several approaches to use the latent space information and reduce the dimensionality like Principal Component Analysis [22], Linear Discriminant Analysis [23], or Autoencoder [24].

3 Energy Forecasting Use Case

In this chapter, we show how data representations affect the forecasting accuracy of a Deep Neural Network (DNN) when forecasting the German electricity demand and using four different data representations, namely *naive*, *naive differences*, *reshaped*, and *reshaped differences*. In the following, we first describe the data for this use case and the data representations. Afterward, we present the selected baselines and DNN architectures to forecast the electricity demand. In the last section, we present the results and compare the evaluated data representations.

3.1 Data

For the electricity demand, we use data from the European Network of Transmission System Operators for Electricity provided by Open Power System Data [25]. We select the electricity demand for Germany from the beginning of 2015 up to the end of 2019, which is the last complete year of data. The data contain typical daily, weekly, and seasonal patterns, which we account for with the help of calendar information. As calendar information, we use hour, day of week, day of year, weekend, and holiday, where the first three are encoded as sine and cosine functions and the last two are encoded as Boolean variables.

3.2 Evaluated Data Representations

This section describes the four data representations chosen to analyze the impact of data representations on the forecasting performance of DNNs, i.e. the

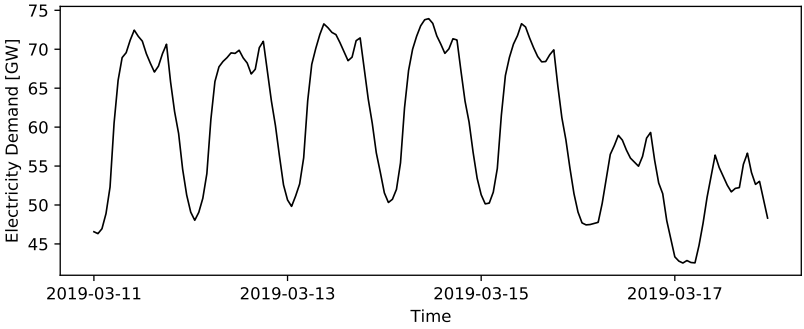


Figure 2: The *naive* data representation that comprises the electricity demand of the past 168 hours shown for the exemplary forecast origin 18.03.2019.

naive, the *naive differences*, the *reshaped*, and the *reshaped differences* data representations. Overall, this selection results in two vector-based and two matrix-based data representations. These data representations are used as input for the evaluated DNN architectures, whose output also depend on these representations.

Naive The *naive* data representation comprises the vector of the last 168 hours of the electricity demand (see Figure 2). It is defined as

$$\begin{bmatrix} x_k \\ \vdots \\ x_{k-167} \end{bmatrix}, \tag{1}$$

where $x \in X$ is the set of historical electricity demand values and k the forecast origin.

Naive Differences The *naive differences* data representation is again comprised of a vector of the last 168 hours of the electricity demand. However, instead of using the raw values as in the *naive* data representation, we now use differences. The lag of these differences depends, in our case, on the forecast

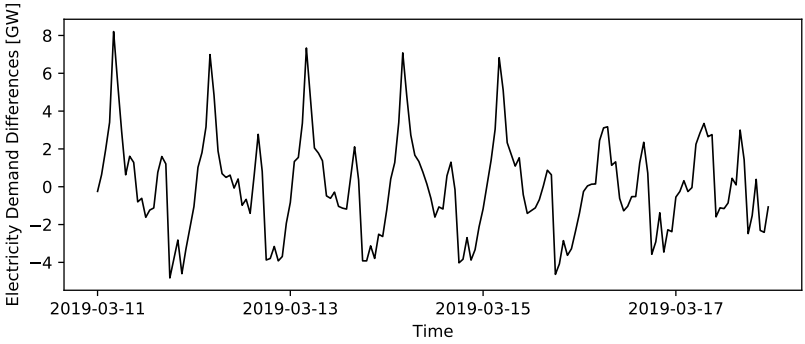


Figure 3: The *naive differences* data representation that comprises the electricity demand differences with $h = 1$ of the past 168 hours for the exemplary forecast origin 18.03.2019.

horizon, e.g., for a one-day ahead forecast, we calculate differences with a lag of one day. An exemplary week is illustrated in Figure 3 and the data representation is then defined as

$$\begin{bmatrix} x_k - x_{k-h} \\ \vdots \\ x_{k-167} - x_{k-167-h} \end{bmatrix}, \tag{2}$$

where $x \in X$ is the set of historical electricity demand values, k the forecast origin, and h the lag used for differencing that we set as the forecast horizon (e.g. 1, 24, 168).

Reshaped The *reshaped* data representation uses the *naive* data representation and reshapes it into a two-dimensional matrix such that each row represents a day. Consequently, the *reshaped* data representation consists of a 7x24 dimensional matrix (see Figure 4) and is defined as

$$\begin{bmatrix} x_k \\ \vdots \\ x_{k-167} \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_k & \dots & x_{k-23} \\ \vdots & \ddots & \vdots \\ x_{k-144} & \dots & x_{k-167} \end{bmatrix}, \tag{3}$$

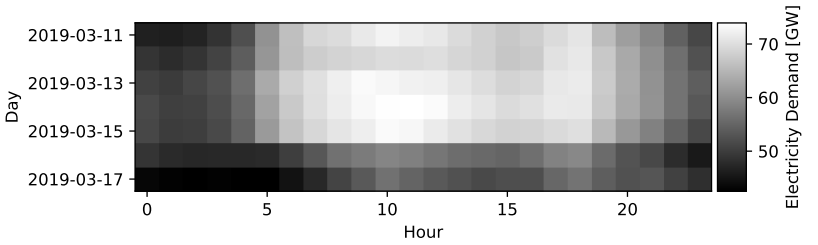


Figure 4: The *reshaped* data representation that comprises the electricity demand of the past 168 hours shown for the exemplary forecast origin 18.03.2019.

where $x \in X$ is the set of historical electricity demand values and k the forecast origin.

Reshaped Differences The *reshaped differences* data representation is equivalent to the *reshaped* data representation but uses the *naive differences* data representation as its basis. The *reshaped differences* data representation, therefore, also consists of a 7x24 dimensional matrix (see Figure 5 for an exemplary week) and is defined as

$$\begin{bmatrix} x_k - x_{k-h} \\ \vdots \\ x_{k-167} - x_{k-167-h} \end{bmatrix} \Leftrightarrow \begin{bmatrix} x_k - x_{k-h} & \dots & x_{k-23} - x_{k-23-h} \\ \vdots & \ddots & \vdots \\ x_{k-144} - x_{k-144-h} & \dots & x_{k-167} - x_{k-167-h} \end{bmatrix}, \quad (4)$$

where $x \in X$ is the set of historical electricity demand values, k the forecast origin, and h the lag used for differencing that we set as the forecast horizon (e.g. 1, 24, 168).

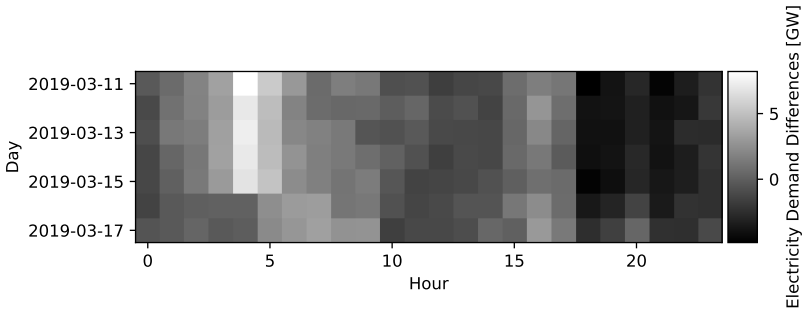


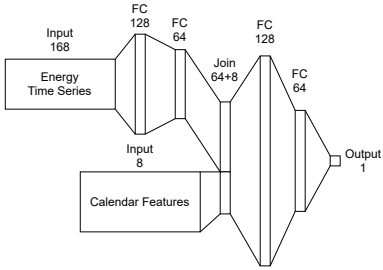
Figure 5: Exemplary week of the *reshaped differences* data representation that reshapes the electricity demand differences with $h = 1$ of the past 168 hours as a 24x7 matrix for the exemplary forecast origin 18.03.2019.

3.3 Experimental Setup

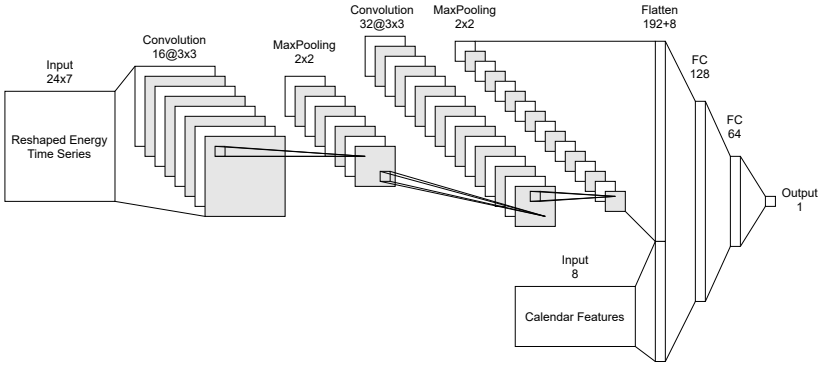
In this section, we introduce the evaluated DNN architectures and the selected baselines before we describe the experimental setup including the train validation test split, the considered forecast horizons, the number of performed runs, the used evaluation metrics, and the implementation.

To investigate the impact of data representations in energy time series forecasting, we use two DNNs (see Figure 6). The first DNN is a Fully Connected Network (FCN). It only consists of fully connected layers, whose input is a vector of historical energy data. This FCN is applied to both naive data representations. The second DNN is a Convolutional neural network (CNN). It consists of convolutional layers followed by fully connected layers that process the energy time series as a matrix. This CNN is applied to both reshaped data representations.

The FCN consists of two parts. The first part processes the energy time series vector input into a 64 dimensional latent vector representation and consists of two layers. The second part joins the 64 dimensional latent energy vector with the calendar feature vector and processes the concluding vector to the single forecast output.



(a) Fully Connected Network (FCN)



(b) Convolutional Neural Network (CNN)

Figure 6: DNN architectures used for forecasting the energy time series, where the numbers indicate the number of neurons in the fully connected layers and the number of features with the corresponding kernel sizes in the convolutional layers (separated with @).

The CNN is also split into two parts. The first part processes the reshaped energy time series with two stacked convolutional layers. This first part results in a latent representation of matrices, which is then flattened and joined with the calendar features. The second part processes the vector-based latent representation of the energy time series and calendar features with three hidden layers.

As baselines for the general forecasting result, we choose two linear regression models that forecasts the electricity demand based on selected past electricity demand or rather electricity demand difference values and calendar features. For the electricity demand, we use the same data as for the *naive* data represen-

tation. Analog for the electricity demand differences, we use the same data as for the *naive differences* data representation. Regarding the calendar features, we use the same calendar features as for the FCN and CNN models. Thus, the linear regression for the electricity demand is defined as

$$\hat{x}_{k+h} = \alpha + \sum_{i=0}^{167} \beta_i x_{k-i} + \sum_{j=1}^{8} \gamma_j c_{k+h,j}, \tag{5}$$

where $x \in X$ is the electricity demand, $c \in C$ the calendar features, and h the forecast horizon. For the electricity demand differences the linear regression is defined as

$$\hat{\Delta}(x_{k+h}, x_k) = \alpha + \sum_{i=0}^{167} \beta_i (x_{k-i} - x_{k-i-h}) + \sum_{j=1}^{8} \gamma_j c_{k+h,j}, \tag{6}$$

where $x \in X$ is the electricity demand, $c \in C$ the calendar features, and h the forecast horizon.

To apply the mentioned architectures and benchmarks, we run the following setup: Regarding the train-validation-test split, we use the years 2015 to 2017 for training, 2018 for validation, and 2019 for testing. With regard to the forecast horizon, we forecast a specific hour for each model, i.e. one-hour, one-day, and one-week ahead. We evaluate the forecast horizons with the Mean Absolute Error (MAE) defined by $\frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{N}$, where y_i is the ground truth and $\hat{y}_i$ the prediction. For each combination of the four evaluated data representations and the three forecast horizons, we run the respective network with ten different seeds. We report the mean and standard deviation of these ten runs and use this mean to calculate the relative advantage in percent compared to the *naive* data representation, i.e. $\frac{MAE_{compare}}{MAE_{naive}} - 1$.

The whole experimental setup is implemented in Python. For this purpose, we use PyTorch [26] for realizing the DNN architectures and pyWATTS [27] for defining a reproducible and reusable pipeline. The implementation is available on GitHub[1].

[1] `https://github.com/KIT-IAI/SmartDataRepresentations`

Table 1: Forecasting MAE results of the four evaluated data representations in GW for the three selected forecast horizons. For the *naive* and *naive differences* data representations, we use the FCN, while we apply the CNN for the *reshaped* and *reshaped differences* data representations.

Forecast Horizon	Data Representations			
	Naive	Naive Differences	Reshaped	Reshaped Differences
One-Hour	0.420 $\pm$ 0.025	**0.376 (-10.3%) $\pm$ 0.027**	0.531 (+26.5%) $\pm$ 0.015	0.385 (-8.3%) $\pm$ 0.003
One-Day	**1.197 $\pm$ 0.128**	1.289 (+7.6%) $\pm$ 0.118	1.209 (+0.9%) $\pm$ 0.017	1.240 (+3.5%) $\pm$ 0.010
One-Week	**1.677 $\pm$ 0.084**	1.775 (+5.8%) $\pm$ 0.102	1.742 (+3.9%) $\pm$ 0.028	1.820 (+8.5%) $\pm$ 0.012

3.4 Results

In this section, we present the results of the evaluated data representations regarding the energy time series forecast. For the 2019 test data, we report the results for the one-hour, the one-day, and the one-week ahead forecast (see Table 1). In addition, for all forecast horizons, we consider the *naive* data representation as a benchmark. More specifically, we compare the mean of all runs of each data representation to this benchmark before comparing the best data representations to the baseline.

For the one-hour ahead forecast, the *naive differences* data representation is best with an improvement of 10%. In contrast to this data representation, the *reshaped* data representation reduces the forecasting accuracy up to 27%. The *reshaped differences* data representation performs similarly as the *naive differences* data representation with an improvement of 8% compared to the *naive* data representation.

For the one-day ahead forecast, the *naive* data representation performs best, while the *reshaped* data representation performs quite similar with a higher MAE of 1%. The *naive differences*, and *reshaped differences* data representations reduce the forecasting performance between 3% to 8%.

For the one-week ahead forecast, the *naive* data representation is the best performing data representation. The *reshaped* data representation has a 4% higher MAE. However, both data representations based on differences, namely the *naive differences* and the *reshaped differences* data representations, perform worse than the *naive* data representation with higher MAEs between 5% and 9%.

Table 2: Forecasting MAE results of the baseline and the best evaluated data representations in GW for the three selected forecast horizons. For the baseline, we employ a linear regression. For the *naive* data representation, we use the FCN, while we apply the CNN for the *reshaped differences* data representation.

Forecast Horizon	Baselines		Best Data Representation	
	Naive	*Naive Differences*		
One-Hour	0.506	**0.470**	0.376 (-20.0%) $\pm$ 0.027	*Reshaped Differences*
One-Day	**1.639**	1.691	1.197 (-27.0%) $\pm$ 0.128	*Naive*
One-Week	**1.975**	2.128	1.677 (-15.1%) $\pm$ 0.084	*Naive*

For all forecast horizons, the best evaluated data representation performs better than the selected baselines. For example, the *reshaped differences* data representation improves the forecasting accuracy by at least 20% for the one-hour ahead forecast. The *naive* data representation obtains a 27% better forecasting accuracy for the one-day ahead forecast compared to the best performing baseline. For the one-week ahead forecast, the *naive* data representation achieves an 15% better forecasting accuracy. We additionally run the *naive* and *naive differences* data representation experiments on a simple Multilayer Perceptron (MLP) with one hidden layer consisting of ten neurons and achieve similar results as for the linear regression model.

4 Discussion

This section discusses the results of the energy forecasting use case. Our results show that the evaluated data representations, despite essentially containing the same information, result in different accuracies in the energy forecasting use case. Although the *reshaped* and the *reshaped differences* data representations are based on a similar concept, only the *reshaped differences* data representation outperforms the naive benchmark in the single case of the one-hour ahead forecast. Furthermore, depending on the forecast horizon, the data representations perform differently. Nevertheless, there is no data representation that offers the best forecasting accuracy for all evaluated forecast horizons. For example, the *naive differences* data representation is the best data representation for one-hour ahead forecasts. However, for the one-day and one-week ahead forecast, the *naive* data representation performs best. As a consequence,

it should be investigated in which way various data representations influence the forecasting accuracy of Deep Neural Networks (DNNs) and how they are affected by different forecast horizons and architectures given a specific use case.

In the evaluated energy forecasting use case, we consider four data representations and two DNN architectures. For these data representations and architectures, our results show that the forecasting accuracy varies. However, the ambiguous results do not allow for a general statement regarding the impact of data representations on the performance of DNNs. Moreover, the considered use case does not provide insights on the transferability of the results to other not yet evaluated data representations and to use cases from other domains. In addition, this work only investigates the impact of data representations on the accuracy of DNNs but does not examine other relevant metrics such as computational effort, robustness, or interpretability. Altogether, one should investigate further data representations, architectures, and use cases with regard to various metrics.

5 Conclusion

The present paper analyzes the impact of data representations on the performance of Deep Neural Networks (DNNs) at the example of energy time series forecasting. Based on an overview of exemplary data representations, we select four different data representations, namely the *naive*, the *naive differences*, the *reshaped*, and the *reshaped differences* data representation. We evaluate these data representations using two different DNN architectures and three forecasting horizons on real-world energy time series.

The results show that, depending on the forecast horizon, the same data representations can have a positive or negative impact on the accuracy of DNNs in the considered energy forecasting use case. Overall, there is no best performing data representation for all forecasting horizons. For example, reshaping the energy time series decreases the forecasting accuracy up to 27% compared to the *naive* data representation. However, reshaping the differences of the energy

time series is beneficial for one-hour ahead forecasts and yields an up to 8% higher forecasting accuracy compared to the *naive* data representation.

In future work, we plan to investigate the impact of various data representations on DNNs for datasets from different domains and other DNN architectures. In the case of energy forecasting, for example, data representations for additional information like weather could be investigated. For datasets from the given and other domains, the impact of data representations could also be evaluated regarding the problem complexity and dataset size as well as other metrics such as computational effort and interpretability. Furthermore, one could verify the results reported in the present paper using different DNN architectures and datasets. Lastly, it could be interesting to also investigate data representations within DNNs and probabilistic data representations.

Acknowledgements

This project is funded by the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI, the Helmholtz Association under the Program "Energy System Design", and the German Research Foundation (DFG) under Germany's Excellence Strategy – EXC number 2064/1 – Project number 390727645.

References

[1] Y. Liu, Y. Zhong, and Q. Qin, "Scene Classification Based on Multiscale Convolutional Neural Network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 7109–7121, Dec. 2018.

[2] P. M. Kamble and R. S. Hegadi, "Handwritten Marathi Character Recognition Using R-HOG Feature," *Procedia Computer Science*, vol. 45, pp. 266–274, Jan. 2015.

[3] H. Zhu, X. Chen, W. Dai, K. Fu, Q. Ye, and J. Jiao, "Orientation robust object detection in aerial images using deep convolutional neural

network," in *IEEE International Conference on Image Processing (ICIP)*, pp. 3735–3739, IEEE, Sept. 2015.

[4] G. Jader, J. Fontineli, M. Ruiz, K. Abdalla, M. Pithon, and L. Oliveira, "Deep Instance Segmentation of Teeth in Panoramic X-Ray Images," in *31st Conference on Graphics, Patterns and Images (SIBGRAPI)*, pp. 400–407, IEEE, Oct. 2018.

[5] T. Scherr, K. Löffler, M. Böhland, and R. Mikut, "Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy," *PLOS ONE*, vol. 15, id. e0243219, Dec. 2020.

[6] K. Faust, S. Bala, R. van Ommeren, A. Portante, R. Al Qawahmed, U. Djuric, and P. Diamandis, "Intelligent feature engineering and ontological mapping of brain tumour histomorphologies by deep learning," *Nature Machine Intelligence*, vol. 1, pp. 316–321, July 2019.

[7] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription," in *IEEE Workshop on Automatic Speech Recognition Understanding*, pp. 24–29, IEEE, Dec. 2011.

[8] D. Zimbra, M. Ghiassi, and S. Lee, "Brand-Related Twitter Sentiment Analysis Using Feature Engineering and the Dynamic Architecture for Artificial Neural Networks," in *49th Hawaii International Conference on System Sciences (HICSS)*, pp. 1930–1938, IEEE, Jan. 2016.

[9] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric Deep Learning on Graphs and Manifolds Using Mixture Model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, IEEE, July 2017.

[10] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[11] P. Jahankhani, V. Kodogiannis, and K. Revett, "EEG Signal Classification Using Wavelet Feature Extraction and Neural Networks," in *IEEE John*

Vincent Atanasoff 2006 International Symposium on Modern Computing (JVA'06), pp. 120–124, IEEE, Oct. 2006.

[12] J. Jia and H.-C. Chua, "Solving two-spiral problem through input data representation," in *Proceedings of ICNN - International Conference on Neural Networks*, vol. 1, pp. 132–135, IEEE, Nov. 1995.

[13] B. Heidrich, M. Turowski, N. Ludwig, R. Mikut, and V. Hagenmeyer, "Forecasting energy time series with profile neural networks," in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, pp. 220–230, Association for Computing Machinery, June 2020.

[14] J. G. Ordiano, S. Waczowicz, V. Hagenmeyer, and R. Mikut, "Energy forecasting tools and services," *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 2, id. e1235, Dec. 2018.

[15] M. Imani and H. Ghassemian, "Sequence to Image Transform Based Convolutional Neural Network for Load Forecasting," in *27th Iranian Conference on Electrical Engineering (ICEE)*, pp. 1362–1366, IEEE, Apr. 2019.

[16] G. Hafeez, K. S. Alimgeer, and I. Khan, "Electric load forecasting based on deep learning and optimized by heuristic algorithm in smart grid," *Applied Energy*, vol. 269, id. 114915, July 2020.

[17] C. Chinrungrueng and C. Sequin, "Optimal adaptive k-means algorithm with dynamic adjustment of learning rate," *IEEE Transactions on Neural Networks*, vol. 6, pp. 157–169, Jan. 1995.

[18] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, pp. 191–203, Jan. 1984.

[19] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," *ACM SIGMOD Record*, vol. 25, pp. 103–114, June 1996.

[20] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “OPTICS: ordering points to identify the clustering structure,” *ACM SIGMOD Record*, vol. 28, pp. 49–60, June 1999.

[21] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp. 226–231, Aug. 1996.

[22] K. Pearson, “On lines and planes of closest fit to systems of points in space,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, pp. 559–572, Nov. 1901.

[23] P. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Psychology Press, 2 ed., Apr. 2014.

[24] M. A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” *AIChE Journal*, vol. 37, no. 2, pp. 233–243, Feb. 1991.

[25] F. Wiese, I. Schlecht, W.-D. Bunke, C. Gerbaulet, L. Hirth, M. Jahn, F. Kunz, C. Lorenz, J. Mühlenpfordt, J. Reimann, and W.-P. Schill, “Open Power System Data – Frictionless data for electricity system modelling,” *Applied Energy*, vol. 236, pp. 401–409, Feb. 2019.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Dec. 2019.

[27] B. Heidrich, A. Bartschat, M. Turowski, O. Neumann, K. Phipps, S. Meisenbacher, K. Schmieder, N. Ludwig, R. Mikut, and V. Hagenmeyer, “pyWATTS: Python Workflow Automation Tool for Time Series,” *arXiv*, id. 2106.10157v1, June 2021.

Systemidentifikation und Simulation nichtlinearer dynamischer Systeme mit Gaußschen Prozessmodellen mit näherungsweiser Rückführung normalverteilter Ausgangsgrößen

Lars Kistner, Andreas Kroll

Fachgebiet Mess- und Regelungstechnik, Universität Kassel
Mönchebergstr. 7, 34125 Kassel
E-Mail: {lars.kistner, andreas.kroll}@mrt.uni-kassel.de

1 Einführung

In diesem Beitrag werden Gaußsche Prozessmodelle (GPM) zur Systemidentifikation und zur Simulation nichtlinearer dynamischer Systeme vorgestellt. Gaußsche Prozessmodelle sind in Standardform für deterministische Regressoren definiert und liefern eine Normalverteilung als Ausgangsgröße. Durch die Varianz einer Prädiktion lässt sich, im Gegensatz zu vielen anderen Methoden aus dem Bereich des maschinellen Lernens und der Systemidentifikation, ihre Vertrauenswürdigkeit abschätzen. Im Falle einer Simulation von dynamischen Systemen in Output-Error-Anordnung mit Gaußschen Prozessmodellen müssen daher Normalverteilungen der Ausgangsgröße verzögert als Eingangsgröße zurückgeführt werden. In diesem Beitrag werden Verfahren aus der Literatur vorgestellt, um bei Modellen mit stochastischen Eingangsgrößen zu arbeiten und diese Konstellation auf die Eignung für die Systemidentifikation hin zu untersuchen. Die Untersuchung erfolgt dabei an einem künstlichen System und einem realen Eintank-Laborsystem mit zwei Eingangsgrößen und einer Ausgangsgröße.

2 Gaußsche Prozessmodelle

Gaußsche Prozessmodelle gehören zur Klasse der kernelbasierten, nichtparametrischen Modelle und sind besonders für hochdimensionale Probleme bei verhältnismäßig wenig Daten geeignet. Mit GPM lassen sich beliebige statische Nichtlinearitäten der Form $y = f(\boldsymbol{x})$ darstellen. Die Annahme dabei ist, dass Ähnlichkeiten im Eingangsraum $\boldsymbol{x}$ sich auch im Ausgangsraum y wiederfinden. Um diese Ähnlichkeit zu beschreiben, wird eine Kernel- oder Kovarianzfunktion verwendet.

Bei GPM wird angenommen, dass jede endliche Menge von N Auswertungen einer Zufallsfunktion $[f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), ..., f(\boldsymbol{x}_N)]$ einer multivariaten Gaußverteilung folgt und als Wahrscheinlichkeitsdichtefunktion geschrieben werden kann [3]:

$$p(\boldsymbol{f}(\boldsymbol{X})|\boldsymbol{X}) = \mathcal{N}\left(\boldsymbol{m}_f, \boldsymbol{\Sigma}_f\right) \tag{1}$$

$\boldsymbol{X}$ ist die Matrix bestehend aus allen Vektoren der Eingangsgrößen ($\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]^T$). $\boldsymbol{f}(\boldsymbol{X})$ ist der Vektor der ungestörten Beobachtungen. $\boldsymbol{m}_f$ sind die Erwartungswerte und $\boldsymbol{\Sigma}_f$ die Kovarianzmatrix des GPM. Eine Funktion, die einen Gauß-Prozess darstellt, wird geschrieben als:

$$\boldsymbol{f}(\boldsymbol{X}) \sim \mathcal{GP}\left(\boldsymbol{m}_f, \boldsymbol{\Sigma}_f\right) \tag{2}$$

Da meistens keine Informationen über den Erwartungswert eines Gauß-Prozesses vorliegen, wird $\boldsymbol{m}_f \equiv \boldsymbol{0}$ angenommen. Die Kovarianzmatrix $\boldsymbol{\Sigma}_f$ wird über die Kernelfunktion $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ erzeugt und dann Kernelmatrix $\boldsymbol{K}$ genannt. Da im Kernel ein normalverteilte Rauschannahme getroffen wird, wird im Zusammenhang mit der Kernelmatrix $\boldsymbol{K}$ im Folgenden $\boldsymbol{y}$ verwendet. Die getroffene Annahme ist $\boldsymbol{y} = f(\boldsymbol{X}) + \mathcal{N}(0, \sigma_n)$. Die einzelnen Beobachtungen werden als y_1 bis y_N notiert. Unter Beachtung von (2) ergibt sich so:

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) \tag{3}$$

Der gebräuchlichste Kernel ist der Squared-Exponential (SE)-Kernel. Wird er mit einem freien Parameter pro Eingangsdimension $d \in 1...D$ ausgestattet, so

wird er als Automatic-Relevance-Determination Squared-Exponential (ARD-SE) Kernel bezeichnet und wird beschrieben durch:

$$k(\boldsymbol{x}_i,\boldsymbol{x}_j) = \sigma_f^2 \cdot \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_{i,d}-x_{j,d})^2}{l_d^2}\right) + \sigma_n^2\delta_{ij} \tag{4}$$

σ_f, l_d und σ_n sind freie Hyperparameter. σ_f ist ein Skalierungsfaktor für den Kernel, l_d ein Skalierungsfaktor pro Eingangsdimension und σ_n^2 beschreibt die geschätzte Varianz, welche auf den Beobachtungen erwartet wird. Ein GPM wird durch Anpassung seiner Hyperparameter trainiert, sodass die Beobachtungen $(\boldsymbol{X},\boldsymbol{y})$ bestmöglich erklärt werden. Eine Vorgehensweise dazu ist die Hyperparameter über Maximierung der Marginal-Log-Likelihood:

$$\ln p(\boldsymbol{y}|\boldsymbol{X}) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(\det\boldsymbol{K}) - \frac{1}{2}\boldsymbol{y}^T\boldsymbol{K}^{-1}\boldsymbol{y} \tag{5}$$

auszulegen. Da alle Beobachtungen und auch die Vorhersage eines GPM als normalverteilte Zufallsvariable aufgefasst werden, gilt folgende Verbundverteilung der Zufallsvariablen $\boldsymbol{y}$ und y_*:

$$\begin{pmatrix}\boldsymbol{y}\\ y_*\end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}k(\boldsymbol{X},\boldsymbol{X}) & k(\boldsymbol{X},\boldsymbol{x}_*)\\ k(\boldsymbol{x}_*,\boldsymbol{X}) & k(\boldsymbol{x}_*,\boldsymbol{x}_*)\end{bmatrix}\right) \tag{6}$$

$$\begin{pmatrix}\boldsymbol{y}\\ y_*\end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{K} & \boldsymbol{k}_*\\ \boldsymbol{k}_*^T & k_{**}\end{bmatrix}\right) \tag{7}$$

$$\begin{pmatrix}y_1\\ y_2\\ \dots\\ y_N\\ y_*\end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}k(\boldsymbol{x}_1,\boldsymbol{x}_1) & k(\boldsymbol{x}_1,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_1,\boldsymbol{x}_N) & k(\boldsymbol{x}_1,\boldsymbol{x}_*)\\ k(\boldsymbol{x}_2,\boldsymbol{x}_1) & k(\boldsymbol{x}_2,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_2,\boldsymbol{x}_N) & k(\boldsymbol{x}_2,\boldsymbol{x}_*)\\ \dots & \dots & \dots & \dots & \dots\\ k(\boldsymbol{x}_N,\boldsymbol{x}_1) & k(\boldsymbol{x}_N,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_N,\boldsymbol{x}_N) & k(\boldsymbol{x}_3,\boldsymbol{x}_*)\\ k(\boldsymbol{x}_*,\boldsymbol{x}_1) & k(\boldsymbol{x}_*,\boldsymbol{x}_2) & \dots & k(\boldsymbol{x}_*,\boldsymbol{x}_N) & k(\boldsymbol{x}_*,\boldsymbol{x}_*)\end{bmatrix}\right) \tag{8}$$

Dabei ist $\boldsymbol{x}_*$ der Vektor der Eingangsgrößen für den zu prädizierenden Wert y_*. Die bedingte Wahrscheinlichkeitsverteilung für y_* gegeben $\boldsymbol{x}_*,\boldsymbol{X}$ und $\boldsymbol{y}$ wird

damit zu:

$$p(y_*|\boldsymbol{x}_*,\boldsymbol{X},\boldsymbol{y}) \sim \mathcal{N}(k(\boldsymbol{x}_*,\boldsymbol{X})k(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{y}, \\ k(\boldsymbol{x}_*,\boldsymbol{x}_*) - k(\boldsymbol{x}_*,\boldsymbol{X})k(\boldsymbol{X},\boldsymbol{X})^{-1}k(\boldsymbol{X},\boldsymbol{x}_*)) \quad (9)$$

$$\sim \mathcal{N}(\boldsymbol{k}_*\boldsymbol{K}^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_*\boldsymbol{K}^{-1}\boldsymbol{k}_*^T) \quad (10)$$

Für einen beliebigen Regressorenvektor $\boldsymbol{x}_*$ ist der Erwartungswert der Prädiktion y_* somit gegeben durch:

$$\mathrm{E}(y_*) = \mu(\boldsymbol{x}_*) = \boldsymbol{k}_*\boldsymbol{K}^{-1}\boldsymbol{y} \quad (11)$$

und die Varianz durch:

$$\mathrm{Var}(y_*) = \sigma^2(\boldsymbol{x}_*) = k_{**} - \boldsymbol{k}_*\boldsymbol{K}^{-1}\boldsymbol{k}_*^T \quad (12)$$

Für eine detaillierte Herleitung und weitreichendere Erklärungen sei auf die Literatur verwiesen [1, 18, 3].

3 Systemidentifikation mit Gaußschen Prozessmodellen

Gaußsche Prozessmodelle sind in Standardform für deterministische Regressoren definiert (vgl. Bild 1 links). Für Eingangsgrößen, welche beim Training mit einem mittelwertfreien Rauschen behaftet sind, kann diese Anforderung verletzt werden ohne größere Auswirkungen auf die Unsicherheitsabschätzung der Ausgangsgröße zu haben. Im Falle einer Simulation oder beim Training in Output-Error (OE)-Anordnung von dynamischen Systemen mit Gaußschen Prozessmodellen müssen jedoch Normalverteilungen der Ausgangsgröße verzögert als Eingangsgröße zurückgeführt werden (vgl. Bild 1 rechts). Eine Vernachlässigung der Unsicherheit durch Einschränkung auf den Erwartungswert als Eingangsgröße führt zu unrealistisch kleinen Unsicherheiten der Prädiktion mit fortschreitender Simulationszeit.

In [6] wurde erstmals eine Taylorreihenentwicklung und eine Monte-Carlo-Integration vorgestellt, um GPM mit normalverteilten Eingängen zu realisie-

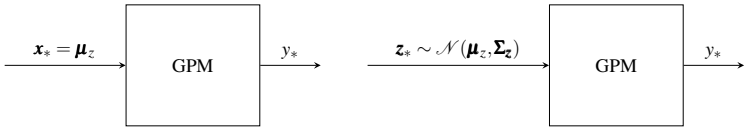


Bild 1: GPM mit **links:** deterministischem Regressor **rechts**: normalverteilter Zufallsgröße als Regressor (stochastisch)

ren. In der Arbeit [5] wurde darüber hinaus eine näherungsweise Lösung mit exakten Momenten vorgestellt. Stochastische Eingänge werden dabei über das Lösen des Integrals:

$$p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) = \int p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{z}_*) p(\boldsymbol{z}_* \mid \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \, d\boldsymbol{z}_* \tag{13}$$

mit:
$$p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{z}_*) = \frac{1}{\sqrt{2\pi\sigma^2(\boldsymbol{z}_*)}} \exp\left(-\frac{1}{2} \frac{(y_* - \mu(\boldsymbol{z}_*))^2}{\sigma^2(\boldsymbol{z}_*)}\right) \tag{14}$$

realisiert (vgl. (10) für den Fall eines deterministischen Regressors). Um den Unterschied zum deterministischen Regressor $\boldsymbol{x}_*$ zu verdeutlichen, wird der Zufallsvektor im Folgenden als $\boldsymbol{z}_*$ mit den Erwartungswerten $\boldsymbol{\mu}_z$ und der Kovarianzmatrix $\boldsymbol{\Sigma}_z$ bezeichnet. $\mu(\boldsymbol{z}_*)$ bezeichnet dabei die Funktion für den Erwartungswert eines GPM gegeben durch (11) und $\sigma^2(\boldsymbol{z}_*)$ die Varianz gegeben durch (12).

Durch die Berechnung des Integrals (13) lässt sich die Information über die Unsicherheit der Prädiktion auch bei der Simulation mittels Einschrittprädiktor näherungsweise erhalten. In Bild 2 wird das Vorgehen für die ersten drei Simulationsschritte im Blockschaltbild gezeigt.

Die neue prädiktive Verteilung ist wegen der Integration über den Regressorenvektor nicht mehr gaußverteilt und daher nur noch approximativ lösbar [5]. Im Folgenden werden drei Verfahren vorgestellt, um (13) zu lösen.

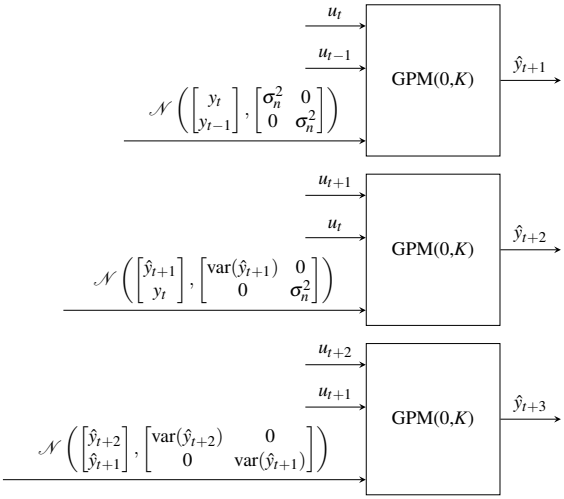


Bild 2: Vorgehen für Propagation der Unsicherheit bei der Simulation eines dynamischen Modells mit zwei verzögerten Ein- und Ausgangsgrößen als Regressoren mittels Einschrittprädiktor für die ersten drei Simulationsschritte

3.1 Näherung mittels Monte-Carlo-Integration

Mittels Monte-Carlo-Integration [8] können hochdimensionale Integrale der Form:

$$\mathscr{A} = \int_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x} \tag{15}$$

approximativ gelöst werden. Wobei $p(\boldsymbol{x})$ das statistische Gewicht und $f(\boldsymbol{x})$ den Wert zum Zustand $\boldsymbol{x}$ aus dem Zustandsraum Ω beschreibt. Im Fall einer Stichprobe $[\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_S] \in \Omega$, welche die Häufigkeitsverteilung von $p(\boldsymbol{x})$ abbildet, gilt für große S:

$$\mathscr{A} \approx \frac{1}{S} \sum_{i=1}^{S} f(\boldsymbol{s}_i) \tag{16}$$

Übertragen auf GPM wird eine Stichprobe S benötigt, welche die Verteilung der Regressoren näherungsweise abbildet:

$$[\boldsymbol{s}_1, \boldsymbol{s}_2, ..., \boldsymbol{s}_S] \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \tag{17}$$

damit kann (13) näherungsweise durch:

$$p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z) \approx \frac{1}{S} \sum_{i=1}^{S} p(y_* \mid \boldsymbol{X}, \boldsymbol{y}, \boldsymbol{s}_i) = \frac{1}{S} \sum_{i=1}^{S} \mathcal{N}\left(\mu(\boldsymbol{s}_i), \sigma^2(\boldsymbol{s}_i)\right) \tag{18}$$

gelöst werden. $\mu(\boldsymbol{s}_i)$ bezeichnet dabei die Formel für den Erwartungswert eines GPM gegeben durch (11) und $\sigma^2(\boldsymbol{s}_i)$ die Varianz gegeben durch (12).

3.2 Näherung mittels Taylorreihenentwicklung

Auch eine Taylorreihenentwicklung kann zur Approximation des Integrals genutzt werden. Wie in [6] gezeigt wird, liefert ein Taylorpolynom erster Ordnung für den Erwartungswert keine zusätzliche Korrektur über der nullten Ordnung, da die Funktionsableitungen zu Null werden. Daher bleibt der Erwartungswert näherungsweise bei:

$$E(y_*) = \mu(\boldsymbol{z}_*) \approx k(\boldsymbol{\mu}_z, \boldsymbol{X}) \boldsymbol{K}^{-1} \boldsymbol{y} \tag{19}$$

Für die Varianz wird in [6] ein Taylorpolynom zweiter Ordnung gewählt:

$$\begin{aligned} \mathrm{Var}(y_*) = \sigma^2(\boldsymbol{z}_*) \approx \sigma^2(\boldsymbol{\mu}_z) + \frac{1}{2} \mathrm{tr} \left(\left. \frac{\partial^2 \sigma^2(\boldsymbol{z}_*)}{\partial \boldsymbol{z}_* \partial \boldsymbol{z}_*^T} \right|_{\boldsymbol{z}_* = \mu_z} \Sigma_z \right) \\ + \left. \frac{\partial \mu(\boldsymbol{z}_*)}{\partial \boldsymbol{z}_*} \right|_{\boldsymbol{z}_* = \boldsymbol{\mu}_z}^{T} \Sigma_z \left. \frac{\partial \mu(\boldsymbol{z}_*)}{\partial \boldsymbol{z}_*} \right|_{\boldsymbol{z}_* = \boldsymbol{\mu}_z} \end{aligned} \tag{20}$$

$$\text{mit:} \quad \frac{\partial \mu(\boldsymbol{z}_*)}{\partial z_{*d}} = \frac{\partial \boldsymbol{k}(\boldsymbol{z}_*)}{\partial z_{*d}} \boldsymbol{K}^{-1} \boldsymbol{y}$$

$$\frac{\partial^2 \sigma^2(\boldsymbol{z}_*)}{\partial z_{*d} \partial z_{*e}} = \frac{\partial^2 k(\boldsymbol{z}_*, \boldsymbol{z}_*)}{\partial z_{*d} \partial z_{*e}} - 2 \frac{\partial \boldsymbol{k}(\boldsymbol{z}_*, \boldsymbol{K})}{\partial z_{*d}} \boldsymbol{K}^{-1} \frac{\partial \boldsymbol{k}(\boldsymbol{K}, \boldsymbol{z}_*)}{\partial z_{*e}} - 2 \frac{\partial^2 \boldsymbol{k}(\boldsymbol{z}_*, \boldsymbol{K})}{\partial z_{*d} \partial z_{*e}} \boldsymbol{K}^{-1} \boldsymbol{k}(\boldsymbol{K}, \boldsymbol{z}_*)$$

Die partiellen Ableitungen müssen nach Regressordimension $d, e \in 1...D$ erfolgen und sind vom Kernel abhängig. Für eine detaillierte Herleitung sei auf [5] verwiesen.

3.3 Näherung mittels exakter Momente

Wie [5] zeigt, ist es für den ARD-SE-Kernel möglich, das Integral (13) für stochastische Regressoren exakt zu bestimmen. Dabei wird das Integral (13) analytisch exakt gelöst, jedoch nicht die Erwartungswert- $\mu(\boldsymbol{z}_*)$ und Varianzfunktion $\sigma^2(\boldsymbol{z}_*)$. Der Erwartungswert ist gegeben durch:

$$E(y_*) = \mu(\boldsymbol{z}_*) = (\boldsymbol{K}^{-1}\boldsymbol{y})^T \boldsymbol{q} \tag{21}$$

$$\text{mit:} \quad \boldsymbol{q}^T = [q_1, q_2, ..., q_N],$$

$$q_i = \sigma_f^2 \sqrt{|\boldsymbol{\Sigma}_z \boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|} \exp\left(-\frac{1}{2} (\boldsymbol{x}_i - \boldsymbol{\mu}_z)^T (\boldsymbol{\Sigma}_z + \boldsymbol{\Lambda})^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_z) \right)$$

Die Varianz kann mit:

$$\mathrm{Var}(y_*) = \sigma^2(\boldsymbol{z}_*) = (\boldsymbol{K}^{-1}\boldsymbol{y})^T \boldsymbol{Q} (\boldsymbol{K}^{-1}\boldsymbol{y}) - \mu(\boldsymbol{z}_*) + \sigma_f^2 - \mathrm{tr}\left(\boldsymbol{K}^{-1}\boldsymbol{Q}\right) + \sigma_n^2 \tag{22}$$

$$\text{mit:} \quad Q_{ij} = \frac{k(\boldsymbol{z}_i, \boldsymbol{\mu}_z) k(\boldsymbol{\mu}_z, \boldsymbol{z}_j)}{\sqrt{|2\boldsymbol{\Sigma}_z \boldsymbol{\Lambda}^{-1} + \boldsymbol{I}|}} \exp\left(-\frac{1}{2} \left(\frac{1}{2}(\boldsymbol{z}_i + \boldsymbol{z}_j) - \boldsymbol{\mu}_z \right)^T \cdot \left(\boldsymbol{\Sigma}_z + \frac{1}{2}\boldsymbol{\Lambda}^2 \right)^{-1} \boldsymbol{\Sigma}_z \boldsymbol{\Lambda}^{-1} \left(\frac{1}{2}(\boldsymbol{z}_i + \boldsymbol{z}_j) - \boldsymbol{\mu}_z \right) \right)$$

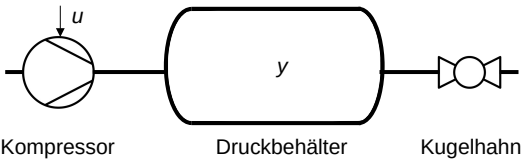


Bild 3: Schematischer Aufbau als Rohrleitungs- und Instrumentierungsschema

$$\text{mit:} \qquad \mathbf{\Lambda^{-1}} = \begin{bmatrix} l_1^{-2} & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & l_D^{-2} \end{bmatrix}$$

bestimmt werden. Für die vollständige Prädiktion, welche Erwartungswert und Varianz beinhaltet, müssen $\boldsymbol{q} \in \mathbb{R}_{N\times 1}$ und $\boldsymbol{Q} \in \mathbb{R}_{N\times N}$ bestimmt werden. $\mathbf{\Lambda}^{-1}$ ist der „Parametervektor“ der Längenparameter des ARD-SE Kernels in Form einer Diagonalmatrix.

3.4 Naive Einschrittprädiktion

Bei der naiven Einschrittprädiktion wird ein GPM, wie bei der klassischen Regression, nur mit deterministischen Regressoren verwendet. Die prädizierten Ausgangsgrößen, welche verzögert als Eingangsgrößen verwendet werden, werden ausschließlich durch ihren Erwartungswert beschrieben.

