Herausgeber

H. SCHULTE
F. HOFFMANN
R. MIKUT

Berlin | 26. – 27. November 2020

PROCEEDINGS **30. WORKSHOP** COMPUTATIONAL INTELLIGENCE

KIT Scientific Publishing

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

Proceedings. 30. Workshop Computational Intelligence

Berlin, 26. – 27. November 2020

PROCEEDINGS **30. WORKSHOP** COMPUTATIONAL INTELLIGENCE

Berlin, 26. – 27. November 2020

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 2020 – Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-1051-2
DOI 10.5445/KSP/1000124139

Inhaltsverzeichnis

Reliable Driver Gaze Classification based on Conformal Prediction

Simone Dari[1,2], Eyke Hüllermeier[2]

[1]Safety Department, BMW Group
Knorrstraße 147, 80788 Munich

[2]Heinz Nixdorf Institute and Department for Computer Science,
Paderborn University
Pohlweg 51, 3098 Paderborn

E-Mail: simone.dari@bmw.de, eyke@upb.de

1 Introduction

Machine learning is increasingly used in practical applications that can be categorized as safety-critical, such as AI-assisted driving. In this context, we recently considered the problem of driver monitoring, which plays an essential part in avoiding accidents by warning the driver in time and shifting the driver's attention to the traffic scenery in critical situations [1]. This may apply for the different levels of automated driving, for take-over requests as well as for driving in manual mode. More specifically, we tackled the problem of predicting the driver's gazing direction. Distinguishing eight different regions, this problem can be formalized as a classification task, in which each region corresponds to a class (cf. Figure 1). We proposed a deep learning approach to predict gaze regions, which is based on informative features such as eye landmarks and head pose angles of the driver. Moreover, we introduced different post-processing techniques that improve the accuracy by exploiting temporal information from videos and the availability of other vehicle signals. Our main interest is to leverage accurate gaze prediction for improved human-computer-interaction. In this regard, it is arguably important to guarantee a certain level

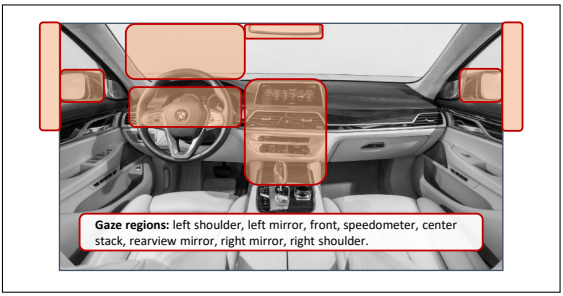


Figure 1: Spatial zones distinguished in the driver gaze classification task.

of awareness of the computer (AI system) of its own certainty or uncertainty in a prediction [3].

In this work, we therefore leverage so-called *conformal prediction* (CP) to increase the reliability of such predictions [10, 12]. Instead of predicting a single class, CP produces a *set-valued* prediction, i.e., a subset of all candidate classes that comprises the true class with high probability. This way, the system is able to express ambiguities (several regions appear plausible, because the driver's gaze cannot be determined precisely) as well as a partial or complete lack of knowledge — for example, the driver may look in a completely different direction, which does not correspond to any of the eight pre-specified regions (thereby producing so-called out-of-distribution data).

Conformal prediction can be seen as a meta-learning technique, which can be put on top of any base learner, i.e., any standard classifier producing "point predictions." It merely requires a measure of (non-)conformity of a (hypothetical) data point, i.e., a measure of how well a combination of feature values and gaze directions fits with the training data seen so far. While the required level of confidence — the predicted set contains the true class with a pre-specified probability (such as 95%) — is guaranteed regardless of the conformity measure, the latter has a strong influence on the precision of predictions, i.e., the (average) size of the predicted sets.

In this work, we evaluate different types of conformity scores to construct conformal predictors for driver gaze classification, including scores derived from kernel density estimation as proposed in [2, 5], and compare them with regard

to the quality of set-valued predictions as well as time efficiency. Moreover, we elaborate on a specific characteristic of our problem, namely the fact that our output space exhibits a natural (topological) structure induced by the spatial relationship between the classes — unlike standard classification problems, where the classes constitute a simple set with no relationships between its elements. As a consequence, there are more meaningful (*viz.* topologically connected) and less meaningful (unconnected) set-valued predictions. To assure semantically meaningful set-valued predictions, we propose an extension of standard CP.

The work is structured as follows. In Section 2, we shortly review the gaze classification system with its results. Section 3 explains the conformal prediction method more closely. In Section 4, we apply the method to the gaze dataset. Section 5 discusses the results while Section 6 concludes with some final remarks.

2 Gaze Classification

The problem of driver monitoring was recently studied in [1]. This section briefly summarizes the method used and the key results obtained. For detailed information, the interested reader is referred to [1].

2.1 Problem Statement and Dataset

The goal of the gaze classifier is to reliably classify the region the driver is looking at, based on an image of the driver. Certain regions are of special interest and are displayed in Figure 1. The underlying dataset was extracted from a naturalistic driving study in which participants were driving a car for several months while being recorded with an RGB camera installed at the A-pillar. Sample images are provided in Figure 2. The examined dataset consists of 75 video snippets from 20 subjects (5 female, 15 males). Driver videos were recorded in size 980×540 at 15 frames per second. The important regions of interest (also *classes*, *labels*) with the number of images available are given in table 1.

Table 1: Number of classes

Class	Abbr.	Number
left shoulder	ls	98
left mirror	lm	967
speedometer	sp	228
front	f	2,296
inner mirror	inm	713
center console	cc	332
right shoulder	rs	72
right mirror	rm	356

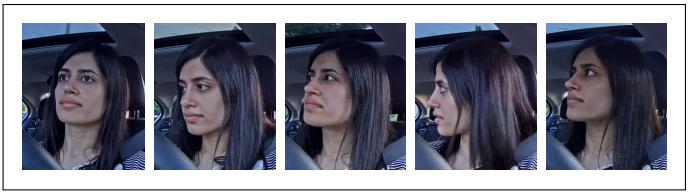


Figure 2: Example images from the dataset.

2.2 Method

The pipeline of the gaze classification system is depicted in Figure 3. In a pre-processing step, meaningful features on the driver's head pose and the eyes are generated from existing image-based methods and then fed into a fully connected neural net. For an inserted image, the driver's face is detected [4] and the three head pose angles are computed [8]. The angles describe the orientation of the head, where the rotation around the x-axis is called *pitch* (i.e. from up to down), around the y-axis *yaw* (i.e. from left to right), and around the z-axis *roll* (i.e. from left to right shoulder). For the eyes, the eye landmark detector by Park et al. [7] is employed. We make use of 15 landmarks per eye that describe the eyelid and the iris. The generated features are fed as input to a neural network architecture. The output of this network is scaled by the softmax-activation function, which produces a vector with probabilities for each class.

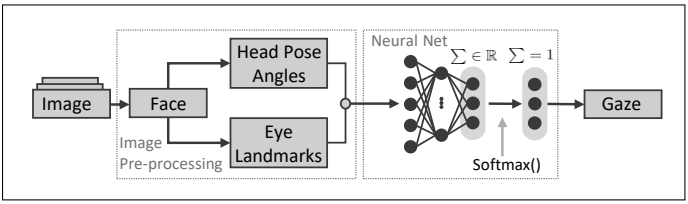


Figure 3: Pipeline of the Gaze classification system.

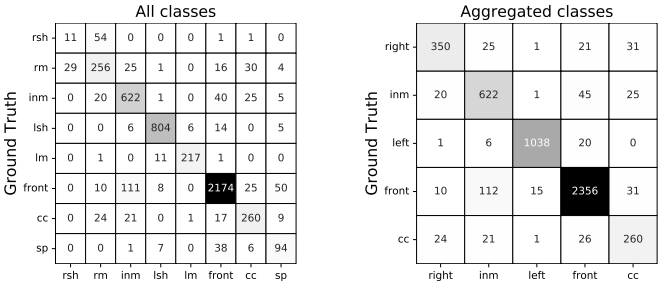


Figure 4: Results after Cross-validation [1].

2.3 Results

As there might be driver-dependent characteristics in the features, we propose training with a leave one-driver out cross-validation, training on 19 drivers, while testing on the remaining driver. The results from the test set after every iteration are aggregated into the confusion matrix given in Figure 4. In total, the model achieves an accuracy of 87.1%. After aggregating the classes from the left and the right side, as well as the *speedometer* with the *front* class, accuracy increases to 91.4%. Misclassification occurs for the classes *front* and *inner mirror*, as well as for *inner mirror*, *front* and the *right side*.

2.4 Discussion

In general, the error rate of 12.9% can be narrowed down to three types of misclassifications: (i) misclassifications between similar classes that can be aggregated together without a higher loss of information (e.g., *right shoulder*

and *right mirror*) (4.3%), (ii) misclassifications between classes far apart (e.g., *left mirror* and *right mirror*) which make up 1.4% and (iii) misclassifications among classes close to one another. Indeed, there is a high number of misclassifications for the classes *front*, *inner mirror*, *right mirror* and *center console*. One can possibly assume that classes away from the camera are harder to perceive. If the driver is looking through the front windshield and directly beneath the inner mirror, e.g., while focusing on a vehicle far ahead on the right side, it becomes difficult from the camera point of view to correctly annotate this situation, for both, the human annotator and apparently also the system.

3 Conformal Prediction

In cases of uncertainty, set-valued predictions are supposed to produce reliable predictions, i.e., subsets of classes comprising the true one with high probability, very much like confidence intervals as known from classical statistics. One way of obtaining such sets is through *conformal prediction* [10, 12], which is based on the idea of reducing prediction to hypothesis testing: Given a query instance, a class label is included as a candidate in the set-valued prediction unless the hypothesis that this label corresponds to the ground truth can be rejected at a pre-specified level of confidence. The test itself relies on assigning each instance/label combination a measure of *non-conformity*, reflecting how "strange" this combination appears in light of the data seen so far. Its counterpart is the *conformity* measure, reflecting the similarity to the data seen so far.

The original idea of CP was introduced for the setting of online learning [12]. Here, we present a version adapted to the standard setting of supervised learning, called *Inductive Conformal Prediction* (ICP) [6]. We only focus on the case of classification, for which we have seen n examples in the training data and seek to predict the label of a new query instance.

Formally, for previous observations $(x_1,y_1),(x_2,y_2),\ldots,(x_n,y_n)$ with $(x_i,y_i) \in \mathscr{Z} = \mathscr{X} \times \mathscr{Y}$, a set-valued predictor $\Gamma^\varepsilon : \mathscr{X} \to 2^{\mathscr{Y}}$ is constructed on the basis of a permutation-invariant *non-conformity measure* $\phi : \mathscr{Z} \times \mathscr{Z}^n \to \mathbb{R}$ that indicates how strange a hypothetical example $z = (x,y) \in \mathscr{Z}$ is compared

to previous examples in a set $A \in \mathscr{Z}^n$. The outputs of the non-conformity measure are called non-conformity scores and are formally described as

$$\begin{aligned} \phi_i &= \phi(z_i, A_i) = \phi((x_i, y_i), \{z_1, \ldots, z_{n+1}\} \setminus \{z_i\}) && (1) \\ &:= \phi(y_i, \hat{f}_{A_i}(x_i)), && (2) \end{aligned}$$

where $\hat{f}_{A_i} : \mathscr{X} \to \mathscr{Y}$ is the point prediction rule learned on A_i. Then, for a new instance x_{n+1}, all possible candidate values $y \in \mathscr{Y}$ are considered for y_{n+1} by testing the hypothesis $H_0 : y_{n+1} = y$ (against $H_1 : y_{n+1} \neq y$) and computing the p-values

$$p_y = \frac{\sum \mathbf{1}\{\phi_i \geq \phi_{n+1}\}}{n+1}. \tag{3}$$

The set-valued prediction is then given by

$$\Gamma^{\varepsilon}(x_{n+1}) = \{y : p_y > \varepsilon\}. \tag{4}$$

Under some technical assumptions[1], it can be shown that this prediction fulfills

$$\mathbb{P}(y_{n+1} \in \Gamma^{\varepsilon}(x_{n+1})) \geq 1 - \varepsilon. \tag{5}$$

In ICP, the dataset is split into three parts: the training set, the *calibration* set and the testing set. The training set is used to train the point predictor $\hat{f}$. The calibration set $\{z_1, \ldots, z_n\}$ is used to compute the non-conformity scores $\{\phi_1, \ldots, \phi_n\}$ only once. For a new instance z_{n+1} from the test dataset, the non-conformity score ϕ_{n+1} is computed as usual:

$$\begin{aligned} \phi_i &= \phi(z_i, A \setminus \{z_i, z_{n+1}\}) \quad \forall i \in \{1, \ldots, n\} && (6) \\ \phi_{n+1} &= \phi(z_{n+1}, A \setminus z_{n+1}). && (7) \end{aligned}$$

Then, the non-conformity score ϕ_{n+1} is compared to the scores from the calibration set to eventually compute its p-value according to (3).

The guarantee (5) holds "on average", that is, when assuming new samples (x_{n+1}, y_{n+1}) to be drawn according to the underlying probability measure on

[1] A key assumption is the condition of exchangeability [12].

Table 2: Non-Conformity Measures

A)	Kernel Density (KDE)	$\phi((x,\tilde{y}),A) = (1+\hat{p}_A(x\|y=\tilde{y})^{-1}$
B)	Distance to mean (DTM)	$\phi((x,\tilde{y}),A) = \|\bar{x}_{A,\tilde{y}} - x\|$
C)	1 Nearest Neighbour (1NN)	$\phi((x,\tilde{y}),A) = \frac{\min\{\|x_{A,y}-x\|, y=\tilde{y}\}}{\min\{\|x_{A,y}-x\|, y\neq\tilde{y}\}}$

$\mathscr{X}$. It does not hold, however, *conditional* to a specific class $\tilde{y} \in \mathscr{Y}$, i.e., under the condition that $y_{n+1} = \tilde{y}$. In other words, predictions might be more valid for some (ground truth) classes and less for others. Therefore, in cases of strong class imbalance, where the set sizes vary too strongly among the classes for the same choice of confidence level $1-\varepsilon$, it appears meaningful to choose $\varepsilon_{\tilde{y}}$ for each class $\tilde{y}$ separately. This is also known as *Mondrian Conformal Prediction* [11].

In the following, we consider several measures for $y_{n+1} = \tilde{y}$ which are given in table 2. There, $\bar{x}_{A,\tilde{y}}$ is the mean over all x-vectors in A labeled with class $\tilde{y}$, and $x_{A,y}$ the instance in A with smallest (Euclidean) distance to x among those with label y. Moreover, $\hat{p}_A$ denotes the class-conditional density, estimated on the set A by means of kernel density estimation with Gaussian kernel learned.

4 Results

In this section, the results for the different non-conformity measures are reported for the gaze dataset introduced earlier. For every new instance in the test set, the p-value p_y for each class $y \in \mathscr{Y}$ is returned. The latter can be used in two different ways: (i) The class with the highest p-value is chosen as a point prediction (the p-value itself is then called the *credibility* of the prediction). (ii) For a given confidence level $1-\varepsilon$, the set of labels Γ^{ε} is returned as a set-valued prediction. For (i) the error of this predictor is reported, while for (ii), the average size of the predicted sets (at different confidence levels) is of specific interest.

4.1 The Gaze Dataset

Similar to [6, 2], we apply the method of conformal prediction by extracting the output of the neural network before applying the softmax function and use it as the input feature for ICP (cf. Figure 3). In this way, we circumvent the disadvantages of the softmax transformation [2]. For calibration, 300 instances per class (100 instances for the classes *left shoulder* and *right shoulder*) are sampled. The test set is a newly annotated dataset that consists of 5 videos with 3,138 frames.

We report the results for the three conformity measures KDE, DTM, and 1NN. A stacked bar plot is employed to visualize the set sizes for each conformity measure. It is provided in Figure 5. The set sizes at confidence level $1-\varepsilon$ are displayed in different colors. The black graph corresponds to the accuracy of the single predictions while the red graph represents the accuracy of all predicted sets (including non-empty sets as well). More information is provided in Table 3 with the average credibility of the class with the highest p-value. Furthermore, the table contains information on the average size of the non-empty sets and the accuracies for all sets at different confidence levels $1-\varepsilon \in \{0.85, 0.90, 0.95, 0.98\}$.

From Table 3, it can be observed that the error of the point predictor is lowest at 10.6% for the KDE measure. The other measures produce error rates between 16.1% and 18.3%, while the error rate for the baseline gaze classifier with the softmax-generated output is at 15.2%. The favorable, i.e., the highest average credibility is reached by KDE and 1NN at 32.19% and 34.14%. From the plots in Figure 5, it can be noticed that there are only slight differences in the number of empty set predictions (colored in green) and single set predictions (in purple). The number of sets with more than one label is highest for all confidence levels for the measure 1NN. Table 3 shows that the average size of non-empty sets is always lowest for KDE. Also, the average set size for non-empty sets is for all three measures similar at lower confidence levels, e.g., $1-\varepsilon = 0.85$. With increasing confidence levels, the sizes vary more strongly, e.g., 4.81 for DTM and 1.69 for KDE at confidence level $1-\varepsilon = 0.98$. The (statistical) guarantee (5) is met for both, 1NN and KDE. DTM misses

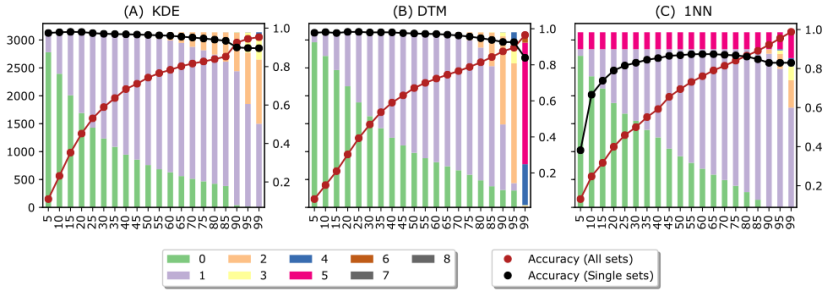


Figure 5: Stacked bar graph that visualizes the set sizes.

the required confidence level by a few percentage points at higher confidence levels.

The average computational time per method, i.e., computing the conformity scores for the calibration set and the testing set, is shortest for DTM with 64 seconds. For the KDE method and 1NN, 379 resp. 723 seconds are needed on average.

4.2 Structure of Prediction Sets

As the label space $\mathscr{Y}$ consists of eight regions in our application, there are $2^8 = 256$ possible prediction sets that might be produced by CP. Even if all these sets are valid in a statistical sense, not all of them appear to be semantically meaningful. In fact, $\mathscr{Y}$ is not just a set of distinct classes. Instead, the classes are spatially related to each other. Intuitively, one would therefore expect that prediction sets correspond to spatially neighbored regions. Or, stated differently, prediction sets that include certain regions while omitting regions "in-between" may appear less meaningful. For example, if *right mirror* and *inner mirror* are included, one would expect *front* to be included, too.

To check for the semantic meaningfulness of CP predictions, we examine the test data further. For the confidence level $1-\varepsilon = 0.96$, the predicted sets and their number are displayed in Table 4. The sets $\{$*right mirror*, *inner mirror*$\}$ and $\{$*inner mirror*, *center console*$\}$ are examples of arguably less meaningful

Table 3: Results Gaze Dataset

Conf-Meas.		KDE	DTM	1NN
Error*		0.106	0.161	0.17
Time (s)		379	64	723
Avg. Cred.		32.19	39.68	34.14
85	Size	1.2	1.28	1.4
	Acc.	0.853	0.849	0.895
90	Size	1.34	1.73	1.44
	Acc.	0.932	0.878	0.931
95	Size	1.56	2.38	1.59
	Acc.	0.947	0.897	0.962
98	Size	1.69	4.81	2.06
	Acc.	0.953	0.965	0.988

*Error of the Baseline model: 0.152.

predictions, as they omit the in-between class *front*. These sets are marked with (*). Only once, the predicted set contains classes which are evidently not meaningful {*right mirror*, *left mirror*, *front*} marked with (**). This combination was produced by 1NN. In total, 48 of the 256 theoretically possible sets are predicted.

5 Discussion

While the statistical guarantee of correctness holds with an increasing number of instances, regardless of the non-conformity measure chosen, this measure has an important influence on the *efficiency* of CP, that is, the size of prediction sets: The more suitably the non-conformity measure is chosen, the smaller these sets will be. In our case, non-conformity scores derived from KDE and 1NN provide significantly smaller sets than DTM, which is in line with the theory presented in [9]. Indeed, one should note that the distance to the mean is a rather crude measure, which ignores a lot of information about the class distributions.

Table 4: Sets at confidence level $1 - \varepsilon = 0.96$

Set Size	1NN	DTM	KDE	Sets
0	0	0	8	{}
1	2307	26	1718	{rm}, {lsh}, {sp}, {inm}, {lm}, {cc}, {fr}
2	344	1007	1009	{rm, cc}, {rm, inm}*, {rm, fr}*, {lsh, fr}*, {fr, cc}, {lm, lsh}, {lm, fr}, {cc, sp}, {inm, sp}*, {inm, cc}*, {fr, sp}, {inm, fr}
3	120	591	397	{lm, cc, sp}, {rsh, rm, cc}, {rsh, inm, fr}*, {rm, lm, fr}**, {rm, inm, fr}*, {rm, inm, cc}*, {inm, fr, cc}, {inm, cc, sp}*, {inm, fr, sp}, {lm, fr, sp}, {rm, fr, sp}*, {inm, lm, fr}, {lm, lsh, fr}, {fr, cc, sp}
4	53	1445	6	{inm, fr, cc, sp}, {rm, lm, fr, sp}*, {lm, lsh, fr, sp}, {lm, fr, cc, sp}, {rsh, rm, fr, cc}, {rsh, rm, inm, cc}, {rm, inm, fr, sp}*, {rm, inm, fr, cc}
5	312	60	0	{inm, lm, fr, cc, sp}, {rm, inm, fr, cc, sp}, {rsh, rm, inm, fr, cc}, {rsh, rm, inm, lm, fr}*
6	2	9	0	{rm, inm, lm, fr, cc, sp}, {rsh, rm, inm, fr, cc, sp}

Number of predicted sets: 48.

As for the semantic meaningfulness of the CP predictions, we observed that CP seems to capture the spatial structure of the classes quite well, with only a few exceptions. In total, 48 of the 256 possible sets were returned as predictions, only 13 of which displayed minor gaps and a single one severe "gaps" in the associated spatial region, i.e., the union of the regions associated with the classes in the set.

CP can also be used as a point predictor. In that sense, regarding solely the error made when choosing the class with the highest p-value, CP with the KDE measure as non-conformity score even outperforms the original gaze classification system, while also providing additional information. This indicates that the relations among the neural net's outputs cannot solely be disclosed with the softmax function, and that the highest value in the output layer does not always correspond to the best prediction.