4 Fallstudie: Kompressor-Druckbehälter-System

Im ersten akademischen Fallbeispiel (Simulationsstudie) wird ein System betrachtet, welches sich an einem Kompressor orientiert, der einen Druckbehälter befüllt. Außerdem befindet sich am Druckbehälter ein teilgeöffneter Kugelhahn. Der Aufbau ist als Rohrleitungs- und Instrumentierungsschema in Bild 3 dargestellt. Die Leistung des Kompressors wird über das Steuersignal u reguliert. Die abhängige Variable ist der Druck y im Druckbehälter. Das folgende Modell hat keinen Anspruch physikalisch akkurat zu sein und ist durch

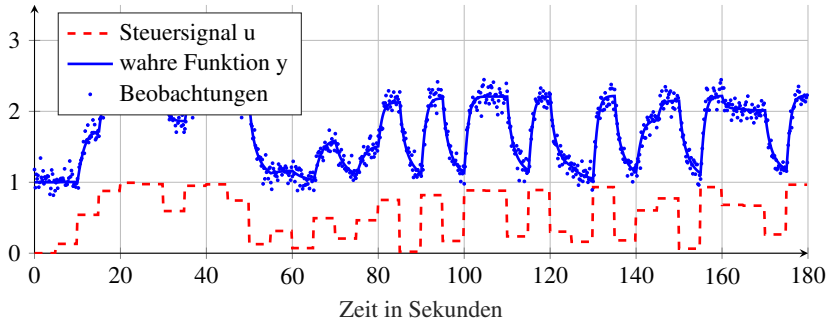


Bild 4: Trainingsdatensatz A, $u \in [0,\ 1]$

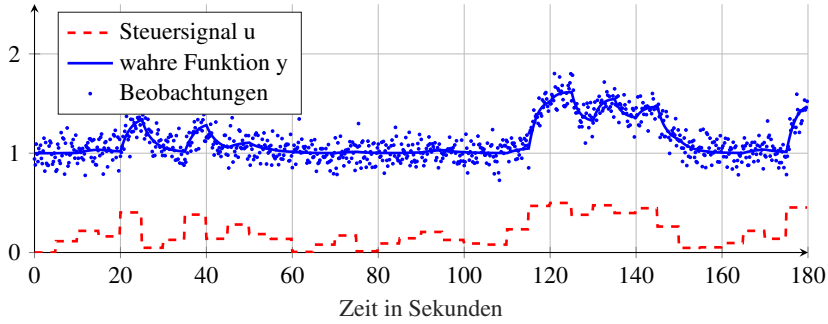


Bild 5: Trainingsdatensatz B, $u \in [0,\ 0{,}5]$

folgende Differentialgleichung gegeben, die bekannte Phänomene beschreiben sollen:

$$\begin{aligned} \dot{y}(t) &= 1 - \exp(-2\pi u(t)^4) - 0{,}1(y(t)^3 - 1) \\ \dot{y}(t) + 0{,}1y(t)^3 &= 1{,}1 - \exp(-2\pi u(t)^4) \end{aligned} \tag{23}$$

Nach der Festlegung auf 6 verzögerte Eingangsgrößen $(y_t, ..., y_{t-6}, u_t, ..., u_{t-6})$ zur Prädiktion von y_{t+1}, Auswahl des ARD-SE-Kernels und des Downhill-Simplex-Suchverfahren zur Optimierung der Kernelparameter werden nun Simulationen anhand von zwei verschiedenen Trainingsdatensätzen verglichen. Die Trainingsdaten wurden erzeugt in dem die Differentialgleichung mit einer Simulationsschrittweite von 0,001 s mit dem expliziten Euler-Verfahren simu-

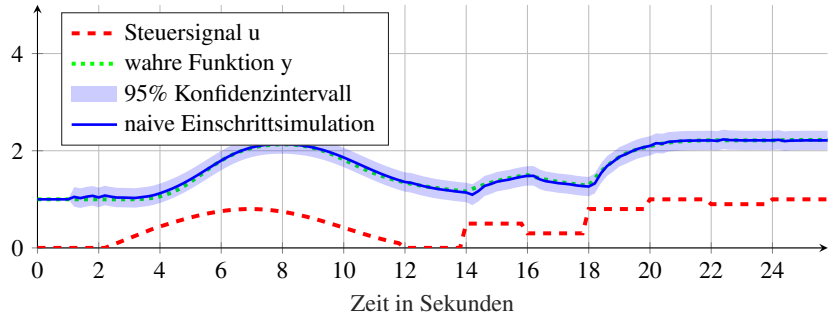


Bild 6: Simulation einer Steuerfolge bei naiver Einschrittsimulation unter Verwendung von Trainingsdatensatz A

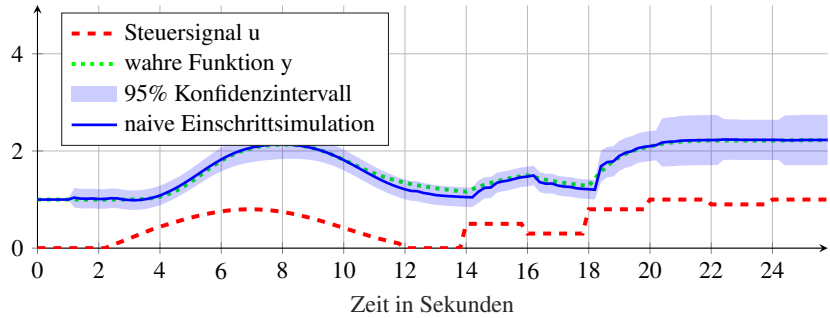


Bild 7: Simulation einer Steuerfolge bei naiver Einschrittsimulation unter Verwendung von Trainingsdatensatz B

liert wurde. Die Datensätze haben eine Länge von 180 s bei einer zeitlichen Auflösung von 0,1 s, die mit einem APRBS-Signal mit einer Haltezeit von 5 s erzeugt wurden. Trainingdatensatz A (siehe Bild 4) deckt einen Wertebereich von $u \in [0,\ 1]$ und Trainingsdatensatz B (siehe Bild 5) von $u \in [0,\ 0{,}5]$ ab. Die Testsignale liegen immer im Bereich von $u \in [0,\ 1]$, sodass beim Trainingsdatensatz B extrapoliert werden muss. Die Simulation für eine gegebene Steuerfolge bei Reduzierung auf den Erwartungswert ist für Datensatz A in Bild 6 und für Datensatz B in Bild 7 zu sehen. Die Modelle verwenden bei allen Betrachtungen die gleichen Kernel-Parameter. Die Erwartungswerte bei den verschiedenen Simulationsverfahren lassen sich rein qualitativ nicht unterscheiden (vgl. Bild 8). Theoretisch sollten die Erwartungswerte von nai-

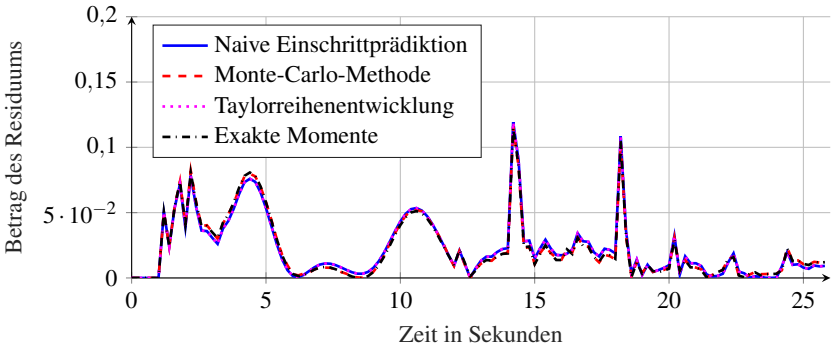


Bild 8: Beträge der Residuen für die verschiedenen Verfahren für die Steuerfolge mit Trainingsdatensatz A aus Bild 6

ver Simulation und Taylor-Approximation einerseits, und exakter Momente- und Monte-Carlo-Methode andererseits die gleichen Ergebnisse liefern, da im ersten Fall dieselben Formeln zugrunde liegen oder wie im zweiten Fall die Monte-Carlo-Methode die Lösung der exakten Momente annähert. Die Verfahren gruppieren sich genauso für beide Trainingsdatensätze, auch wenn die Unterschiede sehr gering sind.

Für einen besseren Vergleich der Auswirkung der Weitergabe der Unsicherheit werden nun vom Datensatz A in Bild 9 und Datensatz B in Bild 10 die 3σ-Unsicherheitsabschätzungen verglichen. Die Bewertung der Unsicherheit gestaltet sich schwieriger. Theoretisch sollte die naive Simulation die geringste Unsicherheit schätzen, dagegen Monte-Carlo-, exakte Momente- und Taylorreihen-Methode näherungsweise die Gleiche, aber immer eine höhere als die naive Simulation. In der Fallstudie schätzt die naive Simulation immer die geringste Unsicherheit, jedoch sind die Unterschiede minimal und stehen in keinem Verhältnis zum Aufwand. Wie klein die Unterschiede sind wird beim Vergleich von Bild 9 und 10 deutlich. Bei der Simulation mit Datensatz B werden bei der Rückführung die Unsicherheiten der Eingangsgrößen besonders groß. Genau in diesem Fall werden die Unterschiede in den Unsicherheiten zwischen den Verfahren besonders klein. Erst bei Betrachtung des Datensatzes A separieren sich die Verfahren wie erwartet (vgl. Abb. 9).

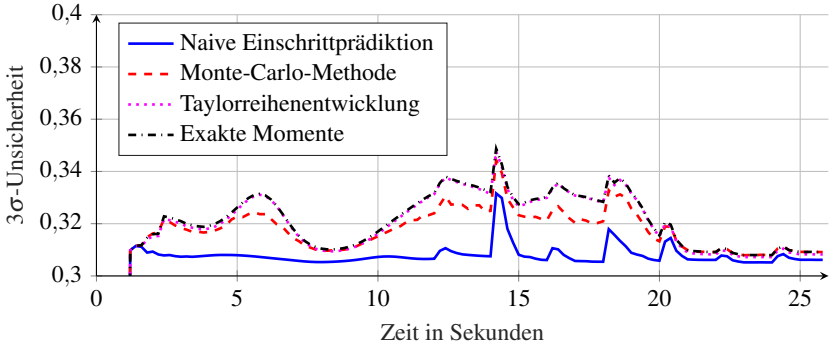


Bild 9: 3σ-Unsicherheitsabschätzung für die verschiedenen Verfahren für die Steuerfolge mit Trainingsdatensatz A aus Bild 6

5 Fallstudie: Eintank-Laborsystem

Für die zweite Fallstudie an einem realen System wurde ein Eintank-System der Modellfabrik μPlant des Fachgebiets Mess- und Regelungstechnik verwendet, welches dem Schema in Bild 11 folgt. Das Wasser wird aus dem Zentraltank mit einer Pumpe in den Tank gefördert. Dabei wird der Durchfluss mit einem magnetisch-induktiven Durchflussmesser (MID) gemessen. Am Ausfluss befindet sich ein Proportionalventil (PV) mit dem der Wasserstrom, welcher zurück in den Zentraltank läuft, eingestellt wird. Der Öffnungsgrad des PV und die Leistung der Pumpe werden über einen Sollwert im Bereich von 0 % bis 100 % angesteuert. Das MID misst den Durchfluss in Litern pro Minute. Die Ansteuerung des PV u_{ventil} und die der Pumpe u_{pumpe} werden als Eingangsgrößen des Eintank-Systems behandelt. Der Zustand des Tanks wird als Volumen des Inventars in Litern erfasst. Die zu prädizierende Größe ist das Volumen des Tankinventars $\hat{y}_{volumen,t+1}$. Untersuchungen haben gezeigt, dass bei dem gewählten Modellansatz (vgl. Bild 12) eine Abtastzeit von $T_0 = 0,7$ s und $p = q = 5$ verzögerte Eingänge gemäß Bild 12 einen guten Kompromiss zwischen der zeitlichen Auflösung, dem Rechenaufwand und der Modellgüte liefern. Bei den Untersuchungen wurde ebenfalls der ARD-SE-Kernel und das Downhill-Simplex-Suchverfahren zur Optimierung der Kernelparameter verwendet. Der

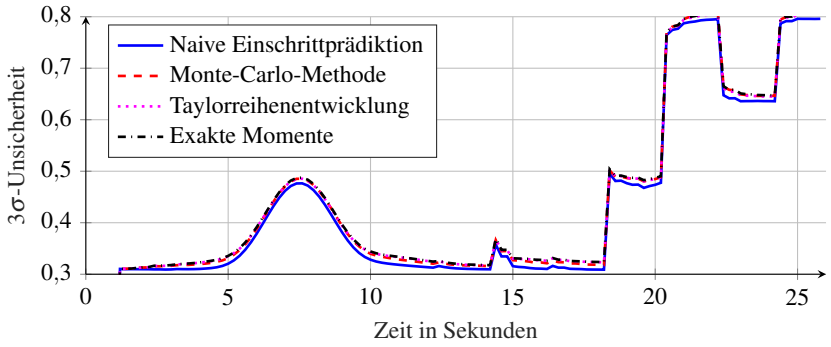


Bild 10: 3σ-Unsicherheitsabschätzung für die verschiedenen Verfahren für die Steuerfolge mit Trainingsdatensatz B aus Bild 7

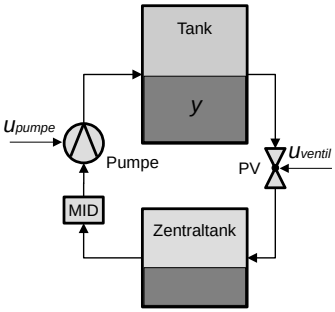


Bild 11: Schematischer Aufbau des Eintank-Systems

Trainingsdatensatz ist 2400 s lang und wurde mittels APRBS-Testsignal mit einer Haltezeit von 10 s erzeugt (vgl. Bild 13).

Als Validierungsdatensatz wurde die Steuerfolge aus Bild 14 verwendet. Die Simulation mit dem 95 % Konfidenzintervall mit einer Rückführung der Unsicherheit mittels Taylorapproximation ist in Bild 15 zu sehen. Die Taylorapproximation ist das einzige Verfahren, bei dem die Beobachtungen im 3σ-Konfidenzintervall bleiben. Die anderen Verfahren unterschätzen die Unsicherheit. Im Gegensatz zur ersten Fallstudie, bei der alle Verfahren plausible Ergebnisse liefern, wird das Verfahren mit den exakten Momenten (auch mit anderen Datensätzen von dem Versuchsaufbau) nach wenigen Schritten numerisch in-

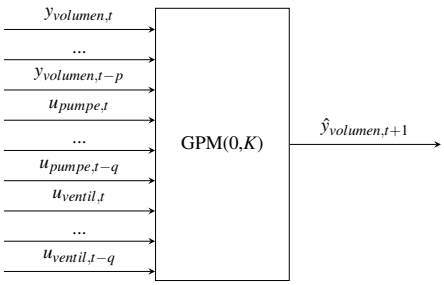


Bild 12: ARX-Modellansatz mit Eingangsgrößen Pumpen-Sollgröße und Ventilöffnung

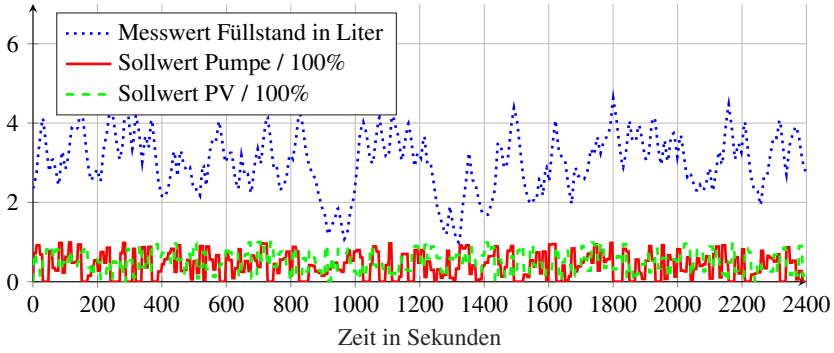


Bild 13: 2400 s Trainingsdatensatz

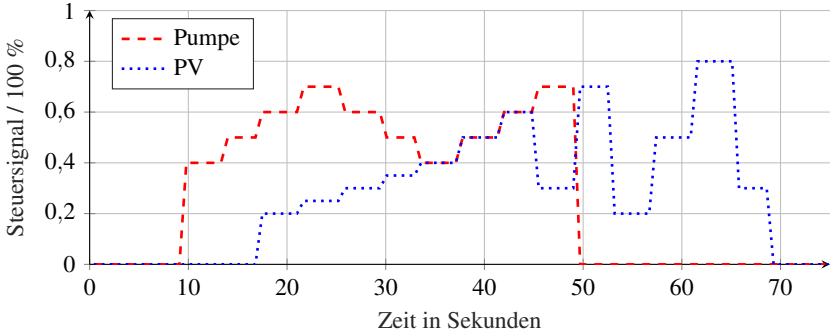


Bild 14: Steuerfolge des Validierungsdatensatzes

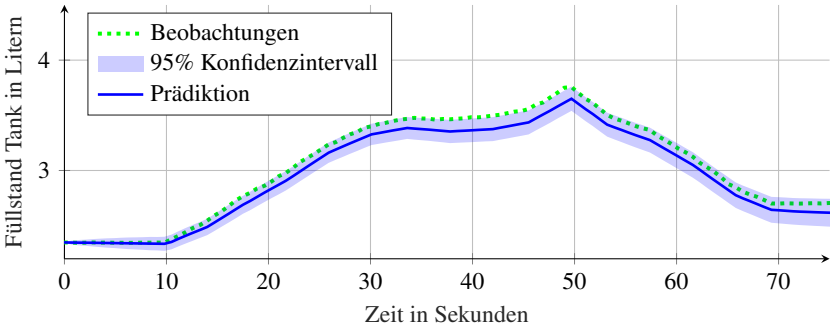


Bild 15: Simulation der Steuerfolge aus Bild 14 bei Rückführung der Unsicherheit mittels Taylorapproximation sowie Messwerte des Füllstands

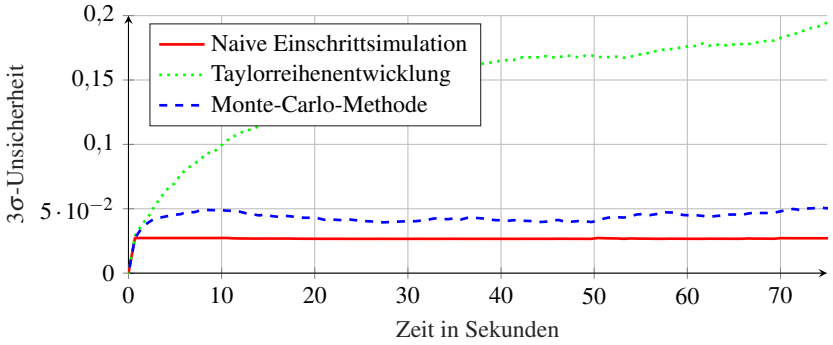


Bild 16: Unsicherheit der verschiedenen Verfahren beim Validierungsdatensatz

stabil. Die 3σ-Unsicherheitsabschätzung der drei anderen Verfahren ist in Bild 16 zu sehen.

6 Zusammenfassung und Ausblick

Im Beitrag wurde in zwei Fallstudien gezeigt, dass Gaußsche Prozessmodelle sich für die nichtlineare Systemidentifikation bei Angabe der Prädiktions-Unsicherheit eignen. Bei der Simulation durch wiederholte Prädiktionen mittels Einschrittprädiktion verliert jedoch die Unsicherheitsabschätzung an Aussagekraft. Die vorgestellten Methoden, die eine Berücksichtigung von Ein-

gangsgrößen mit normalverteilten Unsicherheiten ermöglichen, vergrößern die Unsicherheiten im zweiten Fallbeispiel sehr unterschiedlich, sodass belastbare Aussagen bei der Erkenntnislage nicht möglich sind.

In weiterführenden Arbeiten sollte untersucht werden, ob mit der Rückführung beliebiger Verteilungen (nicht normalverteilte Verteilungsfunktionen), wie sie in [7] vorgeschlagen werden, bessere Ergebnisse erzielt werden können, oder ob die Hyperparameteroptimierung in Output-Error (OE)-Anordnung für die Simulation geeignetere Parameter findet und damit die Aussagefähigkeit der berechneten Unsicherheit verbessert. Aufgrund des Rechenaufwands von GPM und des komplexen Suchraums in OE-Anordnung ist dies ein sehr rechenaufwändiges Problem [4].

Danksagung

Dieser Beitrag wurde von der Deutschen Forschungsgemeinschaft DFG im Rahmen des Schwerpunktprogramms SPP 2086 (KR 3795/8-1) gefördert. Die Autoren danken der DFG für die finanzielle und technische Unterstützung.

Literatur

[1] Carl Edward Rasmussen und Christopher K. I. Williams „Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)“. The MIT Press, 2006, isbn: 026218253X.

[2] Oliver Nelles „Nonlinear System Identification“. 2. Auflage, Berlin Heidelberg: Springer, 2020.

[3] Juš Kocijan „Modelling and Control of Dynamic Systems Using Gaussian Process Models“. Advances in Industrial Control, Cham: Springer International Publishing, 2016, isbn: 3319210203.

[4] Juš Kocijan und Dejan Petelin „Output-Error Model Training for Gaussian Process Models“. Adaptive and Natural Computing Algorithms, Berlin, Heidelberg: Springer, 2011, isbn: 978-3-642-20267-4.

[5] Agathe Girard „Approximate Methods for Propagation of Uncertainty with Gaussian Process“. Dissertation, University of Glasgow, 2004.

[6] Agathe Girard, Carl Rasmussen und Roderick Murray-Smith „Gaussian Process priors with Uncertain Inputs: Multiple-Step-Ahead Prediction“. Advances in neural information processing systems: Proceedings of the 2002 Neural Information Processing Systems Conference, 2002.

[7] Maxim Dolgov und Uwe D. Hanebeck „A Distance-based Framework for Gaussian Processes over Probability Distributions“. arXiv preprint, 2018, doi: arXiv:1809.09193.

[8] Christian P. Robert „Monte Carlo Statistical Methods“. 2. Ausgabe, Springer Texts in Statistics, New York: Springer, 2004, isbn: 9781441919397.

Adaptive Model Predictive Control with Finite Impulse Response Models

Christopher Illg, Tim Decker, Jonas Thielmann, Oliver Nelles

Universität Siegen, Department Maschinenbau, Institut für Mechanik und Regelungstechnik – Mechatronik
Paul-Bonatz-Str. 9-11, 57068, Siegen, Germany
E-Mail: {christopher.illg,tim.decker,oliver.nelles}@uni-siegen.de,
jonas.thielmann@student.uni-siegen.de

Abstract

Controlling nonlinear processes is a challenging task. Model predictive control (MPC) rose in the last decades to the dominating state-of-the-art method for control in the industry. In this work, a closer look at a closed-loop and an open-loop adaptive model predictive control (AMPC) method is taken, since AMPC can deal with nonlinearities in a process, even with linear models of the process. The presented open-loop AMPC method relies on interpolation between several linear finite impulse response (FIR) models, where different methods for the interpolation are investigated. The second method relies on online parameter estimation of a single FIR model via recursive least squares. The presented methods are tested and compared by controlling a single tank system in simulation.

1 Introduction

Since the early successful implementations of model predictive control (MPC) in the 1970s, MPC has risen to be state of the art for many industrial control applications [5, 19, 17]. In MPC, a model of the process is used to calculate an

optimal sequence of future manipulated variables [12]. Early algorithms such as dynamic matrix control (DMC) [15], model algorithmic control (MAC) and internal model control (IMC) are successfully implemented [17] and still have relevance today [4]. These algorithms rely on offline identified time-invariant models of the plant.
For systems with changing process dynamics, these algorithms can fail if the controller does not adapt to the new dynamics. For example: A rocket decreases its total mass by burning fuel. This effects its inertia, propulsion etc. and therefore changes its dynamics. Adaptive model predictive control (AMPC) aims to solve this issue and can be categorized into deterministic and stochastic adaptive control [16], where the difference of stochastic adaptive control takes the uncertainty of parameter estimation into account (as in [8, 11]). In this work, the investigated open-loop and closed-loop adaptive control can be understood as offline and online identification of the model [1].

Both approaches in this work are deterministic and use affine models in AMPC. Adaptive variants of DMC are presented in [6, 21]. These approaches are also referred to as gain scheduling [7]. Here, finite step response (FSR) models are identified for several operating points (OPs). Depending on the output of a process (as scheduling variable), the controller interpolates between these models via triangular membership functions, which can be interpreted as fuzzy logic. The presented open-loop AMPC works similar, but finite impulse response (FIR) models are employed and different membership functions are investigated and compared. FIR models were already successfully used in MPC in the late 1970s [14]. For unconstrained MPC, the arising optimization problem has a closed form solution, which can be solved via least squares (LS). FIR models are inherently stable, their model parameters allow physical interpretability regarding characteristics, such as dead times of the process and they can easily be included [20]. In [10], affine models have been applied in a gain scheduling state controller together with Gaussian radial basis functions. As can be seen in [18], radial basis functions can work well to achieve locally bound validity for affine models. This property is important for these models, since their accuracy is bounded to the proximity to the OP for their estimation. The second investigated approach is a closed-loop AMPC. This approach employs a single FIR model, which is continuously updated via recursive least

squares (RLS). In contrast to the first approach, no OPs and membership functions need to be defined a priori. However, this approach faces the duality of control [2, 3, 13]. The controller drives the process to steady state, but in order to identify the dynamics of a process, it needs to be sufficiently excited by the input signal [18]. Not taking care of this problem can lead to bursting [9]. Therefore, it is worth investigating and comparing both approaches for a nonlinear control problem. As in [21], the control of the fill level in a tank is simulated to investigate both methods.

2 MAC Algorithm

The MAC algorithm belongs to the class of MPC and is based on FIR models. This contrasts the DMC algorithm, which works with FSR models [19, 21]. To model nonlinear processes with a linear model at a given OP in the FIR model, an offset parameter is additionally required. The output $\hat{y}(k)$ at the discrete time step k of the FIR model can be calculated by a linear combination of the given inputs u

$$\hat{y}(k) = g_{\text{off}} + \sum_{j=1}^{n_{\text{FIR}}} g_j u(k-j) + \hat{n}(k). \tag{1}$$

Thereby, the FIR model order n_{FIR}, the predicted disturbance $\hat{n}$, the offset coefficient g_{off} and the impulse response coefficients g_j are used. The main idea of an MPC is to predict the future behavior of a process for different sequences of the manipulated variable. Afterwards, the best sequence according to a predefined objective function is chosen. The manipulated variable is the input of the model u. To predict the future FIR model output $\hat{y}(k+i)$ during time step k, the convolution sum in (1) can be divided into two parts. The first one includes the past sequence of the manipulated variable

$$\hat{y}^{-}(k+i) = g_{\text{off}} + \sum_{j=i+1}^{n_{\text{FIR}}} g_j u(k+i-j), \tag{2}$$

which already influenced the process. The predicted output $\hat{y}^-$ is also called free response of the process. In the second part, for the forced response

$$\hat{y}^+(k+i) = \sum_{j=1}^{i} g_j u(k+i-j), \tag{3}$$

the future sequence of the manipulated variable is considered. It is the task of MAC to optimize this future sequence of the manipulated variable. The whole prediction for time step $k+i$ at the time step k is given by

$$\hat{y}(k+i) = \hat{y}^-(k+i) + \hat{y}^+(k+i) + \hat{n}(k+i). \tag{4}$$

It is assumed, that the disturbance $\hat{n}$ at time step k remains the same over all prediction steps i and is calculated by

$$\hat{n}(k) = y(k) - \hat{y}(k) \quad \text{for } i = 1, \ldots, n_\mathrm{p}\,, \tag{5}$$

with $y(k)$ as the measured process output at the time step k to correct the bias error of the prediction. The output sequence is calculated by

$$\underline{\hat{y}} = \underline{1} g_\mathrm{off} + \underline{H}^- \underline{u}^- + \underline{H}^+ \underline{u}^+ + \underline{\hat{n}}\,, \tag{6}$$

with n_p as prediction horizon, the vector of the past sequence of the manipulated variable $\underline{u}^-$, the proposed sequence of the manipulated variable $\underline{u}^+$, the predicted outputs $\underline{\hat{y}}$ and the disturbance at the predicted time steps $\underline{\hat{n}}$ as

$$\begin{aligned}
\underline{u}^- &= \begin{bmatrix} u(k-n_\mathrm{FIR}+1) \\ u(k-n_\mathrm{FIR}+2) \\ \vdots \\ u(k-1) \end{bmatrix}, \qquad \underline{u}^+ = \begin{bmatrix} u(k) \\ u(k+1) \\ \vdots \\ u(k+n_\mathrm{p}-1) \end{bmatrix}, \\
\underline{\hat{y}} &= \begin{bmatrix} \hat{y}(k+1) \\ \hat{y}(k+2) \\ \vdots \\ \hat{y}(k+n_\mathrm{p}) \end{bmatrix}, \qquad \underline{\hat{n}} = \begin{bmatrix} \hat{n}(k) \\ \hat{n}(k) \\ \vdots \\ \hat{n}(k) \end{bmatrix}
\end{aligned} \tag{7}$$

and the dynamic matrices

$$\underline{H}^{-} = \begin{bmatrix} g_{n_{\mathrm{FIR}}} & g_{n_{\mathrm{FIR}}-1} & \cdots & \cdots & \cdots & g_2 \\ 0 & g_{n_{\mathrm{FIR}}} & \cdots & \cdots & \cdots & g_3 \\ \vdots & \ddots & \ddots & & & \vdots \\ 0 & \cdots & 0 & g_{n_{\mathrm{FIR}}} & \cdots & g_{n_{\mathrm{p}}+1} \end{bmatrix},$$

$$\underline{H}^{+} = \begin{bmatrix} g_1 & 0 & \cdots & \cdots & 0 \\ g_2 & g_1 & \ddots & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_{n_{\mathrm{p}}-1} & g_{n_{\mathrm{p}}-2} & \cdots & g_1 & 0 \\ g_{n_{\mathrm{p}}} & g_{n_{\mathrm{p}}-1} & \cdots & g_2 & g_1 \end{bmatrix}. \tag{8}$$

To optimize the sequence of the future manipulated variable $\underline{u}^{+}$, the difference of the reference values and the prediction

$$\underline{e} = \underline{w} - \underline{\hat{y}} = \underline{w} - \left(\underline{1} g_{\mathrm{off}} + \underline{H}^{-}\underline{u}^{-} + \underline{H}^{+}\underline{u}^{+} + \underline{\hat{n}}\right) \tag{9}$$

has to be calculated, where the vector of the future reference values $\underline{w}$ is defined as

$$\underline{w} = \begin{bmatrix} w(k+1) & w(k+2) & \cdots & w(k+n_{\mathrm{p}}) \end{bmatrix}^{T}. \tag{10}$$

To solve the MPC optimization problem, a cost function

$$J = \underline{e}^{T}\underline{e} + \lambda_{\mathrm{u}}\,\Delta\underline{u}^{+T}\Delta\underline{u}^{+} \tag{11}$$

is introduced. A larger λ_{u} penalizes $\Delta\underline{u}^{+}$ more, which leads to a less aggressive controller. By adjusting λ_{u} (in this work $\lambda_{\mathrm{u}} = 0.2$) this trade-off can be controlled. The change of the manipulated variable $\Delta\underline{u}$ is calculated by

$$\Delta\underline{u} = \begin{bmatrix} u(k) & - & u(k-1) \\ u(k+1) & - & u(k) \\ & \vdots & \\ u(k+n_{\mathrm{p}}-1) & - & u(k+n_{\mathrm{p}}-2) \end{bmatrix} = \underline{T}\left(\underline{u}^{+} - \underline{1}u(k-1)\right) \tag{12}$$

with the transformation matrix

$$\underline{T} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & \ddots & \ddots & \vdots \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}. \tag{13}$$

By minimizing the cost function J with respect to the future sequence of the manipulated variable $\underline{u}^+$, the optimal solution for the MPC optimization problem can be found by solving

$$\begin{aligned} &\min_{\underline{u}^+} && J \\ &\text{subject to} && |\Delta \underline{u}| \leq \underline{1} \cdot \Delta u_{\max} \quad \text{and} \quad \underline{1} \cdot u_{\min} \leq \underline{u} \leq \underline{1} \cdot u_{\max}. \end{aligned} \tag{14}$$

The solution of the unconstrained optimization problem in (14) can be found analytically. By neglecting constraints, the optimum of the cost function J is found by

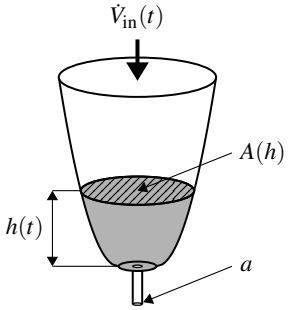
$$\begin{aligned} \underline{u}^+ = &\left(\underline{H}^{+T} \underline{H}^+ + \lambda_{\mathrm{u}} \underline{T}^T \underline{T} \right)^{-1} \\ &\cdot \left(\underline{H}^{+T} \left(\underline{w} - \left(\underline{H}^- \underline{u}^- + \underline{1} g_{\mathrm{off}} + \underline{\hat{n}} \right) \right) + \lambda_{\mathrm{u}} \underline{T}^T \underline{1} u(k-1) \right). \end{aligned} \tag{15}$$

3 Process model

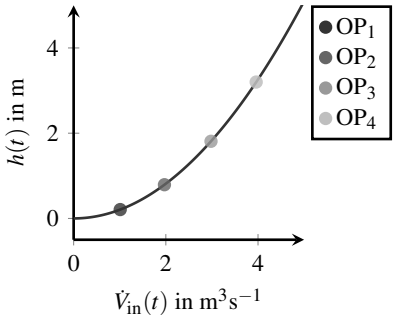
A single tank system, similar to [21], is considered. The task is to control the fill level $y(t) = h(t)$ between 0 and 5 m in a tank with a small hole as an outlet by adjusting the inflow $u(t) = \dot{V}_{\mathrm{in}}(t)$ of the fluid. This setup can be seen in Fig. 1a, where the area of the fluids surface is calculated by

$$A(h) = \left(\sqrt{h} + 1 \right)^2 \pi \, \mathrm{m}^2. \tag{16}$$

The cross-section of the outlet is defined by $a = 0.36\pi\,\mathrm{m}^2$. The fluid is considered to be incompressible. By using this geometry instead of a cylindrical one, the dynamical behavior changes more with different heights. A model of the process can be derived via the law of mass conservation.



(a) single tank



(b) steady state behavior and operating points (OPs) of the single tank system for inflow $\dot{V}_{\mathrm{in}}(t)$

Figure 1: process model with volumetric flow rate as input $u(t) = \dot{V}_{\mathrm{in}}(t)$ and the fill level as output $y(t) = h(t)$

The simplified dynamics for the simulation are described by

$$\dot{h}(t) = \frac{1}{A(h)}\dot{V}_{\mathrm{in}} - \frac{a}{A(h)}\sqrt{2g_{\mathrm{earth}}h(t)} \tag{17}$$

with the discrete equivalent being

$$\begin{aligned} &y(k) = y(k-1) + \\ &T_0\left(-\frac{a}{A\left(y(k-1)\right)}\sqrt{2g_{\mathrm{earth}}y(k-1)} + \frac{1}{A\left(y(k-1)\right)}u(k-1)\right), \end{aligned} \tag{18}$$

where g_{earth} denotes the gravity and T_0 is the sampling time (chosen as 1 s). The nonlinearity of this process can be seen by looking at the steady state behavior of the process in Fig. 1b. This figure also shows the OPs, where the FIR models are identified.

4 AMPC Algorithms

Usually, a linear time-invariant model is used in an MPC to predict the process behavior [19]. This might fail for nonlinear processes, hence an AMPC may be useful. The basic idea of an AMPC is to change the internal model or controller parameters when the OP changes. In this work, an open-loop and a closed-loop adaptive control approach are considered. For both approaches, only the internal model is adapted. This enables better control, not only for nonlinear, but also for time-variant processes.

4.1 Open-loop Adaptive Control

The open-loop AMPC approach requires prior knowledge of the process. There has to be a measurable state variable $\psi(k)$, which correlates with the change of the process. Initially, models have to be estimated at the OPs. Subsequently, the AMPC aims to switch or interpolate between the different models depending on the state variable $\psi(k)$, which is in our case the fill level of the tank $y(k)$. In Fig. 2, the block diagram of an open-loop AMPC is shown. In the past, it

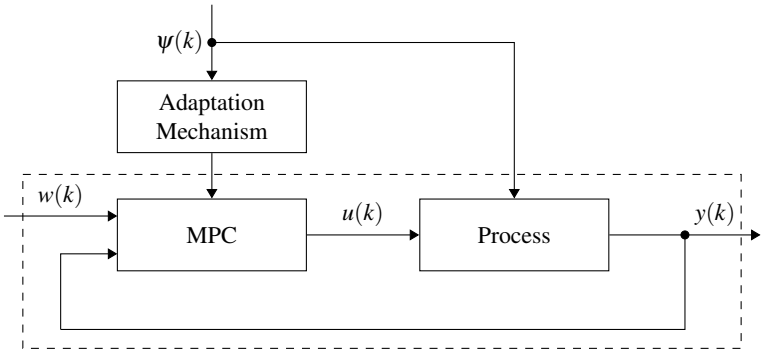


Figure 2: block diagram of a open-loop AMPC approach (adapted from [22])

has been common to adapt only the gain of a simple controller for changing process dynamics. Nowadays, AMPC also encompasses more sophisticated algorithms, which are not restricted to the adaptation of a single parameter. In this work, the FIR coefficients $\underline{g}$ of the internal model are a function of the state

$\psi(k)$. Therefore, the controller behaves more accurately at the OPs, for which a previously identified model is employed. In order to accurately estimate the

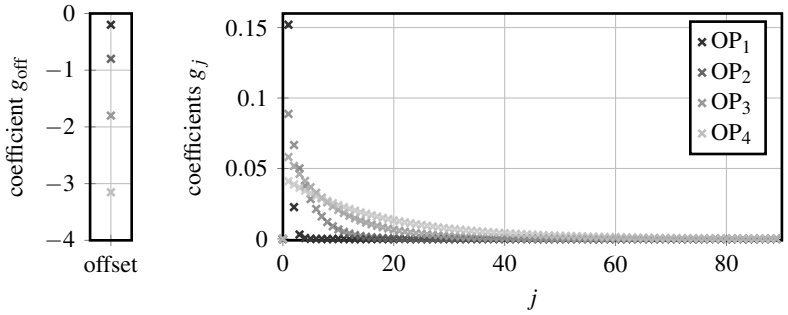


Figure 3: identified FIR coefficients at different operating points (OPs) of the single tank system

FIR coefficients, the system has to be excited at the different OPs. Afterwards, the FIR coefficients $\underline{g}_{\text{OP}_i}$ can be computed from the gathered data with the LS method. They can be seen in Fig. 3 and are defined as

$$\underline{g}_{\text{OP}_i} = [g_{1,\text{OP}_i},\, g_{2,\text{OP}_i},\, \ldots,\, g_{n_{\text{FIR}},\text{OP}_i}]^T \quad \text{for } i = 1,\ldots,n_{\text{OP}}\,, \tag{19}$$

where n_{OP} is the number of OPs. Because such a model only describes the process behavior around its OP, a validity function is needed. The validity function describes how active a model shall be for the given state $\psi(k)$ [18]. Through normalization of membership functions $\mu(\psi)$ (e.g. Gaussians, rectangular, triangular or trapezoid functions), activation functions $\Phi_{\text{OP}_i}(\psi)$ of the different models are calculated according to

$$\Phi_{\text{OP}_i}(\psi) = \frac{\mu_i(\psi)}{\sum\limits_{j=1}^{n_{\text{OP}}} \mu_j(\psi)}. \tag{20}$$

This achieves the partition of unity

$$\sum_{i=1}^{n_{\text{OP}}} \Phi_{\text{OP}_i}(\psi) = 1. \tag{21}$$

The resulting validity functions $\Phi_{\text{Method,OP}_i}$ are shown in Fig. 4. The activation functions for the hard switching case are denoted by $\Phi_{\text{Switch,OP}_i}$. In this method, only a single model is active. The borders for switching are set in the middle between two OPs in the output space.
The linear interpolation with triangular functions has at maximum two local models being active. The validities $\Phi_{\text{Lin,OP}_i}$ are set to be 0.5 for both active models at the previously defined switching borders and reaches 1 at the OP of estimation.
The Gaussian validity functions $\Phi_{\text{Gauss,OP}_i}$ are achieved by placing Gaussian membership functions on the OPs and normalizing them [18]. It is noticeable in Fig. 4 that the Gaussian validity functions do not share the same switching borders with the other validity functions. This is due to tuning of the Gaussians' variances. They are proportional to the span they are supposed to cover, otherwise validity functions in small areas would vanish.

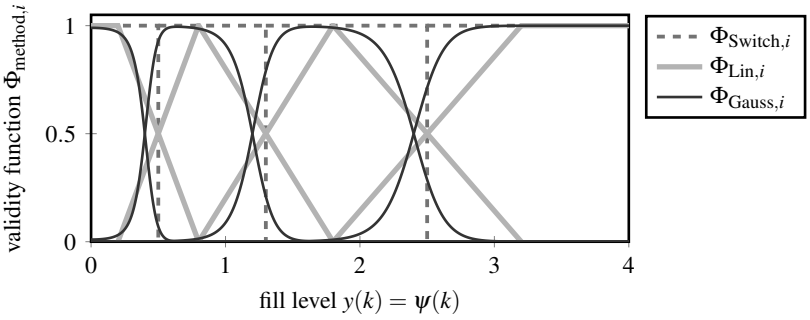


Figure 4: Validity functions $\Phi_{\text{method},i}$ of the open-loop AMPC approaches

The adapted FIR coefficients for the internal model $\underline{g}_{\text{model}}$ are calculated by multiplying the FIR coefficients of the OPs with their validity function and summing them up

$$\underline{g}_{\text{model}}(\psi) = \sum_{i=1}^{n_{\text{OP}}} \underline{g}_{\text{OP}_i} \Phi_{\text{OP}_i}(\psi). \tag{22}$$

4.2 Closed-loop Adaptive Control

A schematic block diagram of the indirect closed-loop AMPC is shown in Fig. 5. The key idea is the online identification of the process. This results in a single FIR model as the internal model for prediction in the AMPC algorithm. To identify the FIR coefficients $\underline{g}(k)$, the RLS method with a forgetting factor

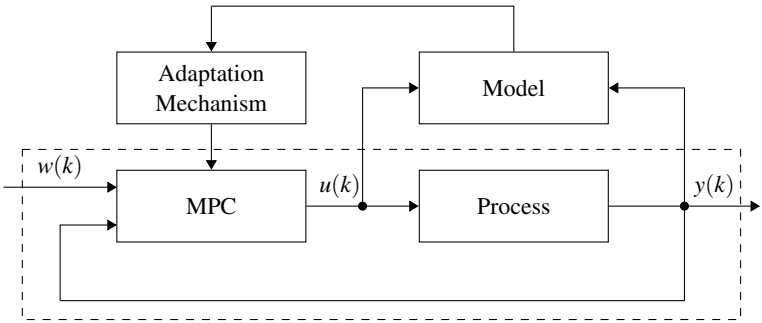


Figure 5: block diagram of an indirect closed-loop AMPC approach (adapted from [22])

λ_{forget}

$$\underline{g}(k) = \underline{g}(k-1) + \underline{\gamma}(k)\left(y(k) - \underline{x}^T(k)\underline{g}(k-1)\right) \tag{23}$$

is used with the adaptation vector

$$\underline{\gamma}(k) = \frac{1}{\underline{x}^T(k)\underline{P}(k-1)\underline{x}(k) + \lambda_{\text{forget}}}\underline{P}(k-1)\underline{x}(k) \tag{24}$$

and the update equation of the covariance matrix

$$\underline{P}(k) = \frac{1}{\lambda_{\text{forget}}}\left(\underline{I} - \underline{\gamma}(k)\underline{x}^T(k)\right)\underline{P}(k-1). \tag{25}$$

When using the RLS method for closed-loop adaptation, new data is collected during closed-loop control of the process. One major issue is that there is no excitation of the system, if the manipulated variable is constant in time. In this case, it is not possible for the RLS method to get new information for the estimation of the FIR coefficients. Hence, persistent excitation of the system is needed. To achieve persistent excitation, white noise v of variance σ^2_{excite} is

added to the optimal manipulated variable u_{opt}

$$u_{\text{excite}} = u_{\text{opt}} + v; \qquad v \sim \mathcal{N}\left(0, \sigma^2_{\text{excite}}\right) \tag{26}$$

as some kind of probing measure [3]. This ensures excitation of the system even at steady-state. The parametrization of the algorithm is executed according to Tab. 1.

Table 1: parametrization of the closed-loop AMPC method

parameter	λ	$\underline{\text{P}}(0)$	σ^2_{excite}	$\text{g}(0)$
value	0.93	$0.1 \cdot \underline{I}$	0.04	$0.01 \cdot \underline{1}$

5 Results

In order to test the presented methods, a single tank system is simulated. Measurement noise is neglected. As reference trajectories, a sequence of steps (see Fig. 6) and a sine wave (see Fig. 7) have been chosen. To measure the quality of the controller performance, the normalized root mean squared error J_{e} and the mean squared input J_{u} are considered. As can be seen from both figures and Tab. 2, all three open-loop adaptive control methods produce results of similar quality. This can be accounted to the fact that the model error is fed back to the controller and compensates errors in the model to some degree. Still, the presented methods outperform the algorithm using a single averaged model, denoted as 'Mean'. When the active model changes in the switching method, a sudden change in the input can be noticed in Fig. 7. This is no desirable property as it increases wear of the machines. The linear and Gaussian method avoid this with their smooth transitions. One advantage of the Gaussian method is that it is more versatile in the design of its validity functions. By adjusting the Gaussians' variances, smoother or sharper transitions can be achieved. Also, multiple active models are possible, whereas with the linear transition only a maximum of two models are active. Due to this versatility, the Gaussian method can be well adapted to any nonlinear process. However, this requires

more effort to work well. Increasing the number of local models improves the overall performance of the algorithm for every method.

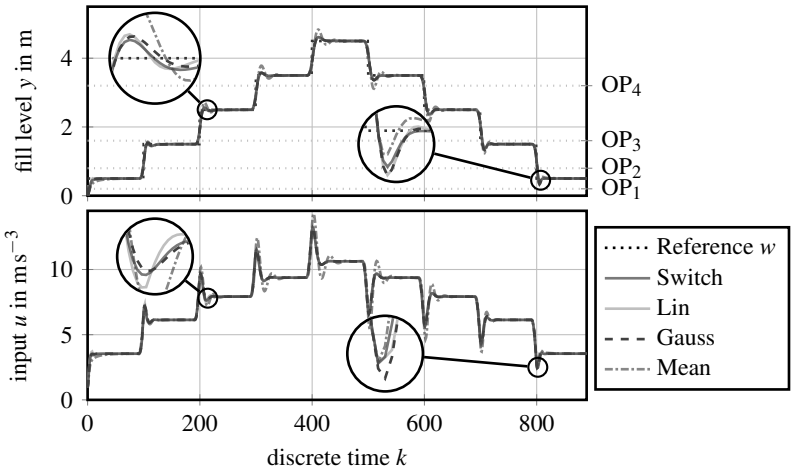


Figure 6: sequence of steps for the open-loop AMPC methods

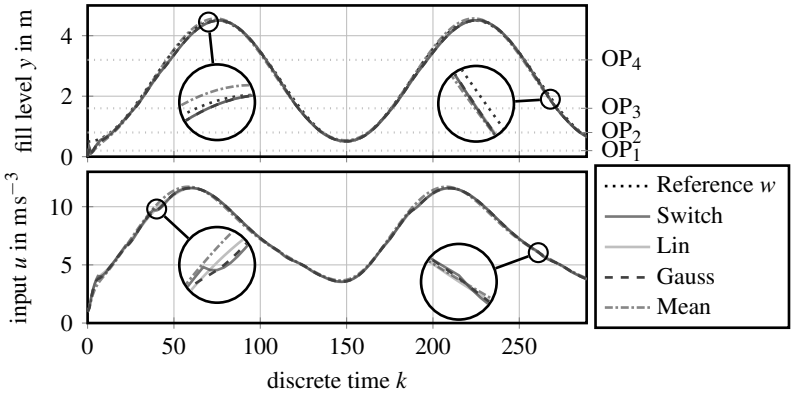


Figure 7: sinusoidal sequence for the open-loop AMPC methods

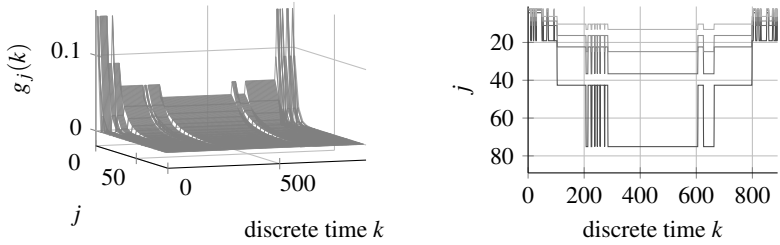
By taking a look at the FIR coefficients in Fig. 8 and Fig. 9, it can be seen how the adapted coefficients of the internal model change over time during simulation. Note that all following plots do not include the offset coefficient g_{off} to

Table 2: measures of performance, J_e and J_u for presented scenarios with open-loop AMPC

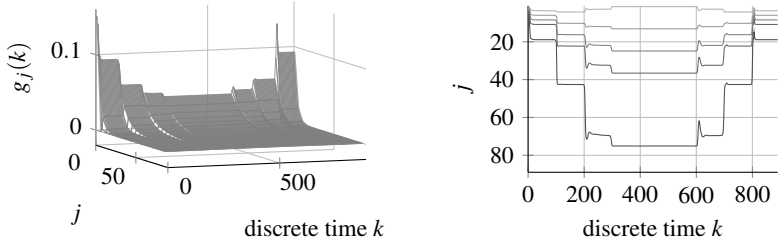
scenario	metric	Switch	Lin	Gauss	Mean
Steps	J_e	0.0699	0.0701	0.0694	0.0852
Steps	J_u	57.88	57.87	57.84	58.12
Sinus	J_e	0.0540	0.0526	0.0549	0.0575
Sinus	J_u	64.58	64.61	64.55	65.35

achieve a better representation. At the start and the end of the simulation of the step sequence, the switching method (Fig. 8a) changes its active model frequently, because the fill level y is near to a switching border. Similar behavior can be noticed in the interval $200 < k < 300$. At the beginning of the simulation, the fill level y starts at 0 m. Since the first step in the reference signal is at 0.5 m, the process is only in a short period of time between fill levels of 0 m and 0.5 m. This causes the FIR coefficients to correspond only at the beginning and during the overshoot at $k = 800$ to the FIR coefficients of OP_1. The process model switches continuously while following the sinusoidal reference. For this, the FIR coefficients in Fig. 9a are only changing by crossing a switching border, while for the Gaussian and linear method the FIR coefficients change is smooth (see Fig. 9b and 9c). This behavior is also presented in the contour plots of the FIR coefficients in Fig. 9 on the right hand side.

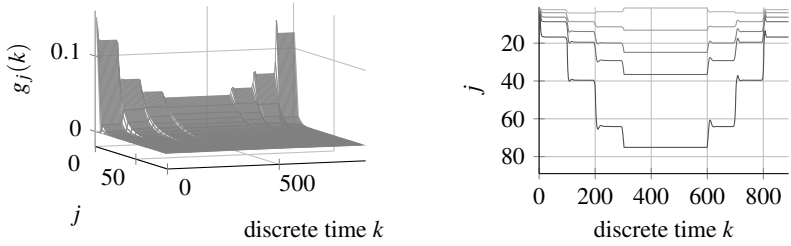
The results of closed-loop AMPC method can be seen in Fig. 10. For evaluating the closed-loop method, only the step sequence is used. With the sinusoidal reference signal, the FIR coefficients have to change too fast and the algorithm can not adapt the model. Due to this issue, the controller gets unstable. Because of the higher adaptation time of the model in Fig. 11, the same step sequence with a five times longer hold time is also considered. In Tab. 3 the measures of performance are given. For the five times longer sequence J_e is mainly lower because of the comparatively few steps in relation to the holding time, but J_u is barely different. The less aggressive control performance in the scenario with the five times longer hold time can be justified by a better model quality. By comparison of Fig. 10 and 11, a reduction of the overshoot is clearly visible. Through persistent excitation, the input u in the closed-loop method is noisier



(a) surface and contour plot for switching method



(b) surface and contour plot for Gaussian method

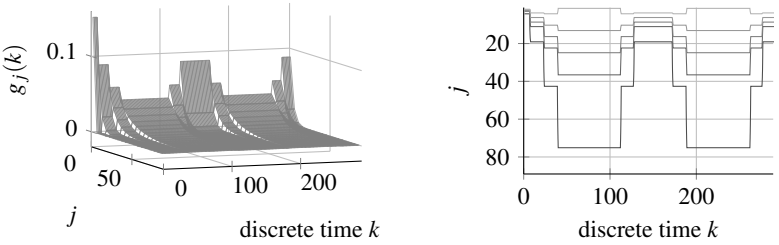


(c) surface and contour plot for linear method

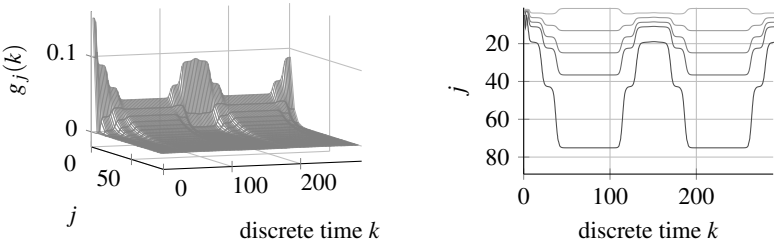
Figure 8: adapted FIR coefficients during simulation of the sequence of steps with the open-loop AMPC

than in the open-loop one. This affects also the fill level y, which also appears noisy, but is needed for a good estimation of the FIR coefficients.

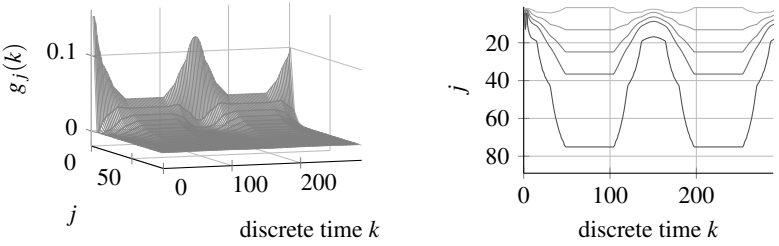
From Fig. 12, it becomes clear that the closed-loop method needs a certain number of time steps to sufficiently learn the FIR coefficients. Whenever the references is changing, the coefficients need to adapt and in the first phase of estimation they become very noisy for a short time until they converge again



(a) surface and contour plot for switching method



(b) surface and contour plot for Gaussian method



(c) surface and contour plot for linear method

Figure 9: adapted FIR coefficients during simulation of the sine wave with the open-loop AMPC

Table 3: measures of performance, J_e and J_u for presented scenarios with closed-loop AMPC

scenario	**metric**	**1x seq. length**	**5x seq. length**
Steps	J_e	0.0843	0.0355
Steps	J_u	57.97	57.25

to the new optimal coefficients. The correction of the bias error fails for highly wrong estimated coefficients and the system excites itself.

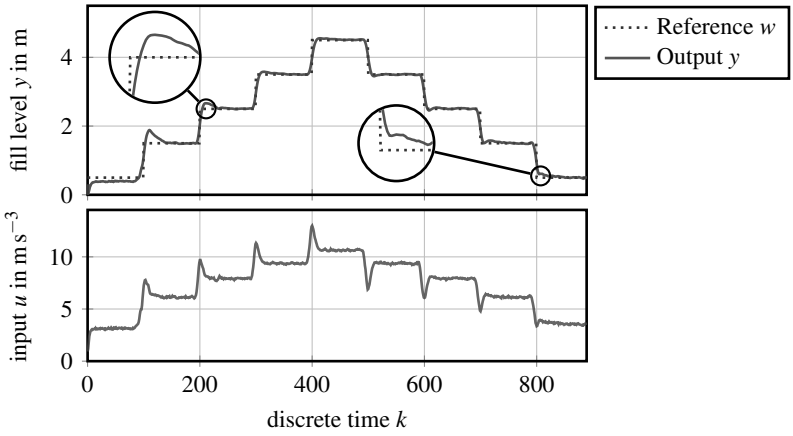


Figure 10: sequence of steps for the closed-loop AMPC methods

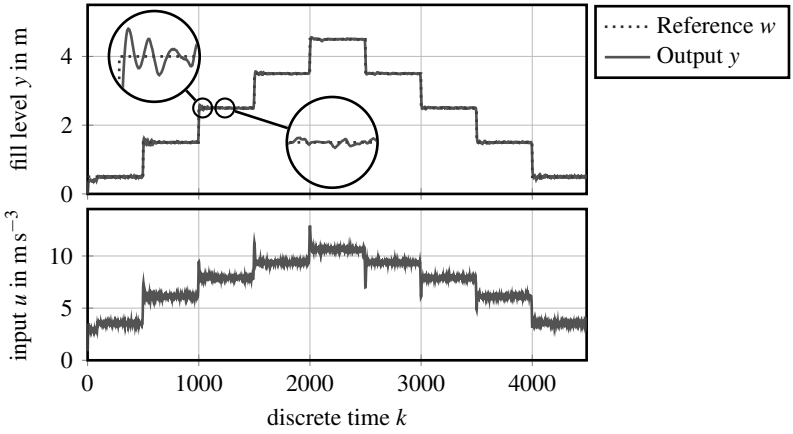
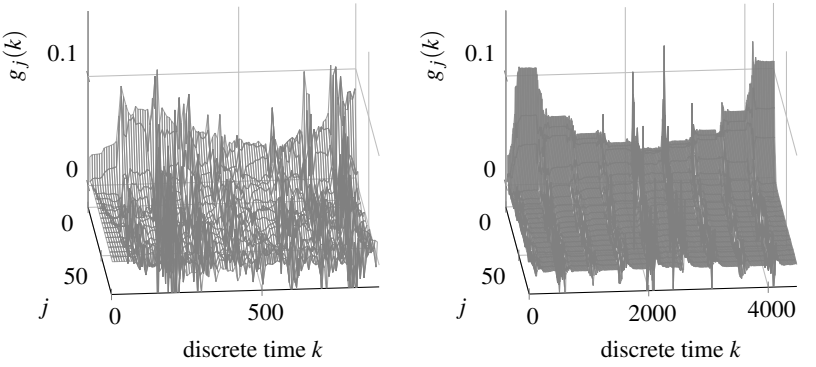


Figure 11: sequence of steps for the closed-loop AMPC methods with the five times longer holding time

6 Conclusion

Two AMPC approaches – an open-loop and a closed-loop – with linear FIR models are investigated. For the open-loop approach, three methods for blending, using the previously estimated models in different OPs, are explained.



(a) surface plot for the normal step sequence

(b) surface plot for the five times longer holding time during the step sequence

Figure 12: adapted FIR coefficients during simulation of the sequence of steps with the closed-loop AMPC

What makes the Gaussian method difficult to compare to the linear and switching methods is the fact that depending on the standard deviations of the Gaussian membership functions, the validity functions do not share the same switching borders. With the possibility of tuning the parameter of each Gaussian separately, this method becomes more suitable for different nonlinear processes, but requires more effort to parameterize. Hard switching can lead to sudden changes in the input, which wears the machines more, otherwise the three open-loop adaptive methods perform similarly well. One major drawback of the closed-loop adaptation is the fact that some time is required to learn suitable FIR coefficients. In contrast to this, using previously learned parameters gives a sufficient accuracy right away and therefore works faster during operation. The open-loop adaptation is more reliable but restricted to operate near to the initial OPs. In addition, the FIR models have to be estimated offline before the control task is started. The closed-loop adaptation requires less prior knowledge and tedious tuning but is restricted to processes and trajectories that give the method sufficient time to estimate an appropriate model. Therefore, if no offline models can be determined, the closed-loop method has to be chosen. A comparison of the closed-loop and open-loop adaptive approach in areas

where the open-loop strategy needs to rely on extrapolation of its models could expose further advantages at the closed-loop scheme.

References

[1] I. D. Landau et al. “Adaptive Control” Springer. 2011.

[2] B. Kouvaritakis and M. Cannon. “Model Predictive Control” Springer. 2017.

[3] S. V. Racović and W. S. Levine “Handbook of Model Predictive Control” Birkhäuser. 2019.

[4] L. Grüne and J. Pannek. “Nonlinear Model Predictive Control” Springer. 2017.

[5] R. Dittmar. “Advanced Process Control” De Gruyter. 2017.

[6] U. Moon and K. Y. Lee. “An Adaptive Dynamic Matrix Control With Fuzzy-Interpolated Step-Response Model for a Drum-Type Boiler-Turbine System” IEEE Transactions of Energy Conversion. 2011.

[7] D. J. Leith and W. E. Leithead. “Survey of gain-scheduling analysis and design” International Journal of Control. 2010.

[8] M. A. P. Ramos et al. “Generalized Minimum Variance Controller with Dynamic Pole Assignment to Improve Performance in Industrial Applications” Technological Developments in Networking, Education and Automation. Springer. 2009.

[9] B. D. O. Anderson and A. Dehghani. “Challenges of adaptive control–past, permanent and future” Elsevier. 2008.

[10] D. Döring. “A Design of Gain-scheduled Control via LMIs” at-Automatisierungstechnik. 2002.

[11] H. Unbehauen. “Adaptive Dual Control Systems: A Survey” Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373). 2000.

[12] J. B. Rawlings. “Tutorial Overview of Model Predictive Control” IEEE Control Systems Magazine. 1999.

[13] B. Wittenmark. “Adaptive dual control methods: An overview” IFAC Adaptive Systems in Control and Signal Processing. 1995.

[14] J. Richalet, A. Rault, J. L. Testud and J. Papon. “Model Predictive Heuristic Control: Applications to Industrial Processes” Automatica. 1978.

[15] C. R. Cutler. “Dynamic Matrix Control an Optimal Multivariable Algorithm with Constraints” Dissertation. 1983.

[16] S. G. Anavatti, F. Santoso and M. A. Garratt. “Progress in Adaptive Control Systems: Past, Present, and Future” International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA). 2015.

[17] C. E. García, D. M. Prett and M. Morari. “Model Predictive Contol: Theory and Practice - a Survey” Automatica. 1989.

[18] O. Nelles. “Nonlinear System Identification”. Springer. 2020.

[19] E. F. Camacho and C. Bordons. “Model Predictive Control”. Springer. 2004.

[20] T. Münker. “Machine Learning with Finite Impulse Response Models”. Dissertation. 2020.

[21] T. Kłopot and P. Skupin. “Adaptive Dynamic Matrix Control with Interpolated Parameters”. 20th International Conference on Methods and Models in Automation and Robotics (MMAR). 2015.

[22] A. Braun. “Optimale und adaptive Regelung technischer Systeme”. Springer. 2020.

Demanded Power Point Tracking of PV Power Plants without Battery Energy Storage

Stephan Kusche, Horst Schulte[1]

[1]Control Engineering Group, Department of Engineering I
University of Applied Sciences Berlin (HTW), Germany
E-Mail: kusche,schulte@htw-berlin.de

Abstract

Future electrical power systems operating up to 100-percent with regenerative energy sources (RES) will need dynamically controllable power plants [1]. The variables to be controlled in power systems are voltage and frequency of the grid. Thereby, the frequency in the grid is changed via the supplied power of the power plants. Until recently, regenerative energy systems have fed as much power as possible into the grid with the objective of optimizing the power by means of maximum power point (MPP) tracking [2]. Therefore, they contribute to grid stability only to a very limited amount. To change the paradigm, control methods like active power tracking control of wind turbines (WT) [3] and photovoltaic (PV) power plants with battery storage have been developed in recent years. However, it is also necessary that PV systems without storage can quickly reduce the power to be supplied. In this paper, a model-based demanded power point (DPP) tracking controller based on Takagi-Sugeno modeling and LMI synthesis is presented. It has the advantage to perform in a wide nonlinear operating range with guaranteed performance, independent of external disturbances such as changes in the irradiation and PV cell temperature.

1 Introduction

Currently the major part of the electrical energy is produced by classical synchronous machines. They have the inherent ability to stabilize the electrical grid due to their inertia. Droops in voltage and grid frequency have to overcome this inertia first in order to make any changes to these quantities.

Until 2020, the photovoltaic electricity share was about 8.7% and 3.9% in Germany and EU28, respectively [4]. In the past, this small share allows renewable energy sources (RES) to simply fed their complete available power into the grid without paying attention to the grid stability. Nevertheless the share of renewable electrical energy is deliberately rising. With higher share, the grid stability becomes a serious issue that has to be addressed by new control concepts. One possible approach is to engineer renewable energy sources in a way that they act like a conventional power plant with an inherent inertia, see e.g. [5, 6]. Optimization can be archived by forming an intelligent compound of these power plants, called dynamic virtual power plant (DVPP) [7]. Participants to the digital virtual power plant need to be able to meet minimum requirements like controllable power generation under varying conditions and they need to supply a proper communication interface.

A usual model scheme for a PV power plant is shown in Figure 1. The primary converter can be identified at the DCDC-converter by the converter model and the converter controller. Its task is to regulate the voltage of the PV cell v_{pv} and to feed the electrical power into the DC link. The MPP respectively DPP controller can be interpreted as part of the power plant controller. The secondary converter can be located at the grid connection. Its task is the power conversion at the grid level and it connects to the PCC. In the following, we will discuss the electrical model of the PV cell and the complete DCDC-converter.

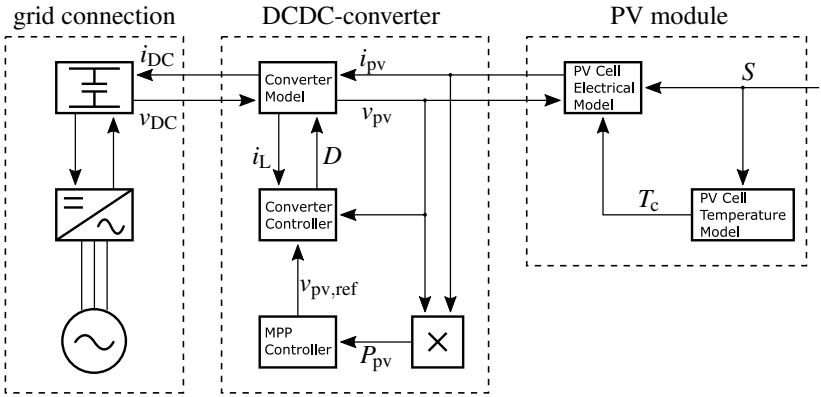


Figure 1: Signal paths for the PV power generator from the physical power source (irradiation S on the right hand side) to the electrical grid (in the left hand side).

2 Model of PV Module

The well known complete single diode model (cSDM) is shown in Figure 2 [8]. It consists of the ideal model of a photovoltaic cell (current source in parallel with a diode) completed by resistors in serial and parallel to accommodate losses. The diode I-V characteristic is described by the theory of Shockley [9]:

$$i_d = i_s \left[\exp\left(\frac{1}{A_n} \frac{v_d}{v_T} \right) - 1 \right], \quad v_T = \frac{k_B T_c}{q_e}. \tag{1}$$

where

$$\begin{cases} A_n \in [1,2], & \text{ideality factor of the diode } [1] \\ i_d, & \text{diode current } [\mathrm{A}] \\ i_s, & \text{diode reverse-bias saturation current } [\mathrm{A}] \\ k_B, & \text{Boltzmann constant } \left[1.38 \cdot 10^{-23}\,\mathrm{J/K}\right] \\ q_e, & \text{elementary charge } \left[1.602 \cdot 10^{-19}\,\mathrm{C}\right] \\ T_c, & \text{absolute temperature of the p-n junction, cell temperature } [\mathrm{K}] \\ v_d, & \text{voltage across the diode } [\mathrm{V}] \\ v_T, & \text{thermal voltage } [\mathrm{V}] \end{cases}$$

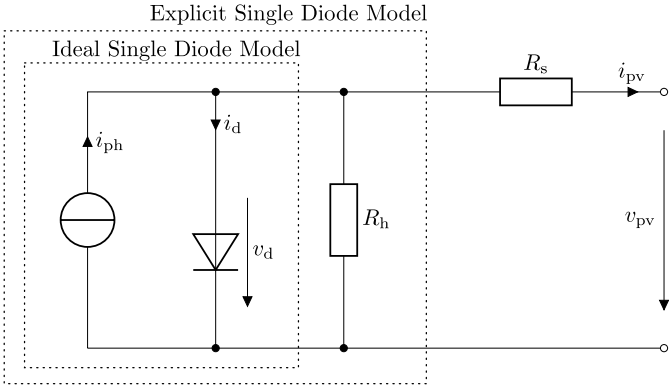


Figure 2: Complete Single Diode Model and simplifications thereupon.

Together with Kirchhoff's first circuit law $i_{pv} = i_{ph} - i_d - v_d/R_h$ and Kirchhoff's second circuit law $v_d = v_{pv} + R_s i_{pv}$ this gives the following relation between i_{pv} and v_{pv}:

$$i_{pv} = i_{ph} - i_s \left[\exp\left(\frac{v_{pv} + R_s i_{pv}}{v_T A_n} \right) - 1 \right] - \frac{v_{pv}}{R_h} - \frac{R_s}{R_h} i_{pv}. \tag{2}$$

where

$$\begin{cases} i_{ph}, & \text{photon current [A]} \\ i_{pv}, & \text{PV cell current [A]} \\ R_h, & \text{shunt resistance [}\Omega\text{]} \\ R_s, & \text{series resistance [}\Omega\text{]} \\ v_{pv}, & \text{PV cell voltage [V]} \end{cases}$$

2.1 Explicit Single Diode Model (eSDM)

The cSDM includes 5 fitting parameters: $(A_n, i_s, i_{ph}, R_h, R_s)$. In dependence on the amount and accuracy of the data provided by the manufacturer about the photo cells I-V-characteristic, it can be hard to identify all 5 parameters. Furthermore, (2) gives only an implicit function for the photon current. To

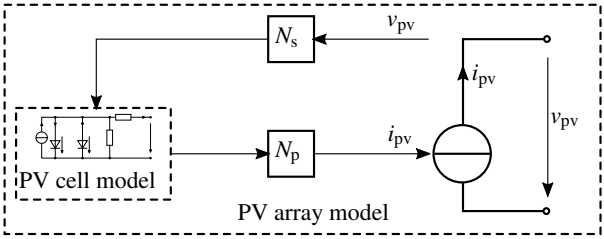


Figure 3: Aggregation of the PV array model from PV cell model.

obtain an explicit form, the cSDM is simplified by setting R_s to zero leading to the eSDM. The governing equation (2) becomes:

$$i_{pv} = i_{ph} - i_s \left[\exp\left(\frac{1}{A_n} \frac{v_{pv}}{v_T} \right) - 1 \right] - \frac{v_{pv}}{R_h} = f(v_{pv}). \tag{3}$$

For the open-circuit $i_{pv} = 0$, it follows from (3) that the diode reverse-bias saturation current i_s can be expressed by the open-circuit voltage $v_{oc} = v_{pv}(i_{pv} = 0)$:

$$i_s = \frac{i_{ph} - \frac{v_{oc}}{R_h}}{\exp\left(\frac{1}{A_n} \frac{v_{oc}}{v_T} \right) - 1}. \tag{4}$$

The model parameters A_n and R_h are assumed to be constant. After this, the dependencies from the irradiance and the cell temperature of the variables i_{ph} and v_{oc}, are addressed.

2.2 Array of Photo Cells

Multiple identical photocells are connected in parallel and series to form one PV array. The number of N_p parallel branches multiply with the photo current of one cell and the number of photocells within one branch N_s multiplies with the voltage of one cell. This gives the accumulated current I_{pv} and voltage V_{pv}: $I_{pv} = N_p i_{pv}$, $V_{pv} = N_s v_{pv}$, see Figure 3. Nevertheless, we use the lower case letters in the following to refer to the PV array values, regardless of the number of PV cells used.

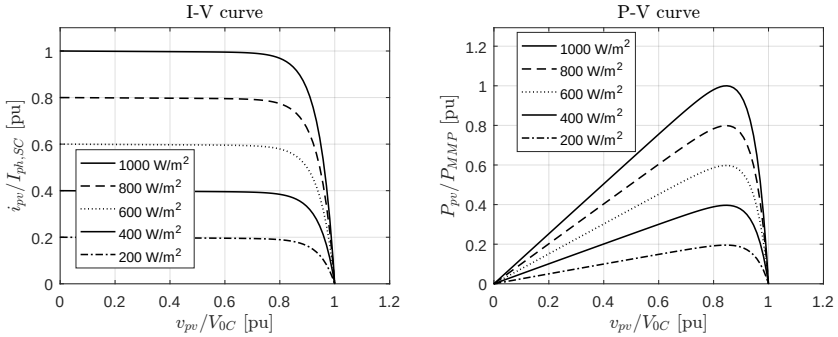


Figure 4: *I-V*-characteristic (left hand side) and *P-V*-characteristic (right hand side) for different irradiance.

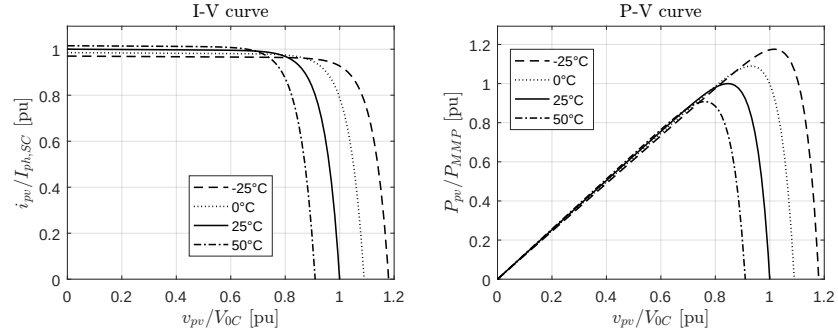


Figure 5: *I-V*-characteristic (left hand side) and *P-V*-characteristic (right hand side) for different cell temperatures.

2.3 Variation with Irradiance and Temperature

Derivations from the STC are addressed by the following correction formulas:

$$\frac{i_{\mathrm{ph}}}{i_{\mathrm{ph}}^{\mathrm{STC}}} = \frac{S}{S^{\mathrm{STC}}}\left[1+\alpha_{\mathrm{T}}(T_{\mathrm{c}}-T_{\mathrm{c}}^{\mathrm{STC}})\right], \quad \frac{v_{\mathrm{oc}}}{v_{\mathrm{oc}}^{\mathrm{STC}}} = 1+\beta_{\mathrm{T}}(T_{\mathrm{c}}-T_{\mathrm{c}}^{\mathrm{STC}}). \qquad (5)$$

In the latter formula, the weak impact on the open-circuit voltage by the change of radiation is neglected. The temperature coefficients α_{T} and β_{T} are usually given by the manufacturer. One can see the resulting characteristics for change of irradiance and cell temperature in Figure 4 and Figure 5 respectively.

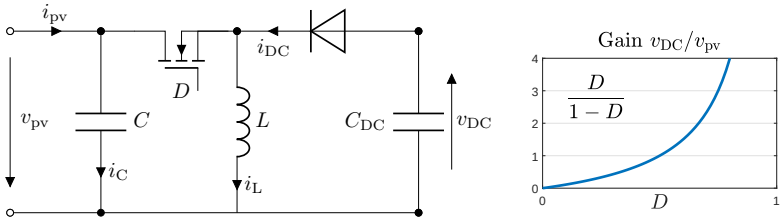


Figure 6: Buck-Boost converter. In switching mode D denotes whether the MOSFET is open ($D = 0$) or closed ($D = 1$). The average value of duty cycle and the corresponding voltage ratio is given in the formula on the right hand side. Consider the definition of v_{DC} in such a way, that v_{DC} has the same sign as v_{pv}.

3 DCDC-Converter

As one can see in (3), the PV cells operating point is controlled via the PV voltage v_{pv}. This voltage is kept constant by a DCDC-Converter which feeds the generated power into a DC link, see Figure 1. Therefore, the converter has to cancel out the disturbances of changing direct link voltage v_{DC} or changes of the incoming PV current i_{pv}.

3.1 Converter Model

We perform our analysis exemplary for three different converters: the buck converter, the boost converter and the buck-boost converter. Circuits and basic equations for these converters can be found in [10], for instance the circuit of the buck-boost converter is given in Figure 6. The converter circuits can be analyzed either for the switche FET (pulse width modulation mode) or as an averaging model. In the first case, D can be either 0 or 1, representing the state of the either closed or opened FET. In the later case $D \in [0,1]$ represents the average over time of open state of the FET, so called duty cycle D. We define the input and output voltage of the converters in such a way, that they are equal in sign (see Figure 6). Then, the averaging models have the following voltage ratios: $v_{\mathrm{DC}}/v_{\mathrm{pv}} = D$ (buck converter), $v_{\mathrm{DC}}/v_{\mathrm{pv}} = 1/(1-D)$ (the boost converter) and $v_{\mathrm{DC}}/v_{\mathrm{pv}} = D/(1-D)$ (buck-boost converter).

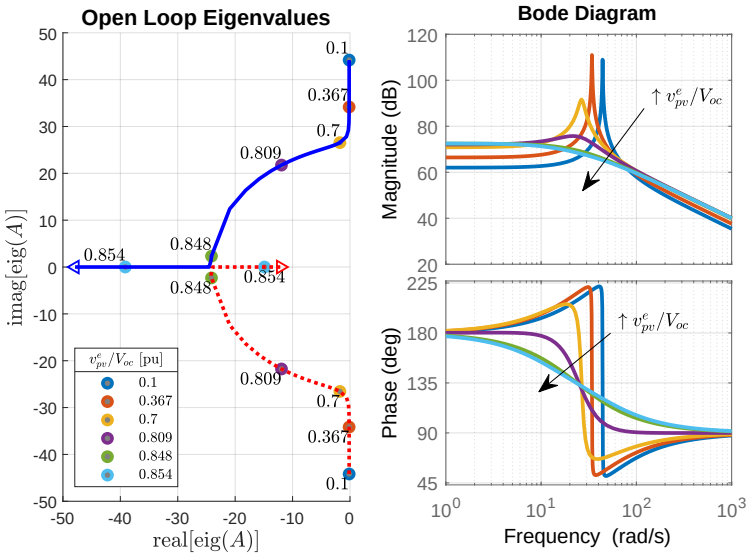


Figure 7: Eigenvalue and Bode diagram for the buck-boost converter coupled with the eSDM.

3.2 Dimensioning of the Electronic parts

The parts used in the converter, namely L and C, are dimensioned in dependence of the nominal operating point. This point is characterized by the maximum power point (MPP) under standard test conditions, i.e. $(v^0_{\mathrm{pv}}, i^0_{\mathrm{pv}})$. On the other side of the converter the nominal direct current link voltage v^0_{DC} gives the nominal duty cycle D^0. Finally the pick-to-pick value of ripple current Δi_{L} and the pick-to-pick value of ripple voltage Δv_{pv} together with the switching frequency f_{sw} are considered:

$$L = \frac{1}{\Delta i_{\mathrm{L}} f_{\mathrm{sw}}} \begin{cases} v^0_{\mathrm{DC}}(1-D^0) \\ v^0_{\mathrm{pv}} D^0 \\ v^0_{\mathrm{pv}} D^0 \end{cases} \qquad C = \frac{1}{\Delta v_{\mathrm{pv}} f_{\mathrm{sw}}} \begin{cases} i^0_{\mathrm{pv}}(1-D^0) & \text{Buck} \\ \Delta i_{\mathrm{L}}/8 & \text{Boost} \\ i^0_{\mathrm{pv}}(1-D^0) & \text{Buck-Boost} \end{cases} \tag{6}$$

3.3 State Space Equation and Linearization

Under the assumption, that the polarity of the switch current or voltage does not change during the whole cycle (continuous conduction mode, see [11]), switching mode and average analysis results in the same describing state space equations for the converter:

$$\frac{dv_{\mathrm{pv}}}{dt} = \frac{1}{C}\begin{cases} i_{\mathrm{pv}} - i_{\mathrm{L}}D \\ i_{\mathrm{pv}} - i_{\mathrm{L}} \\ i_{\mathrm{pv}} - i_{\mathrm{L}}D \end{cases} \quad \frac{di_{\mathrm{L}}}{dt} = \frac{1}{L}\begin{cases} v_{\mathrm{pv}}D - v_{\mathrm{DC}} & \text{Buck} \\ v_{\mathrm{pv}} - v_{\mathrm{DC}}(1-D) & \text{Boost} \\ v_{\mathrm{pv}}D - v_{\mathrm{DC}}(1-D) & \text{Buck-Boost} \end{cases} \tag{7}$$

Define the input as $u = D$, the states as $\mathbf{x} = (v_{\mathrm{pv}}, i_{\mathrm{L}})^{\mathrm{T}}$ and the function $i_{\mathrm{pv}} = f(v_{\mathrm{pv}})$ one gets

$$\frac{dx_1}{dt} = \frac{1}{C}\begin{cases} f(x_1) - x_2 u \\ f(x_1) - x_2 \\ f(x_1) - x_2 u \end{cases} \quad \frac{dx_2}{dt} = \frac{1}{L}\begin{cases} x_1 u - v_{\mathrm{DC}} & \text{Buck} \\ x_1 - v_{\mathrm{DC}}(1-u) & \text{Boost} \\ x_1 u - v_{\mathrm{DC}}(1-u) & \text{Buck-Boost} \end{cases} \tag{8}$$

Taylor linearization around one arbitrary stationary operating point (superscript e) defined by $\underline{x}^e = (v_{\mathrm{pv}}^e, i_{\mathrm{L}}^e)^{\mathrm{T}}$ and $u^e = D^e$ gives the linear system ($\Delta\underline{x} = \underline{x} - \underline{x}^e$, $\Delta u = u - u^e$)

$$\frac{d}{dt}\Delta\underline{x} = \begin{bmatrix} f'(v_{\mathrm{pv}}^e)/C & -a^e/C \\ a^e/L & 0 \end{bmatrix}\Delta\underline{x} + \begin{bmatrix} -b_1^e/C \\ +b_2^e/L \end{bmatrix}\Delta u = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta u, \tag{9}$$

$$a^e = \begin{cases} D^e \\ 1 \\ D^e \end{cases} \quad b_1^e = \begin{cases} i_{\mathrm{L}}^e \\ 0 \\ i_{\mathrm{L}}^e \end{cases} \quad b_2^e = \begin{cases} v_{\mathrm{pv}}^e & \text{Buck} \\ v_{\mathrm{DC}}^e & \text{Boost} \\ v_{\mathrm{DC}}^e + v_{\mathrm{pv}}^e & \text{Buck-Boost} \end{cases} \tag{10}$$

For all converters $y = v_{\mathrm{pv}} = x_1$ applies, and therefore $\Delta y = \mathbf{C}\Delta\mathbf{x}$, $\mathbf{C} = (1,0)$.

A brief analysis of the obtained linear models shows a strong dependency of the model properties from the chosen operating point. Assume standard test conditions for temperature and irradiation, then the eigenvalues and bode plots shown in Figure 7 are obtained for different PV voltages v_{pv}^e.

3.4 DCDC-Converter Control Loop

To meet the predefined reference PV voltage $v_{\mathrm{pv,ref}}$ a I-state-space controller is used:

$$\Delta D = \Delta u = -\mathbf{K}_x \Delta \mathbf{x} - K_\mathrm{I} \int_0^t (v_{\mathrm{pv,ref}} - v_{\mathrm{pv}}) d\tau. \tag{11}$$

The integral on the right-hand side of (11) can be understood as an additional state. Let be $x_\mathrm{I} = \int (v_{\mathrm{pv}}^e - v_{\mathrm{pv}}) d\tau$ and $x_\mathrm{I}^e = \int (v_{\mathrm{pv}}^e - v_{\mathrm{pv,ref}}) d\tau$, then $\Delta x_\mathrm{I} = x_\mathrm{I} - x_\mathrm{I}^e$ and $\Delta \dot{x}_\mathrm{I} = -\mathbf{C}\Delta\mathbf{x}$, under the assumption $v_{\mathrm{pv,ref}} = \mathrm{const}$. Collecting the states as one vector this gives the new state space equation:

$$\frac{d}{dt}\Delta\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{A} & \mathbf{0}_{2\mathrm{x}1} \\ -\mathbf{C} & 0 \end{pmatrix} \Delta\tilde{\mathbf{x}} + \begin{pmatrix} \mathbf{B} \\ 0 \end{pmatrix} \Delta u = \tilde{\mathbf{A}}\Delta\tilde{\mathbf{x}} + \tilde{\mathbf{B}}\Delta u, \quad \Delta\tilde{\mathbf{x}} = \begin{pmatrix} \mathbf{x} - \mathbf{x}^e \\ x_\mathrm{I} - x_\mathrm{I}^e \end{pmatrix}. \tag{12}$$

Following this scheme, the controller can be written as state space controller, namely $\Delta u = -\tilde{\mathbf{K}}\Delta\tilde{\mathbf{x}}$ and $\tilde{\mathbf{K}} = (\mathbf{K}_x^\mathrm{T}, K_\mathrm{I})^\mathrm{T}$.

3.5 LMI Controller Design - Single Model

The controller in form of the matrix $\tilde{\mathbf{K}}$ can be found using linear matrix inequalities (LMIs) conditions. The conditions are in general derived from analysis using the Direct method of Lyapunov [12]. A common Lyapunov function candidate is of the quadratic form, $V = \mathbf{x}^\mathrm{T}\mathbf{P}\mathbf{x}$. Together with the controller and the state space equation this gives the matrix inequality that contains the unknown controller $\tilde{\mathbf{K}}$ and the variable $\mathbf{P}$:

$$\forall \mathbf{x} : \frac{dV}{dt} = \frac{d}{dt}(\mathbf{x}^\mathrm{T}\mathbf{P}\mathbf{x}) = \mathbf{x}^\mathrm{T} \underbrace{\left[(\tilde{\mathbf{A}} - \tilde{\mathbf{B}}\tilde{\mathbf{K}})^\mathrm{T}\mathbf{P} + \mathbf{P}(\tilde{\mathbf{A}} - \tilde{\mathbf{B}}\tilde{\mathbf{K}})\right]}_{\prec 0} \mathbf{x} < 0. \tag{13}$$

Their feasibility can be verified by finding a solution using interior-point methods [13]. In this study we use the Yalmip library for Matlab [14] together with the Mosek solver [15].

The inequation (13) can be extended to add some restrictions about the closed-loop pole [16]. In this, we set a minimum decay rate α, a minimum damping

Table 1: Parameter of the single model

parameter	$v_{\mathrm{pv}}^{e}/v_{\mathrm{pv,MPP}}$	$S^{e}=S^{\mathrm{STC}}$	$T_{\mathrm{c}}^{e}=T_{\mathrm{c}}^{\mathrm{STC}}$	α	Θ	r
value	0.9	1000 W/m^2	298 K	25	20°	51

ratio $\zeta=\cos(\Theta)$, and a maximum undamped natural frequency $\omega_d = r\sin(\Theta)$. The so defined area and the actual area of the poles are marked in Figure 8a.

For the single model, containing only one operational point, the restrictions are given in Table 1. The so found controller has the closed-loop eigenvalues shown in Figure 8a. While the behavior of this controller around its operational point is quite satisfactory, larger deviations from this operational point give poor results in terms of overshoot and oscillations, see Figure 8b. Therefor the Takagi-Sugeno Model will be used in the next steps, utilizing multiple operation points.

3.6 Takagi-Sugeno (TS)-Model

The nonlinear system under investigation is described via the equations:

$$\underline{\dot{x}} = \mathbf{f}(\mathbf{x},\mathbf{u},\theta), \quad \mathbf{x}_0 = \mathbf{x}(t_0), \quad \mathbf{y} = \mathbf{g}(\mathbf{x}), \tag{14}$$

where

$$\begin{cases} \mathbf{f}:\ \mathbb{R}^n \ \rightarrow\ \mathbb{R}^n, & \text{smooth vector-valued function} \\ \mathbf{g}:\mathbb{R}^n \rightarrow \mathbb{R}^p, & \text{smooth vector-valued function} \\ \mathbf{x}\in\mathbb{R}^n, & \text{state space vector} \\ \mathbf{u}\in\mathbb{R}^m, & \text{input vector with controllable and uncontrollable inputs} \\ \mathbf{y}\in\mathbb{R}^p, & \text{output vector} \\ \theta\in\mathbb{R}^d, & \text{time variable parameter vector} \end{cases}$$

Taylor linearization of (14) around $i = 1, 2., ..., N_r$ different equilibrium points $\{\mathbf{x}_i^e, \mathbf{u}_i^e\}$ which fulfill $\mathbf{f}(\mathbf{x}_i^e, \mathbf{u}_i^e) = \mathbf{0}$ gives

$$\Delta\dot{\mathbf{x}}_i = \mathbf{A}_i\Delta\mathbf{x}_i + \mathbf{B}_i\Delta\mathbf{u}_i\,, \quad \Delta\mathbf{y}_i = \mathbf{C}_i\Delta\mathbf{x}_i. \tag{15}$$

All these linear systems share the same state and input vector. The TS model is the weighted sum of these linear systems, giving an interpolation of N_r LTI systems:

$$\underline{\dot{x}} = \sum_{i=1}^{N_r} h_i(\underline{z})\,(\mathbf{A}_i\mathbf{x} + \mathbf{B}_i\mathbf{u} + \mathbf{a}_i)\,, \quad \mathbf{a}_i = -\mathbf{A}_i\mathbf{x}_i^e - \mathbf{B}_i\mathbf{u}_i^e \tag{16}$$

$$\mathbf{y} = \sum_{i=1}^{N_r} h_i(\underline{z})\,(\mathbf{C}_i\mathbf{x} + \mathbf{c}_i)\,, \quad \mathbf{c}_i = -\mathbf{C}_i\mathbf{x}_i^e \tag{17}$$

The functions $h_i : \mathbb{R}^l \to \mathbb{R}$ are the membership functions and fulfill the convex sum condition: $\forall\mathbf{z} : \sum h_i(\mathbf{z}) = 1$. The vector of the so-called premise variables $\mathbf{z}$ may directly contains states x_k, inputs u_k, and time variable parameters θ_k or be a function $\mathbf{z} = \mathbf{z}(\mathbf{x}, \mathbf{u}, \theta)$. In the following we assume, that the equilibrium points are selected by premise variables $\mathbf{z}^e$, arranged on a Cartesian grid. Then we can switch from one linear index i (numbering all models from 1 to N_r) to a total of d Cartesian coordinates $i_1, i_2, ..., i_d$ (number of premise variables is equal to d). The transformation between both ways of indexing is given by:

$$\mathbf{z}_i^e = [z_{1,i_1}^e, z_{2,i_2}^e, ..., z_{d,i_d}^e]^{\mathrm{T}}, \begin{cases} i_k = 1...n_k \\ k = 1...d \end{cases} \Rightarrow i = 1 + \sum_{k=1}^{d} (i_k - 1) \prod_{l=1}^{k-1} n_l. \tag{18}$$

The membership functions are constructed as products of local test-functions

$$h_i(\mathbf{z}) = h_{i_1,i_2,...,i_d}(\mathbf{z}) = \prod_{k=1}^{d} w_{k,i_k}(z_k), \quad \mathbf{z} = [z_1, z_2, ..., z_d]^{\mathrm{T}}. \tag{19}$$

These local test-functions are almost everywhere zero, equals one only at the corresponding supporting point, that is $w_{k,i_k}(z_{k,j_k}^e) = \delta_{i_k,j_k}$ (δ_{ij} is the Kronecker delta), and are interpolated by a given function g between.

Table 2: Parameter defining the operational points of the TS-Model

parameter	values
$v^e_{\mathrm{pv},i}/v_{\mathrm{pv,MPP}}$	$\{0.3, 0.6, 0.9\}$
S^e_i	$\{200, 333, 467, 600, 733, 867, 1000\}$ W/m^2
$T^e_{\mathrm{c},i}$	$\{273, 303, 333\}$ K

Assuming ascending order of the supporting points, $z^e_{k,j_k-1} < z^e_{k,j_k}$, this can be written as

$$w_{k,i_k}(z_k) = \begin{cases} 1 & ,z_k \le z^e_{k,1} \text{ or } z^e_{k,n_k} \le z_k \\ g\left(|z_k - \hat{z}_{k,i_k}|/|z^e_{k,i_k} - z^e_{k,i_k-1}|\right) & ,z^e_{k,i_k-1} \le z_k \le z^e_{k,i_k} \\ g\left(|z^e_k - z^e_{k,i_k}|/|z^e_{k,i_k} - z^e_{k,i_k+1}|\right) & ,z^e_{k,i_k} \le z_k \le z^e_{k,i_k+1} \\ 0 & ,\text{otherwise} \end{cases} . \quad (20)$$

In this we use a very simple function g that satisfy the convex sum condition, namely $g(x) = 1 - x$, which gives triangular shaped test-functions.

Under stationary conditions three parameters are needed to determine the state of the systems, they form the vector of premise variables $\mathbf{z} = (v_{\mathrm{pv}}, S, T_{\mathrm{c}})^{\mathrm{T}}$. The parameters defining the chosen 63 models, are given in Table 2.

3.7 LMI Controller Design - TS-Model

The TS-Model is now used to find controller for each operational point, in such a way that the following control law is applicable:

$$u = D = \sum_{i=1}^{N_r} h_i \left[D^e_i - \mathbf{K}_{x,i}(\mathbf{x} - \mathbf{x}^e_i) - K_{I,i} x_{\mathrm{I}}\right], \quad x_{\mathrm{I}} = \int_0^t (v_{\mathrm{pv,ref}} - v_{\mathrm{pv}}) d\tau. \quad (21)$$

The methodology in applying the Direct method of Lyapunov is the same as before. But now, multiple inequations of the type (13) has to be solved simultaneously for different matrices $\tilde{\mathbf{K}}_i$ but the same variable $\mathbf{P}$. Using the same $\mathbf{P}$ guarantee not only local stability, but global stability. Like before,

additional constraints are applied to the LMIs. They are chosen independently for each model:

$$\alpha_i = 2 \times \lambda_{\text{min,Re},i}^{\text{open-loop}}, \quad r_i = 10 \times \lambda_{\text{max,Abs},i}^{\text{open-loop}}, \quad \Theta = 20^\circ \tag{22}$$

with

$$\begin{cases} \lambda_{\text{min,Re},i}^{\text{open-loop}}, & \text{smallest absolute real part of the open-loop eigenvalues} \\ \lambda_{\text{max,Abs},i}^{\text{open-loop}}, & \text{largest norm of the open-loop eigenvalues} \end{cases}$$

The position of the eigenvalues is exemplary given for two models in Figure 8a. The simulation results shows a much better performance for the TS-controller than for the single controller, see Figure 8c.

4 MPP and DPP Tracking Methods

Classical MPP techniques can be divided into offline techniques, like the fractional open-circuit voltage and the fractional short-circuit current techniques; online techniques, like the perturb-and-observe and the incremental conductance techniques; and advanced (by means of computational effort) techniques [17].

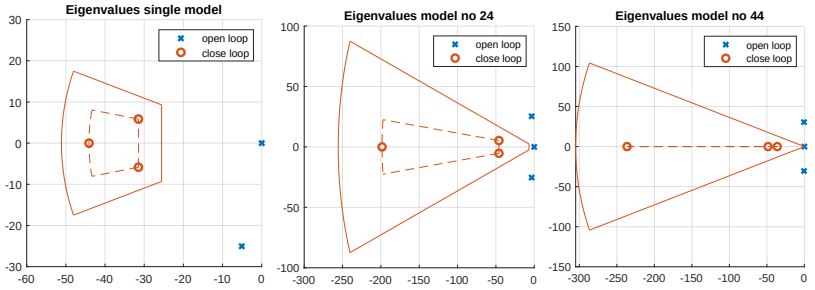
In this we use the perturb and observe (P&O) method, e.g. [18, 19, 20], and extend this method for DPP tracking. The basic idea of this method is to vary (perturb) the PV voltage and to observe the change in the power output. The direction of the voltage steps is kept if the power increases and reverses otherwise. Metaphorically speaking, the operational point moves towards the extremum, hence this method is also called hill climbing method. Once it reached the MPP, the system will oscillate about the MPP. Choice of the step size affects both, the time it takes to reach the MPP and the minimal distance to the MPP due to the oscillations. The first effect makes a large as reasonable step size desirable, the second effect forces the step size to be as small as possible.

The perturb and observe method can fail under changing irradiation conditions. If irradiation increases, then regardless of the direction of the perturbation, the power output is increasing and the algorithm will keep the direction. In [21] a three-point weight comparison P&O method is used to overcome this problem.

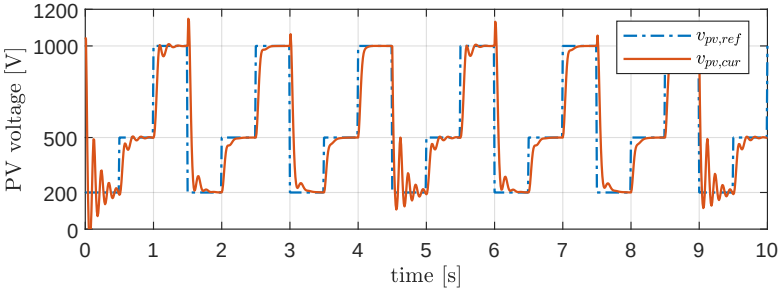
Extension for DPP tracking can be done by replacing the feedback from the produced power $P_{\mathrm{pv}} = v_{\mathrm{pv}} \times i_{\mathrm{pv}}$ to the function $P^*_{\mathrm{pv}} = -|P_{\mathrm{pv}} - P_{\mathrm{dem}}|$. One can see, that this results in a large range for the PV voltage, which makes the advanced TS controller useful if not necessary.

5 Conclusion

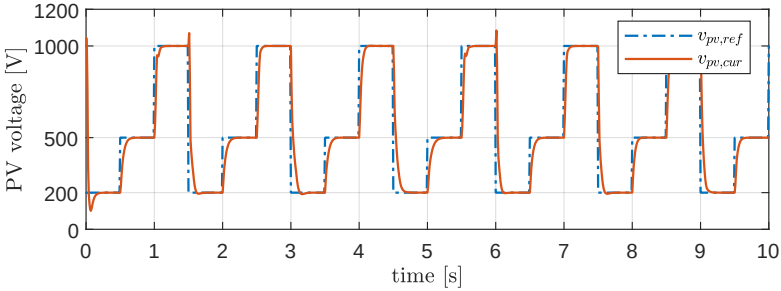
The modelling of a PV power station has been described. Power conversion is performed within two steps, firstly by a buck-boost converter adjusting the power to the DC link level, secondly by an inverter meeting the requirements of the point of common coupling. The first conversion step is described in detail since here a modified Perturb and Observation method is applied to realise a DPP tracking. To cover a large operation range of the converter, the PV voltage solely adjusts the PV current and therefore the produced power, a Takagi-Sugeno Modell is used. The improved performance of the multi-operational point model has been shown in the simulation. This paper shows one way, how PV plants can contribute more to grid stability. An inevitable feature on the market of growing PV power share.



(a) Shown are the Eigenvalues of the open-loop and closed-loop models. The single model (left-hand side) arises from the linearization at one operational point only. On the contrary, the TS model arises from 63 different operational points. Eigenvalues for the TS model number 24 (middle) and 44 (right-hand side) are shown.



(b) Shown is the step response in the PV voltage $v_{\mathrm{pv,cur}}$ for the single model (controller design for one operational point). The values of the reference PV voltage $v_{\mathrm{pv,ref}}$, irradiation S, and cell temperature T_{c} change their values abruptly every 0.5 seconds.



(c) Shown is the step response in the PV voltage $v_{\mathrm{pv,cur}}$ for the TS model (controller design for a total of 63 operational points). The values of the reference PV voltage $v_{\mathrm{pv,ref}}$, irradiation S, and cell temperature T_{c} change their values abruptly every 0.5 seconds.

References

[1] B. Kroposki, B. Johnson, Y. Zhang, V. Gevorgian, P. Denholm, B.-M. Hodge, and B. Hannegan, "Achieving a 100% Renewable Grid: Operating Electric Power Systems with Extremely High Levels of Variable Renewable Energy," *IEEE Power and Energy Magazine*, vol. 15, pp. 61–73, Mar. 2017.

[2] T. Esram and P. Chapman, "Comparison of Photovoltaic Array Maximum Power Point Tracking Techniques," *Energy Conversion, IEEE Transactions on*, vol. 22, pp. 439–449, July 2007.

[3] F. Pöschke, E. Gauterin, M. Kühn, J. Fortmann, and H. Schulte, "Load mitigation and power tracking capability for wind turbines using linear matrix inequality-based control design," *Wind Energy*, vol. 23, no. 9, pp. 1792–1809, 2020.

[4] W. W. Simon Philipps, "Photovoltaics report," tech. rep., Fraunhofer ISE, PSE Projects GmbH, Sept. 2020.

[5] N. Klaes, N. Goldschmidt, and J. Fortmann, "Voltage Fed Control of Distributed Power Generation Inverters with Inherent Service to Grid Stability," *Energies*, vol. 13, p. 2579, Jan. 2020.

[6] S. D'Arco and J. A. Suul, "Equivalence of Virtual Synchronous Machines and Frequency-Droops for Converter-Based MicroGrids," *IEEE Transactions on Smart Grid*, vol. 5, pp. 394–395, Jan. 2014.

[7] B. Marinescu, O. Gomis-Bellmunt, F. Dörfler, H. Schulte, and L. Sigrist, "Dynamic Virtual Power Plant: A New Concept for Grid Integration of Renewable Energy Sources," *arXiv:2108.00153 [cs, eess]*, July 2021. arXiv: 2108.00153.

[8] W. Xiao, *Photovoltaic Power System: Modeling, Design, and Control*. Wiley, 2017.

[9] W. Shockley, "The theory of p-n junctions in semiconductors and p-n junction transistors," *The Bell System Technical Journal*, vol. 28, no. 3, pp. 435–489, 1949.

[10] R. W. Erickson, “DC–DC Power Converters,” in *Wiley Encyclopedia of Electrical and Electronics Engineering*, American Cancer Society, 2007.

[11] R. W. Erickson and D. Maksimovic, *Fundamentals of Power Electronics*. Springer US, 2 ed., 2001.

[12] A. M. LYAPUNOV, “The general problem of the stability of motion,” *International Journal of Control*, vol. 55, pp. 531–534, Mar. 1992.

[13] Y. Nesterov and A. Nemirovskii, *Interior-Point Polynomial Algorithms in Convex Programming*. Studies in Applied and Numerical Mathematics, Society for Industrial and Applied Mathematics, Jan. 1994.

[14] J. Löfberg, “Yalmip : A toolbox for modeling and optimization in matlab,” in *In Proceedings of the CACSD Conference*, (Taipei, Taiwan), 2004.

[15] MOSEK ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 9.0.*, 2019.

[16] M. Chilali and P. Gahinet, “H_∞ design with pole placement constraints: an LMI approach,” *IEEE Transactions on Automatic Control*, vol. 41, pp. 358–367, Mar. 1996.