6 Conclusion

In safety-relevant applications of machine learning, such as AI-assisted driving, a predictive model should produce reliable predictions and be aware of its own uncertainty. In this paper, we considered the problem of predicting the driver's gazing direction and elaborate on the use of conformal prediction to represent uncertainty. Instead of guessing a single class label (gazing direction), even in cases of uncertainty, conformal prediction yields set-valued predictions that are guaranteed to cover the true class with high probability. Our first experimental results with different variants of conformal prediction are rather promising. In particular, we have seen that the extension of our original gaze classification system by means of CP can indeed decrease the error rate of the model while still providing important information on the confidence of the estimates. Especially promising is the non-conformity measure based on kernel density estimation, as it yields the smallest set sizes at high confidence levels.

A possible application of this method, which we seek to investigate in future work, is the handling of out-of-distribution data for classes not covered by the gaze classification system (e.g. blinks). Moreover, we plan to elaborate on

Mondrian Conformal Prediction with class-specific confidence levels, where the confidence levels are determined by the Pareto optimum between different criteria (e.g. average set size and accuracy of single predictions).

References

[1] S. Dari, N. Kadrileev, and E. Hüllermeier. “A Neural-Network Based Driver Gaze Classification System with Vehicle Signals”. In: *IEEE International Joint Conference on Neural Networks 2020*. In press.

[2] Y. Hechtlinger, B. Póczos, and L. A. Wasserman. “Cautious deep learning”. *CoRR*, abs/1805.09460, 2018.

[3] E. Hüllermeier and W. Waegeman. “Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods.” *CoRR*, abs/1910.09457, 2019.

[4] Z. Liu, P. Luo, X. Wang, and X. Tang. “Deeplearning face attributes in the wild”. In: *2015 IEEE InternationalConference on Computer Vision, ICCV 2015,* pages 3730–3738. IEEE Computer Society. 2015.

[5] S. Messoudi, S. Rousseau, and S. Destercke. “Deep conformal prediction for robust models.” In: *18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* , IPMU 2020, pages 528–540. Springer. 2020.

[6] H. Papadopoulos. “Inductive conformal prediction: Theory and application to neural networks.” In: *Tools in Artificial Intelligence,* chapter 18. IntechOpen, Rijeka, 2008.

[7] S. Park, X Zhang, A. Bulling, and O. Hilliges. “Learning to find eye region landmarks for remote gaze estimation in unconstrained settings.” In: *Proceedings of the 2018 ACM Symposiumon Eye Tracking Research and Applications - ETRA ’18,* pages 1–10. ACM Press. 2018.

[8] N. Ruiz, E. Chong, and J. M. Rehg. “Fine-grained headpose estimation without keypoints.” *CoRR, abs/1710.00925,* 2017.

[9] M. Sadinle, J. L., and L. A. Wasserman. “Least ambiguous set-valued classifiers with bounded error levels.” *CoRR,abs/1609.00451,* 2016.

[10] G. Shafer and V. Vovk. “A tutorial on conformal prediction”. *J. Mach. Learn.* vol. 9, pages 371–421, 2008.

[11] P. Toccaceli and A. Gammerman. “Combination of inductive mondrian conformal predictors”. *Mach. Learn.*, vol. 108(3), pages 489–510, 2019.

[12] V. Vovk, A. Gammerman, and G. Shafer. “Algorithmic Learning in a Random World.” Springer-Verlag, New York, 2005.

Siamese Networks for 1D Signal Identification

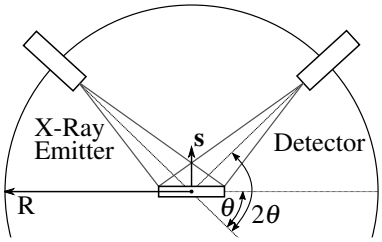
Jan Schuetzke[1], Alexander Benedix[2], Ralf Mikut[1], Markus Reischl[1]

[1] Institute for Automation and Applied Informatics, KIT
Hermann-von-Helmholtz-Platz 1
76344 Eggenstein-Leopoldshafen
E-Mail: jan.schuetzke@kit.edu

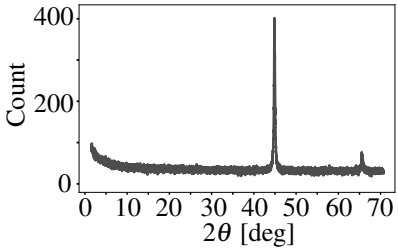
[2] Bruker AXS GmbH
Oestliche Rheinbrückenstraße 49
76187 Karlsruhe
E-Mail: alexander.benedix@bruker.com

Abstract

In material sciences, X-ray diffraction (XRD) or nuclear magnetic response (NMR) are methods to generate one-dimensional signals, describing intensities over an angle or a chemical shift. Each material has a characteristic profile and unknown samples are typically matched to known references. Automatic classification of one-dimensional signal patterns is a non-trivial task due to background noise and varying positions of measured intensities in identical probes. Convolutional Neural Networks prove to be particularly suitable, a limitation, though, is that adding new classes requires retraining. However, continuous discovery of new materials requires possibilities for easy class-extension. Siamese Neural Networks are able to extend data set classes easily and are popular in the field of face recognition, where new faces are constantly added to the database of references. In this paper, we apply Siamese networks to one-dimensional XRD-data for the first time and discuss the opportunities and challenges as well as areas of application. We show that Siamese networks are well suited for the transfer between XRD datasets, achieving an accuracy of 99% for materials not present in the training dataset.



(a) Experimental setup for measuring a XRD scan using the Bragg-Brentano focusing geometry



(b) Resulting XRD scan of an exemplary crystalline sample

Figure 1: Layout and functionality of an instrument for measuring XRD scans and a resulting diffraction pattern

1 Introduction

A typical task in the field of material analysis is to analyze an unknown sample in order to determine the underlying properties or to assign the sample to a known material. For this purpose, a variety of destructive and non-destructive methods exist, which scan the samples with a measuring instrument and then output a one- to multi-dimensional signal, which is subsequently analyzed using software or expert knowledge. Due to the large number of samples to be analyzed and the involvement of experts for the interpretation of the measurement signals, it is especially suitable for this field to utilize automated approaches to speed up the analysis of unknown material samples.

One of the representatives of the methods for the analysis of crystalline samples is the X-ray diffraction (XRD), which utilizes Bragg's Law to deduce the underlying crystal structure of the sample and thus, to determine the corresponding mineral. In the most prevalent technique for measuring XRD signals, called the Bragg-Brentano focusing geometry, the crystalline sample is crushed into a powder and measured using a moving pair of emitter and detector. Figure 1a shows the layout for an instrument that employs the Bragg-Brentano geometry. The powdered sample is placed in the center and X-ray emitter as well as detector are moving upwards on the circular orbit with radius R. The detector measures the intensity of diffracted X-rays subject to incident

angle θ. Figure 1b visualizes the resulting, one-dimensional XRD scan with the measured intensity (count) as a function of the emergent angle, typically described by 2θ [1].

For the XRD scan analysis, each crystalline material forms a distinct diffraction pattern which is subject to the underlying crystal lattice properties. It is of interest where peaks are located and how the intensities are distributed. The X-rays are reflected by atomic planes, so the number and location of peaks depends on the form and shape of the lattice, while the intensity is relative to the comprised atoms. Accordingly, the XRD scans are analyzed by comparing the diffraction pattern to a database of reference minerals. Some natural influences mean that the patterns are not exactly the same, causing small deviations in peak position, shape and intensities [1]. Thus, matching measured and reference patterns is not a trivial task, so professional expertise is required for the manual evaluation of XRD scans.

Similarly, other methods for analyzing unknown material samples, like the proton nuclear magnetic resonance spectroscopy (^{1}H NMR) or infrared spectroscopy, also feature a one-dimensional signal that is characterized by peak locations, shape and intensities. Consequently, a variety of algorithms for the automated analysis have been developed, with a focus on machine and deep learning models in recent years, which gained great popularity especially in the field of image analysis [2, 7, 4]. In many applications, neural networks are employed as classifiers, which have a clear disadvantage for the use with material science data: Once trained for a certain dataset, no further materials can be added without the network (or at least some of its layers) having to be retrained.

A special, extendable type of neural networks are the so-called Siamese networks, which do not learn any classification rules, but distinguish representatives of one class from those of other classes by means of a distance function. Thus, the network deduces the features of two inputs to be compared in two identical, parallel processing lines, allowing the references to be constantly changed and extended, if the type of the calculated features remains the same. Typically, this type of neural network is used in the field of face recognition, where the inputs in the form of faces are always somewhat-similar but the

network extracts characteristic features to distinguish dissimilar from similar faces. Likewise, one-dimensional data such as XRD or NMR scans, which consist of characteristic peaks at specific locations, are potentially suitable for use with Siamese networks.

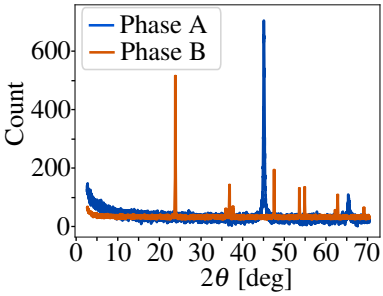
Accordingly, we apply Siamese neural networks for the use with one-dimensional data for the first time, using XRD data as an example. In contrast to regular neural network classifiers, which permanently learn the classification rules in the training phase and are therefore not extendable, the scan to be evaluated can be compared to a dynamically composed set of references for Siamese Networks. Hence, we

1. train a classifier and Siamese network on one dataset
   to compare the baseline performance,
2. apply the trained Siamese network to a second dataset without retraining,
3. retrain the classifier for the second dataset using transfer learning.

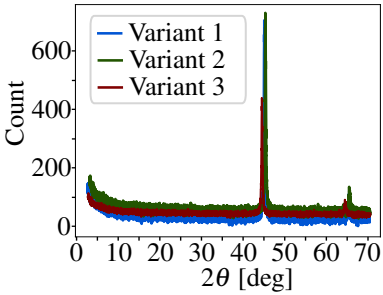
2 Related Work

The XRD scans of distinct phases differ in the number of peaks and the corresponding positions, as demonstrated in Figure 2a. Between diffraction patterns of the identical material, natural influences cause small deviations of peak position, shape and intensities, which is visualized in Figure 2b. For example, stress or small deviations of the lattice lead to a shift of the peaks, varying crystallite sizes cause to wider, flatter peaks and non-ideal preparation of the samples lead to changes of the intensity ratios [1]. The classifiers receive the XRD scans as input and must learn to distinguish between variations within a phase and diffraction patterns of different materials.

Naturally occurring crystalline materials, such as ores, are usually not pure minerals, but rather contain one major phase and small amounts of minor phases. Thus, numerous works employed the Non-Negative Matrix Factorization (NMF) algorithm, which tries to describe the measured signal by combining several components with different proportions [2]. Through a supervised training



(a) XRD scans of exemplary phases A and B, differing in peak numbers and locations



(b) Three XRD scans for various samples of the same material. The peak position, shape and intensities vary between the variants.

Figure 2: Differences between XRD scans of distinct materials and samples of identical phases. The XRD scans serve as an input for the classifiers.

procedure, the model learns to identify the components and depending on the respective fractions, the input sample is assigned to the most similar mineral of the training set. The NMF approach, however, is developed for specific, small applications like the ternary Al-Li-Fe oxide system with 6 phases to be distinguished [2]. It is unclear whether the performance can be transferred for larger datasets, since in the field of material sciences hundreds or thousands of materials frequently have to be distinguished from each other.

Most recently, neural networks gained widespread popularity for use with image data and were also applied to XRD data. First, Park et al. [5] and Oviedo et al. [6] showed that convolutional neural networks work well for predicting crystalline properties like space group and the crystal system. Then, Wang et al. [7] demonstrated that a neural network with a VGG16 like architecture [7] is able to distinguish between 1012 metallic phases. Similarly, convolutional neural networks proved to be reliable for the prediction of unknown one-dimensional NMR signals [4].

In application, the presented networks and algorithms are able to assign an unknown sample to a known reference material. However, it is a requirement that the corresponding material is also present in the training dataset, otherwise the classifier is not able to provide a meaningful result. For the XRD case with

over 1000 learned metallic phases, there is a high probability that the unknown material was also present in the training dataset for a metallic sample, but in a typical diffraction pattern database there are more than 10000 different phases. However, it is unclear whether it is possible to train a single classifier for all possible materials, while still identifying minor differences between similar phases.

Furthermore, in the field of materials science it is common to identify new materials, which would require an extension of the existing classifiers. The presented neural networks [7, 5, 6] are not easily extensible, requiring to train the classifier again in order to add one class to the existing ones. Instead of retraining all of the layers, it is also possible to freeze the weights of the convolutional layers and only retrain the fully connected layers before the output, a method that is usually referred to as transfer learning [8].

Alternatively, a concept called Siamese Neural Networks offers the functionality to extend or limit the classes during the prediction process. Once developed for signature verification [9], Siamese neural networks feature two inputs with parallel processing operations that share its weights. Accordingly, the network rates the similarity of the two processed inputs. For a classification task, one input is the unknown sample to be analyzed and the second one a reference material. After comparing the sample to all references, the networks assigns the class by choosing the highest similarity to a known material. Consequently, the network does not require retraining to add another class, but rather an additional input to be compared is added.

The Siamese network approach is particularly popular in the field of face recognition, where the reference database is continuously supplemented with new faces [10]. Here, the Siamese network uses its convolutional layer structure to extract characteristic features from the 2D inputs to find similarities and distinguish between different faces. In the field of material science, Zhang et al. [11] used Siamese networks for application with 2D NMR data to distinguish between material classes. Two dimensional NMR data, however, is more time-consuming to measure, thus an expert is commonly used to manually evaluate the one-dimensional instead. Accordingly, we employ Siamese networks for

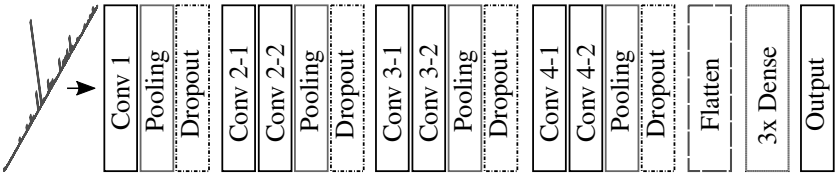


Figure 3: Architecture for VGG16-like classification network with convolutional layer (Conv) stacks, max pooling layers (Pooling) and three fully-connected layers (Dense) before the output

the first time to classify one-dimensional data describing measured materials, for which currently only classifiers exist, that are not extendable.

3 Methods

3.1 Siamese Networks

We want to evaluate whether Siamese networks are suitable for use with one-dimensional data, using XRD data as an example. To assess the performance of the Siamese network, we use the classification network developed by Wang et al. [7] as a reference, as presented in Figure 3. Here, we use diffraction patterns with 3250 datapoints (measured from 5^o to 70^o with $\Delta 2\theta$ of 0.02) as an input for the network and employ an architecture of convolutional layers, max pooling operations and dropout before we reshape the features into an one-dimensional embeddings vector in the flatten layer. For the convolutional operations, we use a 5x1 kernel with 6 to 64 filters from the first to the last convolutional layer, apply max pooling with a pool size of 2 and stride 2 and utilize dropout of value 0.2 between the pooling and convolutional layers of the different stages to reduce overfitting during the training. After the reshaping of the features into an embeddings vector, the three fully-connected (dense) layers learn the classification rules before the networks outputs the prediction scores of the classes in the output layer. Therefore, the number of neurons in the output layer corresponds to the number of eligible classes.

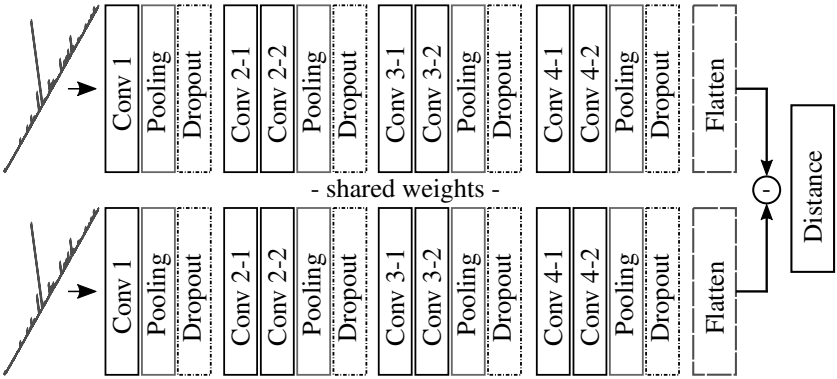


Figure 4: Architecture for VGG16-like classification network

Due to the dense layers, the classification network is not transferable to another set of classes without retraining. One concept to reduce the required retraining time is transfer learning where we freeze the weights of the convolutional layers, so they cannot be changed during the training process.

Accordingly, we try two different tactics to retrain the classification network for different classes:

1. *Full retraining* of all layers, and
2. *transfer learning* approach, where we freeze the weights of the convolutional layers and only retrain the dense layers.

Instead of learning the classification rules in the fully-connected layers, the Siamese network uses the embeddings vectors of the extracted features to rate the similarity of two fed input patterns, as visualized in Figure 4. The Siamese network employs two parallel processing pipelines to transform the input XRD scans into comparable features by sharing the weights between the corresponding layers. Thus, both inputs have to be of the same kind, the network is not able to compare a XRD scan with a crystalline properties vector.

During the training process the Siamese network has to fine-tune the weights to decrease the distance between embeddings of the same class, while increasing

the distance between classes. The distance between two vectors $\mathbf{v}$ and $\mathbf{w}$ is usually calculated using the Euclidean distance

$$d(\mathbf{v},\mathbf{w}) = ||\mathbf{v}-\mathbf{w}|| = \sqrt{\sum_{i=1}^{n}(v_i - w_i)^2}. \tag{1}$$

Thus, the network can be trained by feeding positive and negative examples for each sample and class and adapting the weights to minimize or maximize the distances accordingly. One loss function that combines this functionality into a single function to be optimized is the triplet loss [10]

$$L(A,P,N) = \max(||f(A) - f(P)||^2 - ||f(A) - f(N)||^2 + \alpha, 0). \tag{2}$$

Here, the embedding of the input sample $f(A)$ (anchor) is compared to the embeddings of a positive sample $f(P)$ of the same class and a negative one $f(N)$. The difference between the distances has to be at least equal to the margin α for the loss to be zero, otherwise it is positive. Therefore, the Siamese network is trained by minimizing the triplet loss function.

3.2 Training Data

Measured and labelled data is not available in the required amount since deep learning algorithms require several hundred training examples. Thus, we rely on synthetic XRD data, which we simulate based on a database of measured crystallite properties and physical principles, perfectly imitating measured diffraction data [12]. By simulating the XRD scans, we ensure that the training data contains the relevant deviations of diffraction peaks demonstrated in Figure 2b and every class is represented with the same number of scans. Accordingly, we assemble two datasets:

For the first dataset A, we select 100 random crystallite materials from a reference database (which contains 345 materials in total) and simulate 50 variations for each, resulting in 5000 total synthetic XRD scans. The training, validation and test set are made up with a 50-20-30 split, so for every class 25

variations are used for the training set, 10 for the validation set and 15 for the test set. In total, dataset A contains 2500 samples in the training set and 1000 and 1500 samples in the validation and test set respectively. Accordingly, we ensure that no class is under-represented in any of the sets.

Secondly, we set up dataset B in the same way. We select another 100 random crystallite materials without any overlap between the classes of datasets A and B. Again, we split the 5000 synthetic XRD scans into training, validation and test data using the 50-20-30 split.

Using datasets A and B, we evaluate the performance of a classifier and Siamese networks, as well as transfer learning and the transferability of the Siamese network. Hence, we train the classifier and Siamese network with the train set of dataset A and choose the weights that perform best on the validation set to avoid overfitting. Afterwards, we assess the performance of the VGG16-like classifier and the likewise Siamese network for dataset A using the test set. For the Siamese network, we pick a random scan of the train or validation set as the reference input for each class to calculate the distances for the test samples. As a performance metric, we choose the Top-1 and Top-3 accuracy for the highest prediction scores of the classifiers and the smallest distances for the Siamese network.

Subsequently, we apply the trained Siamese network for the test set of dataset B without retraining. Before we evaluate the performance of the classifier for dataset B, we retrain the network using the two previously described tactics to evaluate the transfer learning approach. Finally, we rate the ability of the Siamese networks for transfer between datasets in comparison to the classifiers performance with full-retraining or transfer learning as an alternative.

4 Results

After training the classification and Siamese network using the train and validation XRD scans, we evaluate the Top-1 and Top-3 accuracy for the 1500 diffraction patterns of the test set. Table 1 presents the performance scores for both networks. Overall, we achieve near perfect results for the classifier with

Table 1: Comparison of Top-1 and Top-3 performance between the VGG16-like classifier and Siamese network for dataset A.

	Network Type	
Accuracy	Classifier	Siamese
Top-1	99.9%	97%
Top-3	100%	99.9%

a 99.9% Top-1 accuracy and can be sure that the correct phase is within the three highest predicted classes (100% Top-3 accuracy). The performance of the Siamese network is slightly worse than the classifier, with scores of 97% and 99.9% Top-1 and Top-3 accuracy, respectively. The results of the Siamese network suggest that there are materials with very similar diffraction patterns in the dataset, which the Siamese network can hardly distinguish. Accordingly, we obtain an almost perfect Top-3 accuracy, because the network does not have to decide correctly. In contrast, the classification network is able to distinguish the subtle differences by means of the dense layers, which means that there is hardly any difference between Top-1 and Top-3 accuracy.

Next we evaluate dataset B, for which we can use the Siamese network without having to retrain. The classification network is retrained using the full retraining and transfer learning approach, where we freeze the weights of the convolutional layers during the training process and only fine-tune the dense layers. Table 2 shows the prediction scores for the two retraining strategies of the classifier and the Siamese network without retraining. For the full retraining approach of the classifier, we achieve almost identical accuracies compared to dataset A, with 98.7% and 99.9% respectively. The slightly worse Top-1 accuracy indicates that the phases of dataset B are harder to distinguish from each other. Notably, the Siamese network achieves an almost identical performance for dataset B with a 99.3% Top-3 accuracy, although it was trained with phases of dataset A only.

However, the transfer learning strategy is only partially suitable in the case of transferring XRD datasets. Although the classifier still manages to predict the correct phase in about 70% of all cases, it is significantly worse than the full retraining approach. This leads to the conclusion that the weights of the

Table 2: Comparison of Top-1 and Top-3 performance between the VGG16-like classifier and Siamese network for dataset B.