[17] A. Eltamaly and A. Abdelaziz, *Modern Maximum Power Point Tracking Techniques for Photovoltaic Energy System*. Aug. 2019.

[18] O. Wasynezuk, “Dynamic Behavior of a Class of Photovoltaic Power Systems,” *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-102, pp. 3031–3037, Sept. 1983.

[19] W. Teulings, J. Marpinard, A. Capel, and D. O’Sullivan, “A new maximum power point tracking system,” in *Proceedings of IEEE Power Electronics Specialist Conference - PESC ’93*, pp. 833–838, June 1993.

[20] N. D’Souza, L. Lopes, and X. Liu, “An Intelligent Maximum Power Point Tracker Using Peak Current Control,” in *2005 IEEE 36th Power Electronics Specialists Conference*, pp. 172–, June 2005. ISSN: 2377-6617.

[21] Y.-T. Hsiao and C.-H. Chen, “Maximum power tracking for photovoltaic power system,” in *Conference Record of the 2002 IEEE Industry Applications Conference. 37th IAS Annual Meeting (Cat. No.02CH37344)*, vol. 2, pp. 1035–1040 vol.2, Oct. 2002. ISSN: 0197-2618.

Approximative Modellierung eines LLC-Resonanzwandlers mit Takagi-Sugeno-Modellen

Alessio Cavaterra[1], Martin Wattenberg[2], Ulf Schwalbe[1], Steven Lambeck[1]

[1]FB Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123, 36037 Fulda
E-Mail: {alessio.cavaterra,ulf.schwalbe,steven.lambeck}@et.hs-fulda.de

[2]Infineon Technologies Austria AG
siemensstr. 2, 9500 Villach
E-Mail: martin.wattenberg@infineon.com

1 Einführung

Elektrofahrräder (kurz „E-Bikes“) sind aus der modernen Mobilität nicht mehr wegzudenken und gewinnen stetig an Bedeutung [1, 2]. Die steigenden Absatzzahlen motivieren Entwicklungsarbeiten an kompakten und leichten Batterieladegeräten. Im F&E-Projekt „SCharger“ des Fachbereichs Elektrotechnik und Informationstechnik der Hochschule Fulda werden solche Verbesserungen untersucht. Aufbauend auf einer Resonanzwandler-Topologie mit einem LLC-Schwingkreis (s. Bild 1) wird erforscht inwiefern abseits von der üblichen gleichstrom- und gleichspannungsorientierten Ladestrategie eine Batterieladestrategie mit einem AC-Batteriestrom und einer DC-Batterieladespannung (AC - engl. „alternating current“, DC - engl. „direct current“) durch Ausbau des üblichen Zwischenkreiskondensators ermöglicht werden kann [3, 4]. Der Verzicht auf dieses Bauteil bewirkt eine pulsierende 100 Hz-Netzspannung, die wiederrum permanente Parameteränderungen hervorruft. Diese und viele weitere nichtlineare Effekte stellen hohe Anforderungen an das Modell und das darauf basierende Steuer- und Regelgesetz.

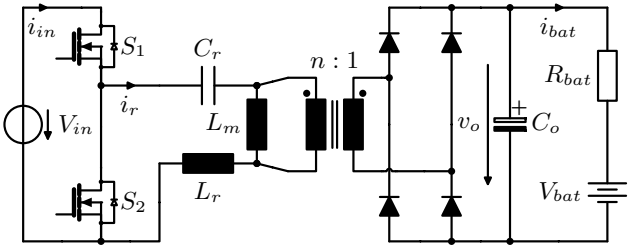


Bild 1: Schaltplan eines LLC-Resonanzwandlers mit ausgangsseitiger Batterie.

Der vorliegende Beitrag zeigt in Kürze die simple Modellierung des LLC-Resonanzwandlers mit Hilfe von Takagi-Sugeno (TS) Fuzzy-Modellen, die einen systematischen Reglerentwurf in künftigen Arbeiten vereinfachen soll. Hierbei werden die Kleinsignalmodelle von Mohammedi et al. [5] als Submodelle innerhalb des TS-Systems verwendet und somit das Großsignalverhalten approximiert. Der Beitrag legt den Fokus auf die Modellierungsergebnisse.

2 Lineare Kleinsignalmodelle für LLC-Wandler

Die mathematische Modellbildung von LLC-Resonanzwandlern kann auf unterschiedlichen Wegen erfolgen. Mohammedi et al. schlagen in [5] eine neue Methode zur Modellbildung eines LLC-Wandlers mit Hilfe des sog. „Homopolaritätszyklus“ vor. Der Homopolaritätszyklus beschreibt hierbei, zu welchem Anteil innerhalb eines Schaltzyklus die Polarität der Inverterspannung v_{inv} und die Polarität der Sekundärkreisspannung v_{sec} das selbe Vorzeichen besitzen. In Kombination mit der Schaltfrequenz f_s (Stellgröße) der Leistungsschalter in der Halbbrücke und der Resonanzfrequenz

$$f_r = \frac{1}{2\pi\sqrt{C_r L_r}} \tag{1}$$

können nun die zwei Arbeitsbereiche $f_s \leq f_r$ (Boost-Modus bzw. Resonanzfall) und $f_s > f_r$ (Buck-Modus) definiert werden.

Für jeden Bereich schlagen die Autoren jeweils zwei lineare Übertragungsfunktionen vor, welche die Ausgangsspannung $\hat{v}_o$ in Abhängigkeit von der

Schaltfrequenz $\hat{f}_s$ sowie der Eingangsspannung $\hat{v}_{in}$ wie folgt zusammenfassen:

$$\hat{v}_o(s) = G_{b,fs}(s) \cdot \hat{f}_s(s) + G_{b,vin}(s) \cdot \hat{v}_{in}(s) \tag{2}$$

$$\hat{v}_o(s) = G_{a,fs}(s) \cdot \hat{f}_s(s) + G_{a,vin}(s) \cdot \hat{v}_{in}(s) \tag{3}$$

Die Indizes b bzw. a deuten den unteren Arbeitsbereich („below“) bzw. den oberen Arbeitsbereich („above“) an. Der Resonanzfall $f_s = f_r$ ist dabei in Gleichung (2) abgedeckt. Aus Platzgründen wird auf die Darstellung der Übertragungsfunktionen verzichtet und auf die Veröffentlichung [5] verwiesen. Alle Übertragungsfunktionen weisen ein schwingungsfähiges Verzögerungsverhalten zweiter Ordnung auf. Die jeweiligen Dämpfungsgrade und Eigenkreisfrequenzen der Übertragungsglieder sind hierbei insbesondere abhängig von der Eingangsspannung V_{in} und der Schaltfrequenz f_s. Wie im nächsten Abschnitt deutlicher gezeigt wird, lassen sich für das Batteriestromsignal sehr ähnliche Aussagen treffen.

3 Approximation über Takagi-Sugeno-Modell

Die Erweiterung des LLC-Resonanzwandlers von Mohammadi et al. [5] um eine Batterielast am Ausgang gestaltet sich vergleichsweise einfach, wenn in Reihe zum ohmschen Lastwiderstand eine DC-Spannungsquelle eingefügt wird (s. Bild1) und zugleich die Annahmen gelten, dass sämtliche Änderungen des Batteriestroms nicht vorhanden und die DC-Batteriespannung konstant ist.

$$i_{bat} = \frac{v_o - V_{bat}}{R_{bat}}, \quad \frac{\partial i_{bat}}{\partial t} = 0, \quad V_{bat} = \text{const.} \tag{4}$$

Über diese Annahmen und den Herleitungen in [5] lässt sich damit explizit eine Gleichung für den Batteriestrom $i_{bat}(s)$ im Laplace-Bereich aufstellen.

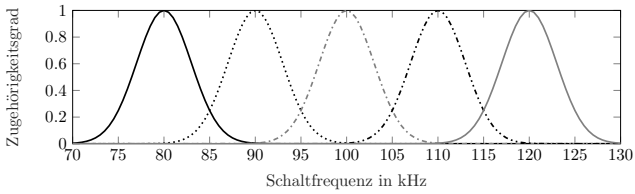


Bild 2: Äquidistante Gauß'sche Zugehörigkeitsfunktionen der Schaltfrequenzen.

Um nicht den Rahmen des Kurzbeitrags zu sprengen, wird im Folgenden auf die Herleitung verzichtet.

$$i_{bat}(s) = G_{i,x,fs}(s) \cdot \hat{f}_s(s) + G_{i,x,vin}(s) \cdot \hat{v}_{in}(s) + G_{i,x,vbat}(s) \cdot V_{bat}(s) + \tilde{a}_y \quad \text{mit} \quad x = \{b, a\} \tag{5}$$

In obiger Gleichung stellt die Übertragungsfunktion $G_{i,x,vbat}(s)$ für alle Bereiche b und a nur eine Verstärkung dar. Dies gilt auch für den affinen Teilterm $\tilde{a}_y$. Die Übertragungsfunktionen für $\hat{f}_s(s)$ sowie $\hat{v}_{in}(s)$ sind jedoch vom Arbeitsbereich abhängig.

$$G_{i,b,fs}(s) = \frac{K_{b,fs} \cdot (\gamma_b s - \delta_b)}{s^2 + \alpha_b s + \beta_b} \qquad G_{i,b,vin}(s) = \frac{K_{b,vin}}{s^2 + \alpha_b s + \beta_b} \tag{6}$$

$$G_{i,a,fs}(s) = \frac{K_{a,fs}}{s^2 + \alpha_a s + \beta_a} \qquad G_{i,a,vin}(s) = \frac{K_{a,vin}}{s^2 + \alpha_a s + \beta_a} \tag{7}$$

In den Gleichungen (6) sowie (7) ist angedeutet, dass das charakteristische Polynom je nach Arbeitsbereich unverändert bleibt. Jedoch sind strukturelle Änderungen im unteren Arbeitsbereich zu beobachten: Die Übertragungsfunktion der Schaltfrequenz im unteren Arbeitsbereich $G_{i,b,fs}(s)$ besitzt eine Nullstelle in der rechten s-Halbebene. Im oberen Arbeitsbereich verschwindet diese Nullstellen hingegen (s. $G_{i,a,fs}(s)$ in Gleichung (7)). Das lokale Systemverhalten eines LLC-Resonanzwandlers verändert sich also in jedem Schaltzyklus nicht nur hinsichtlich des Dämpfungsgrades und der Eigenkreisfrequenz, sondern auch bezüglich der Struktur (Nullstelle).

Im Rahmen des SCharger-Projekts werden alle Arbeitsbereiche innerhalb einer 100 Hz-Periode zyklisch durchlaufen. Die einzelnen Übertragungsfunktionen

müssen daher miteinander verknüpft werden. Die Verknüpfung geschieht über ein TS-Modell mit N Submodellen, welches die Schaltfrequenz f_s als Schedulingvariable in der Prämisse nutzt. Die Gauß'schen Zugehörigkeitsfunktionen $\mu_j(f_s)$ gewährleisten hierbei ein „weiches" Umschalten zwischen den Arbeitsbereichen.

$$i_{bat}(s) = \sum_{j=1}^{N} h_j \cdot \left(G_{i,j,fs}(s) \cdot \hat{f}_s(s) + G_{i,j,vin} \cdot \hat{v}_{in}(s) + G_{i,j,vbat} \cdot V_{bat}(s) + \tilde{a}_y \right) \quad (8)$$

$$h_j = \frac{\mu_j}{\sum_{j=1}^{N} \mu_j} \quad (9)$$

4 Simulationsergebnisse

Die im Paper [5] gegebenen Parameter des LLC-Wandlers werden in einem MATLAB/Simulink-Simulationsmodell übernommen. Der Resonanzwandler besitzt eine Resonanzfrequenz bei $f_r = 96\,\text{kHz}$. Es werden $N = 5$ Zugehörigkeitsfunktionen μ_j mit $j = 1 \ldots N$ um f_r definiert, deren Erwartungswerte bei äquistant-verteilten Stützpunkten der Schaltfrequenz f_s angesetzt werden (s. Bild 2). Die Modellierungsgüte des TS-Modells wird anhand einiger sprungförmiger Anregungen der Schaltfrequenz f_s bzw. der Eingangspannung V_{in} im Bild 3 veranschaulicht.

5 Diskussion und Ausblick

Anhand der Experimente im Bild 3 wird deutlich, dass die Modellierungsgüte im oberen Arbeitsbereich (hohe Schaltfrequenzen f_s) aufgrund von hohen Fehlern von bis zu 2 A gering ist. Allerdings ist die Güte im unteren Arbeitsbereich

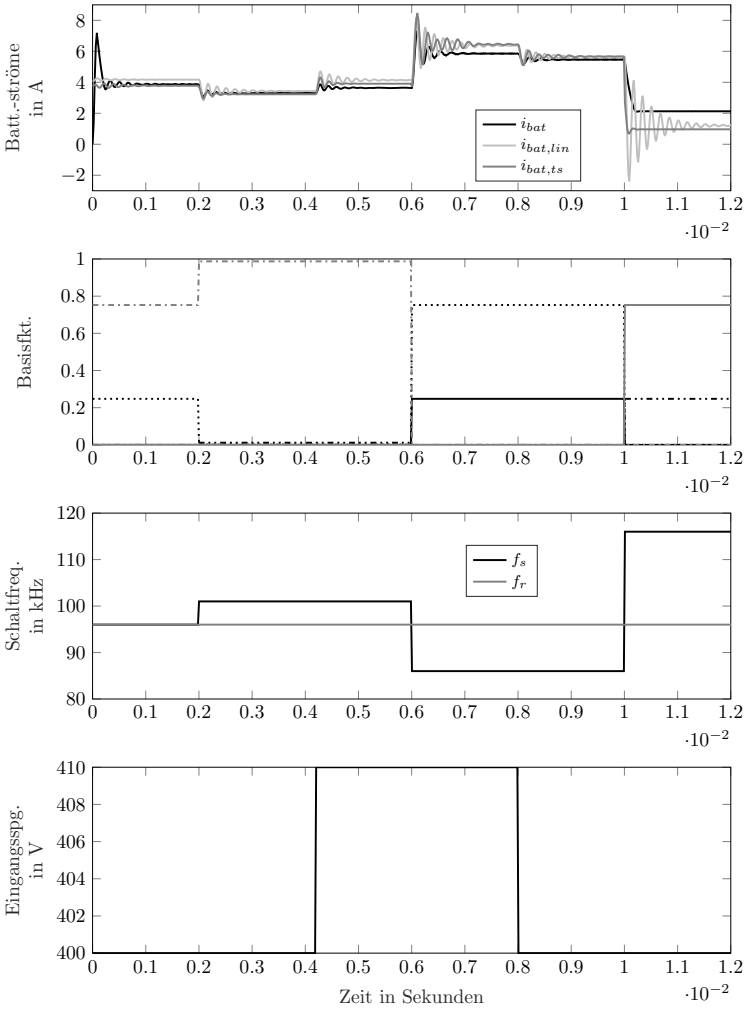


Bild 3: Sprungförmige Anregungen des Simulinkmodells und des TS-Modells über die Eingangsspannung und die Schaltfrequenz. Zur Zuordnung der Basisfunktionen vgl. mit Bild 2.

inkl. des Resonanzfalls ($f_s \leq f_r$) als sehr gut zu bewerten. Anhand der sprungförmigen Anregungen wird zudem deutlich, dass die Übergangsvorgänge hinsichtlich der Dynamik zufriedenstellend abgebildet werden. Währenddessen zeigt der Batteriestrom $i_{bat,lin}$, welcher aus der linearen Übertragungsfunktion für den Resonanzfall $f_r \approx f_s = 96\,\text{kHz}$ resultiert, eine deutliche Schwingneigung auf. Das TS-Modell liefert über den gesamten Arbeitsbereich insgesamt bessere Ergebnisse, als die einfache lineare Übertragungsfunktion. Gleichzeitig trägt es zu einem besseren Systemverständnis bei, das bei weiterführenden Untersuchungen eine wertvolle Grundlage bilden wird.

Weiterführende Arbeiten werden den Prämissenraum um die Eingangsspannung V_{in} erweitern und untersuchen, inwiefern durch eine Erhöhung der Anzahl der Zugehörigkeitsfunktionen je Schedulingvariable sowie über deren Positionierung im Prämissenraum eine Verbesserung der Modellierungsgüte erzielt werden kann.

Die vorliegende Veröffentlichung ist im Rahmen des Förderprogramms „Forschung für die Praxis“ durch das Hessische Ministerium für Wissenschaft und Kunst gefördert worden.

Literatur

[1] Statistisches Bundesamt. „Zahl der Haushalte mit E-Bikes hat sich seit 2015 fast verdreifacht“. *Pressemitteilung Nr. 375*. Wiesbaden. 28. September 2020.

[2] Ernst Brust. „Zahlen – Daten – Fakten zum Fahrradmarkt in Deutschland 2020“. Zweirad-Industrie-Verband (ZIV). Wirtschaftspressekonferenz am 10. März 2021 in Berlin. 2021.

[3] Martin Wattenberg, Ulf Schwalbe and Martin Pfost. „single-Stage LLC Charger with PFC Functionality and Wide Input Voltage Range“. *IEEE Applied Power Electronics Conference and Exposition (APEC)*. 2019.

[4] Wenqi Zhou, Martin Wattenberg and Ulf Schwalbe. „Design Considerations of a single Stage LLC Battery Charger“. *PCIM Europe 2019*. Nuremberg. Mai 2019.

[5] Mehdi Mohammadi, Franco Degioanni, Mohammad Mahdavi and Martin Ordonez. „Small-signal Modeling of LLC Converters Using Homopolarity Cycle“. *IEEE Transactions on Power Electronics*. 35, No. 4. April 2020.

Classification of Pollen by the Means of Circular Transformations

Halil Akcam, Volker Lohweg

inIT-Institute Industrial IT, Technische Hochschule Ostwestfalen-Lippe,
Campusallee 6, 32657 Lemgo, Germany
E-Mail: {halil.akcam, volker.lohweg}@th-owl.de

Abstract

This present paper aims at identifying and classifying pollen grains by using binary 2D projection images of the particles. While the magnitude spectrum of the Fourier transformations and the geometric moments are established methods of classic pattern recognition of binary images, we use the *Generalised Nonlinear Circular Transform* (GNCT), which is a non-linear spectral transformation that generates a feature vector.

In order to use the GNCT approach for the classification of binary 2D pollen projections, a number of receptive fields (a separate area in the image) is placed over the image in a pre-processing step. The points of intersection between the receptive fields and the pixel points are segmented and then converted into a 1D signal. Subsequently, the generalized circular transformation is applied. Thereby, the translation-invariant vectors of the individual receptive fields, which result from the GNCT, are averaged by using group theory and interpreted as features.

It is shown that the feature generation is significant for the accuracy of the classification when using binary 2D projection images. Therefore, a comparison between the classical Fourier transformations and the GNCT approach is conducted. Advantages and disadvantages of both feature generators are discussed with a strong focus on the overall algorithm's robustness and the separability of the naturally occurring pollen particles.

1 Introduction

As one the most common chronic diseases in adults, over 20% of the European population suffer from an allergic reaction that is accompanied by symptoms such as puffy eyes, the urge to sneeze or shortness of breath [1]. One of the main causes is the hypersensitive reaction of the human body to special aerosols such as pollen [2, 3, 4]. It is therefore very advantageous for the affected people to know the air quality with regard to the concentration of pollen. Imaging measurement techniques may provide useful tools in this respect, but pollen identification and classification put high demands on classification systems. Especially the practical image processing and pattern recognition of naturally occurring objects is inevitably subject to various process and signal distortions. Applications which serve to recognise natural objects rely on variation in size, shape, orientation, and spatial position of the particles and generate features for the objects that are invariant in terms of rotation, translation, and scaling. Tools in pattern recognition as well as signal and image processing that meet these requirements are the non-linear one- and two-dimensional spectral transformations. Examples in which the spectra were used to extract features can be found in [5, 6]. In [7] a method is presented, which enables the extraction of translation-invariant features with the help of fast non-linear spectral transformations. The method is a generalised method for calculating circular transformations and is based on the concept of characteristic matrices. Thereby, the circular transformations show a superiority over the magnitude spectrum of the Fourier transformation for 1D binary signals. In [8] an accelerated procedure of this method is presented, which can operate on 2D images. Due to its very low computational complexity, the algorithm is very well suited for implementation on hardware. The presented method from [7, 8] is used in this work to obtain characteristics of naturally occurring particles, here special aerosolls (pollen). For this purpose, binary 2D projections of allergologically relevant pollen are used.

The present paper is structured as follows: In the first part of the next section, the theory of the average group is introduced. Subsequently, the basic idea of the circular transformation is explained before the approach to feature extraction by means of the circular transformation for 2D images is presented. In

the third section, the algorithms of the circular transformation and the magnitude spectrum of the Fourier transformation are applied to the data sets of the binary 2D pollen projections and the results are described. In the fourth section the results are evaluated.

2 Methods

2.1 Average Group

In this section, we explain the basic idea for the construction of invariant features F by using a transform group, which has been presented in [9, 10]. An invariant feature is a complex-valued function, which is invariant with regard to the action of the transformation group g on an image $\mathbf{Y}$, i.e. $F(g\mathbf{Y}) = F(\mathbf{Y}) \forall g \in G$. This approach is based on the averaging of a group, which can be described as

$$F(\mathbf{Y}) = \frac{1}{|G|} \int_G f(g\mathbf{Y})\, dg \quad \text{with } |G| = \int_G f(g\mathbf{Y})\, dg. \tag{1}$$

Hereby, G indicates the group and f refers to the complex-valued function. The complex-valued function f can be chosen arbitrarily and does not necessarily have to be invariant.

In [9], a group of image translations and rotations is used. Given an image $\mathbf{Y} \in \mathbb{R}^{N \times M}$, an element $g \in G$, an angle $\phi \in [0, 2\pi]$, and a translation vector $t = (i_0, j_0)^T \in \mathbb{R}^2$, then the transformation can be expressed as follows

$$(g\mathbf{Y})[i, j] = \mathbf{Y}[k, l] \tag{2}$$

with

$$\begin{pmatrix} k \\ l \end{pmatrix} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} i \\ j \end{pmatrix} - \begin{pmatrix} i_0 \\ j_0 \end{pmatrix}.$$

In this equation, i and j are the pixel coordinates. Using the transformation in (2), equation (1) can be transformed into the following equation:

$$F(\mathbf{Y}) = \frac{1}{2\pi NM} \cdot \int_{i_0=0}^{N} \int_{j_0=0}^{M} \int_{\phi=0}^{2\pi} f(g\mathbf{Y})\, d\phi\, d\, j_0\, di_0. \tag{3}$$

In (3), N and M represent the size of the image.

Due to the discrete image, the integration into (3) can be supplemented by the following summation:

$$F(\mathbf{Y}) \quad = \quad \frac{1}{2\pi NM} \quad \cdot \quad \sum_{i_0=0}^{N-1} \sum_{j_0=0}^{M-1} \sum_{h=0}^{P} f\left(g\left(i_0, j_0, \phi = 2\pi \cdot \frac{p}{P}\right)\mathbf{Y}\right). \tag{4}$$

2.2 Generalised Nonlinear Circular Transform

In [11], the *Generalised Nonlinear Circular Transform* (GNCT) was presented. GNCT is a method which is suitable for the analysis of periodic and transient signals. The method uses non-linear spectral information to generate translation-invariant features. For a signal $S \in \mathbb{R}^P$, the characteristics have the size $ld(P)+1$. This section introduces the main properties of the generalised circular transforms. If $\underline{x} = (x_0, x_1, \ldots, x_{P-1})^T$ is the input vector with the restriction $x_p \in R$ while $\mathbf{A}$ and $\mathbf{B}$ describe the quadratic transformation matrix and its inverse, respectively, then the following equation expresses the corresponding circular transformation CT:

$$\underline{X} = \mathbf{A}_P \cdot \underline{x} \quad \text{and} \quad \underline{x} = \frac{1}{P} \cdot \mathbf{B}_P^T \cdot \underline{X}.$$

Hereby, $\underline{X} = (X_0, X_1, \ldots, X_{P-1})^T$ represents the output.

On the basis of the Hadamard Matrix

$$\mathbf{K} = \begin{bmatrix} +1 & -1 \\ +1 & +1 \end{bmatrix},$$

the transformation matrices are recursively generated.

$$\mathbf{A}_P = diag\left({}^{f}\mathbf{T}_{P/2}, \mathbf{A}_{P/2}\right) \cdot \left[\mathbf{K} \otimes \mathbf{I}_{P/2}\right] \tag{5}$$

and

$$\mathbf{B}_P = diag\left({}^{r}\mathbf{T}_{P/2}, \mathbf{B}_{P/2}\right) \cdot \left[\mathbf{K} \otimes \mathbf{I}_{P/2}\right].$$

$\mathbf{I}$ is the identity matrix while ${}^{f}\mathbf{T}$ and ${}^{r}\mathbf{T}$ are the generalised characteristic matrices. The indices in (5) indicate the dimensions of the square matrices. Depending on the definition of the characteristic matrices, different transformations with different properties are mapped. A derivation of the characteristic matrices is shown in [7]. The characteristic matrices $\mathbf{T}$ are generated by the means of the generalised circular matrices $\mathbf{gC}$. For a fixed integer $k \in \{0,1,\ldots,P-1\}$, the square matrix ${}^{k}\mathbf{J}_m \coloneqq (\delta_{m,n+k})$ is mapped by the means of the Kronecker Symbol

$$\delta_{m,n} = \begin{cases} 1, & m = n \\ 0, & m \neq n \end{cases} \text{ with } m,n \in \{0,1,\ldots,P-1\}.$$

The generalised circular matrices $\mathbf{gC}$ are defined as

$$\mathbf{gC} \coloneqq \gamma \cdot \mathbf{I}_m + \lambda \cdot \left({}^{k}\mathbf{J}_m - {}^{m-k}\mathbf{J}_m\right), \quad \gamma, \lambda \in \mathbb{R}_0.$$

The definition contains

$${}^{f}\mathbf{T}_P = \prod_{i=0}^{ld(P)-1} \left({}^{2^i}\mathbf{gC}\right)^T.$$

In order to generate translation-invariant features of a signal S, the parameters $\gamma, \lambda \in \mathbb{R}_0$ are chosen in a way that the absolute values of the spectrum G also remain unchanged even if the input is translated. This, again, corresponds to

$$G_k = \sum_{j=a}^{b} |X_j|$$

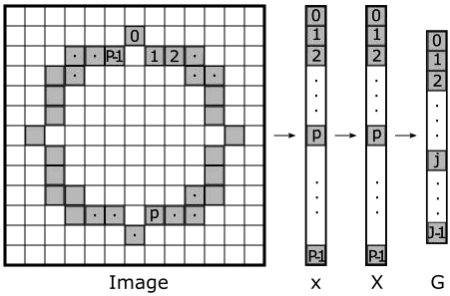


Figure 1: Visual representation of equation 6.

with

$$a = \frac{2^k - 1}{2^k} \cdot P,$$

$$b = \frac{2^{k+1} - 1}{2^{k+1}} \cdot P - 1, \quad \text{and}$$

$$k \in \{0, 1, \ldots, ld(P) - 1\}.$$

2.3 Approach to Pattern Recognition of Binary Images

The approach presented in [9] uses polynomials for the function $f(g\mathbf{Y})$. This leads to a scalar feature for (4). For the pattern recognition of different classes, a global sum would reduce the information required for the differentiation of classes. For this purpose, [8] presents an alternative in which the *GNCT* was used as the nonlinear function $f(g\mathbf{Y})$. Against this background, the following equation can be formulated:

$$\sum_{p=0}^{P} f\left(g\left(i_0, j_0, \phi = 2\pi \cdot \frac{p}{P}\right)\mathbf{Y}\right) \widehat{=} GNCT\left(g\left(i_0, j_0, \phi = 2\pi \cdot \frac{p}{P}\right)\mathbf{Y}\right). \quad (6)$$

While $r \in \mathbb{N}$ indicates the radius, $P \in \mathbb{N}$ represents the number and $p \in \{0, 1, \ldots, P-1\}$ the index of the pixel neighbours under consideration. Figure 1 shows a visual illustration of the corresponding equation.

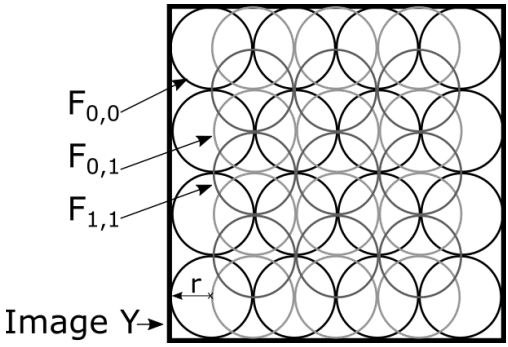


Figure 2: Three different Layers of the receptive field.

In the approach presented in [8], only the translation groups $g_r \in G$ were used. Thereby, $g_r(n,m,\phi) = g(i_0 = 2rn, j_0 = 2rm, \phi)$ while $n,m \in \mathbb{N}_0$. Our research project seeks to explore the question of whether pattern recognition of binary images is possible by applying this method. For this purpose, we expand the approach by using different translation groups g_r to create the invariant features (see Figure 2). In general, for our approach, the invariant features are summarised as follows:

$$F(\mathbf{Y},r) = \left[F_{0,0}(\mathbf{Y},r)^T, F_{0,1}(\mathbf{Y},r)^T, F_{1,1}(\mathbf{Y},r)^T\right]^T.$$

Thereby, the following applies

$$F_{l_1,l_2}(\mathbf{Y},r) = \frac{r^2}{NM} \sum_{n=0}^{\frac{N}{r}-1} \sum_{m=0}^{\frac{M}{r}-1} GNCT\left(g_r\left(n+\frac{1+l_1}{2}, m+\frac{1+l_2}{2}, \phi\right)\mathbf{Y}[r,0]\right).$$

3 Experiment and Results

In order to assess in how far the approach introduced represents naturally occurring particles (pollen), which possess a certain scatter in shape and size,

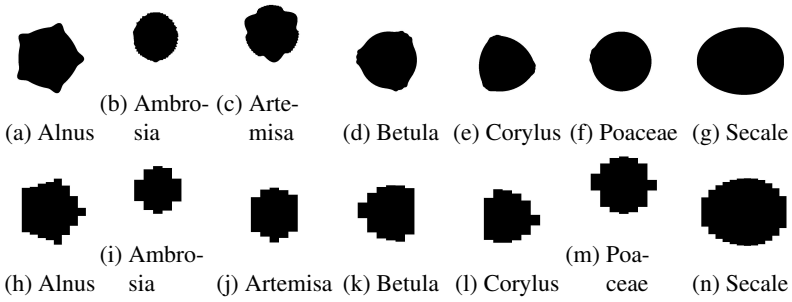
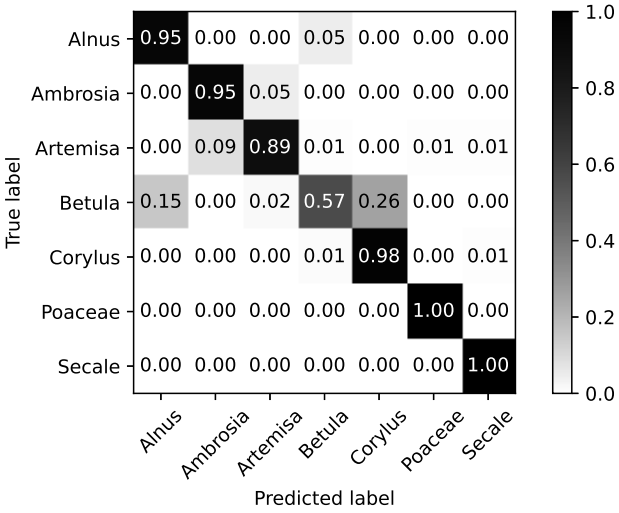


Figure 3: The allergologically relevant pollen (resolution for a-g: $1Pixel = 0.1\mu m \times 0.1\mu m$, resolution for h-n: $1Pixel = 5\mu m \times 0.1\mu m$)

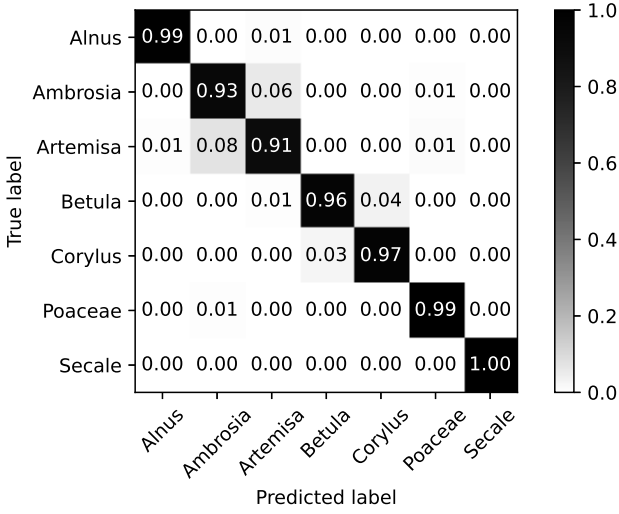
the method needs to be examined for separability and robustness. For this purpose, the binary projections of the allergologically relevant pollen Betula, Alnus, Poaceae, Corylus, Artemisia, Ambrosia, and Secale are used. Usually, the pollen are analysed and classified manually by experts [12]. Thereby, the pollen differ both in terms of both their cellular components (for the present research project, no information in this regard is available to us) and their external morphology, which is mapped with 2D binary projections. Figure 3a-3g shows a visual representation of the 2D projection of the pollen under consideration. A total of 500 projections per pollen are used for training and the subsequent classification, whereby each pollen represented has a different geometric dimension. The images have a size of 800×800 pixels. In addition, also images with lower resolutions (see Figure 3h-3n) are used in order to be able to make better judgements about the robustness of the approach suggested.

For all 1D circular transformations, the parameters $\gamma = 1$ and $\lambda = 1$ and the radii $r \in \{2,5,10\}$ for the circular circles are chosen so that the length of the input vectors $P_2 = 16$, $P_5 = 32$, and $P_{10} = 64$.

The Fourier descriptors are used as a benchmark for assessing the separability and robustness of the method. A detailed derivation of the Fournier descriptors (FD) and their invariance with respect to rotation, translation, and scaling are described in the works [13, 14, 15].

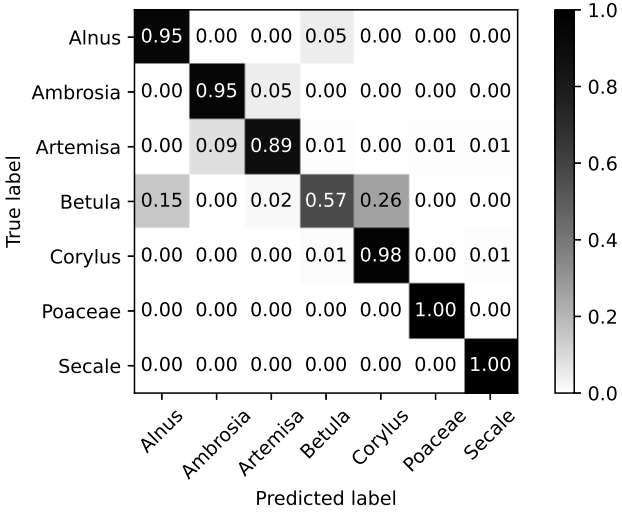


(a) Confusion matrix GNCT

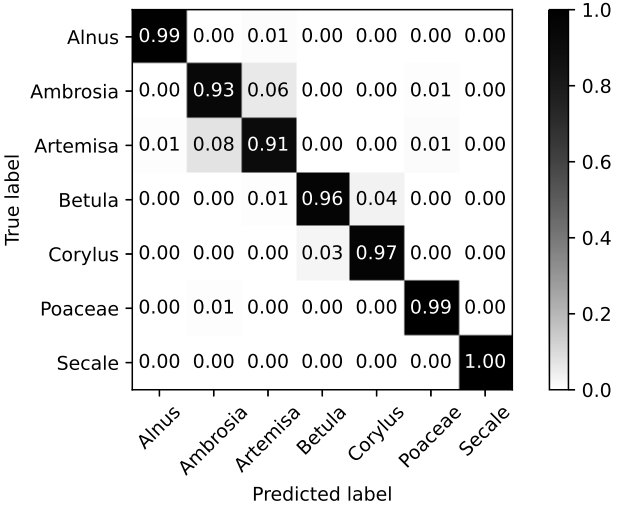


(b) Confusion matrix FD

Figure 4: Confusion matrices with dataset resolution $1Pixel = 0.1\mu m \times 0.1\mu m$



(a) Confusion matrix GNCT



(b) Confusion matrix FD

Figure 5: Confusion matrices with dataset resolution $1Pixel = 5\mu m \times 0.1\mu m$

The results of the experiment with regard to separability are illustrated in Figure 4 in the form of confusion matrices. As Figure 4b shows, for the data set with a high resolution (see Figure 3a-3g), the use of Fournier descriptors yields a success rate of 100%. That is, all particles are correctly classified. Thus, it can be deduced that the individual particles in the binary 2D images have sufficient structural characteristics so that they can be clearly separated. Compared to the Fourier descriptors, the GNCT concept presented has the advantage that no calculation of the contour is necessary in a preprocessing step so that the preprocessing is therefore easier than with the Fourier descriptors. However, the FD algorithm achieves a significantly better separation property in terms of separability (see Figure 4a). Therefore, using the GNCT approach, only 90.6% of the particles are correctly classified. The misclassifications mostly involve the Betula and Artemisa pollen. One reason for this is that the geometric size of the Betula pollen, for example, is roughly in the same range as that of the Alnus pollen. Another explanation is that the external morphology of Betula pollen is very similar to that of Corylus pollen. Since not every pixel value is used in the GNCT approach, information is lost, which, again, makes pollen without strongly distinctive features susceptible to misclassification.

The loss of information and, hence, also the misclassification rate are higher if the resolution is low. Thus, the data set with the lower resolution, which is used to check the robustness (see Figure 3h-3n), contains a higher number of misclassified pollen. That is, even when using Fourier descriptors, only 96.5% of the pollen can be assigned to the correct class (see Figure 5b). When using the GNCT method, the classification success rate of 85.5% for this data set is even lower. Also in this case, it is mostly the Betula and Artemisa pollen that are assigned to the wrong classes (see Figure 5a). As with separability, the misclassification is related to the fact that these two pollen are similar to other pollen in terms of their geometric size and/or their external morphology. In addition, the lower resolution also contributes to misclassifications due to additional information losses.

4 Conclusions

In this paper we have proposed a concept (GNCT concept) based on nonlinear circular transformations, which is to be used for pattern recognition of naturally occurring objects (pollen particles). The concept includes the application of the non-linear circular transformation to so-called receptive fields (separated areas in an image), which are then averaged with group theory. In addition, the Fourier descriptors have been used as a comparison algorithm.

In contrast to other methods, no prior knowledge or information, such as about the contour of the objects, is needed for the application of the GNCT approach. Moreover, as [8] demonstrates, this approach has a low level complexity. Therefore, the application of the GNCT approach is simpler than other methods. However, compared to the FD concept, the GNCT approach provides poorer performance in terms of separability. In terms of robustness, both algorithms show a similar behavior.

Accordingly, the application of the GNCT concept to binary images does not lead to a satisfactory result, which is due to the low information content of binary images. When using binary images, the FD approach is indeed more suitable than the GNCT approach for the classification of pollen. Yet, the GNCT approach shows promising potential. Especially for data with more information content, such as non-binary images, the approach could deliver better results in terms of separability. Therefore, the next step is to apply the GNCT concept to non-binary images.

Acknowledgment

This work was partly funded by the Federal Ministry for Economic Affairs and Energy (BMWi) within the project FeinSys, Grant-No.: ZF4654001GM8.

References

[1] Erika Mutius. *Bioaerosole und ihre Bedeutung für die Gesundheit: Rundgespräch am 27.Oktober 2009 in München*, volume 38 of *Rundgespräche der Kommission für Ökologie*. Pfeil, München, 2010.

[2] H. Behrendt, W. M. Becker, C. Fritzsche, W. Sliwa-Tomczok, J. Tomczok, K. H. Friedrichs, and J. Ring. Air pollution and allergy: experimental studies on modulation of allergen release from pollen by air pollutants. *International archives of allergy and immunology*, 113(1-3):69–74, 1997.

[3] Sabit Cakmak, Robert E. Dales, and Frances Coates. Does air pollution increase the effect of aeroallergens on hospitalization for asthma? *The Journal of allergy and clinical immunology*, 129(1):228–231, 2012.

[4] Gennaro D'Amato, Karl Christian Bergmann, Lorenzo Cecchi, Isabella Annesi-Maesano, Alessandro Sanduzzi, Gennaro Liccardi, Carolina Vitale, Anna Stanziola, and Maria D'Amato. Climate change and air pollution: Effects on pollen allergy and other allergic respiratory diseases. *Allergo journal international*, 23(1):17–23, 2014.

[5] H. Reitboeck and T. P. Brody. A transformation with invariance under cyclic permutation for applications in pattern recognition. *Information and Control*, 15(2):130–154, 1969.

[6] M. Wagh and S. Kanetkar. A class of translation invariant transforms. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(2):203–205, 1977.

[7] Volker Lohweg and Dietmar Müller. Ein generalisiertes Verfahren zur Berechnung von translationsinvarianten Zirkulartransformationen für die Anwendung in der Signal- und Bildverarbeitung. In W. Brauer, Gerald Sommer, Norbert Krüger, and Christian Perwass, editors, *Mustererkennung 2000*, Informatik aktuell, pages 213–220. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[8] T. Henke, T. Ginzel, and V. Lohweg. A simplified scheme for hardware-based pattern recognition. In *2005 International Conference on Image Processing (ICIP): September 11 - 14, 2005, Genova, Italy*, pages I–349, Piscataway, NJ, 2005. IEEE Operations Center.

[9] Hanns Schulz-Mirbach. Invariant features for gray scale images. In Gerhard Sagerer, editor, *Mustererkennung 1995: Verstehen akustischer und visueller Informationen ; 17. DAGM-Symposium, Bielefeld, 13. - 15. September 1995*, Informatik aktuell, pages 1–14. Springer, Berlin, 1995.

[10] Sven Siggelkow and Marc Schael. Fast estimation of invariant features. In W. Brauer, Wolfgang Förstner, Joachim M. Buhmann, Annett Faber, and Petko Faber, editors, *Mustererkennung 1999*, Informatik aktuell, pages 181–188. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[11] Volker Lohweg, Carsten Diederichs, and Dietmar Müller. Algorithms for hardware-based pattern recognition. *EURASIP Journal on Advances in Signal Processing*, 2004(12), 2004.

[12] Stiftung Deutscher Polleninformationsdienst. https://www.pollenstiftung.de, 02.08.2021.

[13] Wilhelm Burger and Mark James Burge. *Digitale Bildverarbeitung*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[14] Xiaoou Tang, Feng Lin, Scott Samson, and Andrew Remsen. Binary plankton image classification. *IEEE Journal of Oceanic Engineering 31*, (3):728–735, 2006.

[15] Saihua Zhu, Junyuan Zhang, Xiangze Lin, and Deying Liu. Classification of rice planthoppers based on shape descriptors. *The Journal of Engineering 2019*, (22):8378–8382, 2019.

Label Assistant: A Workflow for Assisted Data Annotation in Image Segmentation Tasks

Marcel P. Schilling[1], Luca Rettenberger[1], Friedrich Münke[1], Haijun Cui[2], Anna A. Popova[2], Pavel A. Levkin[2], Ralf Mikut[1], Markus Reischl[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: marcel.schilling@kit.edu

Abstract

Recent research in the field of computer vision strongly focuses on deep learning architectures to tackle image processing problems. Deep neural networks are often considered in complex image processing scenarios since traditional computer vision approaches are expensive to develop or reach their limits due to complex relations. However, a common criticism is the need for large annotated datasets to determine robust parameters. Annotating images by human experts is time-consuming, burdensome, and expensive. Thus, support is needed to simplify annotation, increase user efficiency, and annotation quality. In this paper, we propose a generic workflow to assist the annotation process and discuss methods on an abstract level. Thereby, we review the possibilities of focusing on promising samples, image pre-processing, pre-labeling, label inspection, or post-processing of annotations. In addition, we present an implementation of the proposal by means of a developed flexible and extendable software prototype nested in hybrid touchscreen/laptop device.

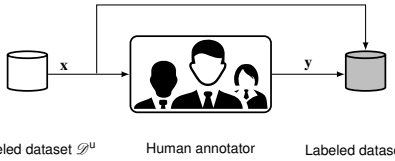


Figure 1: Naïve Workflow: A human annotator iterates over an unlabeled dataset $\mathscr{D}^{\mathrm{u}}$ to sequentially label a sample $\mathbf{x}$ in order to generate labels $\mathbf{y}$ to build a labeled dataset $\mathscr{D}^{\mathrm{l}}$ without any form of assistance.

1 Introduction

Current research in the domain of image processing is focused on Deep Learning (DL) architectures. Deep Neural Networks (DNNs) like for instance Convolutional Neural Networks (CNNs) show very promising results to solve computer vision tasks like image classification or segmentation. For example, AlexNet [1] with more than 80.000 citations (date of statistic: May, 2021) w.r.t. image classification on ImageNet [2] shows the impact of DL in the field of image processing. Walsh et al. [3] argue that DNNs are beneficial to achieve accurate prediction quality in complex scenarios like biomedical applications.

However, the authors in [3, 4] name as one general bottleneck of DL that image annotation[1] is time-consuming and often requires expert knowledge as a bottleneck. Besides, following the arguments of Northcutt et al. [5], label quality can negatively affect model performance. This may lead to a selection of sub-optimal machine learning models since benchmarks with errors in labels are not reliable in general. Karimi et al. [6] argue that especially in small data scenarios like biomedical problems, an erroneous annotation may significantly reduce the performance of DNNs.

The naïve way to generate a labeled dataset $\mathscr{D}^{\mathrm{l}} = \{(\mathbf{x}_i, \mathbf{y}_i) \mid i = 1, \ldots, M\}$ composed of M instances is represented in Figure 1. An annotator adds sequentially corresponding labels $\mathbf{y}_i$ to samples $\mathbf{x}_i$ of the unlabeled dataset $\mathscr{D}^{\mathrm{u}} = \{\mathbf{x}_i \mid i =$

[1] Label and annotation are used as a synonym in this article.

$1,\ldots,N\}$ assembled of $N \geq M$ instances without any form of assistance. The labeled dataset $\mathscr{D}^1$ incrementally increases during labeling.

There are several ideas to enhance image annotation for the development of DL applications w.r.t. decreasing annotation effort and improving annotation quality which will be presented as an overview in Section 2.

Current research predominantly focuses on separate aspects of ways to enhance a naïve generation of annotated datasets. However, to the best of our knowledge, there is no generic workflow summarizing and combing ideas of improving the image annotation procedure. We are structuring the ideas and thereby propose a comprehensive workflow. The proposal is intended to serve as a template that can be used as an initial starting point for DL projects in cases where a labeled dataset for supervised learning is required.

Our key **contributions** are the following:

- a survey of methods/approaches to assist data annotation for DL,
- a generic workflow build on meaningful combinations as well as extensions of them, and
- the introduction of a developed and extendable software prototype which can be used for assisted labeling in practical problems.

Related work is summarized in Section 2. Our workflow and methods are presented in Section 3. Besides, the software implementation is described in Section 4 following obtained results in Section 5. Finally, we conclude our work in Section 6.

2 State of the Art

The requirement of annotated data is an often addressed issue in the context of supervised DL approaches. Data efficient architectures [7, 8], self-supervised learning [9], semi-supervised learning [10], and transfer learning [11] are methods to deal with hurdle of obtaining labeled data from the perspective of network architecture/training. Considering data annotation, there are two

aspects to take into account - *labeling effort* [3, 4] and *label quality* [6, 5]. In general, decreasing manual effort for users while maintaining high label quality is desired.

There are basic **software packages** like LabelMe [12], Pixel Annotation Tool [13], Image Labeling Tool [14] or the basic release of Fiji/ImageJ [15] for annotating images in the context of segmentation like depicted as naïve workflow in Figure 1.

In the context of labeling, **Deep Active Learning** (DAL) surveyed in [16] is proposed as a method to reduce labeling effort. The key concept of the mostly considered pool-based sampling is using a more elaborate sampling strategy in contrast to do a straightforward sequential approach. Based on a criterion, also named as query strategy, the human annotator should focus on the most promising samples instead of annotating without any sampling strategy naïvely. As depicted in [16], criteria can be in terms of model uncertainty or diversity of the dataset (e.g. measured via distances in latent feature space). However, DAL research mainly focuses on a theoretical perspective. Implementations in open-source labeling tools like [14, 17, 18, 19] lack, only few commercial supplier like Labelbox [20] provide interfaces to affect sampling.

A few software tools already have implemented the idea of **pre-labeling**. The general idea of pre-labeling is using a heuristic as an initial guess to simplify labeling. For instance, the Computer Vision Annotation Tool [19] or Fiji/ImageJ plugins presented in [17, 18] implement an interface for using deep learning models in order to do image pre-labeling. However, Fiji/ImageJ is implemented in Java and consequently a deployment of models nested in state-of-the-art python-based frameworks like PyTorch [21] or TensorFlow [22] requires additional effort. Commercial tools like Labelbox [20] also offer an interface to upload pre-labels. Besides, there is a function in terms of automatically creating clusters of pixels based on regional image properties in order to simplify labeling. The tool ilastik [23] enables semi-automatic image segmentation by a combination of edge detection and watershed algorithm [24]. The authors in [25] propose a pipeline for obtaining initial labels based on traditional image processing approaches like Otsu thresholding [26] and watershed algorithm [24], but an open-source software implementation lacks. Moreover,

the tool LabelMe [12] offers functionality to use previous neighboring labels as pre-labels which may be beneficial for 3D/spatial or temporal data.

Furthermore, image **pre-processing** is another form of assistance in the context of image annotation. For instance, Fiji/ImageJ offers a raw image preprocessing with operations like adjustment of the contrast or noise filtering. The software BeadNet [27] is an example for image preparation in the sense that images are resampled in order to simplify labeling.

Karimi et al. [6] and Northcutt et al. [28] address the issue of noisy labels and survey options to handle them. For instance, the authors in [6] present methods like pruning wrong labels, adapting DNN structures, developing more elaborate objectives, or changing training procedures to cope with noisy labels. Northcutt et al. [28] propose Confident Learning, which is a method for pruning wrong labels in a labeled dataset after labeling has finished. Hereby, each sample is ranked concerning the disagreement between predictions of a trained model and corresponding noisy labels. However, the ideas are detached from the actual labeling process and focus on classification.

In particular, the idea of giving direct **feedback** concerning segmentation labels is a concept that is not considered in state-of-the-art approaches. Hence, software tools do neither support the possibility of scoring labels w.r.t. quality nor allow post-processing of them. Only some tools like Labelbox [20] enable manual tagging of images for a review process in order to allow further manual inspection by other annotators.

The toolbox LabelMe [12] allows using watershed algorithm [24] in order to do **post-processing** of coarse annotations. However, state-of-the-art tools lack w.r.t. post-processing functions allowing customization depending on the problem.

Moreover, the general approach is that labeling is performed using a mouse as **input device**. The work of [29] compares mouse devices with touch devices. The experiments of the authors show that in case of bimanual tasks, like fitting a mask on an object, touchscreens are beneficial.

The main open problems/questions of related work can be summarized in: (i) no definition of a comprehensive workflow combining different approaches of

improving image annotation, (ii) lack of smart methods concerning sample selection directly integrated into the annotation process, (iii) no possibility for direct feedback w.r.t. label quality in the annotation process, and (iv) a missing flexible software implementation to make use of combinations of label assistance.

3 Methods

3.1 Properties and challenges in datasets

In order to introduce a workflow, we give a brief overview of properties in datasets and arising challenges as one part of our contribution:

- A dataset may have temporal or spatial relations like videos or 3D images. In this case, neighboring frames are often very similar.
- Related to this, datasets composed of video sequences are often very homogeneous within a scene, but quite heterogeneous when comparing different sequences.
- Dealing with for instance microscopy images, areas of interest may be depending on relative changes in gray value/color channels. Thus, not the whole value range in high-resolution images is relevant.
- Furthermore, noise in datasets may impede image annotation.
- The level of difficulty to solve the task can range from already available heuristics to solve the problem coarsely to hard problems. Here, there are no ways to tackle the problem directly. Besides, within a dataset, there may be a variance in examples w.r.t. difficulty to interpret them.
- Depending on the problem, there is often prior knowledge before starting labeling, e.g. a specific number of segments per sample or the desired property of no holes within a segment.
- Annotations by humans are not guaranteed to be perfect. Intra-observer and inter-observer variance may lead to errors.

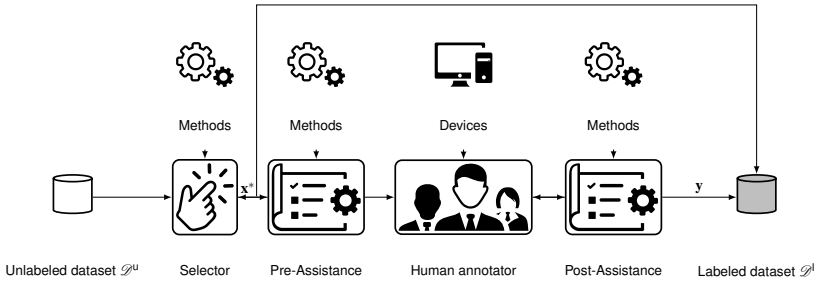


Figure 2: Assisted Labeling Workflow: A selector chooses promising samples $\mathbf{x}^*$ out of the unlabeled dataset $\mathscr{D}^{\mathrm{u}}$. The pre-assistance and post-assistance module guide the human annotator during the labeling procedure. Final labels $\mathbf{y}$ are obtained and the labeled dataset $\mathscr{D}^{\mathrm{l}}$ increases gradually.

The aforementioned properties serve as motivation for following presented approaches and methods included in the workflow proposal (Section 3.2).

3.2 Workflow

Our proposed workflow is represented in Figure 2. Firstly, starting from a unlabeled dataset $\mathscr{D}^{\mathrm{u}}$, a selector (cf. Sec. 3.3) prioritizes between all unlabeled samples and favors the next sample to label, denoted as $\mathbf{x}^*$. The subsequent pre-assistance module can yield assistance in two ways: providing pre-labels (cf. Sec. 3.4.2) as initial guesses as well as pre-processing of samples (cf. Sec. 3.4.1) to simplify annotation. Afterward, the labeling is done by the human annotator. This process can be performed using different input devices as depicted in Section 3.5. Finishing the labeling of the sample, post-assistance is a further part of the workflow. On the one hand, labels can be inspected based on defined metrics in order to provide feedback to the human annotator (cf. Sec. 3.6.1). On the other hand, based on post-processing functions, corrections of the labels are possible (cf. Sec. 3.6.2). Hence, the final label $\mathbf{y}$ is obtained and the number of labeled images in $\mathscr{D}^{\mathrm{l}}$ increases. It should be noted that Figure 2 represents the workflow in total, but in practical applications, the assistance is related to the dataset/task. Hence, in general, not all modules need to be activated.

The following sections are composed of two parts: an introduction of the *concept in general* and *presented methods*. Results of the presented methods can be found in Section 5.

3.3 Selector

General Concept The basic idea of the selector is to allow the user to affect the sampling of images during the labeling procedure and focus on promising samples $\mathbf{x}^*$ instead of labeling all images. Let an abstract query strategy, denoted as $a_j \in \mathscr{A}$, be part of the set $\mathscr{A}$ of A query strategies. Thus, a_j takes all unlabeled samples of $\mathscr{D}^{\mathrm{u}}$ into account and maps to a score $s_j(\mathbf{x}) \in [0,1]$ regarding each sample $\mathbf{x}$. An increasing $s_j(\mathbf{x})$ describes more relevance of a sample. To provide a generic sampling approach, the final score is obtained using weighted averaging

$$s(\mathbf{x}) = \frac{1}{\sum_{j=1}^{A} w_j^a} \sum_{j=1}^{A} w_j^a \, s_j(\mathbf{x}) \tag{1}$$

based on weights $w_j^a \geq 0$ in order to favor query strategies. The weights $w_j^a \geq 0$ are hyperparameters that need to be obtained depending on the underlying problem and query strategies a_j. The next promising sample is obtained by $\mathbf{x}^* = \underset{\mathbf{x} \in (\mathscr{D}^u \setminus \mathscr{D}^l)}{\operatorname{argmax}} \; s(\mathbf{x})$.

Presented Methods Examples for query strategies in the context of DAL can be found in [16], like for instance using model uncertainty or heterogeneity for sampling. Firstly, we present a novel cherry-picking function for users. The annotator could inspect the dataset and assign $s_i(\mathbf{x}) = 1$ for relevant samples $\mathbf{x}$ or $s_i(\mathbf{x}) = 0$ for images which should not be considered directly at the beginning of the labeling. This clears the hurdle of manually creating a list in parallel, to mark relevant samples.

Furthermore, we investigate the potential of an automated selector in the context of a sequential dataset. Thereby, we introduce two additional query strategies apart from the traditional ordered sequential sampling. On the one hand,

random sampling can serve as a query strategy. On the other hand, we propose a sequence-aware sampler. If the Euclidean difference in reduced gray-level feature space between two images is larger than a pre-defined threshold, a new sequence or strong change within a sequence is detected. Afterward, the sampler selects randomly a sample per cluster and only if each cluster is represented in $\mathscr{D}^{\mathrm{l}}$, a cluster is considered multiple times. It should be remarked, that for complex problems a more elaborate feature reduction method is advantageous.

3.4 Pre-Assistance

3.4.1 Image Pre-processing

General Concept The key idea of image pre-processing is not directly displaying the initial raw image during image annotation. Instead of this, a pre-processed image is generated. Abstractly speaking, the image pre-processing module is a generic function h which yields a pre-processed form of the raw sample $\mathbf{x}$ in terms of

$$\tilde{\mathbf{x}} = h(\mathbf{x}). \tag{2}$$

The objective is to accelerate annotation via displaying $\tilde{\mathbf{x}}$ where image understanding is simplified. However, it should always be considered the same pre-processing during labeling a specific dataset since varying image modalities may lead to inconsistent annotation results.

Presented Methods The desired methods are highly correlated to the depicted dataset. Therefore, we limit our presented pre-processing to two example functions h: noise filtering to deal with noisy samples and image normalization to handle high-resolution images with relative changes as depicted in Section 3.1. Custom functions can be easily implemented to find a solution that is suitable for the individual problem.

3.4.2 Pre-labeling

General Concept The main idea in the pre-labeling module is utilizing prior knowledge/heuristics, which can serve as an initial guess. Since a correction of labels is in many cases easier than starting labeling from scratch, we propose pre-labeling to boost the annotation of images. Generally speaking, an initial guess

$$\hat{\mathbf{y}} = l(\mathbf{x}) \tag{3}$$

is proposed applying a pre-label function l. However, it must be considered that pre-labeling is only meaningful if a function exists that solves the problem coarsely. In cases where l predicts mostly wrong labels, correction can slow down annotation in contrast to boost it. To evaluate quality and suitability of a pre-label function, e.g. Dice-Sørensen coefficent [30]

$$DSC\big(\mathbf{y}(\mathbf{x}), \hat{\mathbf{y}}(\mathbf{x})\big) = \frac{2 \mid \mathbf{y}(\mathbf{x}) \cap \hat{\mathbf{y}}(\mathbf{x}) \mid}{\mid \mathbf{y}(\mathbf{x}) \mid + \mid \hat{\mathbf{y}}(\mathbf{x}) \mid} \tag{4}$$

can be utilized as metric comparing initial guess $\hat{\mathbf{y}}(\mathbf{x})$ and ground truth $\mathbf{y}(\mathbf{x})$. Hence, the most suitable pre-label function l or a failure of pre-labeling in total can be determined via (4) evaluating a small set of labeled images.

Presented Methods Pre-labeling functions may be various as presented in Section 2. We present several approaches in our software prototype, which can be extended. Firstly, the traditional Otsu segmentation algorithm [26] is shown in order to assist in easier segmentation problems like enumerated in Section 3.1. Moreover, we present pre-labeling via DNNs which have already been trained on a subset of labeled samples or datasets of adjacent domains. This is beneficial in difficult image processing problems, where no suitable other heuristic exists. Besides, for sequential datasets (e.g. time-series or spatial relations) a pre-labeling is shown where previous adjacent labels are presented. Though, in this case, only a sequential image sampler is meaningful.

3.5 Human Annotator

General Concept Following the results of Forlines et al. [29], the general idea of the proposed workflow w.r.t. human annotation increases flexibility. Hence, the input device is seen as a selectable parameter of the workflow.

Presented Methods The status quo in the context of image annotation is using a mouse as an input device. We present an extension of utilizing a touchscreen for image annotation. Thereby, the touchscreen can be used with a touch pencil and fingers as well to provide a maximum level of flexibility and adaption to annotators' preferences.

3.6 Post-Assistance

3.6.1 Label inspection

General Concept As motivated in Section 2, label inspection addresses noisy labels in datasets. The general idea is to score the annotations based on G metrics $g_j \in \mathscr{G}$ which form a set $\mathscr{G}$. Each metric g_j maps labels $\mathbf{y}$ to quality scores $\gamma_j \in [0,1]$. A warning is thrown, if the final weighted score

$$\gamma(\mathbf{y}) = \frac{1}{\sum_{j=1}^{G} w_j^g} \sum_{j=1}^{G} w_j^g \, \gamma_j(\mathbf{y}) \tag{5}$$

falls below a user defined warning threshold $\gamma_0 \in [0,1]$. Analogously to equation (1), weights $w_j^g \geq 0$ allow to prioritize metrics in the final scoring. The user can reinspect the labels in case of $\gamma(\mathbf{y}) \leq \gamma_0$ and errors may be recognized immediately.

Presented Methods Metrics to inspect labels of human annotators can be various. We present in our software prototype methods which rely on expert knowledge. Thereby, we use these priors in combination with region proposals. Thus, the number of holes within a segment or number of segments serve as

a quality measure. Thereby, we compare the deviation to a target property defined by an expert (e.g. only one segment per sample). Since the metric is highly correlated to the problem, custom metrics can be implemented to extend the software functionality. Moreover, using predictions of a DNN trained on a small set of labeled data for benchmarking purpose may serve as an alternative approach, which is more generic. However, this is currently not implemented in the prototype.

3.6.2 Post-processing

General Concept Practical experiments show that some specific errors are reoccurring. In these cases, post-processing can be meaningful. In general, we propose the opportunity to have an abstract post-processing function in the labeling process in order to tackle the problem of noisy labels. Hence, annotators can use this idea in cases where post-processing of labels may be helpful. Displaying a comparison of labels before and after post-processing ensures that assistance is still supervised by human annotators avoiding unwanted changes in post-processing.

Presented Methods We recognized that especially holes or small noisy segments may come up as reoccurring errors. Thus, we implemented morphological operators as a possibility to post-process segmentation maps. Analogously, the post-processing is depending on the dataset and extensions (considering properties like aspect ratio, size, or area) are possible.

4 Implementation

The whole generic workflow depicted in Figure 2 is transferred to practical application. Therefore, a software prototype is developed following the modular architecture of the presented workflow in Section 3. The proposed concept is implemented in a python package and therefore setup respectively integration via pip is easy to manage for users. Besides, the Graphic User Interfaces (GUIs) are developed using Qt5 [31] and thus are flexible for extensions in

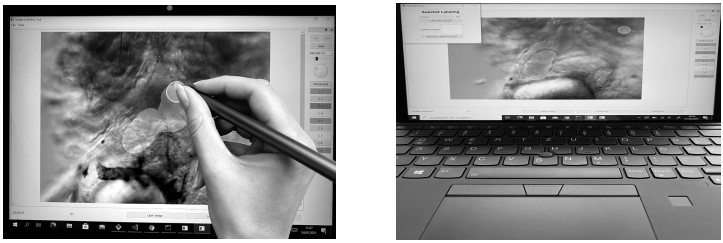
order to do further development. We refer to the Image Labeling Tool [14] for drawing image segmentation masks, since it allows a very flexible way of including pre-labeling without modifying the source code of the tool. Moreover, the publishers provide the tool across different platforms (Linux, Windows). All modules of our proposed workflow include examples concerning processing, scoring, and query functions according to Section 3. However, as mentioned, each module allows the implementation of custom functions in order to gain more flexibility. Consequently, users can customize the proposed workflow to the needs being faced with their individual problem respectively dataset. This may boost the application of the workflow prototype in the research community. Especially, the underlying implementation clears the hurdle to connect the proposed workflow with implementations based on state-of-the-art DL frameworks like TensorFlow and PyTorch [21, 22].

Our software prototype can be used in combination with Windows and Linux operation systems since the implementation is python-based and, using the Image Labeling Tool, relies on a cross-platform segmentation mask drawing tool. We tested it on Windows 10 and Ubuntu 20.04. The system can be used with desktop computers with mouse input devices and tablets as well. Our objective is to provide annotators (e.g. biologists) capsuled hardware, which allows labeling without any installation. Consequently, we deployed our software prototype on a Lenovo X12 Detachable which can be easily handed over to experts as capsuled system. This hardware allows a very flexible usage in terms of offering touch via fingers, touch via a pencil, and laptop mode via keyboard/mouse in parallel. Figure 3 shows the hardware in a practical use-case.

5 Results

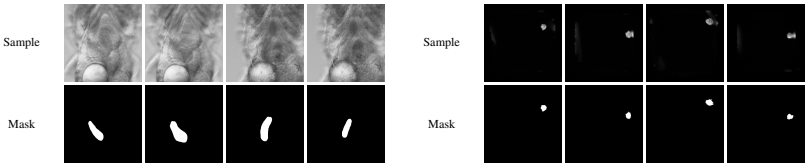
5.1 Datasets

We demonstrate an excerpt of the concept functionalities using two biomedical binary image segmentation datasets depicted in Figure 4.



(a) Tablet mode with pencil. (b) Laptop mode.

Figure 3: Software prototype on Lenovo X12 Detachable.



(a) Medaka [32]. (b) DMA Spheroid [33].

Figure 4: Datasets visualizing exemplary samples and corresponding label masks.

Medaka Dataset The medaka dataset is presented in [32]. It has been released to quantify ventricular dimensions which can be relevant for the understanding of human cardiovascular diseases. An accurate image segmentation of the medaka heart is needed in order to solve this quantification task. The dataset contains 8-bit RGB images and corresponding segmentation masks describing pixels belonging to the ventricle. It includes 565 frames of training data and 165 test samples. Figure 4a illustrates examples and binary segmentation masks. The authors in [32] use the DNN U-Net [34] to solve the image segmentation task. Looking at the example frames, it becomes clear that image segmentation is difficult in this project and thus a simple thresholding algorithm would fail. Furthermore, the dataset is based on roughly 30 video sequences and as presented in Figure 4a neighboring frames may be similar.

Droplet Microarray Spheroid Dataset The spheroid dataset is recorded in a high-throughput Droplet Microarray (DMA) experiment [33]. Currently, the dataset is not publicly available, a description of the experiment is pre-

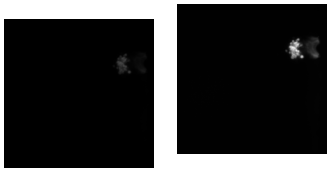
Table 1: Comparison *DSC* of different sampling scenarios (sequential/neighboring, random, sequence-aware) and dataset amounts $|\ \mathscr{D}^{l}_{\text{train}}\ |$ on medaka dataset [32].

	Configurations			
	Sequential/neighboring	Random	Sequence-aware	Baseline
$\|\ \mathscr{D}^{l}_{\text{train}}\ \|$	32	32	32	400
DSC in %	46.50	77.67	80.63	82.70

sented in the work of Popova et al. [33]. DMA experiments intend to do investigations for drug development and therefore accurate segmentation of fluorescence images is needed. It contains 16-bit high-resolution mono images with corresponding labels obtained by an expert. Thereby, it includes 470 frames of training data and 118 test samples. Being faced with this dataset, the main challenge is to distinguish between artifacts at image boundaries and spheroids. Thus, a straightforward thresholding approach like Otsu [26] is not accurate enough. Figure 4b illustrates this problem using example frames respectively segmentation masks.

5.2 Experiments

Selector To present the potential of the selector module, we first utilize the medaka dataset introduced in Section 5.1, which is a composition of different sequences. In order to evaluate the experiment, we compare *DSC* (4) using DNN U-Net [34] trained on different sampled training datasets (subsets of the initial training dataset) evaluated on a fixed test dataset. The baseline experiment uses almost the entire dataset (400 samples). Hereby, we compare the methods presented in Section 3.3. Results are shown in Table 1. A comparison of DNN performance in terms of *DSC* shows that by considering only a small subset, random sampling and sequence-aware sampling (selecting one random image of each sequence) are superior to standard labeling of neighboring frames in an ordered sequential fashion. However, in this example, the more elaborate sequence-aware approach did not outperform random sampling. If there are no strong imbalances w.r.t. the distribution of the dataset as well as no priors concerning the dataset, random sampling is definitely a proper starting point. Moreover, it can be recognized that the gap from an amount



(a) Raw. (b) Pre-processed.

Figure 5: Example pre-processing on DMA data: a raw sample (a) is processed to a normalized image (b) to enhance image understanding.

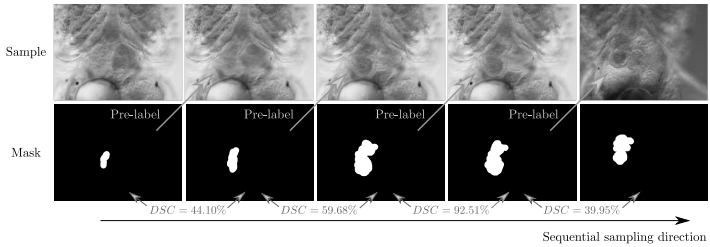


Figure 6: Comparison of sample, corresponding mask, and *DSC* between neighboring frames to illustrate temporal pre-labeling (sequential sampling form left to right) on the medaka dataset.

of $|\mathscr{D}^{l}_{\text{train}}| = 32$ training samples to the baseline with 400 samples is comparatively small. Hence, with an adapted sampling strategy a small amount is sufficient to obtain accurate results shown by a $DSC > 80\%$.

Pre-processing To get an impression of pre-processing, Figure 5 represents an example of the DMA spheroid dataset. Thereby, a raw high-resolution DMA mono image is compared to a pre-processed sample. The pre-processing function normalized the gray levels in the image. Thus, relative changes are visible, image understanding is enhanced, and therefore annotating segmentation masks is simplified.

Pre-labeling Firstly, the potential of the proposed previous label usage is analyzed at the medaka dataset since it is composed of video sequences like presented in Section 5.1. Figure 6 illustrates a sequence of the sequential

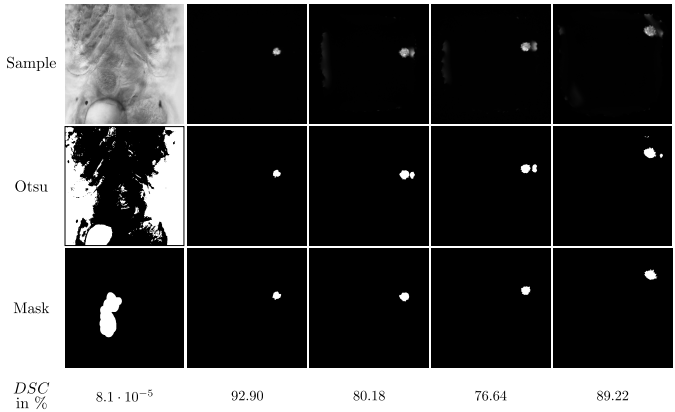


Figure 7: Illustration of visual differences between Otsu pre-labeling and ground truth mask as well as *DSC* to quantify the similarity of masks on medaka (first column) and DMA spheroid samples (remaining columns).

sampling and used pre-labels. In addition to the visual impression, *DSC* (4) is printed to compare neighboring label masks. It can be shown that the first three pre-labels are beneficial since there is a direct relation between frames. Consequently, *DSC* is larger than 40% in each of those frames. Especially, frames 3 and 4 are very similar, which can be demonstrated by a $DSC = 92.51\%$. However, the last frame illustrates a remaining problem in the method if sequences change. Hereby, the displayed pre-label is not helpful in order to do image annotation of the last sample. Figure 7 presents pre-labeling using Otsu thresholding [26]. In order to execute Otsu on RGB medaka images, an upstream transformation to a gray-level image space is done at first. However, the algorithm is not suitable as a pre-labeling strategy for medaka images, which, in addition to visual inspection, a *DSC* tending to zero demonstrates, too. Thus, in this case, pre-labeling would impede annotation instead of simplifying it. Nevertheless, Otsu performs very well on DMA samples shown by $DSC \geq 76\%$. Hence, it provides helpful initial guesses w.r.t. DMA data. Having a closer inspection and comparing it with the ground truth masks, it can be recognized, that there are still small wrong mask segments. However, deleting the wrong mask segment, in this case, is much more efficient than starting image annotation from scratch. The main reason is that curved boundaries of the spheroid are already correctly predicted for the most part.

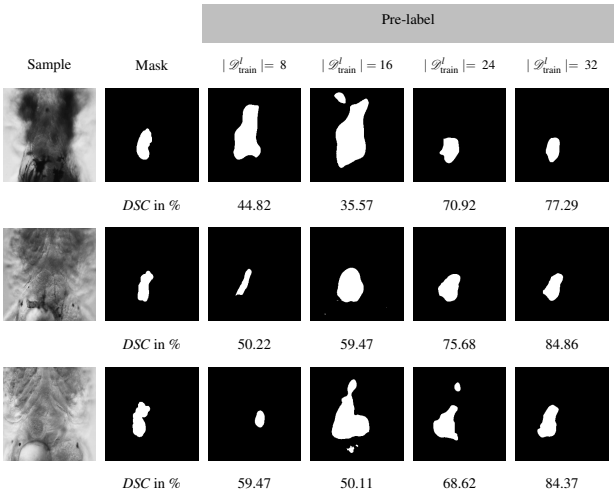
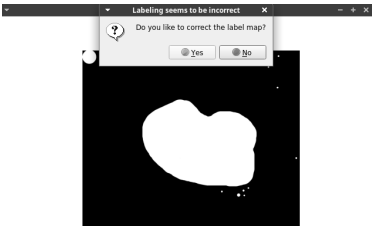


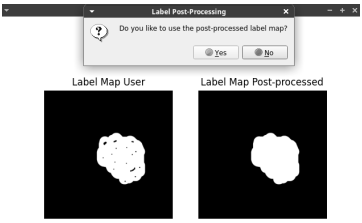
Figure 8: Illustration of DNN pre-labeling performance: comparison different amounts of training data ($|\mathscr{D}^l_{\text{train}}|$) w.r.t. visual impression and *DSC* between ground truth mask and pre-labels respectively DNN predictions.

Since there is no obvious heuristic for medaka dataset, we investigate how DNN U-Net trained on a small labeled dataset can be used as pre-labeling. Results for different amounts of training data $|\mathscr{D}^l_{\text{train}}|$ following random sampling presented in Section 3.3 can be found in Figure 8. We compare pre-labels and ground truth masks of samples $\mathbf{x} \notin \mathscr{D}^l_{\text{train}}$ not represented in the training dataset by visual impression and *DSC* (4) in parallel. Our experiments show, that by using only $|\mathscr{D}^l_{\text{train}}| = 32$ labels, a DNN can serve as a meaningful and generic pre-label strategy on medaka dataset. Furthermore, we offer in our tool the opportunity to export a training job that can directly be sent to data scientists to avoid the requirements of a graphics processing unit on the labeling device. Hence, the annotator only needs to select DNN weights provided by a data scientist. The inference time on the introduced hardware (Intel i3-1110G4) of $t_{\text{inference}} = 0.75$ s is a feasible processing amount during labeling.

Human Annotator User Experience We have presented our implemented software prototype nested in a touchscreen device to several users and have requested feedback concerning labeling comfort. The overall feedback of users



(a) Label inspection.



(b) Label post-processing.

Figure 9: Examples of post-assistance: (a) an inspector warns the user since there is more than one segment labeled, (b) a post-processing can be performed to fill holes.

has been positive. Most of the users named a comfort enhancement during image annotation using a touchscreen. However, very experienced users w.r.t. mouse labeling remark that for them touchscreen labeling is not superior to using a mouse as an input device since they are used to it. Thus, especially for an average user labeling via touchscreen may facilitate access to the procedure. Concluding results, several possibilities of user input maintain the maximum level of adaption to the needs of users.

Post-Assistance Figure 9a illustrates a label inspection evaluating deviations of connected segments to the desired segment number as a quality metric γ_i introduced in Equation (5). Large deviations lead to the presented warning prompt and give users the possibility to relabel images. Consequently, using the feedback mechanism can help to increase attention w.r.t. noisy labels directly during annotation. Post-processing links reoccurring errors with an opportunity to straightforwardly solve them. Figure 9b presents post-processing in form of closing intending to avoid holes in segment masks. Similar to label inspection, the annotator can adopt the post-processing suggestion or reject it avoiding unwanted changes. Therefore, post-processing enables a way of handling common error sources using algorithms like morphological operations or custom functions depending on the underlying problem.

The key results can be summarized the following: A selector can help to reduce the amount of labeled data needed to achieve accurate DNN results. Preprocessing and pre-labeling can facilitate annotation and decrease the effort

needed for labeling an image. Human annotators gain more flexibility by providing different types of input devices. Label inspection and post-processing build awareness of label quality and ways to deal with it.

6 Conclusion

Dealing with Deep Learning (DL), labeling plays an important role. We motivated that assisting annotators during labeling is desired (reducing labeling effort and increasing label quality). Methods to tackle these issues are various, but a summary and combination of those in a general concept is lack. We contribute a summary of properties and challenges in datasets w.r.t. annotation. Besides, we propose a generic workflow combing and extending various ideas of labeling enhancement. Especially, an evolved concept of label inspection and post-processing implemented directly within the annotation process is presented as a novel way to increase label quality. Our contribution is intended to serve as a template, which can be used by the community for practical DL projects where a labeled dataset is required. To make this concept applicable, we present a software prototype implementation as an initial starting point that can be customized. Several functionalities are demonstrated using the prototype processing two biomedical image segmentation datasets. The prototype enables further research on enhancing image annotation and investigations of new underlying methods like more generic feedback approaches or active learning in the proposed pipeline modules. For instance, the initial required amount of labeled data or further quantification of enhancement using an assisted labeling approach may be part of further research.

Acknowledgement

This work was funded in the KIT Future Fields project "Screening Platform for Personalized Oncology (SPPO)" and was performed on the computational resource bwUniCluster 2.0 funded by the Ministry of Science, Research and the Arts Baden-Württemberg and the Universities of the State of Baden-Württemberg, Germany, within the framework program bwHPC.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[2] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[3] Joseph Walsh et al. “Deep learning vs. Traditional computer vision”. In: *Advances in Computer Vision*, pp. 128–144, 2019.

[4] Weicheng Chi et al. “Deep learning-based medical image segmentation with limited labels”. In: *Physics in Medicine & Biology*, 65(23):235001, 2020.

[5] Curtis G. Northcutt, Anish Athalye, and Jonas Mueller. “Pervasive label errors in test sets destabilize machine learning benchmarks”. arXiv:2103.14749 [stat.ML], 2021.

[6] Davood Karimi et al. “Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis”. In: *Medical Image Analysis*, 65(5):101759, 2020.

[7] Tim Scherr et al. “Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy”. In: *PLOS ONE*, 15(12):1–22, 2020.

[8] Fabian Isensee et al. “ nnU-Net: A self-configuring method for deep learning-based biomedical image segmentation”. In: *Nature Methods*, 18(2):203–211, 2021.

[9] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *Proceedings of the 37th International Conference on Machine Learning*, pp. 1597–1607, 2020.

[10] Xiaokang Chen et al. “Semi-supervised semantic segmentation with cross pseudo supervision”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2613–2622, 2021.

[11] Chuanqi Tan et al. “A survey on deep transfer learning”. arXiv:1808.01974 [cs.LG], 2018.

[12] Kentaro Wada. “Labelme: Image polygonal annotation with python”. 2016, Accessed: 2021-05-28. Available: `https://github.com/wkentaro/labelme`.

[13] Amaury Bréhéret. “Pixel Annotation Tool”. 2017, Accessed: 2021-05-27, Available: `https://github.com/abreheret/PixelAnnotationTool`.

[14] Andreas Bartschat. “ImageLabelingTool”. 2019, Accessed: 2021-05-31, Available: `https://bitbucket.org/abartschat/imagelabelingtool`.

[15] Johannes Schindelin et al. “Fiji: an open-source platform for biological-image analysis”. In: *Nature Methods*, 9(7):676–682, 2012.

[16] Pengzhen Ren et al. “A survey of deep active learning”. arXiv:2009.00236 [cs.LG], 2020.

[17] Thorsten Falk et al. “U-net – deep learning for cell counting, detection, and morphometry”. In: *Nature Methods*, 16(1):67–70, 2019.

[18] Réka Hollandi et al. “AnnotatorJ: An ImageJ plugin to ease hand annotation of cellular compartments”. In: *Molecular Biology of the Cell*, 31(20):2179–2186, 2020.

[19] Boris Sekachev et al. “Computer Vision Annotation Tool (CVAT)”. 2020. Accessed: 2021-05-31, Available: `https://github.com/openvinotoolkit/cvat`.

[20] Manu Sharma, Daniel Rasmuson, and Brian Rieger. “Labelbox”. 2021. Accessed: 2021-05-28, Available: `https://labelbox.com`.

[21] Adam Paszke et al. “PyTorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

[22] Martín Abadi et al. “TensorFlow: Large-scale machine learning on heterogeneous distributed systems”. arXiv:1603.04467 [cs.DC], 2016.

[23] Stuart Berg et al. “Ilastik: Interactive machine learning for (bio)image analysis”. In: *Nature Methods*, 16(12):1226–1232, 2019.

[24] Serge Beucher and Christian Lantuéjoul. “Use of watersheds in contour detection”. In: *International Workshop on Image Processing: Real-Time Edge and Motion Detection/Estimation*, pp. 17–21, 1979.

[25] Fabian Englbrecht, Iris E. Ruider, and Andreas R. Bausch. “Automatic image annotation for fluorescent cell nuclei segmentation”. In: *PLOS ONE*, 16(4):1–13, 2021.

[26] Nobuyuki Otsu. “A threshold selection method from gray-level histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[27] Tim Scherr et al. “BeadNet: Deep learning-based bead detection and counting in low-resolution microscopy images”. In: *Bioinformatics*, 36(17):4668–4670, 2020.

[28] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. “Confident learning: Estimating uncertainty in dataset labels”. In: *Journal of Artificial Intelligence Research*, pp. 1373–1411, 2021.

[29] Clifton Forlines et al. “Direct-touch vs. mouse input for tabletop displays”. In: *Conference on Human Factors in Computing Systems*, pp. 647–656, 2007.

[30] Shruti Jadon. A survey of loss functions for semantic segmentation. In: *IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pp. 1–7, 2020.

[31] Tor Arne Vestbø and Alessandro Portale. “Qt”. 2021, Accessed: 2021-05-31, Available: `https://github.com/qt`.

[32] Mark Schutera et al. “Machine learning methods for automated quantification of ventricular dimensions”. In: *Zebrafish*, 16(6):542–545, 2019.

[33] Anna A. Popova et al. “Facile one step formation and screening of tumor spheroids using droplet-microarray platform”. In: *Small*, 15(25):1901299, 2019.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, 2015.

Physik-geführte NARXnets (PGNARXnets) zur Zeitreihenvorhersage

Silas Aaron Selzer[1], Fabian Bauer[2], Sebastian Bohm[1], Peter Bretschneider[2], Erich Runge[1]

[1] Fachgebiet Theoretische Physik I
Technische Universität Ilmenau
Weimarer Straße 25
98693 Ilmenau
E-Mail: {silas-aaron.selzer, sebastian.bohm, erich.runge}@tu-ilmenau.de

[2] Fachgebiet Energieeinsatzoptimierung
Technische Universität Ilmenau
Gustav-Kirchhoff-Straße 5
98693 Ilmenau
E-Mail: {fabian.bauer, peter.bretschneider}@tu-ilmenau.de

Kurzfassung

Durch stark steigende Datenmengen und der Optimierung der Rechentechnik konnten in den letzten Jahren massive Fortschritte in maschinellen Lernverfahren erzielt werden. In vielen Fällen ist die Datenerfassung jedoch mit einem großen Zeit- und Kostenaufwand verbunden. Daher wäre es von Vorteil, bereits aus Teilinformationen valide Schlussfolgerungen ziehen zu können. Für Anwendungen auf naturwissenschaftlich-technische Problemstellungen existiert typischerweise sehr klar definiertes Vorwissen in Form von physikalischen Gleichungen oder empirischen Beziehungen. Eine Synthese von theoretischem Vorwissen verbunden mit datenwissenschaftlichen Methoden erscheint für solche Situationen sehr vielversprechend. In diesem Beitrag wird am Beispiel der Zeitreihenvorhersage an einem chaotischen System gezeigt, dass durch die Berücksichtigung des zugrundeliegenden Differentialgleichungsystems in

der Verlustfunktion eines künstlichen neuronalen so genannten NARX-Netzes die Extrapolationsfähigkeit des Modells verbessert werden kann. Dazu wirken die Physik-geführten Verlustterme in Kombination mit klassischen Verlusten als eine Art Regularisierung. Verwendet man nur Physik-geführte Verluste ist es darüber hinaus möglich, neuronale Netze *label-free* zu trainieren. Dadurch entfällt die kosten- und zeitintensive Beschaffung großer Trainingsdatenmengen.

1 Einführung

Die Zeitreihenprognose ist ein wichtiger Bereich des maschinellen Lernens. Zeitreihenvorhersage bedeutet, dass zukünftige Werte einer Funktion auf der Basis mehrerer historischer Werte ermittelt werden. Neben den zurückliegenden Funktionswerten können weitere exogene Einflussgrößen gegeben sein, welche auf die Zeitreihenprognose einwirken. Anwendung findet die Zeitreihenprognose in vielen Bereichen, vom Energiesektor (z.B. Vorhersage des Energieverbrauchs von Mehrfamilienhäusern [1]) über die Hydrologie (z.B. Vorhersage des Zu- und Abflusses von Stauseen [2]) bis hin zur Finanzwirtschaft (z.B. Vorhersage von Devisenkursen, Aktienindexen oder dem Wirtschaftswachstum [3, 4]). Durch die Analyse des Verhaltens in der Vergangenheit und der Annahme, dass das zukünftige Verhalten deterministisch ist, sollen zukünftige Werte vorausgesagt werden.

Für die Vorhersage von Zeitreihen mittels neuronaler Netze werden beispielsweise sogenannte NARXnets (Nonlinear autoregressive with external input networks) verwendet [5, 6]. Konkret versteht man unter NARXnets rekurrente dynamische neuronale Netze, welche die Vorhersage der gesuchten Größe zum nächsten Zeitschritt $\hat{J}(t_n)$ aus den vorangegangenen k Werten der exogenen Eingangsvariablen $(I(t_{n-1}),...,I(t_{n-k}))$ und zurückliegenden l Werten des vorherzusagenden Ausgangssignals $(J(t_{n-1}),...,J(t_{n-l}))$ berechnen. Die Vorhersage des NARXnet wird nach der Berechnungsvorschrift

$$\hat{J}(t_n) = \text{NARXnet}\,(J(t_{n-1}),...,J(t_{n-k}),I(t_{n-1}),...,I(t_{n-l})) \tag{1}$$

bestimmt. Bei der Anwendung der NARXnets auf naturwissenschaftlich-technische Probleme, wie das Lösen von Differentialgleichungen, erzielen diese vergleichsweise gute Ergebnisse. In Hinblick auf die aufkommende Thematik des Theorie-gesteuerten Lernens [7] stellt sich die Frage, ob die Ergebnisse der NARXnets durch Erweiterung der Verlustfunktion um einen Physik-geführten Verlustterm (Physik-geführte NARXnets) verbessert werden können.

Ziel des Theorie-gesteuerten maschinellen Lernens (auch als Physik-informier- tes, Physik-geführtes oder Physik-inspiriertes Lernen bezeichnet) ist die Schaffung einer Synthese aus den datenwissenschaftlichen und Physik-basierten Modellen. Dadurch wird es möglich, sowohl theoretisches Vorwissen über das System in Form von physikalischen Gleichungen oder empirischen Abhängigkeiten als auch Informationen aus Daten zu nutzen (siehe Abbildung 1) [8, 9]. Durch den massiven Anstieg verfügbarer Daten und Rechenressourcen konnten in den letzten Jahren große Fortschritte bei maschinellen Lernverfahren erzielt werden. Dies spiegelt sich beispielsweise in der Bild- [10] oder auch Spracherkennung [11] wider. Als problematisch bei naturwissenschaftlich-technischen Problemen stellt sich jedoch zumeist eine zeit- und kostenaufwändige Datenerfassung dar. Zudem besteht oft ein Ungleichgewicht zwischen beispielsweise sehr vielen Betriebsdaten, die Systemzustände nahe dem Normalzustand beschreiben, während Vorhersagen besonders wichtig sind für seltene außergewöhnliche Situationen mit starken Abweichungen, in denen letztlich extrapoliert werden muss. Ziel wäre es, bereits aus einer geringen Menge an Teilinformationen valide Schlussfolgerungen zu ziehen. Die maschinellen Lernverfahren zeigen bei geringen Datenmengen oftmals schlechte oder keine Konvergenz. Um diesem Problem entgegenzuwirken, kann theoretisches Vorwissen, welches bisher in ML-Verfahren nicht genutzt wird, in Form von physikalischen Gesetzen oder empirischen Regeln in den Lernprozess eingepflegt werden. Dadurch wird der Lösungsraum beschränkt und man kann bereits mit geringen Datenmengen gute Ergebnisse erzielen. [12]

Für das Theorie-gesteuerte maschinelle Lernen gibt es vielfältige Ansätze: Der von Raissi et al. [12] vorgeschlagene Ansatz zur datengesteuerten Lösung und datengestützten Bestimmung der Koeffizienten in partiellen

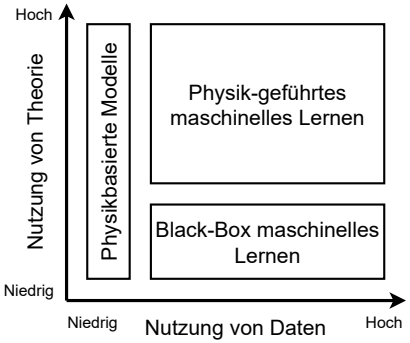


Bild 1: Einordnung des Physik-geführten maschinellen Lernens im Vergleich zu rein datenwissenschaftlichen Black-Box und theoretischen Physik-basierten Ansätzen. [8, 9]

Differentialgleichungen basiert auf der Verwendung der umgestellten Differentialgleichung als Verlustfunktion. Indem die automatische Differenzierung nicht nur für das Backpropagationverfahren genutzt wird, sondern auf die Bestimmung der Ableitungen in der Differentialgleichung in Hinblick auf die Eingangsneuronen ausgedehnt wird, lassen sich die Ableitungen berechnen. Dieser Ansatz wurde auf weitere Probleme aus dem Bereich der Geologie [13], der Nanooptik [14] oder dem Stahlbau [8] übertragen. Weiterhin wurden erste Bibliotheken [15] für diesen Ansatz erstellt.

Ein alternativer Ansatz ist es, die klassischen Verlustfunktionen durch Physik-geführte Verlustterme zu ergänzen. Dadurch verspricht man sich, die Vorteile der Datenwissenschaften mit Vorwissen des Systemverhaltens zu verbinden und somit die Vorteile beider Modellierungsformen zu kombinieren. Beispielsweise kann physikalisches Vorwissen in Form von Differentialgleichungen, Bilanzgleichungen, Monotoniebedingungen oder auch Nichtnegativ-Werten in einen zusätzlichen Verlustterm einfließen. Ein solcher Ansatz wurde von Karpatne et al. [9] zur Vorhersage der Temperatur in Seen genutzt. Auch die Lösung von Eigenwertgleichungen [16] oder die Vorhersage des Schwing-Verhaltens von Gebäuden während bzw. nach Erdbeben [17] wurden mit ähnlichen Ansätzen untersucht. Eine Zeitreihenvorhersage mittels Physik-geführter NARXnets wurde anhand der Dynamik von thermochemischen Energiespeichern betrachtet [18].

In den bisherigen Untersuchungen zur Kombination von theoretischem Vorwissen und den NARXnets zur Zeitreihenvorhersage wurden physikalisches Vorwissen in Form von Bilanzgleichungen, Monotoniebedingungen sowie Nichtnegativ-Werten in die Verlustfunktion mit eingebaut [18] oder aber die Physik in Form einer Strukturänderung aber Beibehaltung der klassischen Verlustfunktion [19] dem neuronalen Netz aufgezwungen. Der hier vorgeschlagene Ansatz zielt darauf ab, die physikalische Konsistenz der Modelle durch Verlustfunktionen zu sichern, welche auf den zugrundeliegenden Differentialoperator des Problems zugeschnitten sind. Es zeigt sich jedoch, dass der Ansatz, die automatischen Differenzierung zur Berechnung der Ableitungen in den Verlustfunktionen (ähnlich der Physik-informierten neuronalen Netze [12]) zu nutzen, deutlich schlechtere Ergebnisse im Vergleich zu klassischen NARXnets aufweist. Grund dafür ist vermutlich, dass die Zeit als zusätzlicher Parameter in den Eingangsneuronen eingespeist werden muss. Das Netzwerk wird dadurch so trainiert, dass die Zeit als Hauptinformationsquelle dient und die zurückliegenden Funktionswerte ignoriert werden. Das NARXnet ist jedoch gerade darauf ausgelegt, Informationen aus den zurückliegenden Funktionswerten zu nutzen, um damit den nächsten Funktionswert vorherzusagen. Durch die Nutzung der zurückliegenden Funktionswerte, welche in die Eingangsneuronen eingespeist werden, lassen sich die Ableitungen in der Verlustfunktion mit linksseitigen Differenzenquotienten abschätzen und somit die Differentialgleichung als Verlustfunktion wählen. Untersucht wurden einmal die Kombination aus klassischen und Physik-geführten Verlusttermen wie auch die alleinige Nutzung der Physik-geführten Verlustterme.

Die Ausarbeitung ist folgendermaßen gegliedert. In Abschnitt 2 wird der Lösungsansatz der Physik-geführten NARXnets allgemein eingeführt. Nachfolgend wird dieser auf das Beispiel einer chaotische Zeitreihe (die modifizierte Van der Pol Gleichung) übertragen. Die Problemstellung wird in Abschnitt 3 erläutert. Die Ergebnisse sind in Abschnitt 4 dargestellt. Abschnitt 5 fasst die Untersuchungen zusammen und gibt einen Ausblick auf weitergehende Fragestellungen.

2 Physik-geführte NARXnets

In den bisherigen Standard-Ansätzen der NARXnets wird als Verlustfunktion

$$\min_{\Theta} Loss_{\text{Std.}} = \min_{\Theta} Loss_{\text{emp.}}(J,\hat{J}) + \lambda_{\text{reg.}} Loss_{\text{reg.}}(\Theta) \tag{2}$$

verwendet, die darauf abzielt, die empirischen Verluste $Loss_{\text{emp.}}(J,\hat{J})$ der Modellvorhersage $\hat{J}$ durch die Veränderung der Modellparameter Θ zu minimieren. Teilweise wird der besagte Ansatz um einen weiteren Term $Loss_{\text{reg.}}(\Theta)$ ergänzt, welcher die Komplexität des Modells bestraft. Ziel ist es, dadurch die Überanpassung an die vorliegenden Daten zu verhindern. Durch den Parameter $\lambda_{\text{reg.}}$ wird das relative Gewicht dieses Terms gegenüber dem empirischen Fehler eingestellt. Wie bereits in der Einführung angeschnitten, hängt die Effektivität dieses Ansatzes maßgeblich von der Größe der Trainingsdatenmenge ab. Weiterhin gibt es keine Garantie, dass solch ein Ansatz die zugrundeliegende Physik korrekt wiedergibt bzw. respektiert. [9]

Der Ansatz der Physik-geführten NARXnets zielt auf Zeitreihenvorhersagen ab, bei denen die zurückliegenden Funktionswerte wie auch exogene Eingangsvariablen in das Netz eingespeist werden sowie das theoretische Vorwissen über den Prozess in Form von Differentialgleichungen oder Differentialgleichungssystemen vorliegt. Das betrachtete physikalische System wird durch die vorherzusagende Variable J, die exogenen Eingangsvariablen I wie auch weitere physikalische Variablen $\boldsymbol{Z}$ mittels einer hier generisch geschriebenen Differentialgleichung

$$\mathscr{F}(I,J,\boldsymbol{Z}) = 0 \tag{3}$$

beschrieben. [9]
Durch die Physik-geführte Verlustfunktion soll nun gemessen werden, in wie weit die Vorhersage von $\hat{J}$ die Differentialgleichung erfüllt. Dazu wird gerade die Nichterfüllung besagter Differentialgleichung

$$Loss_{\text{PG}} = \lambda_{\text{PG}} Loss(\mathscr{F}(I,\hat{J},\boldsymbol{Z})) \tag{4}$$

bestraft, wobei λ_{PG} ein Hyperparameter des Physik-geführten Verlusts darstellt [9]. Als Verlustmaße können etablierte Maße, wie z.B. der mittlere absolute (MAE) oder auch der mittlere quadratische Fehler (MSE) verwendet werden.
Die weiteren Betrachtungen konzentrieren sich auf Differentialgleichungen mit zeitlichen Ableitungen. Da die Zeit nicht als Eingabe in den NARXnets dient, wird in diesem Beitrag vorgeschlagen, die Ableitungen in der Verlustfunktion durch linksseitige Differenzenquotienten abzuschätzen. Für alle exogenen Variablen wie auch die Zielvariable sind die zurückliegenden Funktionswerte bekannt, da diese als Eingangsgrößen in das neuronale Netz dienen. Darüber hinaus müssen die Zeitschrittweite Δt und die Konstanten der Differentialgleichungen gegeben sein. Eine besonders robuste Formel zur Abschätzung einer Ableitung nach [20] ist gegeben durch

$$\left.\frac{\mathrm{d}^m J}{\mathrm{d}t^m}\right|_{t_i=t_0} \approx \sum_{n=0}^{N-1} a_n^{(m,N)} J(t_i+s_n) \quad \text{mit } N>m, \tag{5}$$

wobei m der Ordnung der Ableitung, N der Anzahl der Stützstellen und s_n den Stützstellen relativ zu t_i entspricht. Durch die Lösung des linearen Gleichungssystems

$$\begin{pmatrix} 1 & \cdots & 1 \\ s_1 & \cdots & s_N \\ \vdots & \ddots & \vdots \\ {s_1}^{N-1} & \cdots & {s_N}^{N-1} \end{pmatrix} \begin{pmatrix} a_0^{(m,N)} \\ a_1^{(m,N)} \\ \vdots \\ a_{N-1}^{(m,N)} \end{pmatrix} = m! \begin{pmatrix} \delta_{0,m} \\ \delta_{1,m} \\ \vdots \\ \delta_{N-1,m} \end{pmatrix} \tag{6}$$

können für jedes m die Koeffizienten $a_n^{(m,N)}$ bestimmt werden, wobei $\delta_{i,j}$ das Kronecker-Delta ist [21].