	Network Type		
	Classifier		Siamese
Accuracy	Full Retraining	Transfer Learning	
Top-1	98.7%	70.4%	93.5%
Top-3	99.9%	70.7%	99.3%

convolutional layers were strongly specialized to emphasize the peculiarities of dataset A. Consequently, the network is no longer able to highlight the differences in the phases of dataset B in the transfer learning approach, so that the dense layers can use them for the classification.

Furthermore, the classifier requires more time to be applied to dataset B, regardless of the retraining strategy. Firstly, 2500 synthetic training scans (+ 1500 validation scans to avoid overfitting) must be generated so that the network learns the classification rules for the materials of dataset B. In comparison, the Siamese network requires only one reference per material, so 100 synthetic scans. Secondly, the retraining process for the classifier takes some time[1], while the Siamese network is applied to the second dataset without retraining.

Overall, the Siamese networks prove that they are well suited for the use with XRD data and can also be used for materials that were not represented in the training dataset. Since the diffraction patterns of identical materials can differ significantly, as shown in Figure 2b, it is of interest to know how the Siamese network processes the XRD scans to achieve the small distance between the embedding vectors of identical materials. Hence, we visualize two embedding vectors for XRD scans of the same material that differ in peak position, shape and intensity in Figure 5. Interestingly, the Euclidean distance between the two

[1] The duration of the training process is strongly dependent on the utilized computing power. Since the Siamese networks do not require to be retrained, we refrain from comparing the absolute training times.

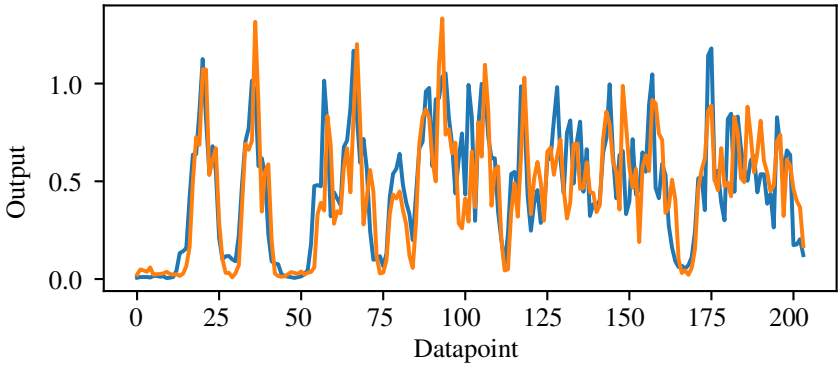


Figure 5: Embedding vector of the Siamese network for two XRD scans of the identical material.

vectors is definitely not zero, as there are clear differences between the network outputs. The output of the network reminds of the input XRD scans, which, however, are shrunk in spatial resolution by the Max Pooling layers. This presumably enables the network to compensate for the possible shifts of the diffraction angles, although the calculated peaks are not completely aligned.

5 Conclusion and Outlook

In summary, we employ Siamese neural networks for the first time with one-dimensional XRD data to distinguish between materials. Therefore we trained a reference classification network on two synthetic datasets and show that the Siamese Network achieves almost identical results. The strength of the Siamese network lies in the fact that it can be applied to materials not included in the training dataset without the need for retraining of the network (in comparison to the classification network). Accordingly, we demonstrate that the Siamese networks are particularly suitable for use in the field of material sciences, where the classes are dynamically composed. The comparable transfer learning approach has not proven to be adequate in our case because the features learned in the classifier were too specific for transferability.

Lastly, we showed how the neural network computes the embedding vectors, and that the Siamese network is not yet able to fully compensate for the dif-

ferences within diffraction patterns of the same material. Therefore, it has to be investigated whether the distance calculation using the pairwise, Euclidean distance for the one-dimensional data can be replaced by a more suitable alternative. In the field of time series analysis, for example, there is a distance calculation using so-called Dynamic Time Warping, which compensates for small displacements on the time axis. It is possible that such a function enables the network to compensate for the differences even better and thus differentiates the materials even more clearly.

The next step is to validate our results with measured XRD scans. While it is necessary that as many real possible variations as possible are represented in the training data and hence, synthetically generated data are possibly essential for the training of a Siamese network, the trained deep learning model should be able to use measured XRD scans as a reference input.

Moreover, we have trained Siamese networks using XRD data as an example, but a deployment with other one-dimensional data would also be imaginable. Especially for similar analytical methods like NMR, for which a classifier based on a neural network has already been developed, a transfer of our results would be conceivable. Accordingly, the next step is to test our Siamese network approach with other one-dimensional data.

References

[1] V. Pecharesky and P. Zavalij. *Fundamentals of Powder Diffraction and Structural Characterization of Materials*. Springer Science & Business Media, 2005.

[2] V. Stanev, V. Vesselinov, A. Kusne, et al. Unsupervised phase mapping of x-ray diffraction data by nonnegative matrix factorization integrated with custom clustering. *npj Computational Materials*, 4:1–10, 2018.

[3] H. Wang, Y. Xie, D. Li, et al. Rapid identification of x-ray diffraction patterns based on very limited data by interpretable convolutional neural networks. *Journal of Chemical Information and Modeling*, 60:2004–2011, 2020.

[4] Bruker Biospin Corporation. Deep learning applications in nmr spectroscopy. In *European Magnetic Resonance Meeting*, 2019.

[5] W. Park, J. Chung, J. Jung, et al. Classification of crystal structure using a convolutional neural network. *IUCrJ*, 4:486–494, 2017.

[6] F. Oviedo, Z. Ren, S. Sun, et al. Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks. *npj Computational Materials*, 5:1–9, 2018.

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[8] S. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[9] J. Bromley, I. Guyon, Y. LeCun, et al. Signature verification using a Siamese time delay neural network. In *Advances in neural information processing systems*, 1994.

[10] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[11] C. Zhang, Y. Idelbayev, N. Roberts, et al. Small molecule accurate recognition technology (SMART) to enhance natural products research. *Scientific Reports*, 7(1):1–17, 2017.

[12] J. Schuetzke. Evaluation of machine learning approaches for crystalline phase identification. *Master's Thesis, Karlsruhe Institute of Technology*, 2019.

CAD2Real: Deep learning with domain randomization of CAD data for 3D pose estimation of electronic control unit housings

Simon Bäuerle[1,2], Jonas Barth[2], Elton Tavares de Menezes[2], Andreas Steimer[2], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics,
Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Robert Bosch GmbH
E-Mail: {simon.baeuerle, ralf.mikut}@kit.edu

Abstract

Electronic control units (ECUs) are essential for many automobile components, e.g. engine, anti-lock braking system (ABS), steering and airbags. For some products, the 3D pose of each single ECU needs to be determined during series production. Deep learning approaches can not easily be applied to this problem, because labeled training data is not available in sufficient numbers. Thus, we train state-of-the-art artificial neural networks (ANNs) on purely synthetic training data, which is automatically created from a single CAD file. By randomizing parameters during rendering of training images, we enable inference on RGB images of a real sample part. In contrast to classic image processing approaches, this data-driven approach poses only few requirements regarding the measurement setup and transfers to related use cases with little development effort.

1 Introduction

An exemplary use case for our approach is the 3D pose estimation of electronic control units (ECUs). The pose of each individual ECU needs to be detected robustly to enable an automated application of sealing materials. Generally, automated image processing is widely used in industrial series production [1, 2]. A commonly used method to detect ECU poses is by applying classic image processing. In some cases, these algorithms rely on fiducial marks imprinted on parts themselves. Setting up this image processing pipeline needs to be done by experts individually for each new product. Substituting classic image processing by fully automatically designed deep learning on our task can potentially save a significant amount of development effort for new product designs. Furthermore, ANNs generally impose much lower requirements regarding camera resolution and surrounding conditions, enabling simpler measurement setups. Adding fiducial markers in industrial applications “may be undesirable” [10]. Dropping the need for fiducial markers prevents changes on the product design, which would involve product engineers. However, deep state-of-the-art ANN architectures require a large number of labeled images, which are usually not available for ECUs. In contrast to real-world images, rendered CAD images are a widely available data source in industrial settings. A network trained solely on CAD data cannot be directly applied to real images though, since those are different with respect to pixel color values (see [3]). This domain gap is generally present on many different settings that involve ANNs. Techniques to overcome this domain gap are called domain adaptation and are subject to current research (e.g. [4, 3]). Instead of adapting the ANN to a different domain, an approach can also include adaptation of the domain itself: Images from the training domain can be randomized to such an extent, that the real-world domain is "just another variation" [5]. This method is called domain randomization and has been tested successfully on other use-cases such as indoor drone flight [6] or robotic grasping and manipulation [5, 7, 8, 9, 10]. Sundermeyer et al. are working on pose estimation by using a denoising autoencoder architecture [11, 12]. In contrast to Sundermeyer et al., we are using state-of-the-art ANN architectures and evaluate different randomization parameters. Tremblay et al., Khirodkar et al. and Hinterstoisser et al. are using domain randomization for object detection [13, 14, 15]. Khirodkar et al. focus

on the use case of detecting cars and also includes a pose estimation. Domain randomization is not limited to image data however. For example, Peng et al. have randomized the dynamic properties of their simulation model to transfer a robotic control algorithm trained by deep reinforcement learning to the real world [16].

In contrast to our setup, pose estimation approaches like BB8 [17], SSD-6D [18] or PoseCNN [19] employ further pose refinement to improve accuracy [20, 21]. Tekin et al. [20] and Do et al. [21] use standardized datasets, which does not target the domain gap that is widely present in real-world use cases. Kleeberger et al. also use domain randomization for pose estimation, but uses depth data instead of RGB images [22].

The generally very promising results on related use cases motivate the deployment of deep ANNs for pose estimation of ECUs. In Section 2 we outline our approach in a general way. This description is made on an abstract level without implementation details. It serves as a template, which can easily be applied to similar use cases. Subsequently, the experimental setup as described in Section 3 includes the details. In contrast to the previous section, we set out specific implementation aspects. A detailed overview is given e.g. over the parameters that we randomize and the way we set up the training datasets. Results for our use case are presented in Section 4. Performance during inference is listed for the different training datasets. We include an estimation of how much errors differ from their mean values. In Section 5 we discuss the results. We analyze the effects of different randomizations of training data. Furthermore, we compare this data-driven approach against classic image processing setups, with a focus on the possible impact on future manufacturing setups. Eventually, in Section 6 the most relevant aspects are summarized and a detailed outlook onto further research opportunities is given.

2 Methods

Figure 1 gives an overview of our approach. To cope with the domain gap between simulated and real-world images we use domain randomization. Unlike domain adaptation methods, domain randomization does not adapt the ANN

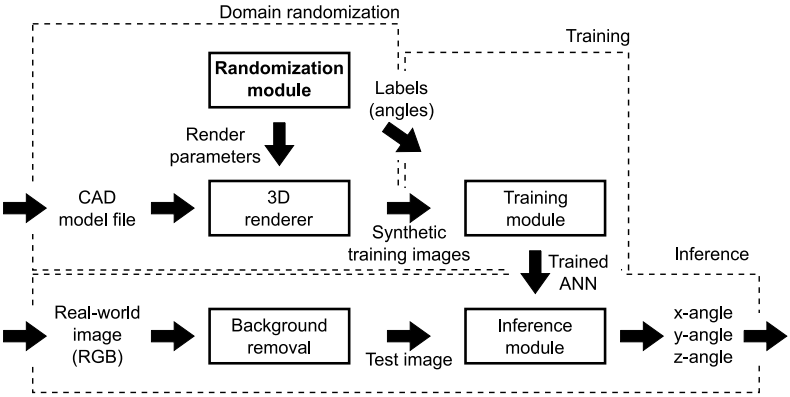


Figure 1: Overview of our approach

to a different domain. Instead, it rather adapts the training domain itself. Our training domain consists of CAD images, which are rendered with arbitrary settings. We generate different datasets of synthetic images automatically. We train state-of-the-art ANNs on synthetic datasets and perform inference on real-world images.

2.1 Domain randomization

We created a generic pipeline for applying domain randomization as depicted in the upper left area of Figure 1 (our specific implementation details are outlined in Section 3). CAD files are loaded into a 3D rendering software suite. A common exchange format is used for transferring CAD files. Geometric features are not changed in any way. Within the 3D renderer, several randomizations are applied. Specifically, we randomize shadows, translations and gray tones. Those are all parameters, which are not causally connected to the labels. For example, gray tones have no causal connection to rotation angles and must not affect inference. The pose of the model is set to a random angle configuration. An arbitrary number of angle configurations with individual randomizations is rendered and output to image files. Randomization parameters and rotation angles (=labels) are set via a scripting interface. Datasets with different randomization parameters can be created in any desired number automatically. This

kind of domain randomization is used to close the domain gap from training on CAD data to inference on real-world data (CAD2Real), saving the need for any labelled real-world images. Once finalized, this setup can be used for different products with minimum effort by replacing the CAD model and rerunning the script. Images for our baseline dataset *CAD_unchanged* are created with this method as well by simply omitting the randomizations.

2.2 ANN inference on real-world images

Real-world images are automatically preprocessed and then fed into the trained ANN. As shown in the lower part of Figure 1, the ANN infers all three rotation angles directly from real-world RGB images. We use a fully automatized preprocessing step to replace the background with a uniform gray tone.

3 Experimental Setup

We described our approach generically in Section 2. Here, we provide specific implementation details. We outline the creation of our different datasets used for training and during inference. Details regarding the ANN are also provided below.

3.1 Datasets

We work with two general types of datasets. On the one hand we have different training datasets. Those are solely based on CAD data and generated automatically. On the other hand we use experimental data for testing purposes. This experimental data is captured from real product samples in a laboratory setting.

Table 1: Parameters and their respective ranges

Parameter	Range
X-/y-angle	[-15 ^{o}, 15 ^{o}]
Z-angle	[-45 ^{o}, 45 ^{o}]
Part translations	[-1.5 cm, 1.5 cm]
Camera translations	[-1.5 cm, 1.5 cm]
Gray tones	[0.05, 0.8]
Light positions	$x \in$ [-1 m, 1 m] $y \in$ [-1 m, 1 m] z = f(x, y)

3.1.1 Training sets

For creation of our training datasets we use the 3D rendering software *Blender*. Blender is open-source software and freely available. It includes a powerful Python API that enables control via automatic scripts. The image generation pipeline is generally depicted in Figure 1. We import the CAD model as STEP-file, a data-format commonly used for CAD data exchange. The geometry itself is not modified. Rotations are applied around the x-/y- and z-axis within a range of -15 to 15 degrees for the x- and y-axis and a range of -45 to 45 degrees for the z-axis. The rotation angles serve as labels and are therefore stored for each image created. Random translations of the model along the x- and y-axis are optionally applied within a range of -1.5 cm and 1.5 cm. Random translations of the camera along the z-axis are optionally applied within a range of -1.5 cm to 1.5 cm. One light source is placed high above the model, emitting uniform lighting. Two additional light sources are optionally included as well. Those are placed randomly above the model, generating random shadows. The color of the model itself is randomized in uniform tones of gray. We are aware that the introduction of additional light sources also makes the part appear brighter. Therefore, we try to limit this effect by keeping the distance of both additional lights to the part on a constant value. For each randomly drawn x- and y-coordinate the z-coordinate is calculated so that the distance to the part

is always equal, effectively placing both lights on an imaginary sphere. All parameters with respective ranges are listed in Table 1.

We are using the training sets described below. During evaluation we compare our domain randomization approach against two datasets:

- *CAD_unchanged*: Images are labeled with rotation angles around x-/y- and z-axis. There are no further modifications made.
- *CAD_augmented*: We take the images from the set *CAD_unchanged* and apply random modifications. We modify brightness, translations and zoom. This data augmentation is applied to already rendered images. Data augmentation of this kind is usually used when not enough labeled training data samples are available.

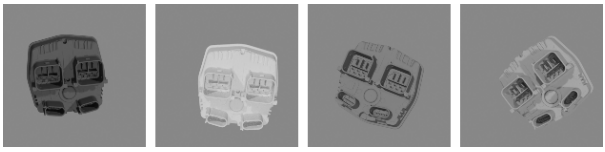
The following datasets include domain randomization. To test the influence of different parameters, we modify the extent of our randomization. Samples for each dataset are shown in Figure 2.

- Dataset *Rand_full* includes all randomizations described above.
- Dataset *Rand_noShadows* is the same as *Rand_full* except for the two additional light sources. Thus, no random shadows are included.
- Dataset *Rand_noTranslations* is the same as *Rand_full* except for the translations of the part and the camera. Part and camera remain at constant positions.
- Dataset *Rand_noGraytones* is the same as *Rand_full* except for randomizing the uniform gray tones. Part color is only affected by the position of both randomly placed lights.

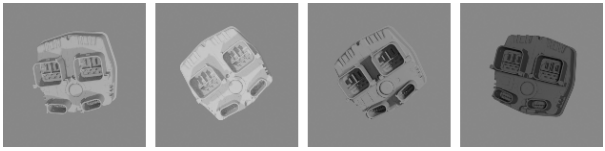
For each dataset described here, we have created 100 000+ training images.

(a) *CAD_unchanged*

(b) *CAD_augmented*

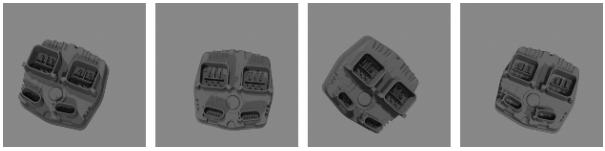


(c) *Rand_full*



(d) *Rand_noTranslations*

(e) *Rand_noShadows*



(f) *Rand_noGraytones*

Figure 2: Example images from training datasets

Figure 3: Example images from our real-world dataset with green background

3.1.2 Real-world dataset

We outline the creation of our real-world dataset, which is used as test set during inference. We use a single sample part from an ECU that is already available during prototype phases (before starting series production). We use the following laboratory setup: The images are recorded with a common SLR camera, since there are no specific requirements regarding the camera. Image resolution is later scaled down during pre-processing to below 300x300 pixels. The camera is mounted onto a fixed frame with two lighting sources on either side. The sample part is placed 0.5 m below the camera on a green cardboard layer. To introduce rotations around the horizontal x- and y-axis we are using 3D-printed wedges. We use multiple wedges with slopes of 2.5, 5 and 10 degrees. Placing the wedges below our sample introduces the respective rotations. We recorded 20 different angle configurations, examples are shown in Figure 3.

3.2 Artificial Neural Network: Training and inference

We use the state-of-the-art architecture *InceptionV3* [23], with weights pre-trained on ImageNet. The architecture including its weights can be imported within Keras [24] in two lines of code. To adjust to our number of labels we append a dense layer with three neurons. Pose estimation is sometimes implemented as classification, with a binning of rotation angle intervals. Binning limits the resolution of angle values to discrete intervals. To avoid this, we

opt for using regression as proposed by e.g. Mahendran et al. [25] instead. We use a linear activation function within the last layer, directly outputting continuous rotation angles around the x-/y- and z-axis. Each ANN is trained on a dataset as described above. We train on each dataset with 10 000 randomly drawn samples for 50 epochs. First runs have shown slightly differing results when re-drawing the samples and re-starting training. Therefore, we execute 30 independent runs on each dataset. For final evaluation we use error metrics as described below. The lower area of Figure 1 shows our setup for inference with the trained ANN. We use preprocessing of the real-world images for removing the background. Since the images are taken on green background, preprocessing can be done automatically without much effort.

3.3 Evaluation metrics

In this section we describe the metrics used for evaluation later on. First, we focus on the evaluation of the training set characteristics (to get an impression of "which training set is best"). We then also estimate the confidence of statements that are made on our limited number of real-world images. For a single test or validation image, an ANN outputs three distinct angle values $\hat{y}_i, i \in \{1,2,3\}$. These are compared against the corresponding ground truth angle values $y_i, i \in \{1,2,3\}$ by calculating error

$$e_i = \| \hat{y}_i - y_i \|_1 \, . \tag{1}$$

Each single ANN is evaluated on validation and test data. We merge all angle values ($\hat{y}_i$ and y_i) into one vector per dataset ($\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$). This is done on validation and test data separately for each ANN. Errors are then averaged over all angles and the respective number of images, yielding an mean error per ANN of

$$\overline{e}_{ANN} = \frac{1}{3N} \| \hat{\boldsymbol{y}} - \boldsymbol{y} \|_1 \, . \tag{2}$$

N is the number of images in either the validation ($N = 500$) or test set ($N = 20$). Since we assess the fitness of the training set characteristics, we calculate the mean error per training set with

$$\overline{e} = \sum_{j=1}^{M} \overline{e}_{ANN,j}, \tag{3}$$

with $M = 30$ ANNs for each training dataset. We further calculate the standard deviation per training set

$$s = \sqrt{\frac{1}{M-1} \sum_{j=1}^{M} (\overline{e}_{ANN,j} - \overline{e})^2}. \tag{4}$$

This yields a standard error of

$$s_{\overline{e}} = \frac{s}{\sqrt{M}}. \tag{5}$$

We further calculate a margin of error for $\overline{e}$. Since our sample size is limited, we use the Student's t-distribution instead of the normal distribution. For our sample size of $M = 30$ and a confidence level of 99% we calculate the margin of error for $\overline{e}$ as

$$MOE = \pm t_{M-1} s_{\overline{e}}, \tag{6}$$

with $t_{M-1} = 2.76$. When working with a normal distribution instead, t_{M-1} would be replaced by z_c with a value of 2.58 for the 99% confidence level (t_{M-1} converges towards z_c for very large sample sizes). The value of t_{M-1} is retrieved from a table for the Student's t-distribution (see e.g. [26], p.206) and depends on the sample size and the confidence level. For our sample size of $M = 30$, we need a t-value that corresponds to M - 1 = 29 degrees of freedom. The values of the distribution function in the table can be used directly when working with a one-sided interval. Our two-sided interval gives an estimation into both directions (upper and lower bound). Therefore, we record t for a value of 0.995 for the distribution function to account for the two-sided interval. Up to now, we have mainly looked at how results differ when re-drawing samples from the training set and retraining the ANN. The limited number of images within the test set is another factor that might affect our results. For our limited

Table 2: Mean error on validation data and real-world data

Training dataset	Validation data $\overline{e}$ [^{o}]	Real-world data $\overline{e}$ [^{o}]
CAD_unchanged	0.4 ± 0.04	11.7 ± 2.4
CAD_augmented	0.4 ± 0.05	2.5 ± 0.6
Rand_full	0.5 ± 0.06	$\mathbf{1.5 \pm 0.2}$
Rand_noTranslations	0.5 ± 0.07	7.7 ± 2.8
Rand_noShadows	0.5 ± 0.1	3.6 ± 0.8
Rand_noGraytones	0.4 ± 0.09	$\mathbf{1.2 \pm 0.2}$

amount of real-world samples we take a similar approach as above, but now on a more isolated scope. For the first ANN trained on dataset *Rand_full*, we evaluate the mean error $\overline{e}_{test}$ for angles around the x-axis on the test set containing 20 images. This is the same calculation as presented in Equation (2), but discarding angles around y- and z-axis. For those 20 error values e_i we also calculate the standard deviation

$$s_{test} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(e_i - \overline{e}_{test})^2} \tag{7}$$

and the margin of error

$$MOE_{test} = \pm t_{N-1}\frac{s_{test}}{\sqrt{(N)}} \tag{8}$$

with $t_{N-1} = 2.86$ for $N = 20$ and a 99% confidence level.

4 Results

We train 30 ANNs on each of the datasets described in Section 3.1. We evaluate the mean absolute error and include a margin of error as described in Section 3.3 for test and validation data on each dataset. Results are shown in Table 2. The outer left column indicates the dataset used during training. We then eva-

luate the performance on validation data and real-world data. Validation data images are from the same domain as the images used during training. Real-world data is taken from product samples and therefore substantially different. This is our target domain used for testing. ANNs trained on *CAD_unchanged* exhibit the lowest error on the validation set, but insufficient performance on real-world test images. For *CAD_augmented* the mean absolute error on real-world images is lower by a factor of approximately five. Another improvement by another factor of almost two over *CAD_augmented* is gained by using *Rand_full*: Angles around all axes are inferred with an mean error of 1.5 degrees. *Rand_full* has full randomization applied. For *Rand_noTranslations* and *Rand_noShadows* we note an error-score inbetween *CAD_unchanged* and *CAD_augmented*. Dropping the randomization of translations affects performance worse than dropping randomization of shadows. Training on *Rand_noGraytones* gives slightly better results than on *Rand_full*, but only by a small margin. Errors on the validation set increase from *CAD_unchanged* to *CAD_augmented* and further rise for the randomized datasets. All randomized datasets show similar errors on validation data. We now take an isolated look onto the limited number of real-world samples as described in Section 3.3. We evaluate the error for rotations around the x-axis only and look at a single ANN trained on *Rand_full*. For our 20 real-world samples the ANN inference has a mean error of $\mathbf{1.6 \pm 0.4}$ **degrees**. This margin of error is calculated for a confidence level of 99%. Our experimental setup as described in Section 3.1 has measurement errors which affect the ground truth labels. All results presented above are naturally limited to measurement tolerances.

5 Discussion

The results presented in Section 4 show a consistent advantage by training on randomized datasets for our application of pose estimation of ECUs. In the later part of this section we also discuss the impact of this research direction on image processing setups in related product applications. But first we look more closely at the effects of how the datasets were set up. First of all, the insufficient performance with the dataset *CAD_unchanged* is not surprising. In this case, the training images differ a lot from the real-world images. This

can be interpreted as a "wide" domain gap, leading to poor transferability from source domain to target domain. A significantly improved performance on real-world images is achieved by applying state-of-the-art data augmentation to the training set. Data augmentation is commonly used to expand the training set size. This is especially useful when dealing with a limited amount of labeled training data. We believe that there is a second benefit of data augmentation. Augmenting training data with changing brightness or translations also increases the diversity within the training set. Increased diversity of features not relevant for inference favors transferability from source to target domain. This effect is exactly the underlying idea of domain randomization. Data augmentation therefore can be seen as a "light version" of domain randomization. With full domain randomization applied, inference quality is further increased by another factor of approximately two. In comparison to data augmentation, domain randomization introduces even more diversity to the training set. This time, the introduced diversity goes beyond simply adjusting images. Modifications of this kind cannot be easily applied to raw images. This is especially clear for the randomization of shadows. Calculating the position and intensity of shadows is an integral part during rendering and not easily possible when working on two-dimensional image data only. Also, translations made within the renderer lead to different outcomes compared to augmentation by translations as well. Translations applied during state-of-the-art data augmentation will not change camera perspective. In contrast, inside the renderer not only the part position changes, but also the perspective view of the part changes. Translations in data augmentation therefore are different from those in domain randomization. However, we believe that the major error reduction is achieved simply by the fact that additional translations are introduced, no matter whether perspective changes or not. The poor performance with the dataset *Rand_-noTranslations* especially motivates the introduction of translations. To get a better impression of the different aspects of domain randomizations, in addition to dropping the added translations in *Rand_noTransla- tions* we dropped the shadow randomization in *Rand_noShadows* and the gray tone randomization in *Rand_noGraytones*. The performance with *Rand_noShadows* is worse compared to *Rand_full* and *CAD_augmented*. It seems that shadows and translations are both relevant factors when dealing with the present domain gap. However, dropping the randomization of gray tones in *Rand_noGraytones* has not caused

deteriorating performance, but even shows slight improvements compared to *Rand_full*. Since we trained 30 different ANNs we do not believe this effect is caused by chance. A possible explanation is that by randomization of gray tones many gray tones are outside of a usable scope (e.g. too light or too dark). This leads to a smaller part of training set being useful for inference, since some images are "too far off". Also, we want to mention that the effects of introducing random shadows and random gray tones overlap in some sense. Both affect the color of the part at a certain position and are not entirely independent of each other. In our opinion, the unexpected behavior on the dataset *Rand_noGraytones* does not hurt the idea of domain randomization in general. It rather motivates further studies on the unique effects of different randomization types. Instead of randomizing gray tones only, e.g. textures could be introduced as well.

Also, the varying error on the validation set from CAD datasets to randomized datasets motivates the analysis of different hyperparameters, mainly training set size and the number of epochs. The most efficient hyperparameters are generally likely to be different depending on the randomization type and extent. With domain randomization we desire to cover all aspects during rendering that make the real-world images different from the CAD images. This does not mean representing reality in simulation exactly however. For example when dealing with differing textures of the part in the real-world, applying textures with random noise to the simulation might be sufficient. The successful application of domain randomization on this use case shows high potential for future setups of image processing pipelines in series production of ECUs. The pipeline that we used can easily be transferred to other products. Other products may be manufactured on different production lines. Subsequent processing of pose information varies between products or production lines. To standardize e.g. the naming of parameters on these interfaces and during further processing steps, an ontology-based approach is useful. Zhou et al. [27] and Svetashova et al. [28] have applied ontologies to other production processes successfully. This approach not only helps during technical setup, but also enables a common understanding of process-specific details across all involved persons [27, 28]. In contrast to many algorithms of classic image processing, our pose estimation approach is not bound to specific product

features. Further improvements aimed at improving inference accuracy can also maintain this product-independent aspect. This advantage is based on the fact, that the features needed for inference are learned by the ANN during the training process and not manually tuned. This data-driven approach has the advantage that for a different problem setting only the problem-specific training data needs to be supplied. We use only data sources that are available without major effort. With our approach, the problem-specific training data can be created automatically from a single CAD model file. Once again we emphasize that training is done on purely synthetic data. Not a single real-world image is needed during training. We see a high potential for a significant reduction of development effort in future image processing setups.

6 Summary and Outlook

We have set out to analyze the applicability of domain randomization to our use case of pose estimation of ECUs. Our goal was to minimize the domain gap and to deploy an ANN trained solely on synthetic data to real-world images. We have shown that applying domain randomization exceeds the effect of data augmentation by a factor of around two. The mean error for inferred rotations around all three axes is only 1.2 degrees on real-world images. The entire pipeline for creating randomized training datasets and training the ANN is fully automatized. The only input needed for creation of all training data is a single CAD model file, which is readily available for all ECUs. We use only a state-of-the-art ANN architecture with a minor adjustment regarding the output dimensionality. Training is done end-to-end, we infer all rotation angles directly from RGB images. No further depth data is needed. We have analyzed the application to our use case and motivated further research directions. The following aspects could make this approach fit for application in production of ECUs:

- We focused on detecting part rotations. For application in series production a post-processing step to determine translational degrees of freedom needs to be appended. Including the translations directly into the labels as well might also be a feasible approach for our use case (directly inferring 6D pose information).
- Our pre-processing currently requires a green-colored background. Randomizing the backgrounds as done in other use cases [5, 10, 13, 11] could make our approach feasible for a background containing workpiece carriers. This would drop the need for using any pre-processing at all.
- We have provided insights into the effect of different randomizations. To further improve accuracy, these influences need to be examined in more detail. Ideally, this simultaneously includes adjustment of hyperparameters for training the ANN as well.

The successful execution of the steps outlined above can reduce the entire pipeline for 6D pose estimation to solely a state-of-the-art ANN architecture. These architectures are conveniently available within the Keras library. This would provide a fully automated pipeline for pose estimation of new ECUs and similar products.

References

[1] A. Korodi, D. Anitei, A. Boitor, and I. Silea, "Image-processing-based low-cost fault detection solution for end-of-line ECUs in automotive manufacturing," *Sensors*, vol. 20, no. 12, p. 3520, 2020.

[2] S.-H. Huang and Y.-C. Pan, "Automated visual inspection in the semiconductor industry: A survey," *Computers in Industry*, vol. 66, pp. 1–10, 2015.

[3] E. R. Tavares de Menezes, "Machine learning and classic image processing - extraction of 3D information from real-world 2D images

of electronic part housings," Masters thesis, TH Köln University for Applied Sciences, 2019.

[4] M. Böhland, T. Scherr, A. Bartschat, R. Mikut, and M. Reischl, "Influence of synthetic label image object properties on GAN supported segmentation pipelines," in *Proceedings 29th Workshop Computational Intelligence*, pp. 289–305, KIT Scientific Publishing, 2019.

[5] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.

[6] F. Sadeghi and S. Levine, "Cad2RL: Real single-image flight without a single real image," arXiv preprint 1611.04201, 2016.

[7] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," arXiv preprint 1809.10790, 2018.

[8] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, and others, "Solving rubik's cube with a robot hand," arXiv preprint 1910.07113, 2019.

[9] S. Grün, S. Höninger, P. M. Scheikl, B. Hein, and T. Kröger, "Evaluation of domain randomization techniques for transfer learning," in *2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 481–486, IEEE, 2019.

[10] X. Ren, J. Luo, E. Solowjow, J. A. Ojea, A. Gupta, A. Tamar, and P. Abbeel, "Domain randomization for active pose estimation," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019 pp. 7228–7234, IEEE, 2019.

[11] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 699–715, 2018.

[12] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel, "Multi-path learning for object pose estimation across domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13916–13925, 2020.

[13] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977, 2018.

[14] R. Khirodkar, D. Yoo, and K. Kitani, "Domain randomization for scene-specific car detection and pose estimation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1932–1940, IEEE, 2019.

[15] S. Hinterstoisser, O. Pauly, H. Heibel, M. Martina, and M. Bokeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[16] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[17] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2017.

[18] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1521–1529, 2017.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," arXiv preprint 1711.00199, 2017.

[20] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 292–301, 2018.

[21] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6Dpose: Recovering 6D object pose from a single RGB image," arXiv preprint 1802.10367, 2018.

[22] K. Kleeberger and M. F. Huber, "Single shot 6D object pose estimation," arXiv preprint 2004.12729, 2020.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] S. Mahendran, H. Ali, and R. Vidal, "3D pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2174–2182, 2017.

[26] J. Puhani, *Statistik: Einführung mit praktischen Beispielen*. Springer Fachmedien Wiesbaden, 2020.

[27] B. Zhou, Y. Svetashova, S. Byeon, T. Pychynski, R. Mikut, and E. Kharlamov, "Predicting quality of automated welding with machine learning and semantics: A Bosch case study," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, p. 8, 2020.

[28] Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, and E. Kharlamov, "Ontology-enhanced machine learning pipeline: A Bosch use case of welding quality monitoring," in *The Semantic Web - ISWC 2020*, 2020.

Application of various balancing methods to DCNN regarding acoustic data

Dominic Schneider[1], Manuel Schneider[1], Maria Schweigel[1], Andreas Wenzel[1,2]

[1]Embedded Diagnostic Systems, Faculty of Electrical Engineering,
Schmalkalden University of Applied Sciences
Blechhammer 9, D-98574 Schmalkalden, Germany
E-Mail: d.schneider@hs-sm.de

[2]Fraunhofer IOSB, IOSB-AST Ilmenau, Fraunhofer Institute of Optronics,
System Technologies and Image Exploitation
Am Vogelherd 90, D-98693 Ilmenau, Germany
E-Mail: andreas.wenzel@iosb-ast.fraunhofer.de

Abstract

This paper describes the application and effects of different balancing methods on the learning behaviour and quality of a DCNN using acoustic data. The aim is to show to what extent these methods have positive as well as negative effects on the use case of the audio data. The evaluation is based on synthetic audio data with multiclass characteristics, because an overlay of effects should be avoided. This serves as preliminary work in order to apply the methodology to the measurement data for the classification of knife sharpness in forage harvesters in later investigations. According to applied balancing methods, the data are represented to the DCNN. The performance and quality shall be measured by formal qualification criteria. It turned out that SMOTE gives the best and most robust results. It shows a higher convergence compared to the other methods. Furthermore the worst results are produced with untreated raw data.

1 Introduction

The application of audio data as learning data set to neural networks can be subject to a wide range of use cases. The use case to be addressed in this paper is classification. This means, in supervised learning, that given class data is learned and subsequently the developed classifier can independently divide represented data into classes [1]. It also means that the aim is to assign the entered data to a class in the neural network. In real use cases there are usually problems with more than two classes. These are defined as multi-class problems. It can also happen that data sets acquired by measurements show unequally distributed class representations. These data sets are called unbalanced. The problem with these lies in the under-represented data, which can lead to a shift in recognition accuracy towards the over-represented data. There are now various methods for dealing with these unbalanced data. The influence of balancing methods has already been investigated on different data and learning structures.[2] They show that there is a probably existing influence on learning behaviour. Thus, the actual goal is to find an optimal balancing method for the real use case of the acoustic data of forage harvesters. This is also a classification problem and should be able to draw conclusions about the sharpness of the knives based on the structure-borne noise of the cutting drum. These audio data are measured values. In order to avoid an accumulation of effects in the evaluation, the effects will be investigated in this paper using synthetic data. This creates a basis which can be applied to the measured data in subsequent work.

The basis of the paper is synthetic data from a self-designed generator. This generator works on the principle of a recursive filter, more precisely the autocorrelation filter. The filter has the order $N = 6$. In total four classes are generated. For each class there is a corresponding representative, after which all class-related data is generated. The representative is audio data with a sampling rate $f_a = 44.1\,kHz$. The number of patterns is chosen very unbalanced so that the following data set is created:

The data for the following investigation are subject to the distribution according to table 1. Most of the articles deal with two-class problems, whereas this one already uses a four-class problem as in the following use case of forage

Table 1: Classification of the synthetic data set

class	sample
1	350
2	2500
3	4000
4	9500

harvesters. The uneven distribution is similar to the first real measurement series, with the sharpest knife condition being strongly underrepresented with class 1 and the dullest knife condition being most with class 4.

2 Methods

2.1 Balancing Methods

The balancing procedures represent the methodological variation of the study. The conventional methods can be divided into five large groups.[3] The first group are the **Non-Heuristic Sampling Methods**. These work randomly and each delete or copy the data to arrive at a certain number of samples. These include Random Over-Sampling and Random Under-Sampling. Random Over-Sampling copies randomly selected data from the under-represented classes and adjusts them quantitatively to the most common class. Random Under-Sampling works the other way round and adjusts the data to the least occurring class. It randomly deletes data samples until the quantities are adjusted.[4]

The second group uses **Synthetic Data Generation**. These also approximate the data samples of all classes of a particular minority or majority. The Synthetic Minority Over-Sampling Technique (SMOTE) and the Adaptive Synthetic Sampling (ADASYN) are used for this. Both mentioned methods are over-sampling methods. The Synthetic Data Generation works in the feature space and less in the data space, like the Non-Heuristic Methods. The SMOTE algorithm over-samples the minority class by forming synthetic samples along

segments of the minority class of k nearest neighbors [5]. The nearest neighbors are determined randomly. Depending on the amount of over-sampling, a sample is generated in the direction of the determined nearest neighbor. Vectorially it is considered to form a difference vector. This is to be multiplied by a random number $\delta = [0,1]$ and appended to the original vector. In this way, a training record S can be considered with a minority class P and other classes N. For each sample $\mathbf{p_i} \in P$ there is a k nearest neighbor $\mathbf{x}$. A new vector can now be defined as described above:

$$\mathbf{x_{new}} = \mathbf{p_i} + (\mathbf{x} - \mathbf{p_i}) \times \delta \tag{1}$$

This causes a selection of a point on a segment between a sample and its nearest neighbor.[6]

ADASYN is motivated by SMOTE and aims to reduce bias including adaptive learning. Looking at the training data set S with m samples, $\{\mathbf{x_i}, y_i\}$, $i = 1, \ldots, m$, we propose x as sample and y as label. Now m_s is described as the number of minority class samples and m_l as the number of majority class samples. Now the degree of imbalance is calculated:

$$d = m_s / m_l \tag{2}$$

Considering that $d = (0,1]$ now determines how many samples must be generated:

$$G = (m_l - m_s) \times \beta \tag{3}$$

The parameter β indicates the degree of balancing and is set to 1 in our case. Now, similar to SMOTE, we have to find the k nearest neighbour. This is created using the euclidian distance in n dimension and the radius is calculated:

$$r_i = \triangle_i / K, i = 1, \ldots, m_s \tag{4}$$

Where $\triangle_i$ describes the number of k nearest neighbours. Furthermore the radius has to be normalized in the following way:

$$\hat{r_i} = r_i / \sum_{i=1}^{m_s} r_i \tag{5}$$

Now it is calculated how many synthetic data samples must be generated per minority sample:

$$g_i = \hat{r}_i \times G \tag{6}$$

Where G is the absolute number of synthetic data examples. Now the following rule can be used to generate the corresponding synthetic samples for each $\mathbf{x_i}$ sample:

$$\mathbf{s_i} = \mathbf{x_i} + (\mathbf{x_{zi}} - \mathbf{x_i}) \times \delta \tag{7}$$

The main difference to the SMOTE method is the use of a density distribution $\sum_i \hat{r}_i = 1$ as a criterion at which point synthetic data must be generated.[3]

The third group uses **Cost-Sensitive Learning**. Here a cost factor for false predictions is determined, which affects the gradient. The data is not changed in this procedure. In this paper, the Weighting Factor is calculated using the samples per class. We assume there are N classes and the ith class has m_i training samples. Then a Classweighting w_i, $i = 1,\ldots,N$ can be determined, which will affect each class according to the number of its samples:

$$w_i = \frac{\sum_i m_i}{m_i \cdot N} \tag{8}$$

The last group represents the **Active Learning**. Here, learning algorithms are applied in advance to balance the data. One application is the methodology of Cluster-Centroids. Depending on the minority class, clusters are formed in the point clouds of the other classes. That means, assuming a learning data set N with m samples, $k \geq 2$ subsets have to be found, which are represented by similar attributes. Thereby the single cluster focal points, but the centroids should differ from each other. With $N = \{x_1,\ldots,x_n\}$ a set of points is defined, which fulfill a similarity condition. It applies $x_i \in \mathbb{R}^d$ and $d \geq 1$. Now μ_j and $M = \{\mu_1,\ldots,\mu_k\}$ is defined as centroid of the cluster $G(j)$ and a cost quantity $P = \{G(1),\ldots,G(k)\}$ is introduced. This describes the cluster problem and is used for optimization:

$$\mathbf{P}: \text{ minimize } z(W,M) = \sum_{i=1}^{n}\sum_{j=1}^{k} w_{ij} d(x_i,\mu_j) \tag{9}$$

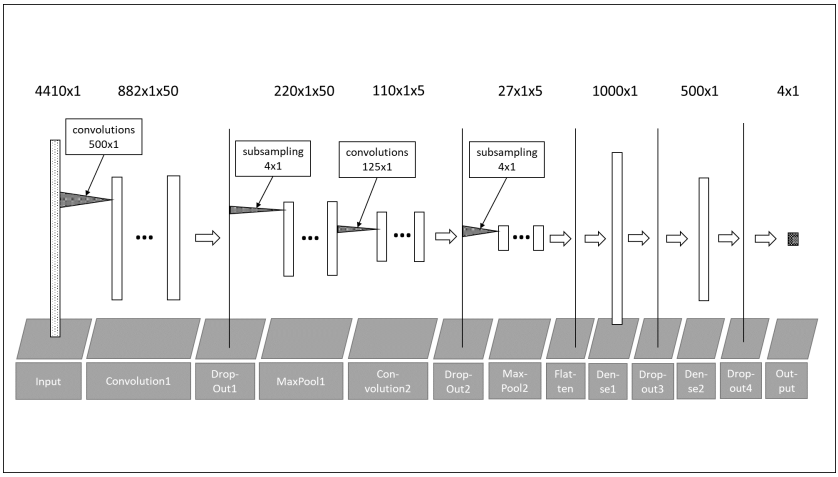


Figure 1: Netsummary of deep convolutional neural network

With $w_{ij} \in \{0;1\}$ it is defined if x_i counts to $G(j)$ and $d(x_i, \mu_j)$ defines the euclidean distance. There are now several methods to solve **P** which will not be discussed in detail. It should only be noted that $k = m_s$ defines itself as a sample set of the minority class. This leads to the consequence that the formation of cluster centroids counts as an under-sampling procedure.[7]

2.2 Deep Convolutional Neural Network

The structure of a deep convolutional neural network was used for the investigation. This structure, as shown in figure 1, has not changed during the analysis. It was proceeded in such a way that there were several initializations, more precisely 100, of the network, because the weights w of the DCNN are initialized stochastically. This set I, in the following called individuals, is generated per balancing procedure and then evaluated. Thus the purely stochastic influence of the weight initialization crystallizes in the results.

The first layer is the **Input layer** of the net. This layer represents the audio samples of length $n_s = 4410$, thus $t_s = 0.1\,s$. Furthermore, this is the first Convolution Layer. It is one-dimensional due to its application to audio vectors.

In the configuration, a filter depth of 50 windows has been set, with each filter window having a width of 500. The activation function is the rectified linear unit.

$$f(x) = max(0,x) \tag{10}$$

The following **Dropout layer** in conjunction with the **Max- Pooling Layer** creates a stochastic selection of the activated output neurons of the convolution layer. Thus, it represents a multinominal distribution during the training period, with the effect of avoiding overfitting in the learning behavior. The dropout rate is 0.4 and the pooling rate is 4.[8] Following this, a second **Convolution Layer** is added. This layer has the same filter width of 500, but only 5 filter windows. The sense of a second convolution layer is the abstract filter behaviour. The first layer should find features on a lower level of abstraction and the second layer on a higher level of abstraction. Dropout and pooling layer are added to this layer to achieve the equivalent effect as with the first convolution layer. The dropout rate and pooling rate are the same as the previous ones with 0.2 and 4. The subsequent **Flatten Layer** is used for dimension reduction and is merely a preparation of the structure for the subsequent **Dense Layer**. These are completely connected layers and are characterized by precise learning behavior. The attached dropout layer also serves to prevent overfitting. The first dense layer has a size of 1000 units with an activation function of the hyperbolic tangent.