Mit diesem Physik-geführten Ansatz lassen sich zwei Typen von Physikgeführ- ten NARXnets ableiten. Zum einen können die Physik-geführten NARXnets allein durch den Physik-geführten Verlust (Gleichung 4) trainiert werden. Ein solches Netz lässt sich ohne die Vorgabe der korrekten Lösung trainieren. Im Gegensatz zu traditionellen Ansätzen wird das Netz somit ausschließlich an nicht gelabelten Datensätzen ausgewertet (engl. *label-free*

learning). Weiterhin kann der Standard-Ansatz (Gleichung 2) um den Physik-geführten Verlustterm (Gleichung 4) ergänzt werden, um den klassischen datenwissenschaftlichen Ansatz mit Vorwissen zu kombinieren.

3 Problemstellung

In diesem Abschnitt wird die Problemstellung eingeführt, an welcher die vorgestellten Ansätze getestet und mit dem Standardansatz verglichen werden. Als Anwendungsbeispiel wird die modifizierte Van der Pol Gleichung verwendet und das Rahmenwerk der Physik-geführten NARXnets auf diese Problemstellung übertragen. Gerade in der modifizierten Form weist die Van der Pol Gleichung ein chaotisches Verhalten auf, sodass die zeitliche Entwicklung zunächst nicht prognostizierbar wirkt, obwohl die mathematische Beschreibung des Systems deterministisch ist.

3.1 Van der Pol Gleichung

Zur Beschreibung des allgemeinen Phänomens von Relaxationsschwingungen wurde 1926 die Van der Pol Gleichung von Balthasar van der Pol eingeführt [22]. Es handelt sich dabei um eine nichtlineare gewöhnliche Differentialgleichung zweiter Ordnung, welche die Form

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \varepsilon\left(1-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t} + x = 0 \tag{7}$$

mit einem Parameter $\varepsilon > 0$ annimmt. Fügt man einen anregenden Term hinzu, ergibt sich

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \varepsilon\left(1-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t} + x = A\cos\left(2\pi f t\right). \tag{8}$$

Für beliebige Werte der Konstanten A und ε erhält man einen Grenzzyklus oder quasiperiodische Lösungen, aber kein chaotisches Verhalten. Dieses tritt erst durch die Modifizierung der Van der Pol Gleichung auf. [23]

3.2 Modifizierte Van der Pol Gleichung

Ueda et al. [24] haben für die Beschreibung des angeregten Oszillators mit negativen Widerstand eine modifizierte Van der Pol Gleichung

$$\frac{\mathrm{d}^2x}{\mathrm{d}t^2} - \varepsilon\left(1-x^2\right)\frac{\mathrm{d}x}{\mathrm{d}t} + x^3 = A\cos\left(2\pi ft\right) \tag{9}$$

formuliert, welche ein chaotisches Verhalten für bestimmte Werte der Koeffizienten A und ε aufweist. Für die weitere Diskussion derselben wird die nichtlineare gewöhnliche Differentialgleichung zweiter Ordnung in ein System aus zwei nichtlinearen gewöhnlichen Differentialgleichungen erster Ordnung

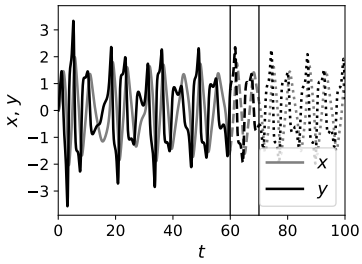
$$\frac{\mathrm{d}x}{\mathrm{d}t} = y \tag{10}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = \varepsilon\left(1-x^2\right)y - x^3 + A\cos\left(2\pi ft\right) \tag{11}$$

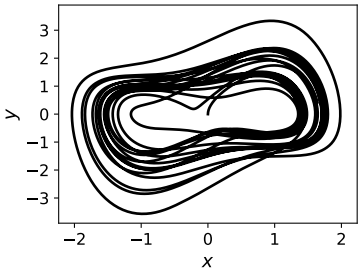
umgewandelt. Je nach Wahl der Koeffizienten A und ε tritt chaotisches oder aber ein reguläres Verhalten auf. [23]
In Abbildung 2 ist die numerisch berechnete Lösung (mittels explizitem Runge-Kutta-Verfahren der fünften Ordnung [25]) gerade für die Wahl der Koeffi- zienten dargestellt, sodass sich ein chaotisches Verhalten ergibt.

Anhand dieser Zeitreihen soll die Performance der Physik-geführten im Vergleich zu den klassischen NARXnets untersucht werden. Als Datengrundlage dient die Zeitreihe mit den Koeffizienten $A = 1$ und $\varepsilon = 1$. Diese wurde in einen Zeitbereich von 0 bis 100 aufgenommen. In Abbildung 2a ist die Aufteilung der Zeitreihe in Trainigsdaten (6000 Werte), Validierungsdaten (1000 Werte) und Testdaten (3000 Werte) gezeigt.
Um neben der numerischen Lösung der Zeitreihe auch das Training mit *realen* Daten zu testen, wird die numerische Lösung verrauscht. Dazu wird auf die numerische Lösung weißes Rauschen nach der Formel

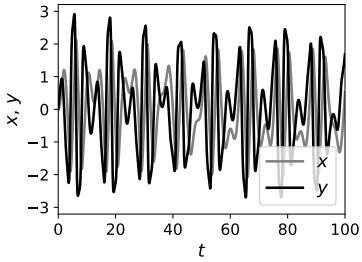
$$x_{\mathrm{real}} = x + \nu \cdot N\left(\mu = 0, \sigma_x\right) \tag{12}$$



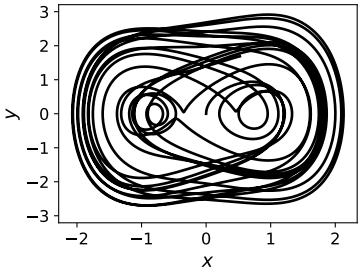
(a) Zeitliche Entwicklung: $A = 1$ und $\varepsilon = 1$



(b) Phasendiagramm: $A = 1$ und $\varepsilon = 1$



(c) Zeitliche Entwicklung: $A = 1$ und $\varepsilon = 0.1$



(d) Phasendiagramm: $A = 1$ und $\varepsilon = 0.1$

Bild 2: Modifizierte Van der Pol Gleichung. Für die Initialbedingung gilt: $x(0) = 0$ und $y(0) = 0$. Die Frequenz wurde mit $f = \frac{1}{2\pi}$ gewählt.

gelegt, wobei ν die Rauschstärke und $N(\mu = 0, \sigma_x)$ die Normalverteilung mit dem Erwartungswert $\mu = 0$ und der Standardabweichung der numerisch berechneten Werte σ_x darstellt.

Die Zeitreihe mit den Koeffizienten $A = 1$ und $\varepsilon = 0.1$ dient als weitere Testdatenmenge um die Generalisierungsfähigkeit der Physik-geführten und klassischen NARXnets zu überprüfen. Für beide Zeitreihen wurden äquidistante Zeitschritte mit identischer Schrittweite gewählt.

3.3 Zeitreihenvorhersage mit neuronalen Netzen

In dem besagten Beispiel ist es das Ziel, mit den unterschiedlichen NARXnet-Ansätzen, die Lösung der modifizierten Van der Pol Gleichung x_n vorherzusagen. Die klassische NARXnet-Struktur

$$\hat{x_n} = \text{NARXnet}\left(x_{n-1}, \ldots, x_{n-k}, y_{n-1}, \ldots, y_{n-l}\right), \tag{13}$$

wird für das Beispiel dahingehend verändert, dass neben x_n auch y_n prognostiziert wird. Dadurch ist es möglich, das Differentialgleichungssystem (Gleichung 10 und 11) in die Verlustfunktion einzupflegen und diese somit auf den zugrundeliegenden Differentialoperator zuzuschneiden (nachfolgend in Unterunterabschnitt 3.3.2 erläutert). Um die Vorhersagen $\hat{x_n}$ und $\hat{y_n}$ zum Zeitpunkt t_n zu bestimmen, werden den neuronalen Netzen die k-Zeitschritte zurückliegenden Funktionswerte von x und y sprich $x_{n-1}, \ldots, x_{n-k}$ und $y_{n-1}, \ldots, y_{n-k}$ als Eingabe vorgegeben.

Für die drei verschiedenen Lösungsansätze (PGNARXnet, PGNARXnetlf und NARXnet; dargestellt in Abbildung 3) werden jeweils identische Architekturen (auch mit identischer Initialisierung der Gewichte und Schwellenwerte), Optimierungseinstellungen und Trainingsdaten verwendet. Das heißt, die drei Netze unterscheiden sich einzig und allein durch ihre Verlustfunktion. Auf die Verlustfunktionen wird nachfolgend eingegangen.

3.3.1 NARXnet

Die zu trainierenden Parameter des NARXnet (Gewichte und Schwellenwerte) werden durch die Minimierung der Kombination des mittleren quadratischen Fehlers der klassischen Verlustfunktionen der beiden Größen x und y

$$Loss_{\text{NARXnet}} = MSE_{\text{klass},1} + MSE_{\text{klass},2} \tag{14}$$

berechnet. Unter der klassischen Verlustfunktion werden die empirischen Verluste, also die Abweichung der Vorhersage des neuronalen Netzes von der vorgegebenen Lösung, verstanden. Da gerade zwei Variablen als Ausgangsgröße

dienen, werden für beide Vorhersagen ($\hat{x}_n$ und $\hat{y}_n$) die klassischen Verlustterme

$$MSE_{\text{klass},1} = \frac{1}{M} \sum_{n=k}^{M+k} (x_n - \hat{x}_n)^2 \tag{15}$$

und

$$MSE_{\text{klass},2} = \frac{1}{M} \sum_{n=k}^{M+k} (y_n - \hat{y}_n)^2 \tag{16}$$

ausgewertet, wobei die Größen, gekennzeichnet mit Dach, die Vorhersage des neuronalen Netzes markieren. M gibt die Anzahl der Vorhersagen an, die im Fall des Trainings der Batchgröße $M = M_{\text{Batch}}$ entspricht. Da die Vorhersage ab dem k-ten Zeitschritt des Datensatzes erfolgt, wird die Summation ab diesem Zeitwert gestartet. Verwendet wird der mittlere quadratische Fehler, um größere Abweichungen stärker zu bestrafen. Durch die automatische Differenzierung wird die Verlustfunktion nach allen zu trainierenden Parametern abgeleitet. Mit dem *Adam*-Optimierer in seiner *AMSGrad*-Ausführung [26] werden die zu trainierenden Parameter mittels eines Gradientenverfahrens mit einer adaptiven Schrittweite verändert.

3.3.2 PGNARXnetlf

Ziel der Untersuchungen ist es zu überprüfen, ob die Ergebnisse der NARXnets durch die Wahl Physik-geführter Verlustfunktionen verbessert werden können. Dazu werden die beiden in Abschnitt 2 vorgeschlagenen Ansätze mit den zuvor gezeigten NARXnet verglichen. Zunächst soll an dieser Stelle das PGNARXnetlf eingeführt werden. Darunter ist ein NARXnet zu verstehen, welches allein durch Physik-geführte Verlustterme trainiert wird. Da keine klassischen Verlustterme bei dieser Netzart verwendet werden, wird das Netzwerk *label-free* trainiert.

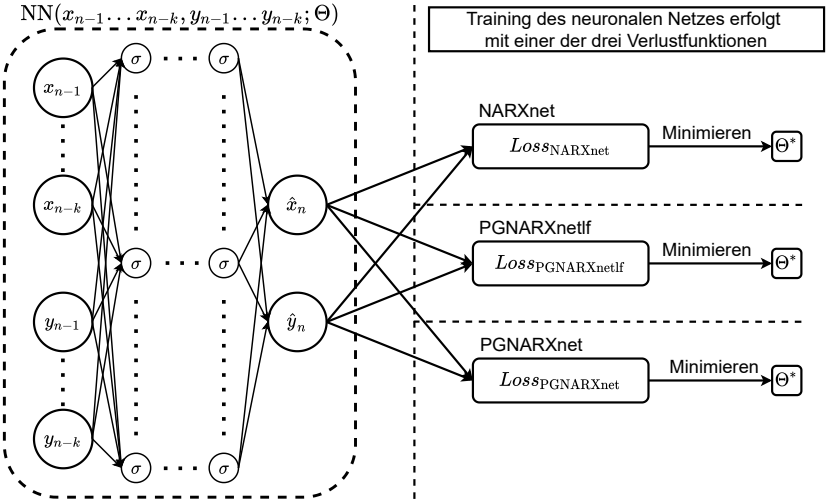


Bild 3: Veranschaulichung der neuronalen Netzstruktur mit drei verschiedenen Lösungsansätzen: NARXnet, PGNARXnetlf und PGNARXnet. Θ stellt die Parameter der neuronalen Netze (Gewichte und Schwellenwerte) dar, wobei Θ^* die optimierten Parameter angibt. σ markiert die Aktivierungsfunktion.

Die Verlustfunktion des PGNARXnetlf setzt sich aus zwei verschiedenen Verlusttermen zu einer kombinierten Verlustfunktion

$$Loss_{\text{PGNARXnetlf}} = MSE_{\text{ODE},1} + MSE_{\text{ODE},2} \tag{17}$$

zusammen. Die Verlustfunktion wird auf das zugrundeliegende Differentialgleichungssystem bestehend aus zwei nichtlinearen gewöhnlichen Differentialgleichungen erster Ordnung (Gleichung 10 und 11) zugeschnitten.
Der erste Verlustterm

$$MSE_{\text{ODE},1} = \frac{\lambda_{\text{ODE},1}}{M} \sum_{n=k}^{M+k} \left(\frac{\mathrm{d}\hat{x}_n}{\mathrm{d}t} - \hat{y}_n \right)^2 \tag{18}$$

spiegelt die erste Gleichung des Differentialgleichungssystems wider, wobei die Ableitung mittels linksseitigem Differenzenquotienten

$$\frac{\mathrm{d}\hat{x}_n}{\mathrm{d}t} \approx \frac{a_0 \cdot x_{n-k} + \ldots + a_{N-2} \cdot x_{n-1} + a_{N-1} \cdot \hat{x}_n}{\Delta t} \tag{19}$$

abgeschätzt wird. N gibt die Anzahl der Stützstellen an. In den Differenzenquotienten fließen die als Eingangsgrößen dienenden zurückliegenden Funktionswerte $x_{n-1}, \ldots, x_{n-k}$ wie auch die bekannte Zeitschrittweite Δt ein. Die Vorfaktoren $a_0, \ldots, a_{N-1}$ werden durch die Lösung des linearen Gleichungssystems (Gleichung 6) bestimmt.
Der zweite Verlustterm

$$MSE_{\mathrm{ODE},2} = \frac{\lambda_{\mathrm{ODE},2}}{M} \sum_{n=k}^{M+k} \left(\frac{\mathrm{d}\hat{y}_n}{\mathrm{d}t} - \varepsilon \left(1 - \hat{x}_n^2\right) \hat{y}_n + \hat{x}_n^3 - A \cos\left(2\pi f t_n\right) \right)^2 \tag{20}$$

zwingt dem neuronalen Netz die zweite Differentialgleichung des Systems und damit die modifizierte Van der Pol Gleichung auf. Die Ableitung wird, wie oben, mittels linksseitigem Differenzenquotienten

$$\frac{\mathrm{d}\hat{y}_n}{\mathrm{d}t} \approx \frac{a_0 \cdot y_{n-k} + \ldots + a_{N-2} \cdot y_{n-1} + a_{N-1} \cdot \hat{y}_n}{\Delta t} \tag{21}$$

abgeschätzt, wobei die zurückliegenden Funktionswerte $y_{n-1}, \ldots, y_{n-k}$ wie auch die Zeitschrittweite Δt verwendet werden. Darüber hinaus müssen die Koeffizienten A, ε und f wie auch die Zeit t_n bekannt sein, da diese in den Verlustterm integriert werden.

Die Vorfaktoren $\lambda_{\mathrm{ODE},1}$ und $\lambda_{\mathrm{ODE},2}$ in den Verlusttermen dienen der Gewichtung, welche für das PGNARXnet entscheidend ist.

3.3.3 PGNARXnet

Neben dem PGNARXnetlf wird eine weitere Netzart eingeführt. In dem PGNARXnet werden die Ansätze des NARXnet und des PGNARXnetlf kombiniert. Die Verlustfunktion setzt sich aus vier Verlusttermen

$$Loss_{\mathrm{PGNARXnet}} = MSE_{\mathrm{klass},1} + MSE_{\mathrm{klass},2} + MSE_{\mathrm{ODE},1} + MSE_{\mathrm{ODE},2} \tag{22}$$

sowohl den klassischen als auch den Physik-geführten Verlusttermen zusammen. Die Gewichtung der Physik-geführten Verlustterme gegenüber den klassischen Verlusttermen kann über die Vorfaktoren $\lambda_{\mathrm{ODE},1}$ und $\lambda_{\mathrm{ODE},2}$ eingestellt werden. Von dieser Art der Verlustfunktion erhofft man sich, dass die Vorteile beider Verlusttermarten überwiegen und somit die Performance der Netze in Hinblick auf die Abweichung der Vorhersage wie auch der Generalisierung verbessert werden. Es ist jedoch zu beachten, dass durch die Hinzunahme der klassischen Verlustterme das Training nicht mehr *label-free* erfolgen kann.

4 Ergebnisse

Die Architektur der neuronalen Netze wird in den Versuchen variiert. Verwendet werden jeweils Netze, bestehend aus zwei verdeckten Schichten. Die Anzahl der Neuronen in den verdeckten Schichten $N_{\mathrm{hidden},1}$ und $N_{\mathrm{hidden},2}$ werden nach den etwas abgewandelten heuristischen Regeln [27, 28, 29] mit

$$N_{\mathrm{hidden},1} = 2 \cdot (2 \cdot d + 1) \qquad \text{und} \qquad N_{\mathrm{hidden},2} = 2 \cdot \sqrt{N_{\mathrm{hidden},1}} \tag{23}$$

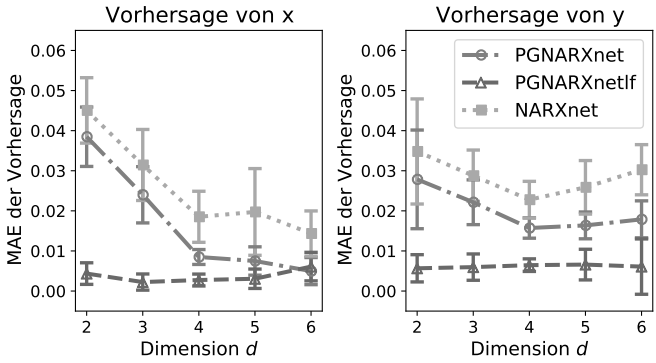
gewählt, wobei d die *Embedding-Dimension* [30] angibt und $N_{\mathrm{hidden},2}$ zur nächsten natürlichen Zahl aufgerundet wird. Aufgrund der zwei Funktionswerte als Ausgabegrößen wurde die Anzahl der Neuronen in den Schichten verdoppelt. Die erste heuristische Regel ist auf das Kolmogorov-Theorem in Bezug auf die Approximation von Funktionen zurückzuführen [31]. Das Kolmogorov-Theorem wurde durch Hecht-Nielsen [32] abgewandelt und auf neuronale Netze übertragen (*Kolmogorov Mapping Neural Network Existence Theorem*). Die zweite heuristische Regel besagt, dass die Anzahl an Neuronen in der zweiten Schicht als Quadratwurzel des Produktes der Dimension der ersten versteckten Schicht und der Dimension der Ausgabeschicht gewählt wird [29]. Die *Embedding-Dimension d* wurde durch Bestimmung des Minimums der *Mutual Information* [33, 34], welches sich in unserem Fall bei einer Verschiebung von $\tau = 85$ befindet, und nachfolgender *false nearest neighbour procedure* [30] mit Hilfe des Programms von Wallot et al. [35] berechnet und beträgt gerade 3. Die Zeitschrittweite $\tau = 85$ wäre in Anbetracht dessen, dass der Fehler des Differenzenquotienten mit der

Tabelle 1: Vergleich der Güte der Vorhersage eines klassischen NARXnet für verschiedene Zeitschrittweiten τ. Die Netze werden mit den Trainingsdaten mit einer Rauschstärke von $\nu = 0.2$ trainiert und die Vorhersage an den Testdaten mit einer Rauschstärke von $\nu = 0$ getestet.

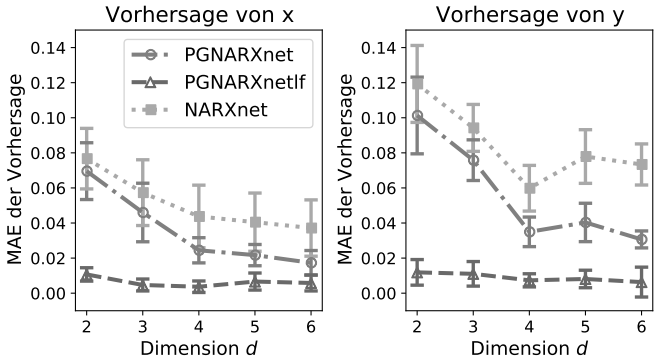
Zeitschrittweite τ	MAE der Vorhersage von x	MAE der Vorhersage von y
1	0.0314 ± 0.0086	0.0288 ± 0.0064
85	0.0221 ± 0.0121	0.0401 ± 0.0148

Zeitschrittweite ansteigt, zu groß und daraus resultiert ein großer Fehler in Hinblick auf die Abschätzung der Ableitungen mit dem Differenzenquotienten. Deswegen wurde die Entscheidung getroffen, keine zusätzliche Verschiebung zu verwenden. Die Vorhersage des NARXnet in Bezug auf die Variable x wird dadurch etwas schlechter, wohingegen der mittlere absolute Fehler der Vorhersage des NARXnet in Bezug auf die Variable y sogar abnimmt (siehe Tabelle 1). Aufgrund der Ergebnisse in der gleichen Größenordnung ist die Entscheidung, $\tau = 1$ zu wählen, legitim.

Zum Training der Netze erfolgt eine Normalisierung der Daten auf den Bereich $[-1,1]$. Als Aktivierungsfunktion wurde die hyperbolische Tangensfunktion verwendet. Die Netze wurden jeweils mit fünf verschiedenen zufälligen Initialisierungen trainiert. Die Ergebnisse zeigen jeweils die Mittelwerte mit den zugehörigen Standardabweichungen. Die Rechnungen wurden mit der *TensorFlow*-Bibliothek [36] und der darauf aufbauenden *Keras*-Bibliothek [37] ausgeführt. Weiterhin sollte erwähnt werden, dass keine vollständige Optimierung des Netzwerks durchgeführt wurde. Im Vordergrund dieser Arbeit waren besagte Lösungsansätze zu testen und miteinander bzw. mit dem Standardansatz zu vergleichen. Um jedoch die beste *Embedding-Dimension* für den Versuch zu finden, wurde diese in den Tests variiert (siehe Abbildung 4). Trainiert wurden die Netzwerke mit einer Rauschstärke ν von 0.2 und der mittlere absolute Fehler anhand der Testdaten wie auch der Zielzeitreihe mit einer Rauschstärke von $\nu = 0$ bestimmt, um das Erlernen der Funktion während des Trainings zu überprüfen.



(a) Testdaten der Trainingszeitreihe: $A = 1$ und $\varepsilon = 1$



(b) Test der Extrapolationsleistung auf neue Daten: $A = 1$ und $\varepsilon = 0.1$

Bild 4: Mittlerer absoluter Fehler der Vorhersage in Abhängigkeit der Dimension d. Die Rauschstärke ν auf den Trainings- und Validierungsdaten betrug 0.2 und die Rauschstärke auf den Testdaten 0.

Mit steigender *Embedding-Dimension* $d = 2 \ldots 4$ lässt sich für das NARXnet und das PGNARXnet eine Abnahme des mittleren absoluten Fehlers der Vorhersage feststellen. Der mittlere absolute Fehler der Vorhersage von y steigt für größere *Embedding-Dimension* sowohl für die Testdaten wie auch die Zielzeitreihe mit neuen Koeffizienten an. Hingegen lässt sich eine weitere Abnahme des mittleren absoluten Fehlers für die Vorhersage von x feststellen. Das PGNARXnet erzielt für alle *Embedding-Dimensions* im Vergleich zum NARXnet einen geringeren Fehler. Durch die Ergänzung des Physik-geführten Ver-

lustterms erfolgt eine Regularisierung sowie eine Einschränkung des Lösungsraums, sodass diese im Vergleich zu den klassischen NARXnets das Differentialgleichungssystem besser erlernen. Von den diskutierten Verläufen weicht der mittlere absolute Fehler des PGNARXnetlf ab. Vor allem bei geringen *Embedding-Dimensions* werden hier bereits sehr gute Werte erzielt. Bei der Bestimmung der *Embedding-Dimension* wird deutlich, die Extrapolationsleistung des PGNARXnetlf ist im Vergleich zu dem klassischen NARXnet-Ansatz wie auch dem PGNARXnet-Ansatz verbessert (siehe Abbildung 4b).
Die Wahl der *Embedding-Dimension* $d = 3$ fiel aufgrund des im Mittel besten Ergebnis für das PGNARXnetlf in Bezug auf die Testdaten. Anhand dieses beispielhaften Netzes werden die nachfolgenden Untersuchungen durchgeführt.

4.1 Performance der Verlustfunktionen

Bei der Diskussion der Verlustfunktionen sind zwei verschiedene Verluste zu unterscheiden: Die Trainings- und Validierungsverluste. Die jeweiligen Verlustfunktionen werden bei den Trainings- und Validierungsverlusten nach jeden Batch ausgewertet. Nach jeder Epoche wird der Mittelwert dieser Verlustwerte berechnet. Die Trainings- und Validierungsverluste unterscheiden sich allein durch die Daten, an welchen diese berechnet werden. Anhand der Trainingsverluste wird das Netz optimiert. Überprüft wird der Optimierungsfortschritt wie auch die Verallgemeinerbarkeit der Ergebnisse durch den Validierungsfehler. Da die prinzipielle Entwicklung der Verlustfunktionen sehr ähnlich sind, wobei die Trainingsverlustwerte typischerweise niedriger ausfallen, sollen an dieser Stelle die Entwicklung der unterschiedlichen Validierungsfehler der verschiedenen Netzarten am Beispiel der *Embedding-Dimension* von $d = 3$ und dem Training mit einer Rauschstärke von $\nu = 0.2$ diskutiert werden (siehe Abbildung 5).

Vergleicht man die Verläufe der unterschiedlichen Netzarten PGNARXnet, PG- NARXnetlf und NARXnet, unterscheiden sich diese stark voneinander. Der Validierungsfehler der NARXnet $Loss_{\text{NARXnet}}$ nimmt nur minimal ab und erreicht bereits nach 114 Epochen sein Minimum. Im Gegensatz dazu verringert sich der Validierungsfehler des PGNARXnetlf $Loss_{\text{PGNARXnetlf}}$ über die 15000 Epochen und erreicht innerhalb dieses Trainingsabschnitts

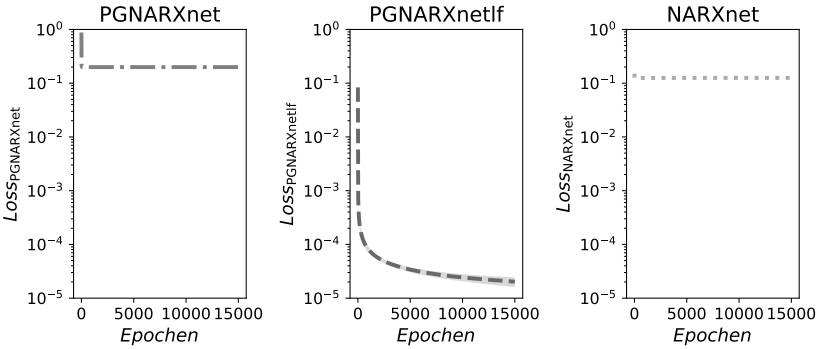


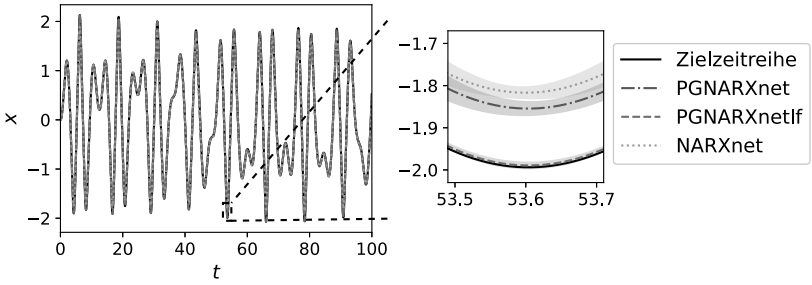
Bild 5: Vergleich der Entwicklung der unterschiedlichen Validierungsfehler der verschiedenen Netzarten. Dargestellt sind die Mittelwerte mit Standardabweichung der Netze mit einer *Embedding-Dimension* von $d = 3$ und dem Training mit einer Rauschstärke von $\nu = 0.2$.

kein Minimum. Bedingt durch den Ausgleich der verschiedenen Verlustterme und die dominierenden klassischen Verluste lässt sich beim PGNARXnet $Loss_{\text{PGNARXnet}}$ ein ähnliches Verhalten wie beim NARXnet feststellen. Ein Minimum lässt sich nach 256 Epochen beobachten.
Anhand dieser Ergebnisse liegt die Vermutung nahe, dass die Physik-geführten NARXnets und dabei primär das PGNARXnetlf die zugrundeliegende Funktion im Vergleich zu den anderen Netzen besser erlernt.

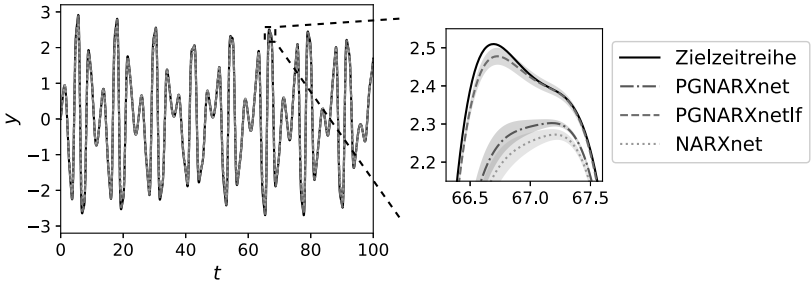
4.2 Vorhersage

Die Vorhersage spiegelt das in vorigen Abschnitt angedeutete Verhalten wider. In Abbildung 6 ist beispielhaft die Vorhersage anhand der Zielzeitreihe mit im Vergleich zur Trainingszeitreihe abgeänderten Koeffizienten für die besagte Netzarchitektur mit einer *Embedding-Dimension* $d = 3$ gezeigt. Im Vergleich zum Training wurde die Rauschstärke der Zielzeitreihe auf $\nu = 0$ gesetzt. Somit kann anhand der Vorhersage überprüft werden, inwiefern das zugrundeliegende Differentialgleichungssystem durch das neuronale Netz erlernt wird.

Anhand der kompletten Zeitreihe lassen sich die Unterschiede der Vorhersage nur schwierig abschätzen. Schaut man sich doch gerade die lokalen Minima und Maxima der modifizierten Van der Pol Gleichung an, stellt man fest, dass



(a) Vorhersage von x



(b) Vorhersage von y

Bild 6: Vergleich der Vorhersagen anhand der Zielzeitreihe mit den Koeffizienten: $A = 1$, $\varepsilon = 0.1$ und einer Rauschstärke von $\nu = 0$. Ziel ist der Test der Extrapolationsfähigkeit.

die Vorhersage des NARXnet am stärksten von der Zielzeitreihe abweicht. Auch die Vorhersage des PGNARXnet weist eine vergleichsweise große Abweichung auf, wobei diese im Vergleich zur Vorhersage des NARXnet geringer ist. Dies resultiert aus der Ergänzung der Physik-geführten Verlustterme, welche hier als eine Art Regularisierung wirken. Mit Abstand die beste Vorhersage für das gewählte Beispiel zeigt das PGNARXnetlf. Tabelle 2 fasst diese Beobachtungen zusammen. Das besagte Beispiel zeigt: Durch die Vorgabe der Differentialgleichungen des zugrundeliegenden Systems kann die Extrapolationsfähigkeit des PGNARXnetlf gegenüber dem klassischen NARXnet erhöht werden, da die Dynamik des Systems explizit durch die Vorgabe der zeitlichen Ableitungen in den Differentialgleichungen und dem Zusammenhang der Ableitung mit den zurückliegenden Funktionswerten aufgezwungen wird.

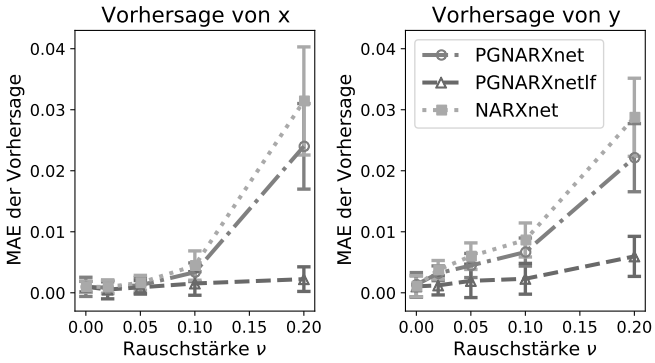
Tabelle 2: Vergleich der mittleren absoluten Fehler der Vorhersage von x und y. Die Netze werden mit den Trainingsdaten mit einer Rauschstärke von $\nu = 0.2$ trainiert und die Vorhersage an der Zielzeitreihe ($A = 1$ und $\varepsilon = 0.1$) mit einer Rauschstärke von $\nu = 0$ getestet.

	Vorhersage von x	Vorhersage von y
$MAE_{\text{PGNARXnet}}$	0.0460 ± 0.0167	0.0758 ± 0.0116
$MAE_{\text{PGNARXnetlf}}$	0.0047 ± 0.0034	0.0110 ± 0.0070
MAE_{NARXnet}	0.0573 ± 0.0187	0.0942 ± 0.0134

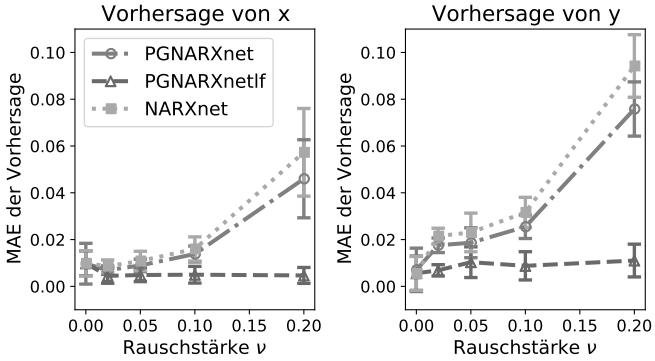
Für das identische Netzwerk wurde weiterhin die Rauschstärke ν auf der Trainingszeitreihe verändert und der mittlere absolute Fehler der Vorhersage anhand der Testdaten sowie der Zielzeitreihe mit veränderten Koeffizienten bei einer Rauschstärke von $\nu = 0$ bestimmt (siehe Abbildung 7). Bei geringen Rauschstärken auf den Trainings- und Validierungsdaten zeigen alle drei verschiedenen Netzarten ähnliche Ergebnisse. Mit steigender Rauschstärke nimmt der mittlere absolute Fehler der Vorhersage des PGNARXnet wie auch des NARXnet deutlich zu, wobei der mittlere absolute Fehler des NARXnet größer ist als der des PGNARXnet. Der mittlere absolute Fehler der Vorhersage für das PGNARXnetlf steigt dagegen nur leicht an bzw. bleibt annähernd konstant.

5 Fazit

Die Physik-geführten NARXnets weisen für analytisch bzw. numerisch berechnete Daten ähnliche Fehlerwerte wie die klassischen NARXnets auf. Der Vorteil besteht jedoch darin, dass die Konsistenz der zugrundeliegenden Physik durch die Physik-geführten Verlustterme gesichert wird. Erhöht man das Rauschen auf den Trainingsdaten, lernen die Physik-geführten NARXnets das zugrundeliegende Differentialgleichungssystem besser. Vor allem spiegelt sich dies bei dem Übertrag der Netze auf neue Zeitreihen mit gleichem Differentialgleichungssystem aber veränderten Koeffizienten wider. Die Extrapolationsleistung kann somit durch die Physik-geführten Verlustterme verbessert werden. Weiterhin ermöglichen die Physik-geführten NARXnets, welche durch die alleinige Nutzung der Physik-geführten Verlustterme trainiert



(a) Testdaten der Trainingszeitreihe: $A = 1$ und $\varepsilon = 1$



(b) Test der Extrapolationsleistung auf neue Daten: $A = 1$ und $\varepsilon = 0.1$

Bild 7: Mittlerer absoluter Fehler der Vorhersage in Abhängigkeit der Rauschstärke ν auf den Trainings- und Validierungsdaten. Die Vorhersage wurde an nicht-verrauschten Daten getestet, um das Erlernen des Differentialgleichungssystems zu prüfen.

werden, sogenanntes *label-free learning*. Außerdem wirkt der Physik-geführte Verlustterm bei den PGNARXnets als eine Art Regularisierung und reduziert zugleich das Problem der Überanpassung.

Um die Vorhersage unter Vorgabe verrauschter Daten zu verbessern, sind derzeit weiterführende Arbeiten geplant. Die Idee ist, zur Optimierung Differenzenquotienten, welche stabil gegenüber Rauschen sind, zu verwenden und einhergehend dazu mehr Stützstellen für diese Differenzenquotienten zu berücksichtigen. Beispielsweise wäre eine Vorschlag Savitzky-Golay-

Differentiatoren zu nutzen, welche auf dem Savitzky-Golay-Filter [38] basieren.

Literatur

[1] R. K. Jain, K. M. Smith, P. J. Culligan und J. E. Taylor. „Forecasting energy consumption of multi-family residential buildings using support vector regression: Investigating the impact of temporal and spatial monitoring granularity on performance accuracy“. In: *Applied Energy*, Band 123, S. 168-178. 2014. https://doi.org/10.1016/j.apenergy.2014.02.057

[2] T. S. Hu, K. C. Lam und S. T. Ng. „River flow time series prediction with a rangedependent neural network“. In: *Hydrological Sciences Journal*, Band 45, Nr. 5, S. 729-745. 2001. https://doi.org/10.1080/02626660109492867

[3] R. Trippi und E. Turban. „Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real-World Performance“. Burr Ridge Ilinois: Irwin Professinal Publishing Co. 1996. https://doi.org/10.1016/S0169-2070(96)00728-5

[4] W. Hunag, K. K. Lai, Y. Nakamori, S. Wang und L. Yu. „Neural networks in finance and economics forecasting“. In: *International Journal of Information Technology & Decision Making*, Band 6, Nr. 1, S. 113-140. 2007. https://doi.org/10.1142/S021962200700237X

[5] S. Chen, S. A. Billings und P. M. Grant. „Non-linear system indentification using neural networks“. In: *International Journal of Control*, Band 51, Nr. 6, S. 1191-1214. 1990. https://doi.org/10.1080/00207179008934126

[6] X. Zhang. „Time series analysis and prediction by neural networks“. In: *Optimization Methods and Software*, Band 4, Nr. 2, S. 151-170. 1994. https://doi.org/10.1080/10556789408805584

[7] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova und V. Kumar. „Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data“. In: *IEEE Transactions on Knowledge and Data Engineering*, Band 29, Nr. 11, S. 2318-2331. 2017. https://doi.org/10.1109/TKDE.2017.2720168

[8] M. A Kraus und A. Taras. „Physik-informierte Künstliche Intelligenz zur Berechnung und Bemessung im Stahlbau“. In: *Stahlbau*, Band 89, Nr. 10, S. 824-832. 2020. https://doi.org/10.1002/stab.202000074

[9] A. Karpatne, W. Watkins, J. Read und V. Kumar. „Physics-guided Neural Networks (PGNN): An Application in Lake Temperature Modeling“. *arXiv-preprint*. 2018. arXiv:1710.11431v2

[10] A. Krizhevsky, I. Stuskever und G. E. Hinton. „Imagenet classification with deep convolutional neural networks“. *Proceedings Advances in Neural Information Processing Systems 25*, S. 1090-1098. 2012. https://doi.org/10.1145/3065386

[11] G. E. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath und B. Kingsbury. „Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups“. *IEEE Signal Processing Magazine*, Band 29, Nr. 6, S. 82-97. 2012. https://doi.org/10.1109/MSP.2012.2205597

[12] M. Raissi, P. Perdikaris und G. E. Kariadakis. „Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations“. In: *Journal of Computational Physics*, Band 378, S. 686-707. 2019. https://doi.org/10.1016/j.jcp.2018.10.045

[13] S. Karimpouli und P. Tahmasebi. „Physics informed machine learning: Seismic wave equation“. In: *Geoscience Frontiers*, Band 11, Nr. 2, S. 1196-2001. 2020. https://doi.org/10.1016/j.gsf.2020.07.007

[14] Y. Chen, L. Lu, G. E. Karniadakis und L. D. Negro. „Physics-informed neural networks for inverse problems in nano-optics and metamaterials“. In: *Optics Express*, Band 28, Nr. 8, S. 11618-11633. 2020. https://doi.org/10.1364/OE.384875

[15] L. Lu, X. Meng, Z. Mao und G. E. Karniadakis. „DEEPXDE: A deep learning libary for solving differential equations“. *arXiv-preprint*. 2020. arXiv:1907.04502v2

[16] M. Elhamod, J. Bu, C. Singh, M. Redell, A. Ghosh, V. Podolskiy, W.-C. Lee und A. Karpatne. „CoPhy-PGNN: Learning Physics-guided Neural Networks with Competing Loss Functions for Solving Eigenvalue Problems“. *arXiv-preprint*. 2020. arXiv:2007.01420v4

[17] R. Zhang, Y. Liu und H. Sun. „Physics-guided Convolutional Neural Network (PhyCNN) for Data-driven Seismic Response Modeling“. *arXiv-preprint*. 2019. arXiv:1909.08118v1

[18] T. Parditia, T. Walser, S. Oladyshkin und W. Nowak. „Improving Thermochemical Energy Storage Dynamics Forecast with Physics-Inspired Neural Network Architecture“. In: *Energies*, Band 13, Nr. 15, S. 3873-3899 2020. https://doi.org/10.3390/en13153873

[19] M. Bolderman, M. Lazar und H. Butler. „Physics-Guided Neural Networks for Inversion-based Feedforward Control applied to Linear Motors“. *arXiv-preprint*. 2021. arXiv:2103.06092v1

[20] B. Fornberg. „Generation of Finite Difference Formulas on Arbitrarily Spaced Grids“. In: *Mathematics of Computation*, Band 51, Nr. 184, S. 699-706. 1988. https://doi.org/10.1090/S0025-5718-1988-0935077-0

[21] C. Taylor. „Finite Difference Coefficients Calculator“. Massachusetts Institute of Technology, https://web.media.mit.edu/~crtaylor/calculator.html. 2019.

[22] B. van der Pol. „On relaxation oscillations“. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Band 2, S. 978-992. 1926. https://doi.org/10.1080/14786442608564127

[23] Y. H. Ku und X. Sun. „CHAOS in Van der Pol's Equation“. In: *Journal of the Franklin Institute, Pergamon Press plc*, Band 327, Nr. 2, S. 197-207. 1990. https://doi.org/10.1016/0016-0032(90)90016-C

[24] Y. Ueda und N. Akamatsu. „Chaotically Transitional Phenomena in the Forced Negative-Resistance Oscillator“. In: *IEEE Transactions on Circuits and Systems*, Band 28, Nr. 3, S. 217-224. 1981. https://doi.org/10.1109/TCS.1981.1084975

[25] J. R. Dormand and P. J. Prince. „A family of embedded Runge-Kutta formulae“. In: *Journal of Computational and Applied Mathematics*, Band 6, Nr. 1, S. 19-26. 1980. https://doi.org/10.1016/0771-050X(80)90013-3

[26] S. J. Reddi, S. Kale und S. Kumar. „On the Convergence of Adam and Beyond“. *arXiv-preprint*. 2018. arXiv:1904.09237

[27] T. Masters. „Practical Neural Network Recipes in C++“. San Diego: Academic Press, Inc. 1993.

[28] R. J. Frank, N. Davey und S. P. Hunt. „Time Series Prediction and Neural Networks“. In: *Journal of Intelligent and Robotic Systems*, Band 31, S. 91-103. 2001. https://doi.org/10.1023/A:1012074215150

[29] J. M. P. Menezes und G. A. Barreto. „Long-term time series prediction with the NARX network: An empirical evaluation“. In: *Neurocomputing*, Band 71, Nr. 16-18, S. 3335-3343. 2008. https://doi.org/10.1016/j.neucom.2008.01.030

[30] M. B. Kennel, R. Brown und H. D. I. Abarbanel. „Determining embedding dimension for phase-space reconstruction using a geometrical construction“. In: *Physical Review A*, Band 45, Nr. 6, S. 3403-3411. 1992. https://doi.org/10.1103/PhysRevA.45.3403

[31] A. N. Kolmogorov. „On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition“. In: *Doklady Akademii Nauk SSSR*, Band 144, S. 679-681. 1957. *American Mathematical Society Translation*, Nr. 28, S. 55-59. 1963.

[32] R. Hecht-Nielsen. „Kolmogorov's Mapping Neural Networks Existence Theorem". In: *First IEEE International Conference on Neural Networks*, San Diego, Band 3, S. 11-14. 1987.

[33] A. M. Fraser und H. L. Swinney. „Independent coordinates for strange attractors from mutual information". In: *Physical Review A*, Band 33, Nr. 2, S. 1134-1140. 1986. https://doi.org/10.1103/PhysRevA.33.1134

[34] W. Liebert und H. G. Schuster. „Proper choice of the time delay for the analysis of chaotic time series". In: *Physics Letters A*, Band 142, Nr. 2-3, S. 107-111. 1989. https://doi.org/10.1016/0375-9601(89)90169-2

[35] S. Wallot und D. Møster. „Calculation of Average Mutual Information (AMI) and False-Nearest Neighbors (FNN) for the Estimation of Embedding Parameters of Multidimensional Time Series in Matlab". In: *Frontiers in Psychology*, Band 9, S. 1679-1689. 2018. https://doi.org/10.3389/fpsyg.2018.01679

[36] M. Abadi et al. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems". https://www.tensorflow.org/. 2015.

[37] F. Chollet et al. „Keras". https://keras.io. 2015.

[38] A. Savitzky und J. E. Golay. „Smoothing and Differentiation of Data by Simplified Least Squares Procedures". In: *Analytical Chemistry*, Band 36, Nr. 8, S. 1627-1639. 1964. https://doi.org/10.1021/ac60214a047

Genetische Algorithmen zur Hyperparameteroptimierung künstlicher neuronaler Netze für die Energiezeitreihenprognose

Tobias M. Fischer, Fabian Bauer, Silas A. Selzer, Peter Bretschneider

Fachgebiet Energieeinsatzoptimierung
Technische Universität Ilmenau
Gustav-Kirchhoff-Straße 5
98693 Ilmenau
E-Mail: {tobias-merlin.fischer, fabian.bauer, silas-aaron.selzer, peter.bretschneider}@tu-ilmenau.de

1 Einleitung

In der Energieversorgung spielen exakte Vorhersagen der Verbrauchslastgänge eine wesentliche Rolle für die optimale Planung und Steuerung des elektrischen Energiesystems. Aufgrund der Digitalisierung des Elektroenergiesystems wird die Anzahl der messtechnisch erfassten Lastgänge und exogenen Einflussgrößen signifikant ansteigen. [1] Zur möglichst genauen Vorhersage ist es notwendig, Modelle zu entwickeln und zu untersuchen, die dem wachsenden Mengengerüst Rechnung tragen und aufgrund der signifikant höheren Anzahl von Beobachtungen zu möglichst verbesserten Modellen führen. Maschinelles Lernen ist eine Anwendung der künstlichen Intelligenz, die Systeme in die Lage versetzt, aus Daten selbständig zu lernen. Bei Verfahren des maschinellen Lernens ist ein entscheidender Faktor die Optimierung der Hyperparameter. Hierzu werden im Rahmen der Arbeit die beiden Verfahren Zufallssuche und genetische Algorithmen untersucht. Dabei zeigen die, auf den Evolutionsprinzipien beruhenden genetischen Algorithmen bei der Suche nach dem globalen

Optimum nichtlinearer Problemstellungen vielversprechende Ergebnisse. Inden durchgeführten Untersuchungen werden Verfahren genetischer Algorithmen angewandt und mit gängigen Methoden verglichen.[2]

2 Hyperparameter

Künstliche neuronale Netze sind informationsverarbeitende Systeme und nutzen zum Training der Netzparameter Methoden des maschinellen Lernverfahrens. Jedes KNN verfügt über Hyperparameter und eine der grundlegendsten Aufgaben der Modelloptimierung ist es, die optimale Hyperparameterkombination über einen definierten Suchraum zu finden. Als Hyperparameter werden alle Parameter bezeichnet, die sich während eines Trainingszyklus des KNN konstant gehalten werden. Hierzu zählen bspw. die Anzahl der Schichten, die Anzahl der Knoten je Schicht oder auch die verwendeten Aktivierungsfunktionen. Besonders in den immer komplexer werdenden KNN-Architekturen ist die richtige Auswahl der Hyperparameter von entscheidender Bedeutung. So lässt sich mit einer effektiven Hyperparameteroptimierung der notwendige Rechenaufwand zum Trainieren des KNN verringern bei gleichzeitiger Steigerung der Performance der Modelle. [2, 3]
Die Frage, wie die verfügbaren Rechenkapazitäten effizient eingesetzt und die Suchräume effektive bearbeitet werden können, führte zu einer Vielzahl von Methoden der Hyperparameteroptimierung. Eine automatisierte Hyperparameteroptimierung bietet dabei mehrere Vorteile. Hyperparameterräume sind häufig komplex, bestehend aus einer Vielzahl von kontinuierlichen, diskreten und kategorischen Hyperparametern und werden deshalb häufig nur abgeschätzt oder heuristisch ermittelt. Dadurch besteht das Risiko, dass die Hyperparameter nicht vollständig optimiert sind und sich in einem lokalen Minimum befinden. Ebenso sind die Wechselwirkungen der Hyperparameter zumeist unbekannt. Durch die algorithmische Ermittlung der Hyperparameter können Unsicherheiten und Eingriffe des menschlichen Beobachters reduziert werden. In der Vergangenheit haben sich eine Vielzahl automatisierter Methoden zu Hyperparameteroptimierung entwickelt. [4, 5]

3 Untersuchte Optimierungsansätze

Die nachfolgenden Problemherleitung der Hyperparameteroptimierung orientiert sich an [6, 7]. Mit $X \subseteq \mathbb{R}^{d_x}$ als Eingaberaum und $Y \subseteq \mathbb{R}^{d_y}$ als Ausgaberaum ergibt sich als Ziel des überwachten Lernens eine Funktion h mit $h(x;\theta) : X \rightarrow Y$, wobei $x \in X$ gilt und h aus einer Schar von Funktionen parametrisiert mit $\theta \in \mathbb{R}^d$ kommt. Im Folgenden wird θ als die Bezeichnung für die Hyperparameter verwendet. Unter Hinzunahme einer Verlustfunktion

$$l(h(x;\theta);y) \tag{1}$$

kann eine Risikofunktion

$$R(\theta) = E_{x,y}[l(h(x;\theta);y)] \tag{2}$$

für ein gegebenes θ als der zu erwartende Fehler über die zugrunde liegende Wahrscheinlichkeitsverteilung $P(x,y)$ definiert werden. Mit Hilfe einer Anzahl μ der möglichen künstlichen neuronalen Netze und den jeweils zugehörigen Lösungen $\{\theta_j\}_{j=1}^{\mu}$ ergibt sich ein Minimierungsproblem der Funktion

$$J_\mu = \frac{1}{\mu}\sum_{j=1}^{\mu} R_n(\theta_j) = \frac{1}{\mu}\sum_{j=1}^{\mu}\left(\frac{1}{n}\sum_{i=1}^{n} l_i(\theta_j)\right), \tag{3}$$

welche den erwarteten durchschnittlichen empirischen Fehlers angibt.