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{11}$$

The following dropout layer has a rate of 0.5. The second dense layer is smaller and has only 500 units with an attached dropout layer and a rate of 0.5. The **Output layer** has the Softmax.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{12}$$

It is one of the most frequently used components and is also applied here.[9] The **cost function** is the categorial crossentropy with applied optimizer ADAM.

$$\mathscr{L}_{CCE}(f(\mathbf{x}),y) = -\mathbf{e}_y \, log \, f(\mathbf{x}) \tag{13}$$

2.3 Qualifying criteria

In the previous section the structure of the classifier was shown. The theoretical basics of the sampling methods used were also discussed in advance. The classification performance is of central importance for this paper, as it is the basis for the evaluation of the balancing methods. In most cases, scalar parameters are determined for this purpose, which have the static consideration of the correctly assigned classes. However, in most cases this does not yield meaningful evaluations. The problem may arise that in case of unevenly distributed classes, i.e. unbalanced data sets, the overrepresented classes are preferred. This way, less existing classes, which have the same valence, would hardly be recognized, but the classifier would still have a comparatively high classification quality. A further problem can arise if the relevance of the correct classification is very different. This is not the case with the data used, but can arise when applying the same methodology to real data. A third problem, which can occur with real data, is uncertainty or noise in the expert specifications. For all measurement data sets, which originate from real applications, the actual measurement signal is faulty. So an expert estimation of the target state of the data can also be faulty. Due to the synthetic data generation this will not be the case in this paper.

With the preceding illustrations of quality problems, the calculations of the used quality parameters shall now be shown. For this purpose, we first look at the feature data with N samples. The assignment of a class to a i data sample is described as CL_i by the classifier. The assignment by the expert is defined as CL_i^s. The element s_j of the vector $\mathbf{s} = [s_1, \ldots, s_N]^T$ is used to define an object of a test set of the class j. After calculation of the classification results an element $r_j \in [0; s_j]$ of the vector $\mathbf{r} = [r_1, \ldots, r_N]^T$, which represents the output of the classification. This vector contains the number of matching objects of a class j. This results in the simplest qualification criterion, which is called q_1:

$$q_1 = \frac{\sum_{j=1}^{N} r_j}{\sum_{j=1}^{N} s_j} \tag{14}$$

This includes the already mentioned problem of preferring the most common class. As a remedy, a criterion must be worked out, which also considers the

weakly occurring classes. For this purpose q_2 is introduced as follows:

$$q_2 = \frac{1}{N} \sum_{j=1}^{N} \frac{r_j}{s_j} \tag{15}$$

By normalizing to the number of samples, all parts with the same weight enter the quality criterion. One way to calculate a measure for the worst recognized class is the geometric mean. This remains independent of the number of objects in a class and is defined as follows:

$$q_3 = \sqrt[N]{\prod_{j=1}^{N} \frac{r_j}{s_j}} \tag{16}$$

With the evaluation of the quotient r_j/s_j the class j can be examined for correct recognition. If the attention is now turned to the generally worst class after

$$q_4 = \min_{i=1,2,\ldots,N} \left(\frac{r_j}{s_j} \right) \tag{17}$$

, the robustness of the classifier can be assessed. However, the parameter q_4 is to be supplemented by a more robust measure. The classification can be regarded as a nominal scale, thus the kappa coefficient is to be used for the judgement agreement. Its starting point is the permutation matrix of two judges, the expert and the classifier. The form presented in the following contains the expert specification as rows of the matrix and the judgements calculated by the classifier as columns. The elements of the swap matrix $\mathbf{K}$ are relative frequencies k_{ij}:

$$\mathbf{K} = \begin{pmatrix} k_{11} & \ldots & k_{1N} \\ \vdots & \ddots & \vdots \\ k_{N1} & \ldots & k_{NN} \end{pmatrix} \tag{18}$$

The frequencies of occurrence of target and actual classification are defined and counted as ka_{ij}. A normalization is done by the set of test data records T after ka_{ij}:

$$k_{ij} = \frac{ka_{ij}}{T} \tag{19}$$

The relative frequency of concordant judgments results along the main diagonal of the matrix K, as follows:

$$p_o = \sum_{i=1}^{N} k_{ii} \tag{20}$$

The proportion of expected concordant judgments is compared to this using the following relative frequency:

$$p_e = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} k_{ij} \right)^2 \tag{21}$$

The above calculation is different from the original definition and uses the squares of the row sums instead of the products of the row and column sums. This is done to achieve an increased weighting of the target classification. The reason for this is that the original kappa coefficient assumes two independent and equal viewers, which is not the case when classifying with a neural network. The kappa coefficient itself is defined as follows:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{22}$$

Finally, there is also the demand for performance in the learning behavior of neural networks. For this purpose a parameter t_i is introduced, where $i = 1, \ldots, I$. This parameter is measured during the calculation and specifies the calculation time for the learning process until a convergence criterion is reached. The background is that with over-sampling or under-sampling the amount of learning data is changed. This should be taken into account when considering the balancing methods. The quality, robustness and performance of the classifier is evaluated with the six criteria described above.[10]

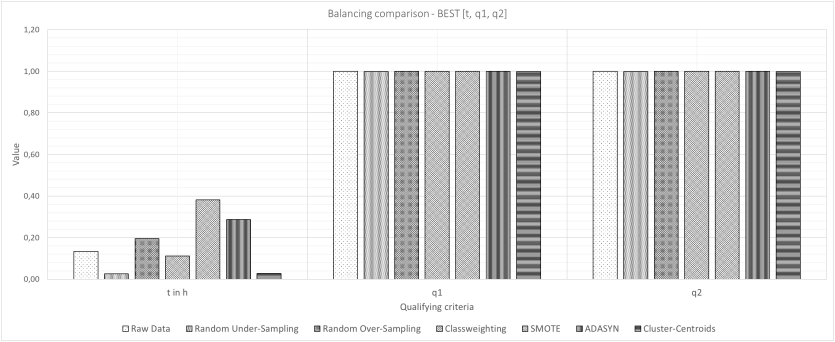


Figure 2: Balancing comparison of the best runs with [t, q1, q2]

3 Results

As already mentioned, training was done with a total of $I = 100$ runs of the same data and the same balancing procedure. The only difference between these individuals lies in their stochastic initialization of their weights. At first the best trainings resulst are to mention. For this purpose the parameters $t, q_1, q_2, q_3, q_4,$ and κ were used.

In figure 2 you can see that there are big differences in the area of training time. The SMOTE and ADASYN methods are the learning methods with the longest training times and the Random Under-Sampling with the Cluster-Centroids the shortest training times. This is due to the creation or removal of samples which actively influence the teaching time. The parameters q_1 and q_2 show that the maximum of 1 is reached as the best run and therefore there is no difference in the balancing procedures.

The analysis of the parameters q_3, q_4 and κ also do not show any differences in the methodologies, as figure 3 shows. Thus, there is a general possible convergence of all six listed sampling methods, including the raw data, towards an optimal classification result. A reason for this result can be the stochastic initialization with the quantity $I = 100$, by which a favourable starting position of the weights w for convergence is given.

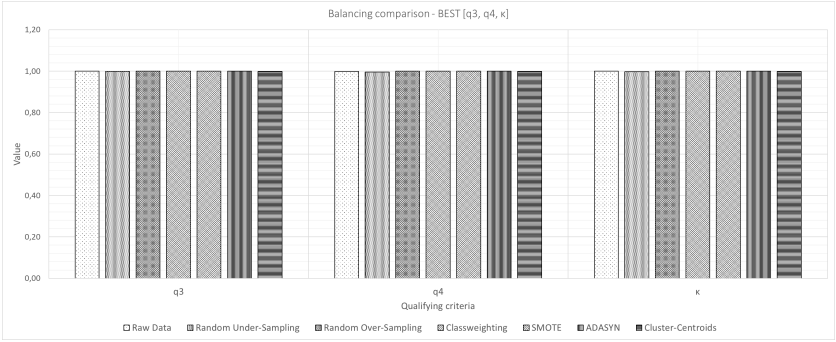


Figure 3: Balancing comparison of the best runs with [q3, q4, k]

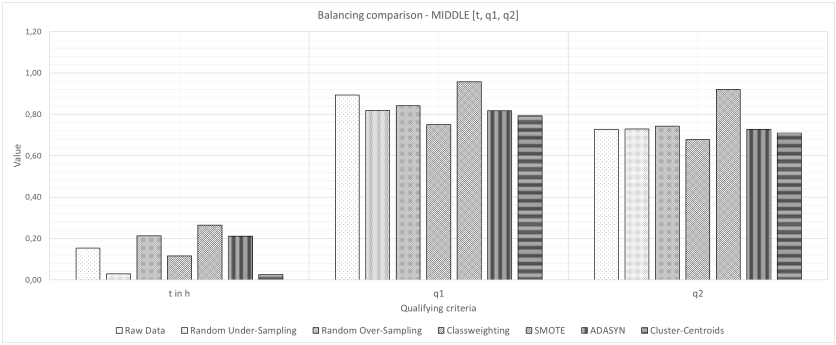


Figure 4: Balancing comparison of the middled runs with [t, q1, q2]

Representations of the best learning outcomes of a set of individuals can provide a static measure of convergence ability, but the statistical mean is a quantity-based measure of how often the set of individuals moves towards convergence. Therefore, the parameters t, q_1, q_2, q_3, q_4 and κ are following formally averaged:

$$\bar{x} = \frac{1}{I} \sum_{i=1}^{I} x_i \tag{23}$$

I represents the set of individuals and x is the formal representative of all used parameters.

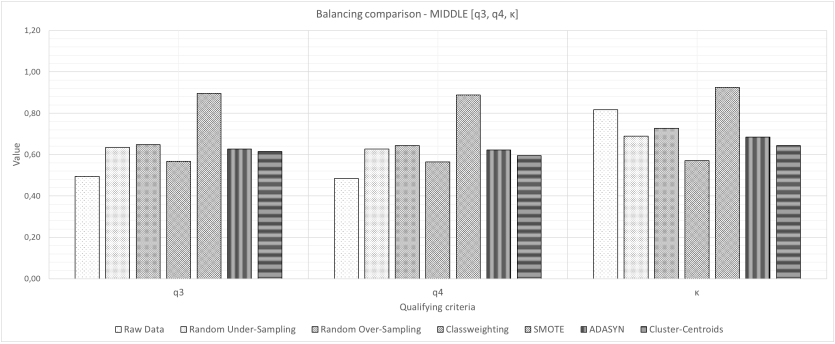


Figure 5: Balancing comparison of the middled runs with [q3, q4, k]

By looking at the parameter t in figure 4 you can see that the behavior of the best runs is reflected. However, the set of balancers which need the longer calculation time is additionally supplemented by Random Over-Sampling. Noticeable with the parameters q_1 and q_2 is that the Classweighting performs worst and SMOTE as synthetic over-sampling is best. Interesting is the difference in raw data between the parameters q_1 and q_2. Since q_1 prefers the most represented class and q_2 the less represented class, this means that the classifier can better learn the over-represented data with pure raw data.

The two parameters q_3 and q_4 are used to evaluate the worst recognized class and have a statement for the robustness of the classifier. Looking at these in figure 5 it becomes immediately clear that SMOTE produces very good results compared to the other methods and that the raw data provide a less robust classification. The results of the κ coefficient are similar to those of the parameter q_1. It shows that the class weights give a relatively poor result and again the SMOTE method generates the best results.

By considering the averaged coefficients, a quantity-related representation is achieved. The distributions of the parameter values are also to be analyzed. Thus, the help of the box plots is used. The boundaries of the box contained in the plots represent so-called quantiles, more precisely $x_{0.25}$, $x_{0.5}$ and $x_{0.75}$. Usually a so-called whisker is specified in connection with this, which is defined

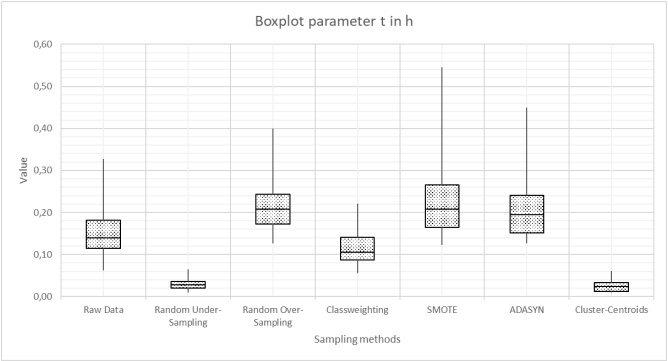


Figure 6: Boxplot of balancing methods regarding training time t

in the following and non-uniform way:

$$w = 1.5 \cdot (x_{0.75} - x_{0.25}) \tag{24}$$

This whisker limits the values up and down. However, if the whisker falls below or exceeds the minimum and maximum values, these are assumed. This procedure was used in the following illustrations.

The variance of the t parameter shows, in figure 6, that the over-sampling procedure has an increased median including larger minima and maxima. This can be attributed to the quantity-related increase in samples. The under-sampling methods, on the other hand, scatter less, as well as the median is lower. A well-balanced means for this is the Classweighting.

The box plots of the parameter q_1 in figure 7 show differences to the conventional appearance of these. First of all, it should be noted that this one, like the following box plots, is limited in $[0;1]$. Furthermore, it can be seen that the scattering measure of all balancing procedures is similar. Only the raw data differ. It is interesting to note that the median $x_{0.5}$ and the 3rd quantile $x_{0.75}$ are very close together. This indicates a high frequency at this point. In addition, it can be seen that the median for the raw data is lower than for the balancing procedures.

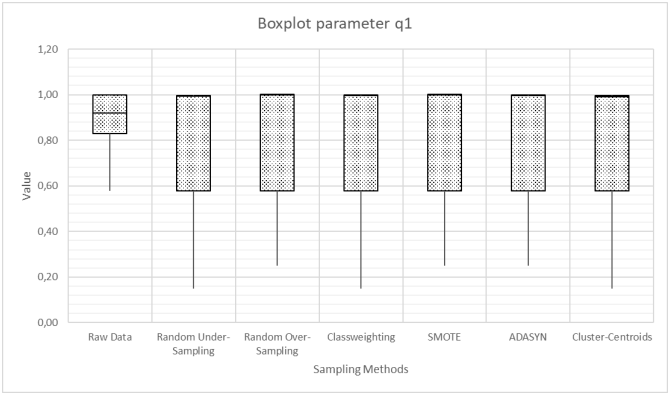


Figure 7: Boxplot of balancing methods regarding parameter q1

The parameter q_2 shows similar behavior in figure 8 as q_1. However, since it prefers the underrepresented classes, there is more variation. Also, the classification is worse when teaching with raw data, which shows its median.

With q_3 a robustness measure is represented in figure 9. The scatter is very high for all methods and covers the whole value range $[0;1]$. For all sampling methods, however, the median is close to 1, which has a high frequency at this point. Only the raw data performs worse. Its median is close to 0.5. This indicates a balanced distribution.

Since q_4 is also a measure for robustness, it can be assumed that a similar form of box plots as in figure 9 can be expected. This is confirmed with figure 10.

As a final parameter κ should be addressed. In previous plots this parameter showed equivalent behavior to q_1 with respect to the best run and the averaged runs. It can be seen in figure 11 that all sampling procedures have a high degree of dispersion. For all of them $x_{0.25}$ is close to 0.25. The minimum values also tend to be as low as 0. The median with the 3rd quantile is close to 1. This again indicates a higher frequency at this point. As an exception the individuals of the classifications with the raw data are shown. These scatter less, because their 1st quantile is around 0.7, but the median is also lower, 0.84 to be exact. This shows the clear influence of the balancing procedures on the stochastic initialization of the weights for the given dataset.

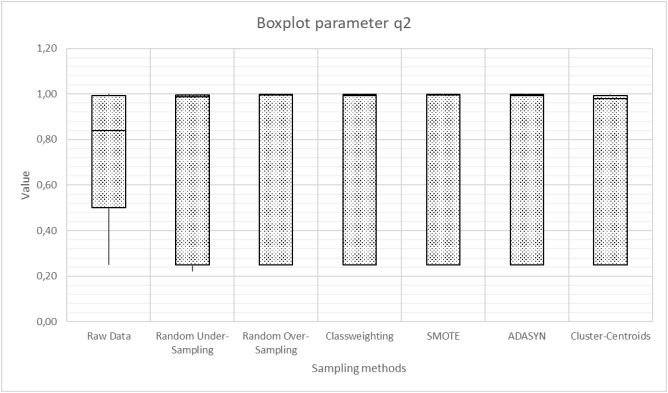


Figure 8: Boxplot of balancing methods regarding parameter q2

4 Conclusion

In this paper the aim was to investigate the influence of different balancing methods on audio data. In order to isolate the effect, it was decided to use synthetic data as a basis. With this data a predefined structure of a deep convolutional neural network was trained. The only variant part of the network was the random initialization of the weights.

With the representation in figure 2 and 3 statements concerning convergence can be made. All balancing methods show, from a static point of view, the convergence ability to solve the problem. The defined parameters all reach the value 1 in $[0;1]$, except the calculation time. This time is proportional to the number of samples used in the learning process. In figure 4 and 5 the averaged values of the parameters is displayed. This shows a dynamic measure, since with each run, i.e. an individual, a different convergence is achieved and the average value is used to consider these individuals. At first it can be seen, that the calculation time is proportional to the given number of samples in the balanced datasets. Furthermore, pure raw data without balancers in the learning behavior prefer the overrepresented classes. With inverse logic this behavior is shown for the underrepresented data. The best middle balancing method is the SMOTE algorithm. This represents itself as extremely convergent with regard to all quality parameters q_1, q_2, q_3, q_4 and κ, whereas the other procedures tend

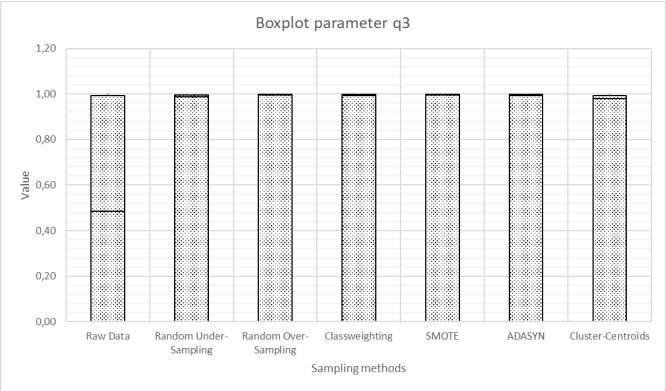


Figure 9: Boxplot of balancing methods regarding parameter q3

to be less optimal. With the analysis of the dispersion of the quality parameters, in figure 6 to 11, high variances are found in all balancing procedures. It shows that, in addition to the convergence, there is also a divergence for each algorithm due to pessimistic initialization of the weights. Likewise, the measure of dispersion itself is similar to most balancers per parameter and decides only on the raw data. A strong binarity between con- and divergence crystallizes.

From the results it can be concluded that the best balancing method is SMOTE. This of course only requires the application to the synthetic audio data. In the following work the effects on real measurement data have to be investigated. Since this contains noisy expert specifications in addition to the noise-superimposed signals, the results have to be awaited. Another effect could also occur when considering robustness as well as convergence.

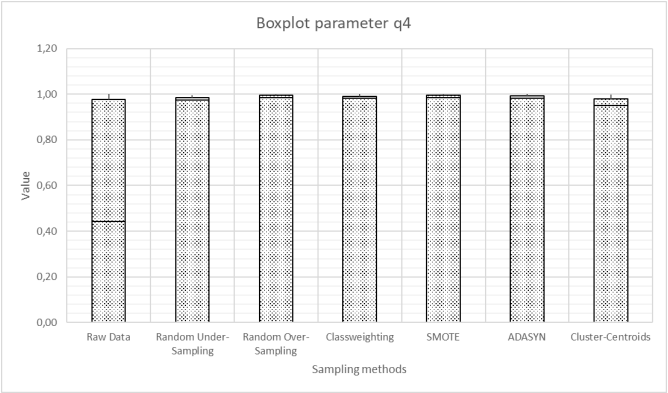


Figure 10: Boxplot of balancing methods regarding parameter q4

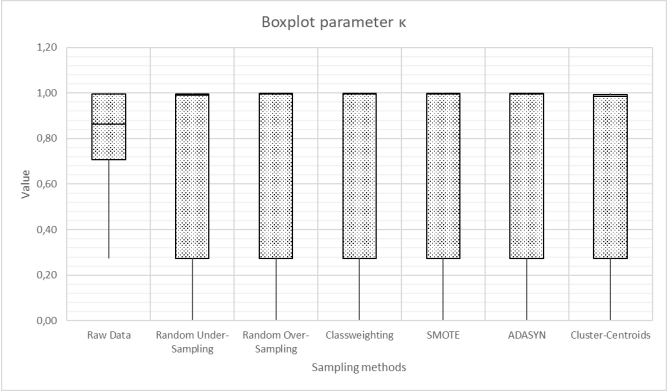


Figure 11: Boxplot of balancing methods regarding parameter k

References

[1] S. Hershey *et al.*: "CNN architectures for large-scale audio classification". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, LA* pp. 131-135 2017.

[2] M. Trommer, M. Schneider, C. Walther: "Auswirkungen von ungleichverteilten und ungenauen Belehrungsdaten auf die Klassifikation der Support Vektor Maschine". In: *Tag der Forschung 2014, Schmalkalden* pp. 35-52 2014.

[3] Haibo He and Yang Bai and E. A. Garcia and Shutao Li: "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)* pp. 1322-1328 2008.

[4] V. S. Spelmen and R. Porkodi: "A Review on Handling Imbalanced Data". In: *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)* pp. 1-11 2018.

[5] Maria Trommer: "Ein Beitrag zur Anwendung von Support-Vektor-Maschinen zur robusten nichtlinearen Klassifikation komplexer biologischer Daten". In: *Beitrag zur Anwendung von Support-Vektor-Maschinen zur robusten nichtlinearen Klassifikation komplexer biologischer Daten* 2017.

[6] T. Maciejewski and J. Stefanowski: "Local neighbourhood extension of SMOTE for mining imbalanced data". In: *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)* pp. 101-111 2011.

[7] Pérez-Ortega, Joaquín AND Almanza-Ortega, Nelva Nely AND Romero, David: "Balancing effort and benefit of K-means clustering algorithms in Big Data realms". In: *PLOS ONE* pp. 1-19 2018.

[8] Haibing Wu and Xiaodong Gu: "Towards dropout training for convolutional neural networks". In: *Neural Networks* pp. 1-10 2015.

[9] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li: “Soft-Margin Softmax for Deep Classification”. In: *Neural Information Processing* pp. 413-421 2017.

[10] Andreas Wenzel: “Robuste Klassifikation von EEG-Daten durch Neuronale Netze”. In: *Robuste Klassifikation von EEG-Daten durch Neuronale Netze* 2005.

Transforming LiDAR Point Cloud Characteristics between different Datasets using Image-to-Image Translation

Felix Berens[1,2], Yannick Knapp[1], Markus Reischl[2], Stefan Elser[1]

[1]Institute for Artificial Intelligence
Ravensburg-Weingarten University of Applied Sciences, Weingarten, Germany
E-Mail: Felix.Berens@rwu.de

[2]Institute for Automation and Applied Informatics
Karlsruhe Institute of Technology, Karlsruhe, Germany
E-Mail: Felix.Berens@kit.edu

Abstract

In recent years several new LiDAR datasets for object detection were published. All these datasets were recorded with different LiDAR setups and at different locations. KITTI, for example, has 64 channels and was recorded in Germany, whereas Lyft (Level 5) has only 40 channels and was recorded in the USA. This leads to different characteristics of the LiDAR point clouds. In this paper, we present and evaluate a way to transform KITTI BEV maps such that they look like Lyft BEV maps. For this transformation we use the state-of-the-art image-to-image translator CycleGAN. The transformation is evaluated by two strategies: Firstly we test if the translated KITTI BEV maps work better for an object detector, which is trained on Lyft. Secondly we test if the characteristic structure of the Lyft dataset (number of channels, location of points) is adopted from the translated point cloud. The conducted experiments showed that after the translation the KITTI BEV maps are more similar to Lyft BEV maps, but the detection got worse.

1 Introduction

Especially in autonomous driving, any method used to evaluate the 3D driving environment must be fail safe. LiDAR, RADAR and ultrasonic sensors are used to obtain 3D information about the surrounding area. While an ultrasonic sensor only works in short range and the 3D information of a RADAR is very sparse, the LiDAR sensor can produce accurate information of the immediate surroundings such as pedestrians or other vehicles.

With the KITTI dataset [1], a dataset for autonomous driving containing LiDAR point clouds is publicly available since 2012 and is still used as a standard dataset. In recent years, new datasets for autonomous driving containing LiDAR point clouds became publicly available e.g. 2018 nuScenes [2], 2019 Lyft Level 5 [3], 2019 Audi A2D2 [4] or 2020 Ford AV [5]. With Astyx HiRes [6] the first dataset for autonomous driving was published, which also contains point clouds from an high resolution RADAR in addition to LiDAR point clouds. As Table 1 shows, all datasets use different LiDAR setups. Significant differences between the datasets are the number of channels and the vertical resolution, which depends on the type of LiDAR that was used. While for most datasets one LiDAR is mounted on the top of the roof, for the Audi dataset multiple LiDARs were used: one at the front of the roof and 4 at the corners of the roof, with a slightly tilt. Due to the different locations of the LiDARs the point clouds of A2D2 look different than point clouds from a single LiDAR. The point clouds in A2D2 have a grid pattern, and not a circular pattern as in the other datasets with one top LiDAR. Also the different LiDAR setups for the datasets lead to a different appearance of the LiDAR point clouds (Fig. 1). Both KITTI and Lyft have a LiDAR mounted on top of the roof and generate a 360 degree vision of the surrounding area by multiple laser channels. Because in KITTI many laser channels scan the close area of the ego vehicle, a point cloud from the KITTI dataset contains much more points near the ego vehicle than a point cloud from Lyft.

[1] While working on this paper only the data of the top LiDAR were available.

Table 1: Comparison of the LiDAR setups of different datasets. Specs marked with * are taken from the datasheet of the LiDAR manufacturer and not from the source of the dataset. T = Top; B=Bumper; F=Front; C=Corner.

	KITTI	Lyft	Audi A2D2	Astyx
Number of LiDARs	1	1 + 2	5	1
Range [m]	120	-	100	100
Channels	64	40/64 [1]	16	16
Azimuthal FOV [°]	360	360	360	360
Vertical FOV [°]	26.8	-	30	30
Azimuth resol. [°]	0.08 - 0.35*	0.2	0.1-0.4 *	0.1-0.4 *
Vertical resol. [°]	0.4*	-	2	2*
Rate [Hz]	10	10	10	10
Position	T	T + 2 B [1]	F + 4 C	T
Intensity	✓	×	✓	✓
Type	HDL-64E	-	VLP-16	VLP-16

These LiDAR datasets are used to train and test methods e.g. for 2D and 3D object detection, segmentation, SLAM or optical flow. Wang et al. [7] showed that an object detector trained on one dataset (source) performs worse on another dataset (target). They concluded that a possible reason is the size of cars in different regions of the world. They applied different strategies that focus on the size of the cars: enlarge or shrink the bounding box and the corresponding point clouds in the training scenes (SN), continue training with some ground truth point clouds from the target dataset (FS) or enlarge or shrink the predicted boxes of the detector (OT). These three methods proved to be effective on the performance of the detector. All these methods base on changing the detection method and not changing the target dataset.

For multiple methods, or other tasks than object detection, it could be difficult and time consuming to modify all methods and retrain them. It would be faster if we could transform the point clouds of the target dataset so that they look like the source dataset on which we trained our methods. So in this paper we will focus on the different LiDAR setups used in the datasets. Such a type of problem is called domain adaption (Sec. 2.2). One part of domain adaptation is unsupervised image to image translation. Methods like CycleGAN [8] or UNIT [9] have shown promising results in the translation between images of

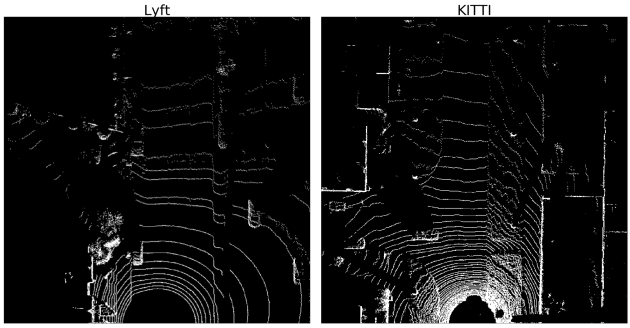


Figure 1: Sample of Bird's-Eye-View for Lyft and KITTI dataset. Both BEV maps are on information level **Position**.

different domains e.g. night to day, summer to winter or simulated to real data. While these methods work well for 2D image domains as seen in [8, 9], they are not designed for the application on 3D point clouds, like the LiDAR point cloud. Approaches for 3D object detection like Simon et al. [10] convert the 3D point clouds to Birds-Eye-View RGB-maps (BEV) and train a 2D object detector on this converted data. They encoded additional information, like height, density and intensity of the point clouds into the RGB color channels of the BEV map.

Sallab et al. [11] and Saleh et al. [12] proposed frameworks to translate synthetically BEV LiDAR maps to real BEV LiDAR maps. The simulated point clouds were generated with the CARLA simulator [13]. Without additional post-processing, the simulated point clouds are too smooth and miss artifacts that are common in real LiDAR point clouds. Sallabs and Salehs approaches used both KITTI as a real world dataset. Sallab did not state which LiDAR setup their simulated LiDAR used and encoded the height information of the 3D point clouds into the BEV maps. But Sallab did not use any additional information about intensity or density of the LiDAR points. The simulated BEV maps were translated with CycleGAN and got used together with real BEV maps from KITTI to train an object detector. Sallab improved the object detector from 65.3 % mAP, when using KITTI point clouds with 100.000 raw simulated point clouds to 71.5 %, when using KITTI point clouds and 100.000 translated simulated point clouds. Saleh used a simulated Velodyne HDL-64E

LiDAR for the simulated point clouds, so they have the same type of LiDAR as for the KITTI dataset (Tab. 1). Saleh mapped the 3D point clouds into BEV without any additional information and used only the location of the points. Saleh trained an object detector with several dataset combinations and tested it on the same split from the KITTI dataset. For KITTI alone they had an AP of 57.26 %, this was improved with the addition of 6.000 simulated point clouds to 59.16 % and with the addition of 6.000 simulated and translated point clouds to 64.29 %. Using only the 6.000 simulated point clouds for training resulted in an AP of 29.93 %, which could also be improved to 34.78 % by translating point clouds. However, this also shows, that the translation alone could not produce perfect real looking data.

We aim to make the BEV maps of two real world LiDAR datasets look more similar. For this we will:

- carry out a translation using CycleGAN between two real world LiDAR datasets for the first time
- use different information levels of BEV maps for the translation (**Position**, **Height**, **Height+Density**, **Height+Density+Intensity**)
- evaluate the quality of this translation with two different strategies. **Usability**: test if the translated target BEV maps work better for an object detector, which is trained on the source. **Structure**: test if the characteristic structure of the Lyft dataset (number of channels, location of points) is adopted from the translated BEV map.

2 Method

2.1 Datasets

We focus on the translation from KITTI to Lyft, because Lyft has a similar LiDAR setup as KITTI, but not totally similar. Both KITTI and Lyft have LiDAR mounted on the roof and generate a 360 degree vision of the surrounding area by multiple laser channels. KITTI has just one top LiDAR and we will also only use the LiDAR point clouds of Lyft's top LiDAR. The circular pattern of

both LiDAR point clouds can be seen in Figure 1. A point cloud from KITTI has much more points near the ego vehicle than a point cloud from Lyft. This can be seen in Figure 1, as more white pixels are located around the ego vehicle of KITTI in the bottom center than in the BEV map of Lyft. Furthermore, in Figure 1 the effect of the different number of LiDAR channels of KITTI and Lyft can be seen. It can be seen that the LiDAR of Lyft produces less white circles than that of KITTI in the BEV map. This is because the LiDAR in KITTI has more laser channels to scan the close area of the ego vehicle, while in Lyft the lasers are such that they focus the scan on the midrange. For the BEV maps the most significant differences between the two datasets can be seen on parts that belong to the street, while in both examples of Figure 1 the cars have a similar L-shape. The number of channels is 64 for KITTI's and 40 for Lyft's top LiDAR (Tab. 1). Hence a translation from KITTI to Lyft should delete some of these wave lines, without destroying the L-shape of a vehicle.

2.2 Domain Adaptation

The problem of training a method on one dataset (*source domain*) and applying it to another dataset (*target domain*), is called domain adaptation. Domain adaptation is a type of transfer learning and aims to bridge the gap between different domains of data [14]. A specific type of domain adaption is the image-to-image translation. In this type an input image from the source domain can be transformed to an image that is similar to the distribution of the target domain. The transformation is observable as the source image adapts the "style" of the target domain. Such a transformation can be studied in two settings: the supervised and the unsupervised setting. In the supervised translation one, every target image has a corresponding source image. In the unsupervised one, no target image has a corresponding source image. As Lyft and KITTI are recorded at different locations and different times, we do not have paired data.

Within the scope of this work, we will use the Cycle-Consistent Adversarial Network (CylceGAN) [8]. By taking pairs of images out of different domains CycleGAN learns how to apply the characteristics of one domain to the images of the other. CycleGAN consists of two generators and two discriminators:

one generator takes images of the first domain and output images of the second domain, the second generator vice versa. The demonstrators determine how plausible the output of the generators are for the domains. In addition, CycleGAN applies the cycle-consistency loss. So in CycleGAN adversarial losses are combined with cycle consistency loss, i.e. evaluating also a possible back-projection of generated data from the target domain into the source domain.

Because CycleGAN[2] is designed to translate 2D data, we will convert the 3D point clouds to 609×609 Birds-Eye-View RGB-maps. We can encode different levels of information into the pixel color values of the BEV. The simplest way of a BEV map (**Position**) uses only the location of the points and ignores the height, such that the pixels are 255 iff there is a LiDAR point at this position and 0 if not. See Figure 1 as an example. In the next level of information (**Height**), we use the height of the points and encode this information into the greyscale value of the pixel. If more than one point is at the same position, we use the maximum height. In the third level of information (**Height+Density**) we map the density of points that lie over each other in addition to the height in the first two color channels of the pixel. In the last level of information (**Height+Density+Intensity**), we also encode the intensity information, which describes how good the laser beam is reflected, into the third color channel of the pixel. If more than one point gets mapped to the same pixel, then the maximum intensity is used.

2.3 Evaluation

We choose two ways to evaluate how good the translation of the point clouds provided by KITTI into the structure provided by Lyft is:

- **Usability**: How good can be the transformed BEV maps be used for task of autonomous driving, when the method is trained on the source dataset.
- **Structure**: How well does the translated BEV maps represent the structure of the LiDAR setup from the source dataset.

[2] Our CycleGAN implementation based on the implementation of Ming-Yu Liu., which can be found on GitHub https://github.com/junyanz/CycleGAN, Jan. 2020.

2.3.1 Usability

To have a quantitative criterion how good the translation fulfills the criterion **Usability** we will use a similar strategy as [11, 12] and test the translation on the detector ComplexYOLO [10]. Instead of retraining with additional BEV maps, we train ComplexYOLO on Lyft and test how the detector performed on the translated KITTI BEV map compared to the original KITTI BEV map.

ComplexYOLO is a real-time 3D object detection network operating on LiDAR BEV maps. It is based on the YOLO framework with the addition of a specific Euler-Region-Proposal approach, that estimates the orientation of objects. The determination of the orientation of an object is necessary in BEV, since an object gets detected from above instead of the common front view and therefore can be rotated.[3]

To improve the detection of objects important structures as the shape of a car should not be destroyed after the translation, while the characteristics of the Lyft BEV maps should be adopted from the translated KITTI BEV maps.

As metrics we will use the well known intersection over union (IoU) with a threshold of 0.5 and the average precision (AP).

2.3.2 Structure

A criterion, which does not mostly depend on geometric properties of a scene, and distinguishes between different LiDAR setups, is required to measure how well the LiDAR BEV map of KITTI is translated to the characteristics of Lyft BEV maps. For this we map the BEV map back to 3D coordinates. In the case **Position** we do not have any height information. So all the height values are set to 0 for all points. These 3D coordinates are transformed to a spherical coordinate system. In a spherical coordinate system every point consists of radial distance, polar angle, and azimuthal angle (r, θ, ϕ). In the case that

[3] Our ComplexYOLO implementation based on the unofficial implementation from Deepak Ghimire, PhD, which can be found on GitHub https://github.com/ghimiredhikura/Complex-YOLOv3, Jan. 2020.

we only have one LiDAR without any tilt and the position of the LiDAR is the origin of the spherical coordinate system the angles can be interpreted as the angles of the laser beam during the scanning of the scene and r as the distance between the object and the LiDAR. So the distribution of ϕ and θ depends on the LiDAR setup. Because the azimuthal LiDAR setup in KITTI and Lyft is similar, the distribution should be similar and only differ mostly due to geometric properties of the scene. Because θ describes the polar angle it depends on the vertical resolution and vertical FOV, besides the scene, the distribution of θ can be used to distinguish between the LiDAR setups. The distribution of r depends both on the scene and also the LiDAR setup. Since in KITTI there are more points around the ego vehicle than in Lyft (see Fig. 1), the distribution has a higher peak by lower r.

After the translation of KITTI to Lyft, the distribution of θ should be more similar to Lyft than to the original KITTI.

As metric to measure the similarity of two distributions we use the well known Wasserstein distance also known as earth mover's distance [15, 16]. The Wasserstein distance comes from the transportation theory and for histograms it can be intuitively interpreted as cost to transform on histogram into the other.

3 Results

We train CycleGAN on a split of 7000 BEV maps for KITTI and 7000 BEV maps for Lyft. The BEV maps are created for the four level of information **Position**, **Height**, **Height+Density** and **Height+Density+Intensity**.

Figure 2 shows qualitative results of the translation for **Position**. It can be seen that in the translated KITTI BEV maps the number of white pixels close to the ego vehicle is decreased, like in Lyft BEV maps. Also some circle lines in the translated BEV maps are deleted and the number of circle lines match the number from Lyft. Further, most of the shapes are preserved for most cars, but for some cars the number of white pixels decrease. So optically these examples show promising results in the translation of the characteristics of the LiDAR BEV maps.

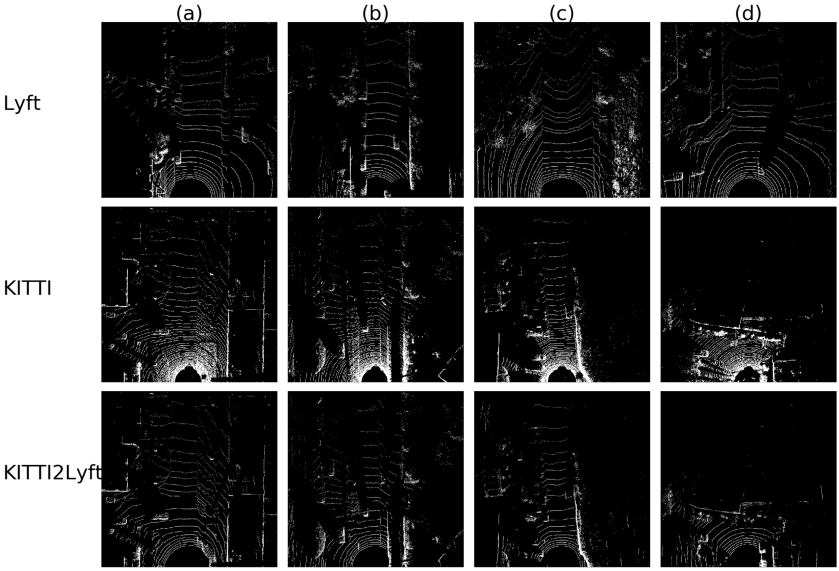


Figure 2: Qualitative results for the translation from KITTI to Lyft, for the information level **Position**.

For quantitative results we use the described evaluation strategies **Usability** and **Structure** (Sec. 2.3).

For **Usability** ComplexYOLO was trained on a Lyft split of 7000 BEV maps for the four different BEVs information levels to detect cars. Table 2 shows the average precision of the detector on test splits from Lyft, KITTI and the same KITTI scenes translated. For real Lyft BEV maps the detector has a good average precision over 0.95. For KITTI BEV maps it drops to an average precision between 0.53 - 0.6. After translation the average precision of the object detector gets a bit worse, probably because some shapes of cars were destroyed. This happened, for example, when translating the BEV map (c) in Figure 2. In the original KITTI BEV map the two cars in Figure 2 (c) are clearly visible, in the corresponding translated map most of the car is missing and could not be detected. Another potential problem is that after the translation pixel with small color values occur around the LiDAR points. These low color artifacts are barely visible to a human. In the first translated

Table 2: Results of the evaluation **Usability**. Average precision (IoU=0.5) for the class car of an object detector that was trained on Lyft Level5 dataset and tested on Lyft Level5, KITTI and KITTI2Lyft translated with CycleGAN.
P = **Position**; H = **Height**; D = **Density**; I=**Intensity**.

	P	H	H+D	H+D+I
Lyft	0.95	0.97	0.97	0.97
KITTI	0.53	0.6	0.54	0.53
KITTI2Lyft	0.52	0.54	0.51	0.47

Table 3: Results of the evaluation **Structure**. Mean Wasserstein distance between different distribution of the spherical coordinates from the datasets.
P = **Position**; H = **Height**; D = **Density**; I=**Intensity**.

	P	H	HD	HDI
$\theta_K \leftrightarrow \theta_L$	0.082	0.12	0.12	0.12
$\theta_{K2L} \leftrightarrow \theta_K$	0.12	0.17	0.17	0.17
$\theta_{K2L} \leftrightarrow \theta_L$	0.06	0.096	0.096	0.093
$\phi_K \leftrightarrow \phi_L$	0.18	0.18	0.18	0.18
$\phi_{K2L} \leftrightarrow \phi_K$	0.20	0.21	0.21	0.20
$\phi_{K2L} \leftrightarrow \phi_L$	0.20	0.21	0.21	0.20
$r_K \leftrightarrow r_L$	4.44	5.35	5.35	5.35
$r_{K2L} \leftrightarrow r_K$	7.18	5.60	6.41	5.25
$r_{K2L} \leftrightarrow r_L$	4.07	3.60	3.88	3.42

map of Figure 2 25454 pixel have a color value between 1 and 10 and 25415 pixel a color value greater than 10. This number of pixels containing small values decreases if we encode more information in the color channels, like in **Height**, **Height+Density** or **Height+Density+Intensity**, but the performance of the detector does not increase as Tab. 2 shows. Also if we set all pixels with a color value smaller than 50 to 0, the AP only growth only a little to 0.53 in the case **Position**. So the influence of these small values can be neglected.

For the criterion **Structure** we translate the coordinate system such that the LiDAR is nearly the origin of the coordinate system and transform the coordinate system to spherical coordinates. We compare the distribution of 100 KITTI BEV maps, the corresponding translated BEV maps and Lyft BEV maps. The

BEV maps are reprojected and transformed to a spherical coordinate system and calculate the mean Wasserstein distance between the distribution of the spherical coordinate values (Tab. 3). According to the results of Table 3 the distribution of r for the translated KITTI BEV maps is always more similar to Lyft than to KITTI and also closer to Lyft, than Lyft to KITTI. The distribution of ϕ does not significantly change after the translation. Interestingly the addition of the information density and intensity has only little influence of the performance.

4 Conclusions and Outlook

In this paper we analyzed the possibility of unsupervised domain adaptation between two LiDAR datasets with different LiDAR setups. The demonstrated method of translating LiDAR BEV maps from KITTI to the structure of the Lyft LiDAR setup using CycleGAN showed optical promising results (Fig. 2). With our evaluation strategy **Structure** we could show that the KITTI point clouds really adopts the LiDAR setup characteristics of Lyft. Nevertheless **Usability** showed that the performance of our tested car detector could not be improved, even got a little bit worse, because some shapes of cars were damaged during the translation process. This confirms the assumption of Wang et al. [7] that the size of cars plays a more important role for the performance of object detectors, than the setup of the LiDAR. To improve the performance one has to improve the preserving of shapes that are interesting for the task, e.g. cars L-shape. But without the additional information to enlarge or shrink the size of cars, the performance of an object detector will likely not get much better. To analyze the influence of different LiDAR setups on object detection further study is needed. The CARLA [13] simulation environment can be used for the study of the setup influence, as we can choose in CARLA the number of channels and thus can record the same scene, with different LiDAR setups. So the object detector can be tested on the same scenes and only the LiDAR setup will change, hence if the setup has no influence the object detector should perform roughly as well, whether the same LiDAR setup is selected as in the training or a different LiDAR setup.