3.1 Rastersuche

Die Rastersuche versucht dieses Minimierungsproblem zu lösen, indem jedem Hyperparameter eine Auswahl diskreter Werte, oder Vielfache dieser Werte, vorgegeben wird und der Algorithmus evaluiert das kartesische Produkt dieser Menge. Zur Rastersuche werden ebenso heuristische Verfahren gezählt, wie etwa die Auswahl der Hyperparameter nach Erfahrungswerten. Bei h Hyperparametern mit n Werten wächst die Anzahl der sequenziell verarbeiteten Trainings- und Bewertungsversuche mit h^n.

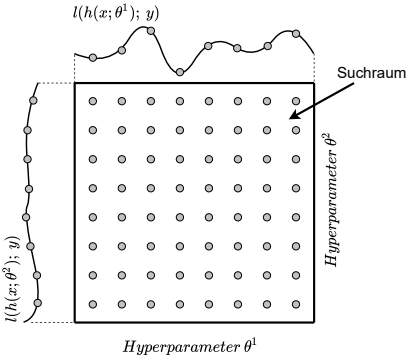


Bild 1: Darstellung der Suchraumabdeckung bei Anwendung der Rastersuche mit zwei Hyperparametern

Die Rastersuche ist deshalb nur für Anwendungen rentabel mit drei oder weniger Hyperparametern, hat jedoch den Vorteil einer gleichverteilten Suchraumabdeckung (siehe Abbildung 1). [3, 5]

3.2 Zufallssuche

Eine Erweiterung bilden zufallsbasierte Methoden der Hyperparameteroptimierung. Die Zufallssuche entspricht dem Vorgehen einer Monte-Carlo Simulation. [8] Im diskreten Hyperparameterraum werden mittels einer Randverteilung zufällige Hyperparameterkombinationen ausgewählt (siehe Abbildung 2). Im Gegensatz zur Rastersuche wird bei der Zufallssuche eine definierte Anzahl von Parameterkombinationen aus der spezifizierten Verteilung gezogen, was die Wahrscheinlichkeit verringert, viel Berechnungszeit in einem niedrig performanten Bereich des Hyperparameterraums aufzuwenden. [3] In einer unlimitierten Suche entspricht der Suchraum der Zufallssuche einer vollständigen Rastersuche, da jede Hyperparameterkombination mindestens einmal vorkommt und konvergiert damit automatisch zum globalen Optimum. In der Praxis bestehen allerdings limitierenden Faktoren wie die Rechenkapazität und -zeit, woraus der Vorteil der Zufallssuche resultiert. Bei nicht gleichverteilten Hyperparametern ermöglicht die Zufallssuche eine bessere Suchraumab-

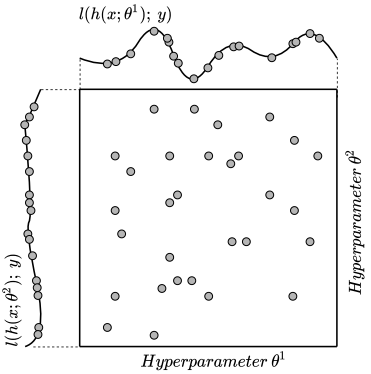


Bild 2: Darstellung der Suchraumabdeckung bei Anwendung der Zufallssuche mit zwei Hyperparametern

deckung als vergleichbare Optimierungsmethoden und steigert so die Wahrscheinlichkeit über die Optimierungszeit das globale Optimum zu finden. Als Nachteil ist zu nennen, dass die Zufallssuche ebenso wie die Rastersuche eine gewisse Anzahl unnötiger Evaluierungen durchführt, da performante Bereiche des Hyperparameterraumes nicht tiefergehend untersucht werden. [4, 5]

3.3 Genetischer Algorithmus

Die Konzeption und Struktur des in dieser Arbeit verwendeten genetischen Algorithmus orientiert sich an [7]. Für eine tiefergehende Analyse des Aufbaus verschiedener genetischer Algorithmen wird der interessierte Leser auf [3], [6] und [9] verwiesen. Die zentralen Themenfelder der Hyperparameteroptimierung sind eine möglichst große Suchabdeckung im Hyperparameterraum sowie die tiefergehende Untersuchung erschlossener lokaler Optima. Als Nebenbedingung sind die Optimierungszeit und Rechenkapazität aufzuführen. Die Zufallssuche, als Weiterentwicklung der Rastersuche, löst zwar das Problem der Suchraumabdeckung, bietet aber keine Möglichkeit regelnder Maßnahmen während der Optimierung.

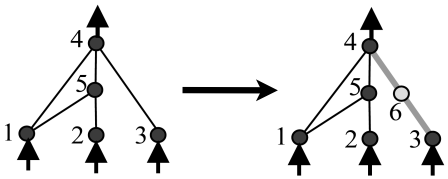


Bild 3: Mutation durch Entstehung eines neuen Neurons [10]

Der genetische Algorithmus löst diesen Zielkonflikt der Hyperparameteroptimierung, indem eine absteigende Einstufung der Verlustfunktionen

$$l\left(\theta_{1:\mu}\right) \leq l\left(\theta_{2:\mu}\right) \leq \ldots \leq l\left(\theta_{\mu:\mu}\right). \tag{4}$$

über die zufällig geschaffene Population $\Psi_\mu = \left\{\theta_1, \ldots, \theta_\mu\right\}$ mit $\theta_{k:\mu}$ als die k-besten Individuen und der Anzahl der Individuen μ erstellt wird.

Für ein Individuum ergeben sich alle Möglichkeiten an Hyperparametern innerhalb des Suchraumes

$$\Psi_\mu^h = \begin{pmatrix} \theta_1^1 & \cdots & \theta_\mu^1 \\ \vdots & \ddots & \vdots \\ \theta_1^h & \cdots & \theta_\mu^h \end{pmatrix}, \tag{5}$$

dargestellt als Matrix, wobei θ_μ^h den h-ten Hyperparameter des μ-ten Individuum bezeichnet. Für die weiteren Berechnungen wird zur Vereinfachung weiterhin Ψ als die Menge über die Population definiert. Zur Minimierung von J_μ verwendet der genetische Algorithmus Mutation, Selektion und Kreuzung. Die Mutation beschreibt die Wahrscheinlichkeit, dass ein Hyperparameter zufällig innerhalb seines Suchraumes verändert wird, siehe Abbildung 3.

Formal kann die Mutation durch die Vereinigung zweier Untermengen, der Menge der zufällig geschaffenen Ausgangspopulation

$$\Psi_{\mu,neu} = \Psi_{m\mu:\mu,alt} \cup \Psi_{1:m\mu,mutiert} \tag{6}$$

beschrieben werden, wobei m die Mutationsrate, $\Psi_{m\mu:\mu,alt}$

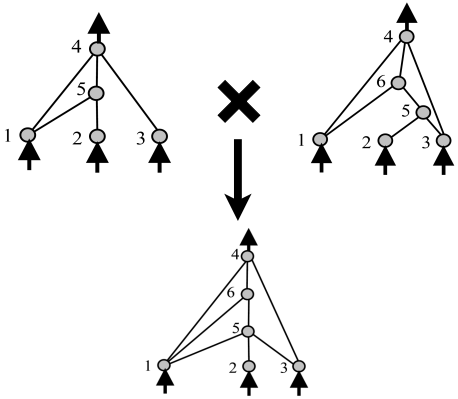


Bild 4: Die Kreuzung zweier Eltern zu einem Ableger [10]

den nicht-mutierten und $\Psi_{1:m\mu,mutiert}$ den mutierten Anteil der zufälligen Ausgangspopulation markiert. Jedoch ist zu beachten, dass die Mächtigkeit der Populationsmenge

$$\left|\Psi_{\mu,neu}\right| = \left|\Psi_{m\mu:\mu,alt}\right| + \left|\Psi_{1:m\mu,mutiert}\right| \tag{7}$$

erhalten bleibt.[7]

Als Selektion wird die Auswahl der ersten k-besten Individuen aus der Abfolge der Verlustfunktionen bezeichnet. Die Kreuzung stellt die Vermehrung von Genen innerhalb der Population dar. Aus jedem ausgewählten Individuenpaar werden zwei Nachfolger mittels der zufälligen Rekombinationsrate r erstellt. Abbildung 4 stellt die Kreuzung beispielhaft für zwei Eltern und einen Ableger dar. Für den genetischen Algorithmus ist es wichtig, dass die Gene vollständig und komplementär an die Ableger abgegeben werden, d.h. es gilt:

$$\left|\Psi^{h}_{\mu,kid1}\right| = \left|\Psi^{rh:h}_{\mu,parent1}\right| + \left|\Psi^{rh}_{\mu,parent2}\right| \tag{8}$$

sowie

$$\left|\Psi^{h}_{\mu,kid2}\right| = \left|\Psi^{rh:h}_{\mu,parent2}\right| + \left|\Psi^{rh}_{\mu,parent1}\right|. \tag{9}$$

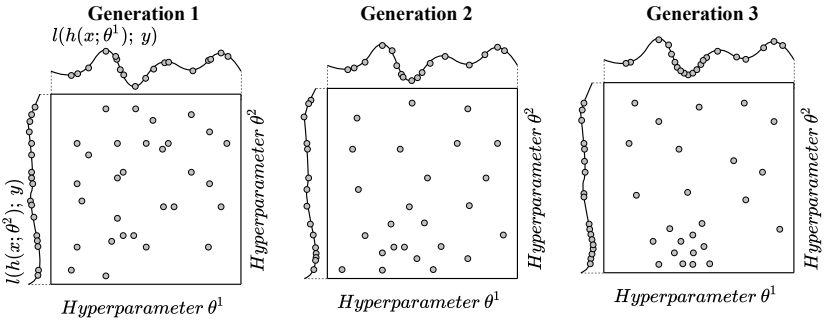


Bild 5: Darstellung der Suchraumabdeckung und Entwicklung der Hyperparametersuche über drei Generation mit genetischem Algorithmus für zwei Hyperparameter

Eine systematische Darstellung der Suchraumabdeckung und die Entwicklung der Hyperparameter über drei Generationen hinweg ist in Abbildung 5 exemplarisch dargestellt. Ziel des genetischen Algorithmus ist es, mittels der oben genannten evolutionären Techniken die Vorteile einer Zufallssuche zu erhalten, aber die Hyperparameteroptimierung dahingehend zu verbessern, dass die vorhandene Rechenkapazität auf vielversprechende Teil- oder Unterräume konzentriert wird. [7]

4 Modellierung

4.1 Datengrundlage

Die Datengrundlage für die durchgeführten Untersuchungen umfasst plausible und vollständige Energiezeitreihendaten im Umfang von drei Kalenderjahren mit einer Abtastzeit von 15 Minuten. Für die Energielastprognose ist es wichtig die Zusammenhänge zwischen verschiedenen Messgrößen als Muster zu identifizieren, weshalb die Untersuchung sowohl univariat als auch multivariat durchgeführt wurde. Für die multivariate Untersuchung standen zusätzlich die Zeitreihendaten für die Lufttemperatur, der Windgeschwindigkeit und der Solareinstrahlung zur Verfügung, ebenfalls mit einer Auflösung von

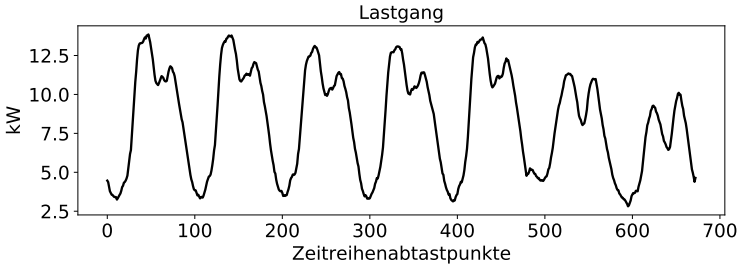


Bild 6: Auszugsweise Darstellung der Energielastzeitreihe über einen Zeitbereich von sieben Tagen

15-Minuten-Mittelwerten. Die Energielastzeitreihe ist auszugsweise in Abbildung 6 dargestellt. Deutlich zu sehen ist der Wochen- und Tageszyklus, wobei 672 Messpunkte einer Woche und 96 einem Tag entsprechen. Der verwendete Datenbestand wurde für das Anlernen der KNN-Modelle in Trainings- und Validierungsdaten aufgeteilt. Der Trainings- und Validierungsdatensatz umfasst die ersten beiden Kalenderjahre. Der anschließende Test der Prognosemodelle wird mit dem dritten Kalenderjahr durchgeführt.

4.2 Modellansatz

Als Modellansatz für die Prognosemodelle werden künstliche neuronale Netze mit den Architekturen Multi-Layer-Perceptron (MLP) und Temporal Convolutional Netzwork (TCN) verwendet.
In der MLP-Architektur bilden die künstlichen Neuronen die funktionellen Grundeinheiten. Sie sind in mehreren Schichten angeordnet und miteinander verbunden (siehe Abbildung 7). Die Schichten unterteilen sich in die Eingangsschicht, verdeckte Schicht und Ausgangsschicht. Die Neuronen einer Schicht sind mit den Neuronen der vorherigen und der nachfolgenden Schicht verbunden. Den Verbindungen zwischen den Neuronen werden Gewichte zugeordnet, welche die Relevanz einer Verbindung kennzeichnen. Die angelegten Signale der Eingabeschicht werden an die direkt verbundenen Knoten weitergegeben. In den Knoten werden die Gewichte aufsummiert, an eine Aktivierungsfunktion übergeben und der Ausgabewert berechnet. Die Informationsverarbeitung

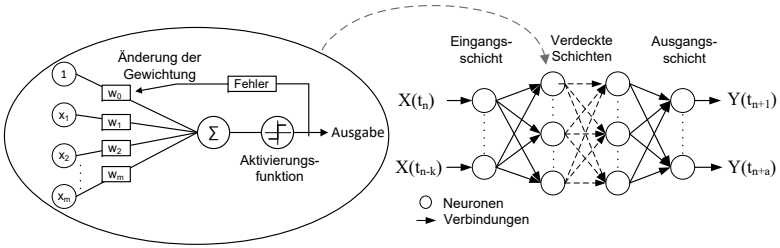


Bild 7: Aufbau eines künstlichen neuronalen Netzes in Anlehnung an [11]

erfolgt somit in den verdeckten Schichten von der Eingangs- zur Ausgangsschicht. Dabei sind die Anzahl der verdeckten Schichten, die Anzahl der enthaltenen Neuronen und die verwendete Aktivierungsfunktion frei zu wählende Hyperparameter bei der Konfiguration des KNN. [2]

Bei der TCN-Architektur hingegen wird die Eingabesequenz nicht aufsummiert, sondern mittels Faltungsoperationen verarbeitet. Der grundlegende Unterschied zwischen MLPs und TCNs ist, dass ein TCN nicht mehr nur aus Neuronen, sondern aus residualen Blöcken (RB) besteht. Diese Blöcke weisen eine eigene Netzstruktur auf, in der die Eingangssequenz zuerst eine Faltung (Dilated Causal Conv) durchläuft. Nach der Faltung erfolgt eine Gewichtsnormalisierung, darauffolgend die Verarbeitung über die Aktivierungsfunktion und zuletzt eine Regularisierung über das Dropout Verfahren. Die Gewichtsnormalisierung dient zur Rechenzeitbeschleunigung sowie dazu, explodierende Werte zu verhindern. Da sich die Struktur der Neuronen eines TCN von denen eines MLP unterscheiden, wird auch von Filtern gesprochen. Zur Bildung eines TCNs werden die residualen Blöcke aufeinandergestapelt. In Abbildung 8 ist der Aufbau eines zweischichtigen residualen Blocks zu sehen. Sollte die Eingabesequenz eine Länge unterschiedlich von der Ausgabe der Faltung haben, wird optional auf die Eingabesequenz eine 1x1 Faltung angewendet, die diese Differenz aufhebt. Durch die Faltung der Eingangsdaten können eine große Anzahl an Vergangenheitswerten berücksichtigt werden, wodurch sich TCNs besonders zur Zeitreihenprognose eignen. [12]

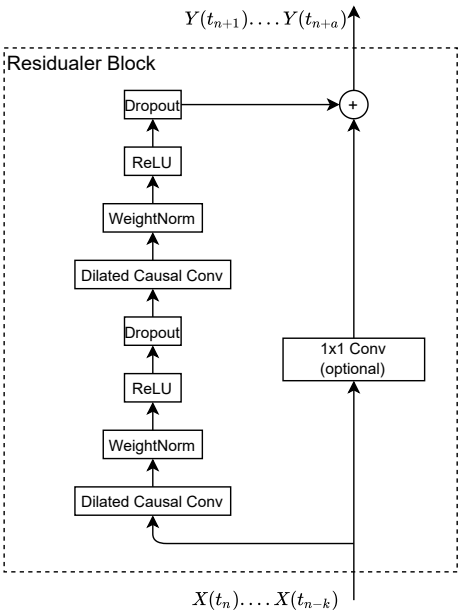


Bild 8: Aufbau eines residualen Blocks eines Temporal Convolutional Netzwork in Anlehnung an [12]

Für das Trainieren der beiden KNN-Modelle, also dem Ändern der gewichteten Verbindungen, wird der Backpropagation-Algorithmus als Lernverfahren verwendet. Dabei erfolgt die Korrektur der Netzgewichte mittels Gradientenabstiegsverfahren zur Fehlerminimierung. Für tiefergehende Erläuterungen wird auf [11] und [13] verwiesen.

Für die Lösung des Minimierungsproblems der Hyperparameteroptimierung wird in den nachfolgenden Untersuchungen der entworfene genetische Algorithmus und die Zufallssuche verwendet. Der genetische Algorithmus verwendet eine Anfangspopulation an Modellen und optimiert diese mittels Mutation, Selektion und Kreuzung.

Für alle Untersuchungsszenarien wird ein Day-Ahead Prognosehorizont festgelegt, was einer 36-Stunden-Prognose entspricht (siehe Abbildung 9).

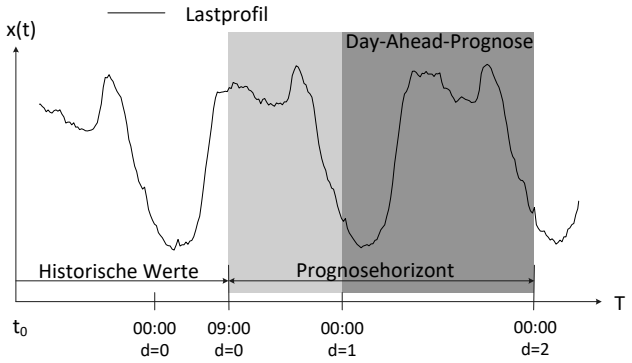


Bild 9: Prognosehorizont für die Untersuchungen

4.3 Hyperparameteroptimierung

Im Rahmen dieses Beitrags werden die Zufallssuche und ein genetischer Algorithmus über den gleichen Hyperparameterraum sowohl für eine univariate als auch eine multivariate Zeitreihenvorhersage verglichen. Als eingesetztes Framework dient Tensorflow [14], wobei die Zufallssuche mittels der Erweiterung Talos [15] durchgeführt wurde. Der Ablauf des genetischen Algorithmus ist in Abbildung 10 dargestellt. Eine zufällig generierte Startpopulation wird fünf Epochen trainiert, um in der Evaluierungsphase eine absteigende Auflistung der Tauglichkeit der Individuen zu erstellen. Von der Startpopulation werden die besten $k = 15$ % der Individuen selektiert. Um der Tatsache Rechnung zu tragen, dass weniger taugliche Modelle eventuell durch geringfügige Mutationen deutlich bessere Ergebnisse erzielen können und damit in der Rangfolge aufsteigen, wird eine Überlebensrate von 10 % eingesetzt. Diese gibt die Wahrscheinlichkeit an, dass ein eigentlich ausselektiertes Modell weiterverwendet wird. Nach der Selektion besteht die Population aus den 15 % besten Individuen und 10 % zufällig ausgewählten. Die Hyperparameter der zufällig ausgewählten Individuen werden mit der Mutationsrate $m = 30$ % zufällig verändert. Zur Erhaltung und Entwicklung der Population werden die besten Individuen zufällig gekreuzt, um Ableger zu rekombinieren. Jeweils zwei Individuen erstellen durch zufällige Rekombination zwei weitere Ableger, wobei der zweite Ableger die Hyperparameter komplementär zum ersten

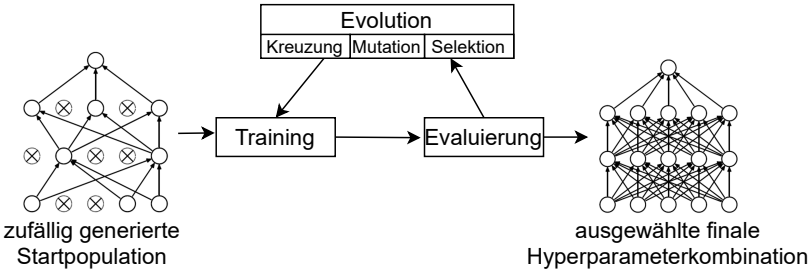


Bild 10: Schematischer Ablauf des genetischen Algorithmus mit Evolutionsstufen [16]

erhält. Die Hyperparameter beider Ableger werden ebenfalls mit $m = 30\ \%$ mutiert. Ein Durchlaufen dieser Schleife wird eine Generation genannt. Der genetische Algorithmus beendet die Hyperparameteroptimierung automatisch nach acht Generationen.

Für eine Speicherplatzoptimierung und Rechenbeschleunigung wird für die Speicherung der Individuen der Genotyp, für die Darstellung der Phänotyp verwendet. Der Genotyp ist die Gesamtheit der Gene eines Individuums bzw. dessen Erbanlagen, der Phänotyp beschreibt die sichtbaren Eigenschaften. Der Genotyp ist eine Liste, Genom genannt, mit allen Hyperparametern und deren Ausprägung eines Individuums. Dies senkt die Zugriffzeiten, da Selektion, Mutation und Kreuzung nicht in der für Tensorflow typischen als Graph dargestellten Netzarchitektur durchgeführt werden müssen. Der Phänotyp entspricht den als final Netzarchitektur dargestellten Hyperparametern, exemplarisch in Abbildung 10 dargestellt. [5, 17]

5 Simulative Untersuchung und Evaluierung

Die in Abschnitt 4 dargestellten und konzipierten Modelle mit den jeweiligen KNN-Architekturen wurden in umfangreichen simulativen Untersuchungen mit den dargestellten Hyperparameteroptimierungsmethoden (vgl. Abschnitt 3) bezüglich der Prognosequalität und der benötigten Rechenzeit miteinander verglichen. Für die Konfiguration des genetischen Algorithmus wurden folgende Einstellungen gewählt:

- Anzahl der Generationen $G = 8$
- Größe der Populationen $\mu = 30$
- Überlebensrate $s = 0,1$
- Selektionsrate $k = 0,15$
- Mutationsrate $m = 0,3$

Zur Sicherstellung der Vergleichbarkeit der Optimierungsergebnisse wird der Umfang bei der Zufallssuche auf 350 Stichproben festgesetzt.

Zur Bewertung der Modellgüte während der Zufallssuche sowie zur absteigenden Sortierung während des genetischen Algorithmus werden alle Modelle fünf Epochen trainiert.

Für die Bewertung werden die Optimierungszeit und die Prognosequalität herangezogen. Die dargestellten Fehlermaße sind der *RMSE* (Wurzel der mittleren Fehlerquadratsumme), *MAPE* (mittlerer absoluter prozentualer Fehler) und der *ME* (mittlere Fehler). Zur Erhöhung der Nachvollziehbarkeit erfolgt die Darstellung der Ergebnisse gegliedert nach der verwendeten Netzarchitektur.

MLPs verfügen als einfache vollverschaltete Netzarchitekturen über keine komplexen inneren Funktionen, wie rekursive Verbindungen oder Regularisierung. Daraus resultiert eine vereinfachte Hyperparameteroptimierung für die Zufallssuche. Hingegen ist die Netzarchitektur von TCNs aus Stapeln von residualen Blöcken aufgebaut. Folglich erhöht sich die Komplexität und Wechselwirkung der Hyperparameterauswahl. Dementsprechend wird die Performance des genetischen Algorithmus an dem einfachen Beispiel der MLP-Architektur getestet und anschließend auf das komplexere Problem der TCN-Architekturen ausgeweitet. [7, 12]

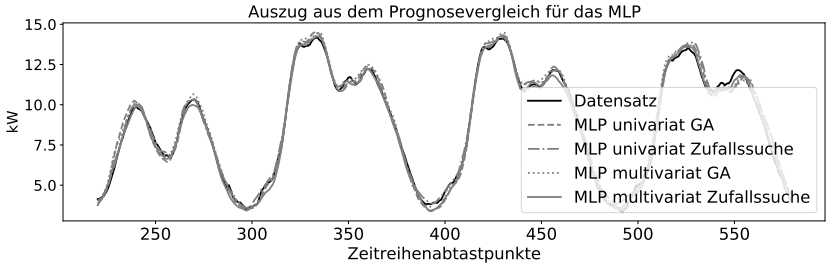


Bild 11: Vergleich der Prognosen der optimierten MLP-Netze anhand eines Ausschnitts der Energiezeitreihe von vier Tagen

5.1 MLP-Architektur

Die Hyperparameter θ werden zur Minimierung der Prognosegüte der Vorhersage mittels MLP-Architektur innerhalb des Suchraums

$$\theta \begin{cases} \text{Verdeckte Schichten} \in [1,\ldots,5] \\ \text{Neuronen} \in [1,\ldots,500] \\ \text{Aktivierungsfunktionen} \in [relu, elu, tanh, selu, sigmoid] \\ \text{Optimierer} \in [rmsprop, adam, sgd, adadelta] \end{cases}$$

variiert, wobei der Suchraum für die verschiedenen Hyperparameteroptimierungsalgorithmen, Zufallssuche und genetischer Algorithmus, identisch gewählt wird.

In Abbildung 11 sind die Prognosen beispielhaft an jeweils einem der optimierten MLP-Netze dargestellt. Die Vorhersagen stimmen zumeist gut mit der Zielzeitreihe überein. Abweichungen der Zeitreihenprognose werden vor allem an den lokalen Minima und Maxima der Energiezeitreihe deutlich.

Zur Abschätzung der Performance der Algorithmen der Hyperparameteroptimierung sind in Tabelle 1 die Prognosegüte anhand verschiedener Fehlermaße sowie die Berechnungszeit aufgetragen. Dabei ist der Durchschnittswert von drei verschiedenen Optimierungsdurchläufen aufgeführt.

Tabelle 1: Mittelwerte der Ergebnisse der unterschiedlichen Hyperparameteroptimierungen für die MLP-Netze

	Univariat		Multivariat	
Methode	Zufallssuche	gen. Algorithmus	Zufallssuche	gen. Algorithmus
RMSE	0,307	0,304	0,3285	0,327
MAPE	3,71	3,69	4,005	3,96
ME	−0,038	−0,033	−0,047	−0,053
Zeit (mm:ss)	45 : 06	33 : 03	56 : 35	52 : 02

Vergleicht man die unterschiedlichen Fehlermaße zunächst für die univariate Zeitreihenprognose, stellt man fest, dass die Fehlerwerte in der gleichen Größenordnung liegen. Dies spricht dafür, dass die beiden Optimierungsmethoden die Umgebung des Minimums des Gütegebirges finden. Es fällt dabei jedoch auf, dass die Vorhersagen der Netze optimiert mit dem genetischen Algorithmus jeweils Verbesserungen zeigen. Der Vorteil der genetischen Algorithmen wird beim Vergleich der benötigten Rechenzeit deutlich. Durch die Verwendung genetischer Algorithmen konnte die Optimierungszeit bei besserer Prognosegüte um mehr als 25 % reduziert werden.

Die Optimierungsergebnisse der multivariaten Zeitreihenvorhersage zeigen ein vergleichbares Verhalten. Auch hier sind die Absolutwerte der Fehlermaße *RMSE* und *MAPE* für den verwendeten genetischen Algorithmus im Vergleich zur Zufallssuche geringer. Der Absolutwert des mittleren Fehlers *ME* ist für die Vorhersage des Netzes optimiert mit dem genetisches Algorithmus minimal größer. Die Zeitersparnis durch Verwendung des genetischen Algorithmus beläuft sich auf ca. 8 %.

Bereits bei der Hyperparameteroptimierung der MLP-Architektur wird deutlich: Beide Algorithmen finden die Umgebung des Minimums der Hyperparameter, jedoch scheint der genetische Algorithmus im Vergleich zur Zufallssuche näher am Minimum zu liegen bzw. stabil zu konvergieren.

5.2 TCN-Architektur

Die Hyperparameter θ werden zur Minimierung der Prognosegüte der Vorhersage im folgenden Suchraum

$$\theta \begin{cases} \text{Filteranzahl} \in [1,\ldots,256] \\ \text{Filtergröße} \in [1,\ldots,16] \\ \text{Anzahl der Stapel} \in [1,\ldots,5] \\ \text{Dilatation} \in [[1,2],[1,2,4],[1,2,4,8],[1,2,4,8,16]] \\ \text{Aktivierungsfunktionen} \in [relu, elu, tanh, selu, sigmoid] \\ \text{Optimierer} \in [rmsprop, adam, sgd, adadelta] \end{cases}$$

verändert. Für die beiden Hyperparameteroptimierungsalgorithmen Zufallssuche und genetischer Algorithmus, wird auch für den Fall der TCN-Architektur der Suchraum konstant gehalten.

Zum Vergleich der Prognosen der unterschiedlich optimierten Netze sind die Vorhersagen beispielhaft an einem trainierten Netz in Abbildung 12 dargestellt. Unterschiede und Abweichungen der Vorhersagen untereinander stellt man vor allem an den lokalen Maxima und Minima der Energiezeitreihe fest.

Die Qualität der Algorithmen der Hyperparameteroptimierung kann durch die verschiedenen Fehlermaße der Prognosegüte sowie die Berechnungszeit aufgetragen in Tabelle 2 abgeschätzt werden. Wie schon für die MLP-Architektur sind die Durchschnittswerte von drei unterschiedlichen Optimierungsdurchläufen angegeben.

Sowohl bei der univariaten als auch bei der multivariaten Zeitreihenprognose stellt man vergleichbare Abhängigkeiten fest. Die Fehlermaße *RMSE* und *MAPE* der Prognose liegen für die Netze optimiert nach der Zufallssuche wie auch nach dem genetischen Algorithmus in einer gleichen Größenordnung, wobei die Fehlerwerte jeweils für die Netze optimiert nach dem genetischen Algorithmus geringer sind. Zunächst scheint der *ME* ein konträres Bild zu vermitteln. Die Absolutwerte der Fehlermaße der Netze optimiert nach der

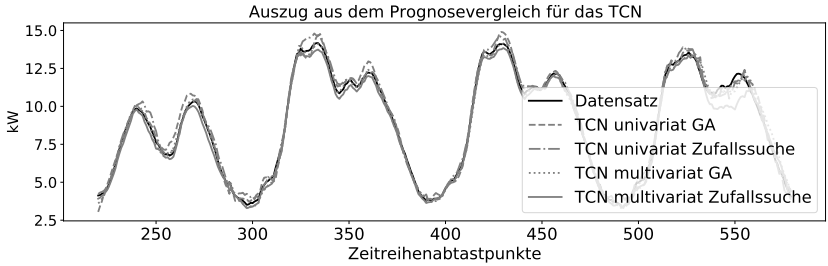


Bild 12: Vergleich der Prognosen der optimierten TCN-Netze anhand eines Ausschnitts der Energiezeitreihe von vier Tagen

Zufallssuche sind hier geringer. Dies lässt sich jedoch durch den Aufbau des Fehlermaß *ME* erklären. Anstatt die absoluten Abweichungen bzw. die quadrierten Werte zu mitteln, werden beim *ME* die Fehler vorzeichenbehaftet gemittelt. Dadurch können sich negative sowie positive Fehlerwerte ausgleichen. Folglich können rauschbehaftete Vorhersagen einen geringeren mittleren Fehler im Vergleich zu Vorhersagen mit einem Offset aufweisen. Gerade der Ausschnitt der Zeitreihe (siehe Abbildung 12) zeigt, dass die univariate Zeitreihenvorhersage optimiert nach dem genetischen Algorithmus die Zielzeitreihe überschätzt (TCN univariat GA). Durch den Offset entsteht ein vergleichsweise großer mittlerer Fehler. Gleiches gilt für die multivariate Zeitreihenvorhersage optimiert nach dem genetischen Algorithmus (TCN multivariat GA). Hingegen weisen die Prognosen mit der Optimierung nach der Zufallssuche keinen klaren Offset auf. Man kann eine Überschätzung und auch Unterschätzung der Zielzeitreihe feststellen, sodass der mittlere Fehler im Vergleich geringer ist.

Neben der Prognosegüte spielt auch die benötigte Rechenzeit der Hyperparameteroptimierung eine entscheidende Rolle. Durch die Verwendung des genetischen Algorithmus kann die Optimierungszeit bei der univariaten Zeitreihenprognose um ca. 25 % und bei der multivariaten Zeitreihenprognose um mehr als 55 % reduziert werden.

Auch anhand der TCN-Architektur konnte damit gezeigt werden: Durch die Verwendung eines genetischen Algorithmus zur Zeitreihenvorhersage kann die Prognosegüte in Bezug auf *RMSE* und *MAPE* bei gleichzeitiger signifikanter Reduktion der Optimierungszeit erhöht werden.

Tabelle 2: Mittelwerte der Ergebnisse der unterschiedlichen Hyperparameteroptimierungen für die TCN-Netze

	Univariat		Multivariat	
Methode	Zufallssuche	gen. Algorithmus	Zufallssuche	gen. Algorithmus
RMSE	0,464	0,443	0,497	0,493
MAPE	5,8	5,3	5,83	5,56
ME	−0,16	0,22	−0,056	0,195
Zeit (mm:ss)	2196 : 36	1647 : 11	2124 : 42	948 : 36

6 Zusammenfassung und Ausblick

Beim Entwickeln und Trainieren von Prognosemodellen basierend auf KNN ist ein entscheidender Faktor die Optimierung der Hyperparameter um leistungsfähige und präzise Modelle zu generieren. Bestehende automatisierte Verfahren zur Hyperparameteroptimierung verwenden zeitaufwändige Raster- oder Zufallssuchen. Speziell bei der Zufallssuche besteht das Risiko lediglich des Auffindens eines lokalen Extrempunktes. Demgegenüber wird bei der Rastersuche zwar der Suchraum größtmöglich abgedeckt, jedoch überschreitet die Optimierungszeit schnell ein akzeptables Maß bei einer steigenden Anzahl an Hyperparametern. Der in diesem Beitrag durchgeführte Vergleich der Methoden zur Hyperparameteroptimierung für KNN zur Energielastprognose hat gezeigt, dass durch die Verwendung von genetischen Algorithmen die Prognosegüte bei gleichzeitiger Reduzierung der Optimierungszeit gesteigert wird. Durch die Verwendung von Evolutionsprinzipien zur Optimierung der Hyperparameter erfolgt durch den Algorithmus eine Entwicklung in der Suchraumabdeckung über die Generationen hinweg, hin zu performanten Bereichen im Suchraum, ohne dabei die Vorteile der Zufallssuche zu verwerfen. In der dargestellten Anwendung der Energielastprognose trägt der genetische Algorithmus dazu bei, die bestmögliche Architektur bzw. Hyperparameterauswahl weitestgehend autonom zu ermitteln und somit die Modelle schneller, effektiver und effizienter zu trainieren. Damit kann dieses Verfahren einen wichtigen Beitrag

leisten hinsichtlich der Reproduzierbar-/ Vergleichbarkeit von Modellen und wissenschaftlichen Untersuchungen. Modellansätze basierend auf KNN können nur adäquat miteinander verglichen werden, wenn diese ein gleiches Maß an Feinabstimmung erhalten. Eine stabile, schnelle und zuverlässige Methode zur automatisierten Hyperparametereinstellung würde also nicht nur die Optimierung erleichtern und die Performance der Modelle steigern, sondern könnte auch durch eine einfachere Handhabbarkeit zur weiteren Verbreitung solcher Anwendungen führen.

Anknüpfungspunkte an die vorliegende Forschungsarbeit bestehen in der Ausweitung der Untersuchung auf weitere und größere Netzarchitekturen wie z.B. Ensemble Learning Verfahren und die Anwendung auf komplexere Aufgabenstellungen mit einer größeren Anzahl exogener Einflüsse. Eine Weiterentwicklung des genetischen Algorithmus könnte daraus bestehen, die trainierte und sortierte Anfangspopulation in Spezies zu unterteilen. Eine Spezie vereint dabei Individuen mit ähnlichen genetischen Eigenschaften und ähnlichen Verlustwerten. So könnte verhindert werden, dass performante Individuen die Population dominieren und gleichzeitig erhalten weniger performante eine längere Entwicklungszeit, da sie nur mit genetisch und leistungsfähig ähnlichen Individuen verglichen werden.

Literatur

[1] O. D. Doleski. „Herausforderung Utility 4.0.“. Wiesbaden: Springer Fachmedien Wiesbaden. 2017.

[2] I. Goodfellow, A. Courville und Y. Bengio. „Deep Learning: Das umfassende Handbuch: Grundlagen, aktuelle Verfahren und Algorithmen, neue Forschungsansätze“. Frechen: Verlags GmbH & Co. KG. 1. Auflage. 2018.

[3] F. Hutter, L. Kotthof und J. Vanschoren. „Automated machine learning: Methods, systems, challenges“. Springer International Publishing. 2019.

[4] T. Yu und H. Zhu. „Hyperparameter Optimization: A Review of Algorithms and Applications“. *arXiv-preprint* 2020. arXiv:2003.05689

[5] L. Yang und A. Shami. „On hyperparameter optimization of machine learning algorithms: Theory and practice“. In: *Neurocomputing*, Band 415, S. 295-316. 2020. https://doi.org/10.1016/j.neucom.2020.07.061

[6] I. Loshchilov und F. Hutter. „CMA-ES for Hyperparameter Optimization of Deep Neural Networks“. *arXiv-preprint* 2016. arXiv:1604.07269

[7] X. Cui, W. Zhang, Z. Tüske und M. Picheny. „Evolutionary Stochastic Gradient Descent for Optimization of Deep Neural Networks“. *arXiv-preprint* 2018. arXiv:1810.06773

[8] S. Raychaudhuri. „Introduction to Monte Carlo simulation“. In: *2018 Winter Simulation Conference*, S. 91-100. 2008. https://doi.org/10.1109/WSC.2008.4736059

[9] N. M. Aszemi and P. D. D. Dominic. „Hyperparameter Optimization in Convolutional Neural Network using Genetic Algorithms“. In: *International Journal of Advanced Computer Science and Applications*, Band 10, Nr. 6, S. 269-278. 2019. http://dx.doi.org/10.14569/IJACSA.2019.0100638

[10] K. O. Stanley und R. Miikkulainen. „Evolving neural networks through augmenting topologies“. In: *Evolutionary Computation*, Band 10, Nr. 2, S. 99-127. 2002. https://doi.org/10.1162/106365602320169811

[11] S. Raschka. „Machine Learning mit Python: Das Praxis-Handbuch für Data Science, Predictive Analytics und Deep Learning“. Frechen: MITP. 1. Auflage. 2017.

[12] S. Bai, J. Z. Kolter und V. Koltun. „An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling“. *arXiv-preprint* 2018. arXiv:1803.01271v2

[13] G. D. Rey und K. F. Wender. „Neuronale Netze: Eine Einführung in die Grundlagen, Anwendungen und Datenauswertung“. Bern: Huber. 2. Auflage. 2011.

[14] M. Abadi et al. „TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems“. https://www.tensorflow.org/. 2015.

[15] „Autonomio Talos“. http://github.com/autonomio/talos. 2020.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever und R. Salakhutdinov. „Dropout: a simple way to prevent neural networks from overfitting“. In: *The Journal of Machine Learning Research*, Band 15, Nr. 1, S. 1929-1958. 2014. https://doi.org/10.5555/2627435.2670313

[17] P. Taylor und R. Lewontin. „The Genotype/Phenotype Distinction“. In: *The Stanford Encyclopedia of Philosophy*, E. N. Zalta (ed.). 2021. https://plato.stanford.edu/archives/sum2021/entries/genotype-phenotype/

Assistenzsystem zur Qualitätssicherung von IoT-Geräten basierend auf AutoML und SHAP

Jan Ewerszumrode, Marvin Schöne,
Stephan Godt, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Schulstraße 10, 33330 Gütersloh
E-Mail: {jan.ewerszumrode, marvin.schoene,
stephan.godt, martin.kohlhase}@fh-bielefeld.de

Kurzfassung

Die Qualitätssicherung stellt einen wesentlichen Bestandteil der Produktentwicklung und Produktion dar, bei der die Potenziale der Daten aus IoT-Geräten bislang wenig Beachtung finden. IoT-Geräte ermöglichen eine Erfassung der tatsächlichen Gerätenutzung sowie auftretender Fehlerfälle, die in Summe als IoT-Gerätenutzungsdaten bezeichnet werden können. In diesem Beitrag wird ein Konzept und die Evaluation eines KI-basierten Assistenzsystems zur verbesserten Qualitätssicherung basierend auf IoT-Gerätenutzungsdaten vorgestellt. Das Konzept vereint eine kontinuierliche Fehlerüberwachung mittels deskriptiver Datenanalysen, eine automatisierte Modellbildung zum Erlernen von Zusammenhängen zwischen Gerätenutzung und auftretenden Fehlern mittels AutoML, und die Modellinterpretation mittels Shapley-Werten zur Bereitstellung hypothetischer Ursachen. Die Evaluation des Konzepts erfolgt anhand realer IoT-Gerätenutzungsdaten von über 40 Tsd. vernetzten Waschmaschinen. Als Ergebnis der Evaluation konnte eine zuvor unbekannte hypothetische Ursache für einen relevanten Fehlerfall auf Grundlage der Gerätenutzung identifiziert werden. Das Assistenzsystem unterstützt somit Domänenexpert:Innen des Qualitätsmanagements bei der explorativen Untersuchung von Kausalitäten zwischen Nutzung und Fehlern, wodurch

sich Verbesserungsmaßnahmen in Bezug auf die IoT-Geräte ableiten lassen können.

1 Einleitung

Zur Erfüllung bzw. Einhaltung aller Anforderungen an ein Produkt ist das Qualitätsmanagement (QM) bzw. die Qualitätssicherung (QS) einer der wesentlichen Bestandteile der Produktion. Quantitative Methoden der QS, wie z.B. *Six Sigma*, stützen sich dabei auf die Produktionsdaten aus MES- und ERP-Systemen. Hierbei wird jedoch der Großteil des Lebenszyklus von Produkten nicht berücksichtigt: Die tatsächliche Nutzung beim Endkunden. Veranschaulichen lässt sich dies anhand von Bild 1, in dem der Status quo und die Vision einer QS von IoT-Geräten gegenübergestellt werden. Neben den Produktionsdaten bieten sich Nutzungs- und Fehlerdaten der gefertigten Produkte bzw. Geräte als zusätzliche Datenquelle für die QS an. Das Internet of Things (IoT) ermöglicht es, diese Daten direkt aus den IoT-Geräten zu erheben. Ein Beispiel für IoT-Geräte sind vernetzte Waschmaschinen, die Daten zu den gewählten Waschprogrammen der Endkunden sowie aus ihrer geräte-internen Fehlerdiagnose erfassen (vgl. Bild 1b) und 1c)). Diese Daten lassen sich als IoT-Gerätenutzungsdaten bezeichnen, mit deren Hilfe eine kundenorientierte Qualitätssicherung (vgl. Bild 1a)) nach den Grundsätzen des Total-Quality-Managements erreicht werden kann.

Mithilfe dieser zusätzlichen Gerätenutzungsinformationen können hypothetische Ursachen für einen auftretenden Fehlerfall identifiziert werden. Die Domänenexpert:Innen des QM werden dabei besser und zuverlässiger unterstützt, unbekannte Kausalitäten aufzudecken, bekannte Kausalitäten zu bestätigen und neues Domänenwissen zu generieren. Der Vergleich einer QS auf Basis von IoT-Gerätenutzungsdaten mit dem Status quo zeigt jedoch Unterschiede in den Eigenschaften der Datenquellen und -basen (vgl. Bild 1c) und 1d)). Diese äußern sich in Form großer, heterogener und hochdimensionaler Datenmengen in einem kontinuierlichen Datenstrom. Durch Verfahren der *künstlichen Intelligenz* (KI) bzw. des *Machine Learnings* (ML) können diese Datenmengen ausgewertet werden (vgl. Bild 1e)).

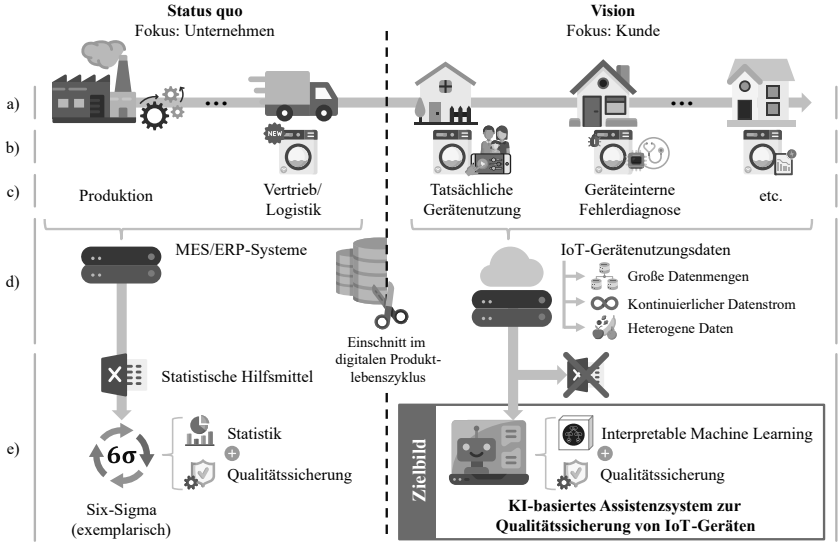


Bild 1: Status quo und Vision einer QS bzgl. des a) Lebenszyklus von b) IoT-Geräten als zusätzliche c) Datenquelle. Anhand von IoT-Gerätenutzungsdaten als d) Datenbasis ergeben sich neue Potenziale zur e) Auswertung mittels KI und IML.

Infolge der Anforderungen an Präzision in der QS werden nachvollziehbare Entscheidungen in der Datenauswertung benötigt, wodurch sich Verfahren des *Interpretable Machine Learnings* (IML) anbieten, um die zumeist intransparenten Entscheidungen des ML erklärbar zu machen. Vergleichbare technische Systeme in [1, 2, 3] lösten bereits ähnliche Aufgaben zur Generierung neuen Wissens mittels IML-Verfahren. Eine derartige Anwendung im Bereich der QS für IoT-Geräte steht jedoch noch aus. Das Potenzial von IML wird zudem im Gesundheitswesen ersichtlich [4, 5], wo ebenfalls ein hohes Maß an Präzision und Sorgfalt gefordert ist.

In diesem Beitrag wird ein Konzept für ein Assistenzsystem zur QS basierend auf IoT-Gerätenutzungsdaten und Verfahren des ML sowie IML vorgestellt, welches anhand realer Daten vernetzter Waschmaschinen evaluiert wird. Ziel des Assistenzsystems ist es, Domänenexpert:Innen aus dem QM bei der explorativen Untersuchung von Fehlerfällen, die während der Gerätenutzung auftreten, zu unterstützen. In den Daten enthaltene Zusammenhänge zwischen

Gerätenutzung und auftretenden Fehlerfällen werden in ML-Modelle mittels *Automated Machine Learning* (AutoML) verdichtet. Anschließend werden die erlernten Zusammenhänge zur Bereitstellung neuer hypothetischer Fehlerursachen durch SHAP aus den ML-Modellen extrahiert und grafisch dargestellt.