Next to the application of the translation between two datasets with different LiDAR setups, the method could also be used to transfer between different types of sensors. With Astyx HiRes a dataset with high resolution RADAR data is published. But since this is the only existent high resolution RADAR dataset at the moment and contains only a few scenes, training methods like object detection is challenging. With the demonstrated framework of translating the BEV maps, one could try to translate between the RADAR and LiDAR sensors. Because Astyx also provides LiDAR point clouds in addition to RADAR point clouds, one could also use supervised domain adaption methods.

References

[1] A. Geiger, P. Lenz, C. Stiller et al. “Vision meets robotics: The kitti dataset”. In: *The International Journal of Robotics Research* 32.11. S. 1231-1237. 2013.

[2] H. Caesar, V. Bankiti, A. H. Lang et al. “nuscenes: A multimodal dataset for autonomous driving”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* S. 11618-11628. 2020.

[3] R. Kesten, M. Usman, J. Houston et al. “Lyft Level 5 Perception Dataset 2020”. URL: https://level5.lyft.com/dataset/ 2020.

[4] J. Geyer, Y. Kassahun, M. Mahmudi et al. “A2D2: Audi Autonomous Driving Dataset”. In: *arXiv preprint arXiv:2004.06320*. 2020.

[5] S. Agarwal, A. Vora, G. Pandey et al. “Ford Multi-AV Seasonal Dataset”. In: *arXiv preprint arXiv:2003.07969*. 2020

[6] M. Meyer and G. Kuschk. “Automotive radar dataset for deep learning based 3d object detection”. In: *2019 16th European Radar Conference (EuRAD)* S. 129-132. 2019

[7] Y. Wang, X. Chen, Y. You et al. “Train in Germany, Test in the USA: Making 3D Object Detectors Generalize”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* S. 11710-11720. 2020.

[8] J. Zhu, T. Park, P. Isola et al. “Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* S. 2242-2251. 2017.

[9] M. Y. Liu, T. Breuel and J. Kautz. “Unsupervised image-to-image translation networks”. In: *Advances in neural information processing systems*. S. 700-708. 2017.

[10] M. Simon, S. Milz, K. Amende et al. “Complex-YOLO: An Euler-Region-Proposal for Real-Time 3D Object Detection on Point Clouds”. In: *Computer Vision – ECCV 2018 Workshops* S. 197-209. 2019.

[11] A. E. Sallab, I. Sobh, M. Zahran et al. “Unsupervised Neural Sensor Models for Synthetic LiDAR Data Augmentation”. In: *arXiv preprint arXiv:1911.10575* 2019.

[12] K. Saleh, A. Abobakr, M. Attia et al. “Domain Adaptation for Vehicle Detection from Bird’s Eye View LiDAR Point Cloud Data”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* S. 3235-3242. 2019.

[13] A. Dosovitskiy, G. Ros, F. Codevilla et al. “CARLA: An Open Urban Driving Simulator”. In: *Conference on Robot Learning* S. 1-16. 2017.

[14] Y. Ganin, E. Ustinova, H. Ajakan et al. “Domain-adversarial training of neural networks”. In: *The Journal of Machine Learning Research* 17.1 S. 2096-2030. 2016.

[15] E. Levina and P. Bickel “The Earth Mover’s distance is the Mallows distance: some insights from statistics”. In: *Proceedings Eighth IEEE International Conference on Computer Vision (ICCV)* 2 S. 251-256. 2001.

[16] Y. Rubner, C. Tomasi and L. J. Guibas “A metric for distributions with applications to image databases”. In: *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)* S. 59-66. 1998.

Potential of Ensemble Copula Coupling for Wind Power Forecasting

Kaleb Phipps, Nicole Ludwig, Veit Hagenmeyer, Ralf Mikut

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

Abstract

With the share of renewable energy sources in the energy system increasing, accurate wind power forecasts are required to ensure a balanced supply and demand. Wind power is, however, highly dependent on the chaotic weather system and other stochastic features. Therefore, probabilistic wind power forecasts are essential to capture uncertainty in the model parameters and input features. The weather and wind power forecasts are generally post-processed to eliminate some of the systematic biases in the model and calibrate it to past observations. While this is successfully done for wind power forecasts, the approaches used often ignore the inherent correlations among the weather variables. The present paper, therefore, extends the previous post-processing strategies by including Ensemble Copula Coupling (ECC) to restore the dependency structures between variables and investigates, whether including the dependency structures changes the optimal post-processing strategy. We find that the optimal post-processing strategy does not change when including ECC and ECC does not improve the forecast accuracy when the dependency structures are weak. We, therefore, suggest investigating the dependency structures before choosing a post-processing strategy.

1 Introduction

As the share of renewable energy sources in the energy system increases, wind power forecasts become essential to guarantee balanced supply and demand. However, wind power highly depends on the chaotic weather system as well as other stochastic features and thus modelling uncertainty in these forecasts is important [1]. Probabilistic wind power forecasts aim to capture the uncertainty inherent in the model parameters and input features. Capturing this uncertainty is not easy and weather predictions, for example, are known to be biased and underdispersed. Meteorologists have therefore been post-processing ensemble weather predictions to describe the uncertainty more accurately [2, 3, 4, 5, 6, 7, 8, 9, 10]. In this post-processing the ensemble weather predictions are calibrated to the actual historical weather observations, eliminating some of the model biases. Transferring this approach to wind power forecasts yields promising results for handling the uncertainty in wind power forecasting models with uncertain weather features and forecast horizons of 3h-24h [11].

Phipps et al. [11] show that post-processing only the resulting wind power forecast is the best strategy with respect to forecast accuracy. However, their approach ignores dependencies between the weather variables as the post-processing is done using Ensemble Model Output Statistics (EMOS). This method involves fitting parametric distributions to the ensemble forecasts and sampling from them to generate post-processed forecasts. Random sampling causes inherent correlations, so-called dependency structures, between variables to be lost, which could affect the forecast performance. The present paper, therefore, extends the previous post-processing strategies by including Ensemble Copula Coupling (ECC) to restore the dependency structures between the variables and investigates whether including the dependency structures changes the optimal post-processing strategy. In contrast to other approaches, we do not investigate the temporal [12] or spatiotemporal [13] dependencies. We also use a parametric approach unlike the non-parametric methods used in [14].

The extended post-processing strategy is evaluated on two data sets with both a linear regression and an artificial neural network used as forecasting models. Both the ability of ECC to restore the dependency structures between the vari-

ables, as well as the resulting forecast accuracy are considered as performance measures.

The remainder of the present paper is structured as follows. Firstly we introduce theoretical concepts including ECC in Section 2. We then discuss the altered post-processing strategies (Section 3) before evaluating the effect of ECC in Section 4. We discuss the results in Section 5 before Section 6 concludes.

2 Background

Before evaluating the effect of ECC on wind power ensemble post-processing, we introduce EMOS, ECC, and also discuss the forecasting models we use.

2.1 Ensemble Post-Processing

The weather ensembles from the Ensemble Prediction System (EPS) are known to be biased and underdispersed, and thus need to be calibrated [2]. Not accounting for this bias or under estimated variance could lead to false wind power forecasts which affect the stability of the energy system. The present paper applies EMOS developed by Gneiting et al. [6] to perform this calibration. EMOS is based on non-homogenous regressions and performed for each weather variable individually given a single origin and set forecast horizon.

The standard EMOS approach is designed for ensemble members that are individually distinguishable. Ensemble members from the European Centre for Medium-Range Weather Forecasts (ECMWF) are, however, classified as exchangeable, thus representing equally likely future scenarios without distinguishing features [15, 16]. Given exchangeable ensembles $x_1, \ldots, x_M$, we apply EMOS from Gneiting and Katzfuss [5], where the weather variable y with an assumed normal distribution is modelled as

$$y|x_1, \ldots x_M \sim \mathcal{N}\left(a + b\sum_{m=1}^{M} x_m, c + dS^2\right), \tag{1}$$

with a, b, c and d being regression coefficients, and the variance S^2 being a linear function of the ensemble spread with

$$S^2 = \frac{1}{M} \sum_{m=1}^{M} \left(x_m - \frac{1}{M} \sum_{m=1}^{M} x_m \right)^2 . \tag{2}$$

We also use the same approach with a truncated normal distribution. Since we post-process each weather variable for a single origin and set forecast horizon, the EMOS coefficients change when any one of these three parameters are altered. We apply EMOS using a rolling calibration window of 40 days with the help of the *scoringRules*[1] package. For more information on EMOS and its application in meteorology see Gneiting et al. [6] or Gneiting [17].

2.2 Ensemble Copula Coupling

With the EMOS method introduced above, we now have ensembles that are calibrated to the past data. All weather variables are modelled with a univariate distribution. We could sample from each of these distributions independently and input the information into the wind power forecasting scheme. However, the original Numerical Weather Prediction (NWP) includes information on the weather variables dependencies among each other as well as in space and time. This information gets lost with the univariate EMOS approach. In order to retain these dependencies, several empirical copula-based approaches have been developed, which we want to introduce in more detail in this section.

A d-dimensional Copula is a multivariate cumulative distribution on the unit cube $[0,1]^d$ with uniform margins [18, 19]. The importance of copulas in restoring dependency structures is based on the theorem of Sklar [20]. He states that for any multivariate cumulative distribution function F with margins $F_1, \ldots, F_M$ there exists a copula C, that is unique on the range of the margins and has the form

$$F(x_1 \ldots, x_M) = C(F_1(x_1), \ldots F_M(x_M)), \tag{3}$$

[1] https://cran.r-project.org/web/packages/scoringRules/scoringRules.pdf

for $x_1, \ldots, x_m \in \mathbb{R}$. With regards to ensemble calibration, we already have the uniform margins $F_1, \ldots, F_M$ in the form of the EMOS univariate distribution. Therefore, Sklar's theorem states that as long as an appropriate copula is defined, univariate ensemble post-processing techniques can be used to accommodate any dependency structure.

The ECC approach, is based on the mathematical concepts defined above with the appropriate copula in the form of a *reordering* process. The idea is that given a dependence structure "template" [21], the samples drawn from the univariate EMOS distributions can be reordered in such a way that they resemble the initial correlation structures that were given in the NWP. The templates are based on the raw ensembles, where we assume that the ensembles capture the correlations. While several variants exist, we take a closer look at random ECC (ECC-R), quantile ECC (ECC-Q), and transformation-based ECC (ECC-T). These methods differ from each other in the way they sample from the distributions and whether they include a reordering step. Firstly, we take a look at an ECC approach based on random draws. Here, we sample with the independent standard uniform random variates $u_1, \ldots, u_M$, such that

$$\tilde{x}_1 = F^{-1}(u_1), \ldots, \tilde{x}_M = F^{-1}(u_M). \tag{4}$$

In this ECC-R approach, the samples have to be reordered according to the template depicted by the raw ensembles. This template is based on the rank structure. Given each time horizon, location and variable and following the notation by Schefzik et al. [21], the raw ensembles $x_1, \ldots x_M$ and their order statistics $x_{(1)} \leq \ldots \leq x_{(M)}$ construct a ranking permutation π, with $\pi(m) := \text{rank}(x_m)$ for $m \in \{1, \ldots, M\}$. After drawing the samples from the distribution, they are reordered according to π. The ECC approach based on quantiles involves the same reordering step as in the ECC-R approach, the sampling method is however different. In the case of ECC-Q, the samples are drawn from equally spaced quantiles, such that

$$\tilde{x}_1 = F^{-1}\left(\frac{1}{M+1}\right), \ldots, \tilde{x}_M = F^{-1}\left(\frac{M}{M+1}\right) \tag{5}$$

and then reordered as before. In contrast to the ECC-R and ECC-Q approach, ECC-T relies on a transformation and does not require an additional reordering step. The samples are drawn such that

$$\tilde{x}_1 = F^{-1}(S(x_1)), \ldots, \tilde{x}_M = F^{-1}(S(x_M)), \tag{6}$$

where S is the fit of a cumulative distribution function to the raw ensembles [21]. The choice of S depends on the variables in question and in the case of temperature, pressure and wind component vectors, S can be assumed as normal with mean equal to the ensemble mean and variance equal to the ensemble variance [3].

2.3 Forecasting Models

The present paper focuses on the effect of ECC on optimal post-processing strategies and not developing state-of-the-art wind power forecasts. Therefore, we use the same forecasting models (a linear regression model and a neural network), and the same forecasting strategy as in [11]. Both are described in the following together with how we measure the forecasting accuracy.

2.3.1 Linear Regression

The simplest models, which we use to forecast wind power, are linear regression models. These linear regression models can be described with

$$y_{t+h} = \beta_0 + \alpha y_{t+h-24} + \sum_{k=1}^{K} \beta_k W_{t+h}^k + \sum_{j=1}^{J} \gamma_j D_{t+h}^j + \varepsilon_{t+h}, \tag{7}$$

where y_t is the dependent variable, which in this case is the wind power, y_{t-24} is the actual wind power a day before, W^k are weather time series, such as wind speed and temperature, and D^j are dummy variables, such as the season, the month and the year. The models are fitted for each forecast horizon $h = h_1 \ldots h_H$ with $h \leq 24$ using actual historical weather data in order to describe the real relationship among the variables and remove any bias fitting

on historical weather forecasts or ensembles could introduce. Each ensemble $x_1 \dots x_M$ from the EPS is then used in a separate prediction run for each forecast horizon to generate an ensemble of wind power predictions with the previously fitted regression coefficients

$$\hat{y}_{t+h}(x_1, \dots x_M) = \hat{\beta}_0 + \hat{\alpha} y_{t+h-24} + \sum_{k=1}^{K} \hat{\beta}_k \widehat{W}^k_{t+h}(x_1, \dots x_M) + \sum_{j=1}^{J} \hat{\gamma}_j D^j_{t+h} + \hat{\varepsilon}_t, \tag{8}$$

where $\widehat{W}^k_{t+h}$ is the weather forecast made at time t for the forecast horizon h.

2.3.2 Neural Network

In order to better forecast non-linear dependencies, we implement a neural network. We tested multiple neural network configurations before selecting a configuration with two hidden layers of 10 and 7 neurons respectively and trained it with the resilient backpropagation algorithm. This network architecture was selected because it is the simplest we found, that still returns accurate forecasts. The chosen activation function is a hyperbolic tangent given by

$$\sigma(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \in [-1, 1]. \tag{9}$$

The input features remain the same as for the linear regression model explained above. Again, the parameters (i. e.weights) are fitted using the actual historical weather data and each ensemble member is passed through the network to get an ensemble wind power prediction. The neural networks are implemented in R with the *neuralnet* package[2].

[2] https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

2.3.3 Forecasting Accuracy

To evaluate the forecasting approaches, we use the Continuous Ranked Probability Score (CRPS). This error measure is used to assess the calibration and sharpness of the probabilistic forecast and can be described as follows

$$\mathrm{CRPS}(F,y) = \int_{\mathbb{R}} \left(F(z) - \mathbb{1}\{y \leq z\}\right)^2 dz, \tag{10}$$

with F being the wind power generations predictive cumulative distribution function, y the verifying observation and $\mathbb{1}$ denoting an indicator function. We report the score over all time steps $t = 1, \ldots N$ in the test set

$$\mathrm{CRPS} = \frac{1}{N} \sum_{t=1}^{N} \mathrm{CRPS}(F_t, y_t). \tag{11}$$

3 Post-Processing Strategies

The present paper focuses on determining whether including ECC into the post-processing of wind power forecasts affects the performance of these strategies and changes the optimal post-processing strategy. Four post-processing strategies were identified by Phipps et al. [11] and these are shown with the addition of ECC in Figure 1. We identify two strategies that can be extended to include ECC, whilst two remain the same:

Raw. Using the raw weather ensembles directly in the forecast is not affected by ECC. Here we take all available M ensemble members from the EPS for multiple weather variables to generate the wind power forecast. Thus, the resulting wind power forecast consists of M members.

One-Step-P. The second strategy identified by Phipps et al. [11] is also unchanged by ECC. The output of the wind power forecasting model is post-processed, without any previous calibration of the input variables. This assumes that post-processing the wind power ensembles also accounts for the biases in the weather ensembles. For a detailed description of the One-Step-P approach see Phipps et al. [11].

One-Step-W. This strategy involves calibrating the raw weather ensembles before they are used as inputs in the wind power forecast model. We initially post-process the weather ensembles analogue to Phipps et al. [11]: Each weather variable (temperature, wind speed, wind component vectors etc.) is considered separately and post-processed using EMOS with a rolling calibration window. This results in probability distributions for each weather variable and we form new post-processed weather ensembles by sampling from these distributions. It is in this resampling stage that we apply ECC. Depending on the method (see Section 2.2) we either reorder the EMOS samples or also consider a different sampling strategy or transformation. As a result the strategy now has multiple variants. We use the resulting post-processed ensembles with restored dependency structures as inputs into the wind power forecast model. The resulting ensemble of wind power forecasts is not processed further.

Two-Step-WP. The final strategy is also altered when we include ECC. In this strategy both one-step post-processing approaches are coupled together. We first post-process the weather ensembles using the altered One-Step-W strategy including ECC. These ensembles are used to generate a wind power ensemble forecast and then we again post-process the result as in the One-Step-P method. Since there is only one set of wind power ensembles, therefore not dependencies present, we are unable to apply ECC after the first EMOS application step.

In the present paper we compare these approaches, focusing on discovering if the inclusion of ECC in One-Step-W and Two-Step-WP improves the forecast performance. Specifically through an evaluation on two data sets, we investigate if ECC leads to one of these two strategies outperforming One-Step-P, which was previously found to perform best [11].

4 Evaluation

We use two different data sets to evaluate the effect of ECC on the performance of the post-processing strategies described above. In this section we briefly introduce those data sets and present the results of our evaluation.

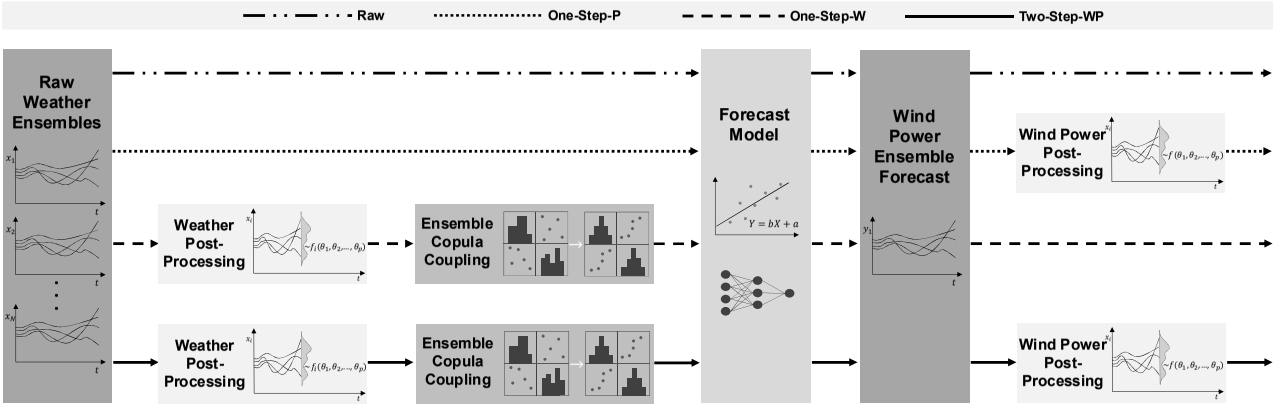


Figure 1: An overview of the post-processing strategies compared. Whilst no post-processing (Raw) and post-processing only the power ensembles (One-Step-P) are unaffected by adding ECC, we extend the other two strategies. Therefore we add ECC after the EMOS stage, when we post-process only the weather ensembles (One-Step-W). We also include ECC after the EMOS stage for the weather variables but before the post-processing of the power ensembles in the Two-Step-WP strategy.

4.1 Data

We evaluate the post-processing strategies on two data sets: A benchmark data set including both an onshore and offshore wind park, and real data from bidding zones 3 and 4 in Sweden. This section briefly introduces the data used.

4.1.1 Benchmark Data

The benchmark data set is based on simulated wind power data based on real wind parks located in Germany. This data was simulating using the *renewable ninjas*[3] API, with the input data being selected to mimic real onshore and offshore wind parks in Germany as closely as possible. Staffell and Pfenninger [22] verify that the simulation and bias-corrections implemented in the renewable ninjas API are capable of reproducing accurate wind power time series. A detailed description of the parameters used in the simulation is provided by Phipps et al. [11].

We access open source weather ensemble data through The International Grand Global Ensemble (TIGGE) archive[4]. TIGGE archive is a result of *The Observing System Research and Predictability Experiment* which aimed to combine ensemble forecasts from leading forecast centres to improve probabilistic forecasting capabilities [23]. Due to damaged tapes in TIGGE we only use data from February 2017 until August 2018, including the parameters two-meter temperature, surface pressure, 10m-U-Component of wind, 10m-V-Component of wind and wind speed. Weather data is downloaded for the same location as the synthetic wind park generated through renewable ninjas. We use the ERA5 reanalysis data for the ground truth historical weather data [24]. We download the identical weather parameters for the same timespan via the Copernicus Climate Data Store (CDS) API [5]. Data from 2017 is used for training the forecast models and from 01.2018-08.2018 for evaluation. For

[3] www.renewables.ninja.
[4] https://apps.ecmwf.int/data sets/data/tigge/
[5] https://cds.climate.copernicus.eu/home

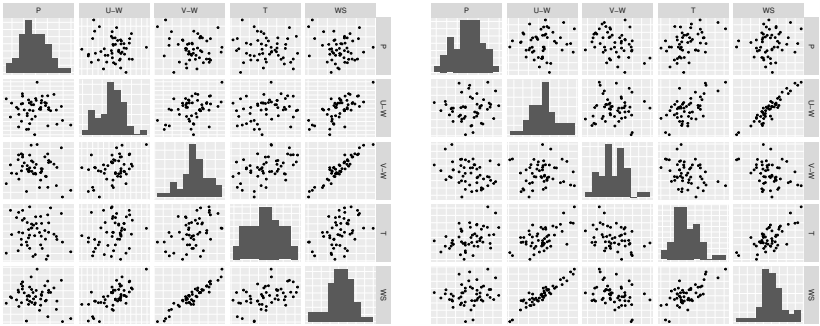
more detailed information on the benchmark data set, including information on how to replicate it, see Phipps et al. [11]

4.1.2 Swedish Data

The Swedish electricity system is divided into four sub-areas or bidding zones with the present paper focusing only on the area contained in bidding zones 3 and 4. We download the wind power generation data aggregated on a bidding zone level through the open-source transparency platform which is operated by the European Network of Transmission System Operators (ENTSO-E) [25]. This data is available at an hourly resolution, but due to limitations in the weather data we can only forecast every 3h. The weather data for bidding zone 3 and 4 is made up of the ECMWF EPS Molteni et al. [15] and the ERA5 reanalysis data C3S [24]. The ERA5 data again serves as the ground truth for post-processing, whilst we use the EPS for the ensemble forecasts. We download the parameters two-meter temperature, surface pressure, 100m-U-Component of wind, 100m-V-Component of wind and wind speed. Since the weather data only comes in a grid-based format and we perform forecasts for an entire bidding zone, this weather data must be aggregated. We use a weighted average method to perform this aggregation. We use data from 2015-2017 for training our forecast models and from 01.2018-08.2019 for evaluation. A detailed description of this data set, including the weighted average aggregation method is provided by Phipps et al. [11].