2 Theoretische Grundlagen & Stand der Technik

Wesentlich für das in diesem Beitrag vorgestellte Konzept sind das AutoML zur Approximation der Funktion $f : \mathscr{X} \rightarrow \mathscr{Y}$ zwischen Gerätenutzung $\mathscr{X}$ und Fehlerfall $\mathscr{Y}$ der IoT-Geräte, sowie das IML zur Untersuchung des erlernten Zusammenhangs von f.

Automated Machine Learning (AutoML). Unter AutoML ist ein Prozess für die automatisierte Entwicklung von ML-Pipelines zur Approximation von f aus einem dedizierten Datensatz $\mathscr{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ zu verstehen. Der Datensatz besteht aus N gelabelten Instanzen, für die $\mathbf{x}^{(i)} \in \mathscr{X}$ und $y^{(i)} \in \mathscr{Y}$ gilt. Der Prozess beinhaltet die Auswahl, Kombination sowie Parametrisierung von ML-Algorithmen. Formalisieren lässt sich dieser Prozess als Optimierungsproblem, das als *Combined Algorithm Selection and Hyperparameter optimization* (CASH) bezeichnet wird [6]:

$$A^*_{\lambda^*} \in \underset{A^{(k)} \in \mathscr{A}, \lambda \in \Lambda^{(k)}}{\operatorname{argmin}} \; E[L(A^{(k)}_{\lambda}, \mathscr{D}_{\text{train}}, \mathscr{D}_{\text{valid}})] \tag{1}$$

mit λ als Hyperparameter aus den Hyperparameterräumen $\Lambda^{(1)}, ..., \Lambda^{(K)}$ und den zugehörigen ML-Algorithmen $\mathscr{A} = \{A^{(1)}, ..., A^{(K)}\}$ für $k = 1, ..., K$. Zudem bezeichnet $L(A^{(k)}_{\lambda}, \mathscr{D}_{\text{train}}, \mathscr{D}_{\text{valid}})$ die Fehlerfunktion für einen auf den Trainingsdatensatz $\mathscr{D}_{\text{train}} \subset \mathscr{D}$ angewendeten und am Validierungsdatensatz $\mathscr{D}_{\text{valid}} \subset \mathscr{D}$ getesteten Algorithmus $A^{(k)}_{\lambda}$. Eine gängige Methode zur Lösung des CASH-Problems besteht darin, die Algorithmenauswahl als zusätzlichen Hyperparameter zu betrachten, wodurch etablierte Hyperparameteroptimierungen verwendet werden können, wie z.B. SMAC [7], *Hyperband* [8] und BOHB [9]. Unter Berücksichtigung der natürlichen Hierarchie bzgl. Auswahl und Parametrisierung von ML-Algorithmen ($A^{(k)}$ bedingt $\Lambda^{(k)}$) stellen hierarchische Suchansätze, wie z.B. *ML-Plan* [10], eine

weitere Lösung des CASH-Problems dar. Des Weiteren bieten sich Verfahren des *Reinforcement Learnings* an, in denen der Agent geeignete $A^{(k)}$ und λ auswählt [11].

Interpretable Machine Learning (IML). IML bezeichnet Methoden, die das Verhalten und die Vorhersagen von ML-Modellen für den Menschen verständlich machen [12]. Während sich die lokale Interpretierbarkeit auf die Erklärung einzelner Prädiktionen bezieht, ist unter einer globalen Interpretierbarkeit die Erklärung des gesamten Modellverhaltens zu verstehen. Wesentlich für die Interpretierbarkeit von ML-Modellen ist der Einfluss eingehender Merkmale auf die resultierenden Prädiktionen [13]. Gegenüber inhärent interpretierbaren ML-Modellen (z.B. *Decision Trees*) eignen sich besonders modellagnostische post-hoc Methoden, welche die Interpretation vom ML-Modell separieren und eine schwerpunktmäßige Betrachtung der Modellgenauigkeit ermöglichen. Shapley-Werte bieten dazu aufgrund hoher Übereinstimmungen mit der menschlichen Intuition und einer fundierten theoretischen Grundlage [14] gegenüber vergleichbaren Methoden, wie LIME [15] oder PFI [20], ein großes Potenzial. Ausgehend von einer lokalen Prädiktion $\hat{f}(\mathbf{x}^{(i)})$ lassen sich die Shapley-Werte $\phi_j^{(i)}$ für jedes Merkmal j über eine gewichtete Summe darstellen, die den Einfluss jedes zum ML-Modell hinzugefügten Merkmals wiedergibt, gemittelt über alle Kombinationen verfügbarer Merkmale [14]:

$$\phi_j^{(i)} = \sum_{\mathscr{S} \subseteq \mathscr{S}_{\text{all}} \setminus \{j\}} \frac{|\mathscr{S}|!(M - |\mathscr{S}| - 1)!}{M!} \cdot (\hat{f}_{\mathbf{x}}(\mathscr{S} \cup \{j\}) - \hat{f}_{\mathbf{x}}(\mathscr{S})) \qquad (2)$$

mit der Merkmalsanzahl M, der Menge aller Merkmale $\mathscr{S}_{\text{all}}$, und der Prädiktion $\hat{f}_{\mathbf{x}}(\mathscr{S}) = E[\hat{f}(\mathbf{x}^{(i)}) | \mathbf{x}_{\mathscr{S}}]$ einer ausgewählten und auf die Teilmenge $\mathscr{S}$ beschränkten Kombination an Merkmalen $\mathbf{x}_{\mathscr{S}}$ des Eingabevektors. Des Weiteren besitzen Shapley-Werte folgende Eigenschaften: a) Die Summe der Shapley-Werte aller Merkmale ist gleich der Differenz aus $\hat{f}(\mathbf{x}^{(i)})$ minus der mittleren Prädiktion $E[\hat{f}(\mathbf{x})]$ einer zufälligen Instanz $\mathbf{x} \in \mathscr{X}$; b) $\phi_j^{(i)} = 0$, wenn das Merkmal j keinen Einfluss auf die Prädiktion hat; c) wenn die Werte zweier Merkmale über alle $\mathscr{S}$ hinweg eine symmetrische Auswirkung haben, ergeben sich die gleichen Shapley-Werte und d) ihre lokalen Einflüsse sind über $\mathbf{x}^{(i)}$ hinweg additiv [13]. Die exakte Berechnung der Shapley-Werte äußert sich jedoch aufgrund 2^M möglicher Teilmengen für $\mathscr{S}$ als NP-schwer. Sampling-basierte Ap-

proximationen der Shapley-Werte, wie *Kernel*-SHAP [14], ermöglichen zwar die Berechnung lokaler Interpretationen, scheitern jedoch an einer globalen Interpretation für große Datensätze [16]. Die Anpassung von *Kernel*-SHAP für baumartige ML-Modelle führt zu *Tree*-SHAP und ermöglicht eine Reduktion der ursprünglich exponentiellen auf eine polynomielle Berechnungszeit. Die zugrundeliegende Idee von *Tree*-SHAP besteht darin, den Anteil aller möglichen Teilmengen in jedes der Blätter des Baums rekursiv zu verfolgen. Da die hierdurch erzeugten Shapley-Werte auf bedingte Erwartungswerte beruhen, ist eine gesonderte Behandlung abhängiger Merkmale erforderlich, welche die Interpretation verfälschen können (z.B. die Schätzung von $\phi_j^{(i)} \neq 0$ für Merkmale ohne Einfluss) [12]. Unter Berücksichtigung eines Backgrounddatensatzes lässt sich diese Abhängigkeit nach den Regeln der zufälligen Inferenz beheben, wodurch die zuvor aufgeführten Eigenschaften weiterhin gültig sind und sich zusätzlich eine Berechnungszeit proportional zur Größe des Backgrounddatensatzes ergibt [17]. Aufgrund der Komplexitätsreduktion ermöglicht *Tree*-SHAP eine globale Interpretierbarkeit basierend auf vielen lokalen Interpretationen.

3 Konzept des KI-basierten Assistenzsystems

Der Nutzen des KI-basierten Assistenzsystems liegt in der Bereitstellung hypothetischer Ursachen für Fehlerfälle $F\#_l$, die aufgrund der Gerätenutzung auftreten. Diese Ursachen sollen den Domänenexpert:Innen helfen, unbekannte Kausalitäten zu identifizieren und bekannte Kausalitäten zu bestätigen. Die Anzahl und Art der Fehlerfälle $F\#_l$ werden von dem jeweiligen Fehlerdiagnosesystem der IoT-Geräte vorgegeben, wie z.B. *Kurzschluss im Gerät.* Das Konzept dieses KI-basierten Assistenzsystems zur QS von IoT-Geräten ist in Bild 2 dargestellt. Als Eingabe in das System dienen IoT-Gerätenutzungsdaten (bestehend aus Nutzungs- und Fehlerdaten der IoT-Geräte), während als Ausgabe Diagramme zur Erklärung des erlernten Zusammenhangs zwischen Gerätenutzung und Fehlerfällen dienen. Hierbei vereinigt das Konzept die drei Schritte 1) einer kontinuierlichen Fehlerüberwachung mittels Methoden der deskriptiven Datenanalyse, namens QS-Watchdog, 2) einer automatisierten Modellbildung zur Klassifikation eines Fehlerfalls anhand zugehöriger Nutzungsdaten mittels

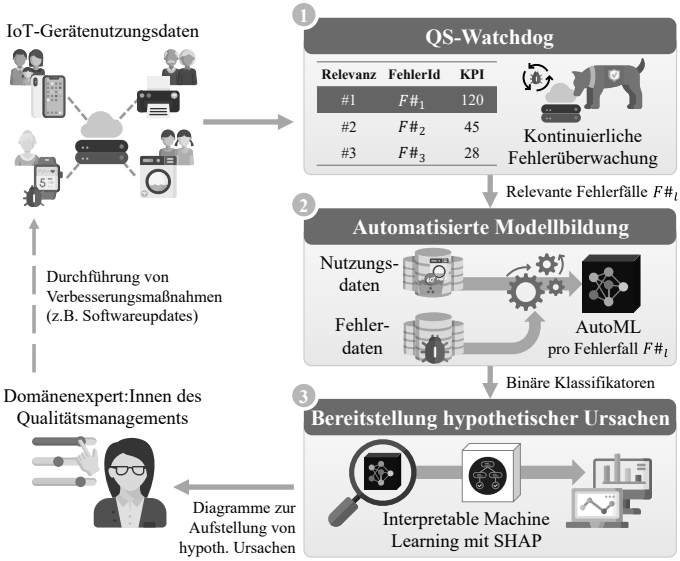


Bild 2: Assistenzsystem zur QS mittels IoT-Gerätenutzungsdaten als Eingabe, einer 1) kontinuierlichen Fehlerüberwachung, 2) automatisierten Modellbildung mittels AutoML und 3) Bereitstellung hypothetischer Ursachen durch IML für Domänenexpert:Innen des QM

AutoML und 3) die globale Interpretation zuvor trainierter Modelle mittels Shapley-Werten in Form von aussagekräftigen Diagrammen.

Qualitätssicherungs-Watchdog (QS-Watchdog). Aufgrund eines starken Ungleichgewichts zwischen intakten und defekten Geräten, bedarf es einer Selektion von Fehlerfällen, um eine stichhaltige Menge an Labels in Form von betroffenen Geräten für die weiteren Schritte zu gewährleisten. Gleichzeitig ist die Beobachtung von Trends bei relevanten Fehlerfällen, die den Kundennutzen beeinträchtigen, für die Domänenexpert:Innen des QM von Interesse, was den Einsatz einer kontinuierlichen Fehlerüberwachung der IoT-Geräte notwendig macht. Hierfür ist es erforderlich, eine Kennzahl zu definieren, die relevante von nicht-relevanten Fehlerfällen unterscheidet und somit eine Priorisierung ermöglicht. Aus dem Bereich der Zuverlässigkeitsanalyse bieten sich Kennzahlen an, welche die Häufigkeit relevanter Fehler mit der Betriebszeit in Bezug setzen. Eine der hierbei wichtigsten Kennzahlen stellt die Fehlerrate $\lambda(t)$ dar. Für

praktische Anwendungen kann $\lambda(t) = \lambda$ angenommen werden, wodurch sich λ über $\hat{\lambda} = n/T$ schätzen lässt, wobei T die kumulierte Betriebszeit und n die absolute Anzahl an Fehlern ist [18]. Als Kennzahl für den QS-Watchdog ergibt sich demnach die geschätzte Fehlerrate $\hat{\lambda}_{F\#_l}$ pro Fehlerfall $F\#_l$ für ein zuvor festgelegtes Zeitintervall:

$$\hat{\lambda}_{F\#_l} = n_{F\#_l}/T. \tag{3}$$

Wichtig zu erwähnen ist, dass sich n auf die Anzahl repräsentativer Fehler bezieht, die dadurch definiert werden, dass sie auf eine Unterbrechung der Gerätefunktionalität (z.B. Stopp eines Waschprogramms) folgen müssen.

Automatisierte Modellbildung. Für eine präzise Erklärung des Zusammenhangs zwischen Gerätenutzung und Fehlerfällen basieren die zugrundeliegenden ML-Modelle auf einer pro Fehlerfall $F\#_l$ konzentrierten binären Klassifikation $y_{F\#_l} \in \{0 := F\#_l \text{ trat nicht auf}, 1 := F\#_l \text{ trat auf}\}$. Anstelle eines universellen Klassifikators ist es somit erforderlich, mehrere binäre Klassifikatoren je Fehlerfall zu trainieren. Um das Training der Klassifikatoren dennoch domänenunabhängig und automatisiert zu gestalten, erfolgt dieser Schritt mit Hilfe von AutoML. In diesem Beitrag wird das AutoML von Databricks [19] verwendet, welches die folgenden ML-Algorithmen beinhaltet: *Logistic Regression*, *Decision Trees*, *Random Forests* [20], *XGBoost* (XGB) [21] und *LightGBM* (LGBM) [22]. Zusätzlich ergibt sich durch die Beschränkung auf binäre Klassifikationen eine deutlich verringerte Komplexität des Suchraums nach (1), die sich positiv auf die angestrebte Generalisierbarkeit auswirkt [23]. Zur Evaluation der AutoML-Modelle werden die IoT-Gerätenutzungsdaten in Testdaten $\mathscr{D}_{\text{test}}$ und Trainingsdaten aufgeteilt (25%, 75%), wobei die Trainingsdaten jeweils vom AutoML in separate Trainingsmengen $\mathscr{D}_{\text{train}}$ und Validierungsmengen $\mathscr{D}_{\text{valid}}$ stratifiziert unterteilt (75%, 25%) werden. In Anbetracht der stark ungleich verteilten Klassen findet die Beurteilung erfolgreicher Klassifikatoren mittels des *F1-Scores* F_1 für $\mathscr{D}_{\text{test}}$ statt. Ab einem $F_1 > 0.9$ für $\mathscr{D}_{\text{test}}$ wird dem Klassifikator eine ausreichende Generalisierbarkeit des Datensatzes für eine anschließende Interpretation zugesprochen. Andernfalls wird der Klassifikator für weitere Interpretationen zurückgewiesen.

Bevor AutoML angewendet werden kann, müssen die binären Klassen aus der Gerätemenge extrahiert werden, die keine Fehlerfälle beinhalten und somit das Normalverhalten charakterisieren, sowie aus defekten Geräten bzgl. des betrachteten Fehlerfalls bestehen. Aufgrund des Ungleichgewichts zwischen intakten und defekten Geräten, bieten sich Under- und Oversampling-Verfahren zur Bereinigung der ungleichen Verteilung an. Während beim Undersampling eine zufällige Untermenge aus der Klasse intakter Geräte ausgewählt wird, werden beim Oversampling neue synthetische Daten aus der Klasse defekter Geräte, z.B. mittels SMOTE bzw. SMOTE-NC [25] generiert, wodurch sich bei beiden Verfahren ein ausbalancierter Trainingsdatensatz ergibt. Um möglichst signifikante Merkmale zu verwenden, werden diese mittels FRESH [26] anhand von p-Werten auf ihre Signifikanz gegenüber der Klasse getestet und selektiert. Angesichts der Betrachtung von Geräten über die gesamte Betriebszeit kann durch eine gesonderte Betrachtung der Nutzungsverlauf defekter Geräte bis zum *ersten relevanten Fehlerfall* hergestellt werden. Hierdurch ließen sich bspw. Nutzungsmuster in den Anfangsphasen des Gerätes bis zur ersten Reparatur identifizieren.

Bereitstellung hypothetischer Ursachen. Zur Interpretation des erlernten Zusammenhangs zwischen Nutzung und Fehler für eine Menge an intakten und defekten Geräten bedarf es einer globalen Interpretation des erlernten Modellverhaltens. Dieses lässt sich über den Einfluss eingehender Merkmale auf das Prädiktionsergebnis mittels Shapley-Werten beschreiben und interpretieren. Die Ermittlung der Shapley-Werte erfolgt durch *Tree*-SHAP, wodurch die globale Interpretation über eine Vielzahl lokaler Interpretationen der zuvor separierten Testmengen $\mathscr{D}_{\text{test}}$ erfolgt. Da es sich bei der automatisierten Modellbildung um binäre Klassifikatoren handelt, dient hierbei die prädizierte Wahrscheinlichkeit $\hat{f}(\mathbf{x}^{(i)})$ für ein defektes Gerät als Ausgabe. Für einen idealen binären Klassifikator ($F_1 = 1.0$) mit ausbalancierten Daten ergibt sich $E[\hat{f}(\mathbf{x})] = 0.5$. Wie in Bild 3a) schematisch dargestellt, kann $\hat{f}(\mathbf{x}^{(i)})$ für eine lokale Interpretation ausgehend von $E[\hat{f}(\mathbf{x})]$ durch Aufsummieren aller $\phi_j^{(i)}$ über j nachvollzogen werden. Zur Bestimmung des globalen Merkmalseinflusses (*Feature Importance*) I_j eignet sich der Mittelwert der absoluten Shapley-Werte pro

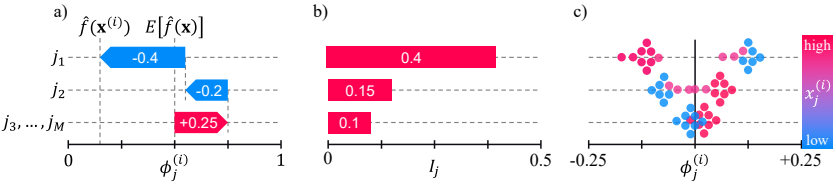


Bild 3: Schematische Diagramme zur Interpretation mittels Shapley-Werten für a) lokale Interpretationen, b) globale Merkmalseinflüsse und c) einer Übersichtsabbildung.

Merkmal:

$$I_j = \sum_{\{i \in \mathbb{N} \mid (\mathbf{x}^{(i)}, y^{(i)}) \in \mathscr{D}_{\text{test}}\}} \frac{|\phi_j^{(i)}|}{|\mathscr{D}_{\text{test}}|} \tag{4}$$

für alle Instanzen aus $\mathscr{D}_{\text{test}}$. Für eine erste globale Interpretation lässt sich I_j, wie in Bild 3b), über ein Balkendiagramm darstellen. In Verbindung mit den tatsächlichen Merkmalswerten $x_j^{(i)}$ zur Gerätenutzung jeder lokalen Interpretation und den zugehörigen Shapley-Werten $\phi_j^{(i)}$, lassen sich globale wechselseitige Beziehungen zwischen der Gerätenutzung und auftretenden Fehlerfällen in einer sogenannten Übersichtsabbildung identifizieren. Diese Übersichtsabbildung ist in Bild 3c) schematisch dargestellt. Hierbei gibt jeder Punkt eine lokale Interpretation pro Merkmal auf der y-Achse wieder, während die Shapley-Werte auf der x-Achse aufgetragen sind und der Merkmalswert über die Farbe der Punktfüllung beschrieben wird. Für einen Überblick über die Verteilung werden überlappende Punkte in Richtung der y-Achse gestapelt. Außerdem wird die Beschreibung dieser Merkmale in allen Diagrammen auf maximal drei beschränkt, da eine für den Menschen gute Erklärung weniger durch eine allumfassende als vielmehr durch eine präzise Erklärung ausgewählter Merkmale erreicht wird [24]. Der Einfluss aller restlichen Merkmale wird über diese Beschränkung hinaus zusammengefasst.

4 Evaluation

Zur Evaluation des Konzepts dienen reale IoT-Gerätenutzungsdaten von vernetzten Waschmaschinen, die im nachfolgenden Abschnitt 4.1 vorgestellt werden. Darauf aufbauend erfolgt in Abschnitt 4.2 eine schrittweise Evaluation der

einzelnen Bestandteile des Konzepts sowie in Abschnitt 4.3 eine Vorstellung und Diskussion der Evaluationsergebnisse.

4.1 Rahmenbedingungen & Versuchsaufbau

Der Evaluationsdatensatz des verwendeten Versuchsaufbaus umfasst reale IoT-Gerätenutzungsdaten von $>$ 40 Tsd. vernetzten Waschmaschinen, die Daten zu $>$ 10 Mio. Waschprogrammen beinhalten und über einen Zeitraum vom August 2019 bis Mai 2021 erhoben wurden. Hinter jedem dieser Waschprogramme verbergen sich ereignisdiskrete Daten, welche sich in Nutzungs-[1] und Fehlerdaten[2] aufteilen lassen. Die Vielzahl dieser Ereignisse führt zu einem Datenstrom mit fortlaufend neu hinzukommenden Geräten, gerätespezifischen Zeitreihen vereinzelter Nutzungs- und Fehlerdaten sowie Inkonsistenzen realer Datenproduzenten. Um diesen Datenstrom effizient zu verwalten und über alle Geräte hinweg vergleichbar zu machen, bietet sich eine aggregierte Sicht der IoT-Gerätenutzungsdaten an [26]. In dieser aggregierten Sicht werden die gerätespezifischen Zeitreihen der Nutzungs- und Fehlerdaten zusammengefasst, sodass jedes Element die bisherige Betriebszeit eines Gerätes beschreibt, inklusive einer Auflistung bereits aufgetretener Fehlerfälle. Ein Gerät wird als intakt bezeichnet, wenn während der bisherigen Betriebszeit keinerlei Fehlerfälle aufgetreten sind. Die Aggregation der Nutzungsdaten erfolgt über die Extraktion etablierter deskriptiver Merkmale[3] aus den dynamischen Variablen eines Waschprogramms. Die Extraktion der booleschen Programmextras erfolgt anhand der aktiven Nutzung dieser Extras im Verhältnis zur Anzahl durchgeführter Waschprogramme pro Gerät[4]. Für jeden Fehlerfall $F\#_l$ wird nach diesem

[1] Nutzungsdaten lassen sich in vier Gruppen clustern: 1) *Geräteeigenschaften* beinhaltet den Gerätetyp und die Softwareversion; 2) *Programmanwahl* beinhaltet das Waschprogramm (z.B. Feinwäsche, etc.), zusätzliche Einstellparameter (Temperatur, Schleuderdrehzahl) und Programmextras (z.B. Stärken/Weichspülen, etc.); 3) *Programmzustand* beinhaltet den Energie-, Wasserverbrauch und die Programmdauer abgeschlossener Waschprogramme; 4) Die *automatische Dosierung* beinhaltet die Dosierungsmenge und den verbleibenden Waschmittelinhalt.

[2] Fehlerdaten beinhalten Informationen zu aufgetretenen Fehlerfällen während eines Waschprogramms, die von der internen Gerätefehlerdiagnose bestimmt werden.

[3] Folgende deskriptive Merkmale werden berechnet: Arithmetischer Mittelwert, Median, Varianz, Standardabweichung, Minimum, Maximum, RMS und Summe aller Werte.

[4] Bsp.: Das Programmextra Stärken/Weichspülen wird zum Merkmal Stärken/Weichspülen%.

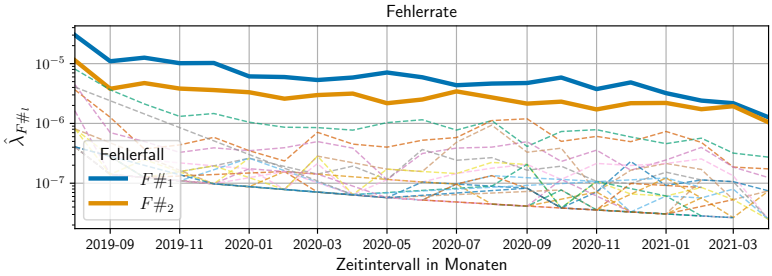


Bild 4: Fehlerrate aller spezifizierten Fehlerfälle pro Monate vom 08.2019 bis zum 04.2021, inklusive der hervorgehobenen Fehlerfälle $F\#_1$ und $F\#_2$.

Schema eine aggregierte Sicht der Nutzungsdaten aus intakten und defekten Geräten erzeugt, woraus sich die entsprechenden Datensätze $\mathscr{D}_{F\#_l}$ ergeben.

4.2 Schrittweise Evaluation des Konzepts

Evaluation des QS-Watchdogs. Mittels der Fehlerdaten des Evaluationsdatensatzes lässt sich die kontinuierliche Fehlerüberwachung für jeden Fehlerfall $F\#_l$ nachbilden, wobei ein Zeitintervall von einem Monat betrachtet wird. Die Fehlerrate $\hat{\lambda}_{F\#_l}$ lässt sich nach (3) bestimmen. In Bild 4 ist die Fehlerrate über den betrachteten Zeitraum pro Fehlerfall $F\#_l$ dargestellt, wobei $\hat{\lambda}_{F\#_l}$ logarithmisch abgebildet wird. Insgesamt ist ein deutlicher Rückgang von $\hat{\lambda}_{F\#_l}$ zu erkennen, welches jedoch auf die anfänglich geringe Betriebszeit weniger IoT-Geräte im Evaluationsdatensatz zurückzuführen ist. Auffällig sind hierbei die Fehlerfälle $F\#_1$ und $F\#_2$, die sich von der Gesamtmenge an Fehlern absetzen, eine demnach hohe Relevanz aufweisen und zur weiteren Betrachtung der automatisierten Modellbildung herangezogen werden.

Evaluation der automatisierten Modellbildung. Basierend auf den identifizierten Fehlerfällen $F\#_1$ und $F\#_2$ des QS-Watchdogs findet im Folgenden die Modellbildung mittels AutoML zur Generierung von $\hat{f} : \mathscr{X} \rightarrow \mathscr{Y}$ anhand der Evaluationsdatensätze $\mathscr{D}_{F\#_1}$ und $\mathscr{D}_{F\#_2}$ statt. Durch die stratifizierte Aufteilung dieser Datensätze ergibt sich für jeden der Fehlerfälle ein Testumfang von > 10 Tsd. Geräte, wobei $F\#_1$ einen Anteil von $4,7\%$ und $F\#_2$ von $2,2\%$ defekter

Tabelle 1: Evaluation der automatisierten Modellbildung anhand der besten AutoML-Runs für alle Datensatzkonfigurationen der Fehlerfälle $F\#_1$ und $F\#_2$ mittels separatem Testdatensatz. Erfolgreiche Runs mit $F_1 > 0.9$ sind fett markiert.

Datensatzkonfiguration					Testergebnisse der besten AutoML-Runs				
Fehlerfall		#S	#B	#nM	Klassif.	F_1	*PPV*	*TPR*	*MMC*
$F\#_1$	$\mathscr{D}_{F\#_{1.1}}$	-	-	126	LGBM	**0.984**	0.998	0.971	0.984
	$\mathscr{D}_{F\#_{1.2}}$	U	-	100	LGBM	0.541	0.374	0.979	0.578
	$\mathscr{D}_{F\#_{1.3}}$	O	-	133	LGBM	**0.987**	0.992	0.981	0.986
	$\mathscr{D}_{F\#_{1.4}}$	-	✓	112	XGB	**0.958**	0.993	0.926	0.957
	$\mathscr{D}_{F\#_{1.5}}$	U	✓	120	LGBM	0.206	0.115	0.984	0.263
	$\mathscr{D}_{F\#_{1.6}}$	O	✓	131	XGB	**0.957**	0.996	0.922	0.956
$F\#_2$	$\mathscr{D}_{F\#_{2.1}}$	-	-	123	LGBM	0.105	0.341	0.062	0.138
	$\mathscr{D}_{F\#_{2.2}}$	U	-	91	LGBM	0.045	0.023	0.712	0.012
	$\mathscr{D}_{F\#_{2.3}}$	O	-	141	LGBM	0.192	0.578	0.115	0.251
	$\mathscr{D}_{F\#_{2.4}}$	-	✓	100	XGB	0.281	0.621	0.181	0.329
	$\mathscr{D}_{F\#_{2.5}}$	U	✓	112	XGB	0.064	0.033	0.854	0.087
	$\mathscr{D}_{F\#_{2.6}}$	O	✓	134	LGBM	0.446	0.723	0.323	0.476

#S-Samplingart: kein Sampling (-), Undersampling (U), Oversampling (O);
#B-Betriebszeit bis Fehler: nein (-), ja (✓); #nM-Merkmalsanzahl

IoT-Geräte aufweist. Zur Adressierung dieses Ungleichgewichts kommen die zusätzlichen Maßnahmen des Under- und Oversampling zum Einsatz. Hinzu kommt die zusätzliche Betrachtung der Gerätebetriebszeit bis zum ersten Fehlerfall, wodurch sich insgesamt sechs verschiedene Datensatzkonfigurationen in Form der aggregierten IoT-Gerätenutzungsdaten ergeben: Kein Sampling, Undersampling, Oversampling jeweils pro gesamter Betriebszeit sowie bis zum ersten Fehlerfall. Aufgrund der Merkmalsauswahl mittels FRESH variiert die Anzahl der Merkmale in jeder Datensatzkonfiguration. Für jede der insgesamt 12 Datensatzkonfiguration wird ein AutoML-Run gestartet, welcher 200 Versuche beinhaltet. Die Evaluationsergebnisse der besten AutoML-Runs mittels der separierten Testmengen sind in Tabelle 1 aufgeführt. Neben F_1 als primäre Bewertungsmetrik werden hier *Precision PPV* und *Recall TPR* als zusätzliche Metriken aufgeführt. Um Fehlinterpretationen aufgrund des starken Klassenungleichgewichts zu vermeiden, wird auf die Evaluation mittels *Accuracy* verzichtet und der *Matthews-Correlation-Coefficient MCC* zur Betrachtung der gesamten Konfusionsmatrix verwendet [27]. Die Evaluation aus

Tabelle 1 zeigt, dass vier Klassifikatoren für den Fehlerfall $F\#_1$ in der Lage sind, eine ausreichende Generalisierbarkeit von $\hat{f}$ für $\mathscr{D}_{F\#_1}$ zu erlernen. Auffällig ist zudem, dass beim Undersampling zwar ein guter *Recall* in allen Fehlerfällen erzielt werden konnte, der jedoch zu Lasten der *Precision* fällt. Für den Fehlerfall $F\#_2$ konnte diese Generalisierbarkeit jedoch nicht erzielt werden, wodurch dieser von einer darauffolgenden Interpretation zurückgewiesen wird. Zur weiteren Evaluation der Bereitstellung hypothetischer Ursachen werden demnach die erfolgreichen Klassifikatoren der Datensatzkonfigurationen $\mathscr{D}_{F\#_{1.1}}$, $\mathscr{D}_{F\#_{1.3}}$, $\mathscr{D}_{F\#_{1.4}}$ und $\mathscr{D}_{F\#_{1.6}}$ für den Fehlerfall $F\#_1$ mittels Verfahren des IML untersucht.

Evaluation der Bereitstellung hypothetischer Ursachen. Zur Bereitstellung hypothetischer Ursachen des Fehlerfalls $F\#_1$ werden die Shapley-Werte der vier erfolgreichen Klassifikatoren mittels *Tree*-SHAP für die zuvor bestimmten Testdaten berechnet. Aufgrund des Klassenungleichgewichts wird ein zusätzliches Undersampling durchgeführt, um die Klassen der intakten Geräte zu reduzieren und eine Fehlinterpretation der Modelle zu vermeiden. Der für die Berechnung verwendete Backgrounddatensatz entspricht hierbei den reduzierten Trainingsdaten der Datensatzkonfigurationen, während für die tatsächliche Schätzung der Shapley-Werte die jeweiligen reduzierten Testdaten verwendet werden. Für diese ausbalancierten Testdaten ergibt sich für die betrachteten Klassifikatoren eine Genauigkeit von ≥ 0.961, die eine stichhaltige Interpretation dieser Shapley-Werte erlaubt.

In Bild 5 sind drei lokale Interpretationen des Klassifikators aus $\mathscr{D}_{F\#_{1.1}}$ für verschiedene Waschmaschinen gegenübergestellt. Der Klassifikator besitzt einen Erwartungswert von $E[\hat{f}(\mathbf{x})] = 0.497$ und gibt für die Waschmaschine in 5a) eine Wahrscheinlichkeit von $\hat{f}(\mathbf{x}^{(1)}) = 0.001$ bzgl. des Fehlerfalls $F\#_1$ aus. Der Shapley-Wert des Merkmals Stärken/Weichspülen% gibt hierbei den größten Einfluss mit einem Merkmalswert von 0% wieder. Genau gegensätzlich verhält es sich mit der Waschmaschine aus 5b), die zu 100% Stärken/Weichspülen verwendet hat und zu $\hat{f}(\mathbf{x}^{(2)}) = 0.999$ als defekt klassifiziert wird. Bei der Waschmaschine in 5c) stellt sich die Ausgabe als weniger eindeutig heraus. Während ein Großteil der Merkmale dazu führt, dass der Klassifikator zu einem defekten Gerät tendiert, senkt der Verzicht auf Stärken/Weichspülen diese Tendenz von ursprünglich 0.915 auf $\hat{f}(\mathbf{x}^{(3)}) = 0.585$.

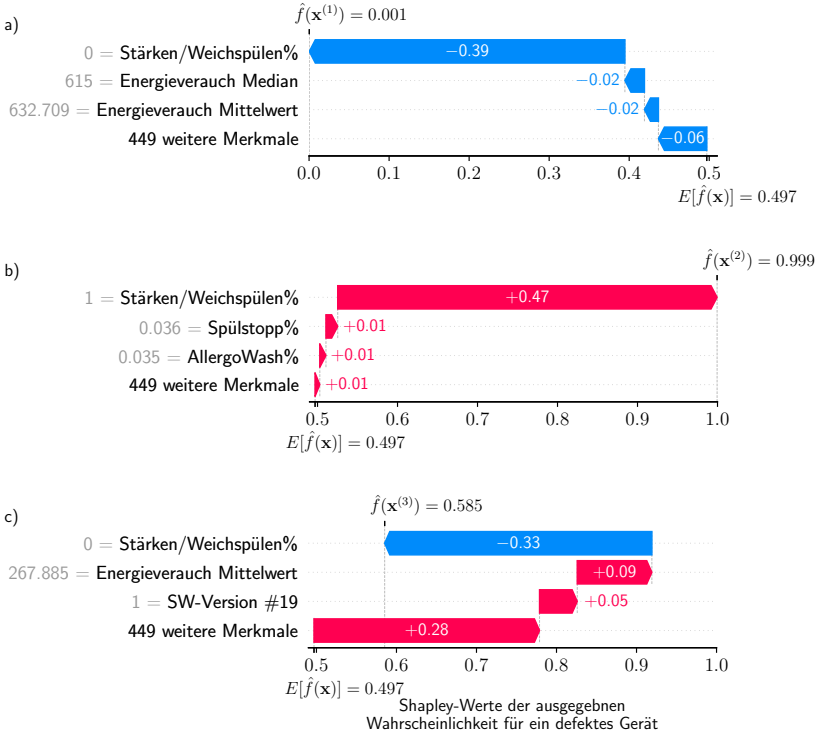


Bild 5: Lokale Interpretationen der ausgegebenen Wahrscheinlichkeit für ein defektes Gerät bzgl. des Fehlers $F\#_1$ mittels Shapley-Werten für drei verschiedene IoT-Geräte mit a) einem intakten Gerät, b) einem defekten Gerät und c) einer unsicheren Prädiktion.

Für eine erste globale Interpretation erfolgt die Ermittlung des Einflusses pro Merkmal nach (4) unter Verwendung der zuvor berechneten Shapley-Werte der Testdatensätze. In Bild 6 sind diese globalen Merkmalseinflüsse für jeden erfolgreichen Klassifikator gegenübergestellt. Dieses zeigt den signifikanten Einfluss des Merkmals Stärken/Weichspülen% und unterstützt die Beobachtungen der zuvor aufgeführten lokalen Interpretationen aus Bild 5. Insbesondere der Klassifikator aus $\mathscr{D}_{F\#_{1.1}}$ zeigt einen hohen Einfluss von 0.42, während der Klassifikator aus $\mathscr{D}_{F\#_{1.3}}$ mit Oversampling auf mehrere Merkmale angewiesen ist. Die Klassifikatoren aus $\mathscr{D}_{F\#_{1.4}}$ und $\mathscr{D}_{F\#_{1.6}}$ mit Betriebszeit bis zum Fehler äußern eine Relevanz des Merkmals Waschmittelverstärker%, das bei Betrach-

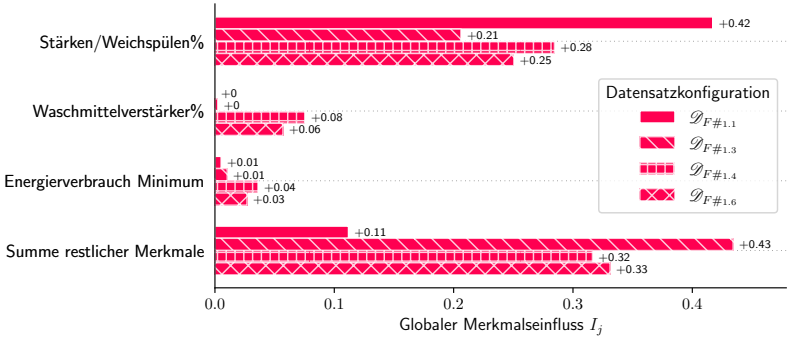


Bild 6: Balkendiagramm zur Darstellung des globalen Merkmalseinflusses mittels Shapley-Werten für die erfolgreichen Klassifikatoren der AutoML-Runs des Fehlerfalls $F\#_1$.

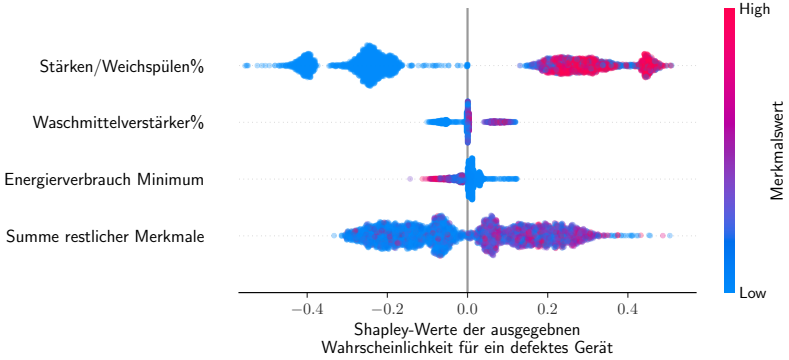


Bild 7: Übersichtsabbildung der Shapley-Werte für erfolgreiche Klassifikatoren aus $\mathscr{D}_{F\#_{1.1}}$, $\mathscr{D}_{F\#_{1.3}}$, $\mathscr{D}_{F\#_{1.4}}$ und $\mathscr{D}_{F\#_{1.6}}$ des Fehlerfalls $F\#_1$ zur Darstellung des Merkmalseinflusses gegenüber ihrer Merkmalswerte.

tung der gesamten Betriebszeit keine Bedeutung hat. Zusätzlich zum reinen Einfluss sind in Bild 7 die gesamten lokalen Interpretationen inklusive der jeweiligen Merkmalswerte in einer Übersichtsabbildung dargestellt. Zu erkennen ist hierbei, dass alle Waschmaschinen, die Stärken/Weichspülen verwendet haben (rote Punkte), positive und hohe Shapley-Werte aufweisen, während die Mehrzahl der Waschmaschinen, die kein Stärken/Weichspülen verwendet haben (blaue Punkte), negative Shapley-Werte aufweisen. Die Shapley-Werte der restlichen Merkmale äußern ein deutlich geringeren Einfluss. So tendieren

bspw. Prädiktionen für Waschmaschinen mit einem hohen Wert des Merkmals minimaler Energieverbrauch oder einem geringen Wert des Merkmals Waschmittelverstärker%, geringfügig zu einem intakten Gerät.

4.3 Ergebnisse & Diskussion

Anhand der Ausgabe des Assistenzsystems, in Form der Bilder 5, 6 und 7, lässt sich Stärken/Weichspülen% aufgrund seiner Signifikanz als alleinige hypothetische Ursache basierend auf der Gerätenutzung für den Fehlerfall $F\#_1$ identifizieren. Hypothese: *Der Fehlerfall $F\#_1$ tritt bei Waschmaschinen auf, bei denen das Programmextra Stärken/Weichspülen verwendet wurde*. Die restlichen Merkmale bieten hingegen einen zu geringen Einfluss, um weitere hypothetische Ursachen abzuleiten. Durch die Aufdeckung des unbekannten Zusammenhangs zwischen dem Stärken/Weichspülen und dem Auftreten des Fehlerfalls $F\#_1$, ist die Evaluation insgesamt positiv zu bewerten. Hinzu kommt die erfolgreiche Bestimmung der Relevanz von $F\#_1$ und $F\#_2$, sowie die Feststellung einer unzureichenden Grundlage des $F\#_2$ für weitere Interpretationen.

Trotz einer erfolgreichen Evaluation des Konzepts ist nicht sichergestellt, dass die identifizierten hypothetischen Ursachen einen kausalen Zusammenhang zum untersuchten Fehlerfall aufweisen. Eine eindeutige Fehlschlussfolgerung wäre z.B., dass der Energieverbrauch der Geräte erhöht werden muss, um die Wahrscheinlichkeit für den Fehler $F\#_1$ zu verringern. Demnach bedarf es weiterhin Domänenexpert:Innen, welche die vom Assistenzsystem aufgezeigten Korrelationen kritisch überprüfen und in kausale Zusammenhänge überführen müssen. Anhand von kausal widerlegbaren Hypothesen können Verbesserungen an der Datenbasis von IoT-Gerätenutzungsdaten, anstatt an den IoT-Geräten selbst, vorgenommen werden. Bspw. ließen sich durch widerlegbare Hypothesen unbekannte Inkonsistenzen oder bislang unbeobachtete *Confounder* in den Daten identifizieren. Eine weitere Beschränkung ergibt sich aus der Begrenzung des eingehenden Datensatzes auf die Nutzung der Geräte, während ein ganzheitlicher Datensatz über die Lebenszeit (Produktion + Nutzung) umfassendere Zusammenhänge sowie Schlussfolgerungen zulässt. Des Weiteren äußerte sich die Durchführung mehrerer AutoML-Runs je Fehlerfall für die binären Klassifikationen zwar

als präzise, was jedoch zu Lasten der Skalierbarkeit des Konzepts auf die Anwendung mehrere Fehlerfälle geht.

5 Zusammenfassung & Ausblick

In dieser Arbeit wurde das Konzept eines KI-basierten Assistenzsystems zur QS von IoT-Geräten vorgestellt, das relevante Fehlerfälle identifiziert, Zusammenhänge zwischen Gerätenutzung und Fehlerfällen mittels AutoML approximiert, und diese Zusammenhänge den Domänenexpert:Innen mittels SHAP zugänglich macht. Als Ergebnis der Evaluation basierend auf realen IoT-Gerätenutzungsdaten vernetzter Waschmaschinen konnte die Verwendung eines Programmextras als hypothetische Ursache für einen Fehlerfall identifiziert werden, so dass sich eine insgesamt positive Evaluation ergab. Im Anschluss an die Evaluation erfolgte abschließend eine Diskussion über das erarbeitete Konzept, welches insbesondere die Diskrepanz zwischen Korrelationen und tatsächlichen Kausalitäten sowie die Grenzen des Konzepts thematisiert.

Zukünftig gilt es, die Potenziale und Grenzen des Konzepts mithilfe der Domänenexpert:Innen des QM weiter auszuarbeiten, die identifizierten Hypothesen auf Kausalitäten zu überprüfen und in Verbesserungsmaßnahmen zu überführen. Zur Unterstützung bei der Schätzung von Kausalitäten bietet sich zudem die Verwendung von *Double/Debiased Machine Learning* an, wodurch *Confounder* in den Daten sichtbar werden. Nach dem ersten Erfolg in der Evaluation bietet sich die Betrachtung eines Mehrklassen-Problems zur verbesserten Skalierbarkeit und die Einbeziehung von Produktionsdaten an.

Literatur

[1] J. Chatterjee, N. Dethlefs. „Temporal Causal Inference in Wind Turbine SCADA Data Using Deep Learning for Explainable AI“. In: *Jour. of Phys.: Conf. Ser. 1618*. 2020.

[2] B. Steurtewagen, D. Van den Poel. „Adding interpretability to predictive maintenance by machine learning on sensor data“. In: *Comp. & Chem. Eng. 152*, 107381. 2021.

[3] R. Chen, F. Jankovic, N. Marinsek, et al. „Developing Measures of Cognitive Impairment in the Real World from Consumer-Grade Multimodal Sensor Streams“. In: *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, S. 2145-2155. 2019.

[4] S.M. Lundberg, B. Nair, M.S. Vavilala, et al. „Explainable machine-learning predictions for the prevention of hypoxaemia during surgery“. In: *Nat Biomed Eng 2*, S. 749-760. 2018.

[5] S.M. Lauritsen, M. Kristensen, M.V.Olsen, et al. „Explainable artificial intelligence model to predict acute critical illness from electronic health records“. In: *Nat Commun 11*, 3852. 2020.

[6] C. Thornton, et al. „Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms“. In: *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, S. 847–855. 2013.

[7] F. Hutter, H.H. Hoos, K. Leyton-Brown. „Sequential modelbased optimization for general algorithm configuration“. In: *International conference on learning and intelligent optimization*, S. 507–523. 2011.

[8] L. Li, K. Jamieson, G. DeSalvo, et al. „Hyperband: A novel bandit-based approach to hyperparameter optimization“. In: *The Journal of Machine Learning Research 18.1*, S. 6765-6816. 2017.

[9] S. Falkner, A. Klein, F. Hutter. „BOHB: Robust and efficient hyperparameter optimization at scale“. In: *International Conference on Machine Learning*, S. 1437-1446. 2018.

[10] F. Mohr, M. Wever, E. Hüllermeier. „ML-Plan: Automated machine learning via hierarchical planning“. In: *Machine Learning 107*, S. 1495–1515. 2018.

[11] B. Zoph, Q.V. Le. „Neural architecture search with reinforcement learning“. In: *ICLR 2017*.

[12] C. Molnar. „Interpretable machine learning. A Guide for Making Black Box Models Explainable“. 2019.

[13] E. Štrumbelj, I. Kononenko. „Explaining prediction models and individual predictions with feature contributions“. In: *Knowl. and info. sys. 41.3*, S. 647-665. 2014.

[14] S.M. Lundberg, L. Su-In. „A Unified Approach to Interpreting Model Predictions“. In: *Adv. Neural Inf. Process Syst. 30*, S. 4765-4774. 2017.

[15] M.T. Ribeiro, S. Singh, C. Guestrin. „Why should I trust you?: Explaining the predictions of any classifier“. In: *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*. 2016.

[16] S.M. Lundberg, E.G. Gabriel, L. Su-In, et al. „From local explanations to global understanding with explainable AI for trees“. In *Nature Machine Intelligence volume 2*, S. 56-67. 2020.

[17] D. Janzing, et al. „Feature relevance quantification in explainable AI: A causal problem“. In: *Int. Conf. on AI. and Stat.*, S. 2907-2916. 2020

[18] Birolini, Alessandro. „Reliability engineering“ Springer Berlin. 2017.

[19] Databricks AutoML. URL: https://databricks.com/product/automl. [Zugriff am: 01.09.2021].

[20] L. Breiman. „Random Forests“. In: *Machine Learning 45*, S. 5–32. 2001.

[21] T. Chen, C. Guestrin. „"Xgboost: A scalable tree boosting system“. In: *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*. 2016.

[22] G. Ke, Q. Meng, et al. „LightGBM: A Highly Efficient Gradient Boosting Decision Tree“. In: *Adv. Neural Inf. Process Syst. 30*, S 3149-3157. 2017.

[23] M. Wever, A. Tornede, F. Mohr, E. Hüllermeier. „AutoML for Multi-Label Classification: Overview and Empirical Evaluation“. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence*. 2021.

[24] T. Miller. „Explanation in artificial intelligence: Insights from the social sciences“. In: *Artificial Intelligence 267*, S. 1-38. 2019.

[25] N.V. Chawla, K.W. Bowyer, L. O.Hall, W.P. Kegelmeyer. „SMOTE: synthetic minority over-sampling technique“. In: *Journal of artificial intelligence research*, S. 321-357. 2002.

[26] M. Christ, A.W. Kempa-Liehr, M. Feindt. „Distributed and parallel time series feature extraction for industrial big data applications“. *arXiv preprint arXiv:1610.07717*. 2016.

[27] D. Chicco, G. Jurman. „The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation“. In: *BMC genomics 21.1*, S. 1-13. 2020.

Dieser Tagungsband enthält die Beiträge des 31. Workshops „Computational Intelligence“ des Fachausschusses 5.14 der VDI/VDE-Gesellschaft für Mess- und Automatisierungstechnik (GMA) und der Fachgruppe „Fuzzy-Systeme und Soft-Computing“ der Gesellschaft für Informatik (GI), der vom 25. – 26.11.2021 in Berlin stattfindet.

Der GMA-Fachausschuss 5.14 „Computational Intelligence“ entstand 2005 aus den bisherigen Fachausschüssen „Neuronale Netze und Evolutionäre Algorithmen“ (FA 5.21) sowie „Fuzzy Control“ (FA 5.22). Der Workshop steht in der Tradition der bisherigen Fuzzy-Workshops, hat aber seinen Fokus in den letzten Jahren schrittweise erweitert.

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 978-3-7315-1131-1