4.2 Results

We evaluate both the ability of ECC to restore the dependency structures and the effect this has on forecast performance. This section presents the results of the analysis for both data sets introduced above. When analysing performance of ECC with scatter plots we only consider April 14, 2018 at 6:00am. The results for all other dates and forecasts horizons are similar and therefore not discussed in detail.



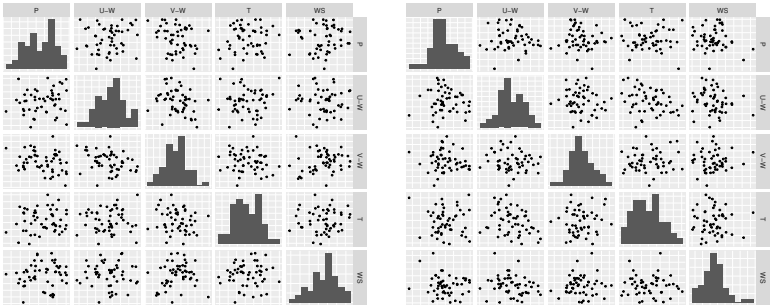
(a) Onshore Benchmark (b) Offshore Benchmark

Figure 2: Scatter plots showing the raw dependency structures for both benchmark data sets on April 14, 2018 at 6am. Pressure (P), the U-component of wind (U-W), the V-component of wind (V-W), temperature (T) and wind speed (WS) are shown. Despite correlation between wind components and wind speed, the dependencies between the weather variables are not particularly strong.

4.2.1 Benchmark Data

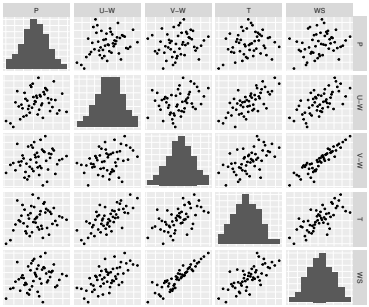
First we consider the effect of ECC on restoring the dependency structures in the data. We see the dependencies between various weather variables in the form of scatter plots in Figure 2. These plots show a histogram of the empirical distribution of individual weather ensembles along the diagonal and scatter plots of their dependencies in the other positions. In the onshore benchmark there is a clear correlation between the V-component of wind and wind speed, and also a slight correlation between wind speed and the U-component of wind. There are no strong dependencies shown between any other weather variables. For the offshore benchmark data the only noticeable correlation is between the U-component of wind and wind speed.

Figures 3 and 4 show how these dependency structures change after applying ensemble post-processing and various ECC methods. Figure 3 details the onshore benchmark and we see, that only applying EMOS removes all dependency structures. ECC-R is also not effective, with no dependencies being visible. Both the ECC-Q and ECC-T methods show improvement. The quantile sampling based ECC-Q leads to an almost symmetric marginal distribution,

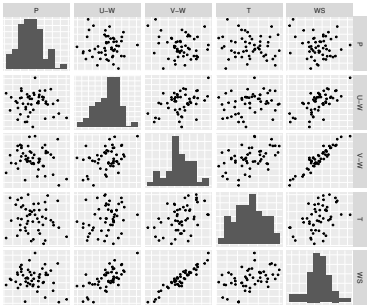


(a) EMOS

(b) ECC-R



(c) ECC-Q

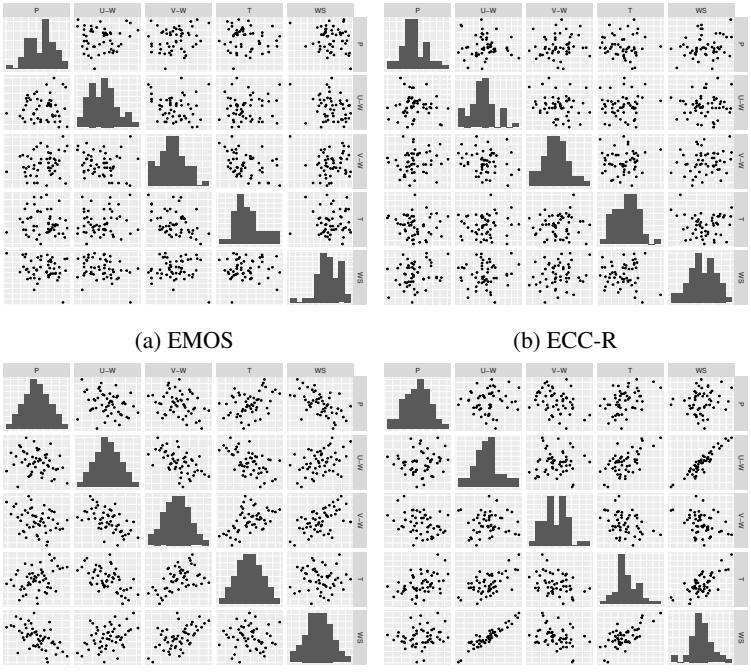


(d) ECC-T

Figure 3: Scatter plots showing the dependency structures with various post-processing strategies that aim to restore the raw dependency structure of the onshore benchmark on April 14, 2018 at 6am. Pressure (P), the U-component of wind (U-W), the V-component of wind (V-W), temperature (T) and wind speed (WS) are shown. We see that EMOS destroys all dependencies and ECC-R is not effective in restoring the structures. ECC-Q and ECC-T both restore the dependency structures effectively.

but also accurately recreates the dependencies between the weather variables. Since ECC-T is based on a transformation it is not surprising that this method recreates the dependencies with the most accuracy. The results are similar for the offshore benchmark in Figure 4, with the exception of the ECC-Q method. Although we see some dependency structures being recreated, ECC-Q is not as effective here as in the onshore data set.

In order to assess the effect of ECC on forecast accuracy we consider plots of the mean CRPS for each forecast horizon on each benchmark data set. Figure



(a) EMOS (b) ECC-R

(c) ECC-Q (d) ECC-T

Figure 4: Scatter plots showing the dependency structures with various post-processing strategies that aim to restore the raw dependency structure of the offshore benchmark on April 14, 2018 at 6am. Pressure (P), the U-component of wind (U-W), the V-component of wind (V-W), temperature (T) and wind speed (WS) are shown. We see that EMOS destroys all dependencies and ECC-R is not effective in restoring the structures. ECC-Q shows some improvements, but ECC-T restores the dependency structures the best.

5 compares the means CRPS of One-Step-P against variants of One-Step-W. Figure 6 also plots the mean CRPS scores, but this time for variations of Two-Step-WP against One-Step-P. The One-Step-P strategy is almost always slightly more accurate than the variations of One-Step-W and very similar to the Two-Step-WP variations. We also see, that there is almost no difference between the various variations One-Step-W and Two-Step-WP based on different ECC methods. The neural networks perform worse than the linear models on the benchmark data. This is mainly due to a lack of training data but

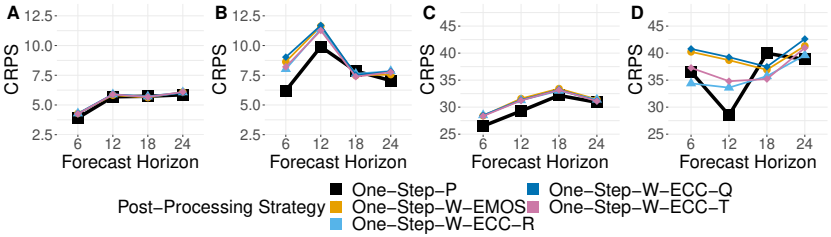


Figure 5: Plot comparing the average CRPS score for the test data on the benchmark data set for each forecast horizon using the One-Step-W variants to the One-Step-P strategy. In (A) we see the linear model for the onshore data and (B) the neural network. (C) and (D) show the evaluation of the linear model and neural network on the offshore data set.

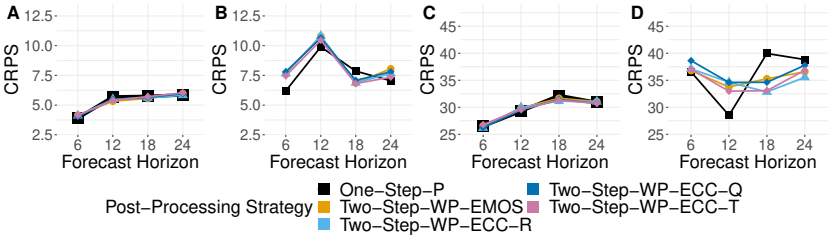


Figure 6: Plot comparing the average CRPS score for the test data on the benchmark data set for each forecast horizon using the Two-Step-WP variants to the One-Step-P strategy. In (A) we see the linear model for the onshore data and (B) the neural network. (C) and (D) show the evaluation of the linear model and neural network on the offshore data set.

also suggest that a linear model is more than capable of delivering accurate forecasts.

4.2.2 Swedish Data

The effect of ECC on the dependency structures in the Swedish data is similar to the benchmark data set and therefore we do not discuss it here in detail. The mean CRPS values for bidding zone 3 are shown in Table 1 and for bidding zone 4 in Table 2. Again all One-Step-W variations deliver similar results and perform noticeably worse than the One-Step-P strategy. The Two-Step-WP performs similarly to the One-Step-P method, but again applying ECC has little effect. On a whole, we see that as with the benchmark data set ECC has

Table 1: Summary of mean CRPS for the test data in bidding zone 3 in Sweden. The best prediction for each strategy and each forecast model is highlighted in bold.

Data Set		6h	12h	18h	24h
	Linear Raw	97.12	84.86	91.97	94.28
	Linear One-Step-P	64.12	68.90	64.62	69.27
	Linear One-Step-W-E	107.95	95.28	103.85	100.43
	Linear One-Step-W-R	107.95	95.26	103.21	100.24
	Linear One-Step-W-Q	108.21	95.48	103.67	100.27
	Linear One-Step-WP-T	107.89	95.43	103.60	100.17
	Linear Two-Step-WP-E	63.70	**67.26**	**63.28**	67.38
	Linear Two-Step-WP-R	64.50	67.40	63.61	**67.02**
	Linear Two-Step-WP-Q	64.55	67.41	63.65	68.10
Bidding	Linear Two-Step-WP-T	**63.58**	67.78	64.04	68.03
Zone 3	Neural Raw	64.35	70.73	72.55	66.99
	Neural One-Step-P	**61.05**	63.02	67.13	**59.54**
	Neural One-Step-W-E	73.40	64.81	63.26	74.45
	Neural One-Step-W-R	74.60	65.29	65.34	74.11
	Neural One-Step-W-Q	74.85	65.52	65.07	75.45
	Neural One-Step-W-T	74.79	66.18	64.52	74.38
	Neural Two-Step-WP-E	65.69	**59.37**	56.79	69.38
	Neural Two-Step-WP-R	67.23	60.81	57.12	69.65
	Neural Two-Step-WP-Q	66.27	61.04	57.29	70.87
	Neural Two-Step-WP-T	67.51	62.23	**56.22**	70.29

almost no effect on the forecast accuracy, with all ECC variants performing similarly to post-processing strategies where only EMOS is used. In the case of the Swedish data we also note, that the neural network performs slightly better than the linear model in bidding zone 3.

5 Discussion

The fact that the ECC variants perform differently is not surprising as similar results are observed by Schefzik et al. [3]. Across all data sets applying EMOS destroys the dependency structures and ECC-R is relatively ineffective

Table 2: Summary of mean CRPS for the test data in bidding zone 4 Sweden. The best prediction for each strategy and each forecast model is highlighted in bold.

Data Set		6h	12h	18h	24h
Bidding Zone 4	Linear Raw	58.50	67.00	56.51	51.54
	Linear One-Step-P	45.13	51.90	50.35	44.26
	Linear One-Step-W-E	59.48	60.39	55.97	50.73
	Linear One-Step-W-R	59.48	60.78	56.33	51.28
	Linear One-Step-W-Q	59.71	61.01	56.35	50.71
	Linear One-Step-W-T	59.46	60.80	56.33	50.72
	Linear Two-Step-WP-E	45.21	52.69	49.89	43.92
	Linear Two-Step-WP-R	45.00	52.75	50.48	43.48
	Linear Two-Step-WP-Q	44.54	52.38	**50.11**	43.41
	Linear Two-Step-WP-T	**44.43**	**51.44**	50.23	**43.34**
	Neural Raw	52.74	**46.70**	51.02	**43.98**
	Neural One-Step-P	**49.55**	47.80	**48.11**	46.22
	Neural One-Step-W-E	82.29	90.97	100.03	82.26
	Neural One-Step-W-R	81.34	88.97	98.62	80.29
	Neural One-Step-W-Q	78.32	86.29	95.83	78.22
	Neural One-Step-W-T	74.79	82.88	89.79	73.70
	Neural Two-Step-WP-E	52.06	58.04	63.87	49.42
	Neural Two-Step-WP-R	51.79	56.92	62.82	49.39
	Neural Two-Step-WP-Q	52.48	57.19	63.47	49.00
	Neural Two-Step-WP-T	53.27	56.67	61.52	49.35

in rebuilding dependency structures. Since ECC-Q relies on quantile sampling, it produces marginal distributions that are always close to symmetric. With regards to dependency structures, ECC-Q delivers mixed results. For the onshore benchmark (and also bidding zone 3) it recreates the dependency structures almost as accurately as ECC-T, but not for the other data sets. This could be due to ECC-Q still relying on sampling from the EMOS distribution, which we obtain by considering the last 40 days. Therefore it is possible that the EMOS parameters vary in accuracy which could lead to a ECC-Q sampling that is also slightly worse. ECC-T on the other hand is based on a monotonic transformation and it is therefore expected that it always accurately recreates the dependency structures of the raw ensembles. ECC-T therefore

unsurprisingly performs the best in this regard, but the trade-off is the marginal distributions, which are not as symmetrical as those from ECC-Q.

The key focus of the present paper is however, evaluating how the inclusion of ECC affects post-processing strategies. Although, as discussed above, the performance of the various ECC variants differs, this appears to have no effect on the forecast performance. The only instance in which the ECC variations noticeably diverge is when the neural network is used in bidding zone 4 in Sweden with the One-Step-W strategy. In this case the One-Step-P method performs far better than any One-Step-W variation, and the EMOS variant of One-Step-W performs better than all ECC strategies. In this case ECC didn't lead to any improvement, but actually caused the forecasts to be slightly worse. For other post-processing strategies and forecast horizons there is no noticeable difference between the One-Step-W or Two-Step-WP variations with and without ECC. For the forecast horizons and data sets considered, we therefore find that dependency structures do not play a significant role with regards to wind power forecast accuracy.

We consider two possible explanations for this. Firstly, the dependency structures present in both our data sets are not strong. Although Schefzik et al. [21] reported improved results when using ECC, they were working with data sets that showed clear dependencies. Since there are no strong dependency structures for ECC to restore in our datasets, it makes sense that this does not lead to an improvement. We therefore suggest to investigate the dependency structures before deciding whether ECC is included in the post-processing strategy. The second explanation is that the wind power forecast model is capable of implicitly restoring these dependencies. When the weather ensembles are used to generate wind power forecasts an implicit calibration (due to the training of the model with historical weather variables and observed/simulated wind power generation) occurs and this could be sufficient to negate the effect of the missing dependency structures.

6 Conclusion

The present paper investigates whether including Ensemble Copula Coupling (ECC) into different post-processing strategies affects which of the strategies is optimal. We show that ECC, particularly ECC-Q and ECC-T, restores the dependency structures effectively, but this does not affect the forecast performance. The strategies post-processing the weather variables only (One-Step-W) or both the weather variables and the wind power (Two-Step-WP) deliver almost identical results, regardless of whether ECC is used or not. Given these results, we conclude, that ECC does not change the optimal post-processing strategy for wind power forecasts. Due to the smaller number of post-processing steps required and superior or similar forecast accuracy, the strategy post-processing only the resulting wind power (One-Step-P) remains optimal. However, our data does not contain strong dependency structures which limits the potential of ECC. Therefore, we recommend investigating the dependency structures before selecting a post-processing strategy.

Future work should focus on investigating data with different dependency structures to understand when ECC plays a role. Additionally, while the present paper focuses solely on dependencies between weather variables, future work should focus on recreating the spatial and temporal dependencies with ECC and analysing their effect on forecast performance.

Acknowledgements

This work was partly funded by the German Research Foundation (DFG) Research Training Group 2153 “Energy Status Data – Informatics Methods for its Collection, Analysis and Exploitation” and Helmholtz AI. The research detailed in the current paper was based on data from the ECMWF obtained through an academic licence for research purposes.

References

[1] P. Pinson and J. W. Messner, "Application of postprocessing for renewable energy," in *Statistical postprocessing of ensemble forecasts* (S. Vannitsem, D. S. Wilks, and J. Messner, eds.), pp. 241–266, Amsterdam: Elsevier, 2018.

[2] T. L. Thorarinsdottir and T. Gneiting, "Probabilistic forecasts of wind speed: ensemble model output statistics by using heteroscedastic censored regression," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 173, no. 2, pp. 371–388, 2010.

[3] R. Schefzik, T. L. Thorarinsdottir, and T. Gneiting, "Uncertainty quantification in complex simulation models using ensemble copula coupling," *Statistical Science 2013, Vol. 28, No. 4, 616-640*, 2013.

[4] R. Schefzik, "Ensemble calibration with preserved correlations: unifying and comparing ensemble copula coupling and member-by-member postprocessing," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 999–1008, 2017.

[5] T. Gneiting, *Calibration of medium-range weather forecasts*. 2014.

[6] T. Gneiting, A. E. Raftery, A. H. Westveld, and T. Goldman, "Calibrated probabilistic forecasting using ensemble model output statistics and minimum CRPS estimation," *Monthly Weather Review*, vol. 133, no. 5, pp. 1098–1118, 2005.

[7] T. Gneiting, K. Larson, K. Westrick, M. G. Genton, and E. Aldrich, "Calibrated probabilistic forecasting at the stateline wind energy center," *Journal of the American Statistical Association*, vol. 101, no. 475, pp. 968–979, 2006.

[8] C. Gilbert, J. W. Messner, P. Pinson, P.-J. Trombe, R. Verzijlbergh, P. Dorp, and H. Jonker, "Statistical post–processing of turbulence–resolving weather forecasts for offshore wind power forecasting," *Wind Energy*, vol. 23, no. 4, pp. 884–897, 2020.

[9] C. Sweeney and P. Lynch, “Adaptive post-processing of short-term wind forecasts for energy applications,” *Wind Energy*, vol. 14, no. 3, pp. 317–325, 2011.

[10] A. Bossavy, R. Girard, and G. Kariniotakis, “Forecasting ramps of wind power production with numerical weather prediction ensembles,” *Wind Energy*, vol. 16, no. 1, pp. 51–63, 2013.

[11] K. Phipps, S. Lerch, M. Andersson, R. Mikut, V. Hagenmeyer, and N. Ludwig, “Evaluating ensemble post-processing for wind power forecasts,” http://arxiv.org/abs/2009.14127v1.

[12] N. Ludwig, S. Arora, and J. W. Taylor, “Probabilistic load forecasting using post-processed weather ensemble predictions,” *Journal of the Operational Research Society (submitted)*, 2020.

[13] S. Späth, L. von Bremen, C. Junk, and D. Heinemann, “Time-consistent calibration of short-term regional wind power ensemble forecasts,” *Meteorologische Zeitschrift*, vol. 24, no. 4, pp. 381–392, 2015.

[14] M. Taillardat, O. Mestre, M. Zamo, and P. Naveau, “Calibrated ensemble forecasts using quantile regression forests and ensemble model output statistics,” *Monthly Weather Review*, vol. 144, no. 6, pp. 2375–2393, 2016.

[15] F. Molteni, R. Buizza, T. N. Palmer, and T. Petroliagis, “The ECMWF ensemble prediction system: Methodology and validation,” *Quarterly Journal of the Royal Meteorological Society*, vol. 122, no. 529, pp. 73–119, 1996.

[16] C. Fraley, A. E. Raftery, and T. Gneiting, “Calibrating multimodel forecast ensembles with exchangeable and missing members using bayesian model averaging,” *Monthly Weather Review*, vol. 138, no. 1, pp. 190–202, 2010.

[17] T. Gneiting and M. Katzfuss, “Probabilistic forecasting,” *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014.

[18] H. Joe, *Multivariate models and multivariate dependence concepts*. CRC Press, 1997.

[19] R. B. Nelsen, *An introduction to copulas*. Springer Science & Business Media, 2007.

[20] M. Sklar, *Fonctions de Répartition À N Dimensions Et Leurs Marges*. Université Paris 8, 1959.

[21] R. Schefzik, "Ensemble calibration with preserved correlations: unifying and comparing ensemble copula coupling and member-by-member postprocessing," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 703, pp. 999–1008, 2017.

[22] I. Staffell and S. Pfenninger, "Using bias-corrected reanalysis to simulate current and future wind power output," *Energy*, vol. 114, pp. 1224–1239, 2016.

[23] R. Swinbank, M. Kyouda, P. Buchanan, L. Froude, T. M. Hamill, T. D. Hewson, J. H. Keller, M. Matsueda, J. Methven, F. Pappenberger, M. Scheuerer, H. A. Titley, L. Wilson, and M. Yamaguchi, "The TIGGE project and its achievements," *Bulletin of the American Meteorological Society*, vol. 97, no. 1, pp. 49–67, 2016.

[24] Copernicus Climate Change Service (C3S), "Era5: Fifth generation of ecmwf atmospheric reanalyses of the global climate," 2017.

[25] European Network of Transmission System Operators (ENTSO-E), "Transparency platform (tp)," 2019.

On the Detection of SARS-CoV-2 induced Pneumonia in X-Ray Thorax Images with Convolutional Neural Networks

Patrick Kurz[1], Paul Kaufmann[1], Roman Kalkreuth [2], Jannis Born[3], Roman Klöcker[4], Felix Hahn[4], Felix Döllinger[5], Timo A. Auer[5]

[1]Chair for Computational Intelligence
Johannes Gutenberg University Mainz, Germany
E-Mail: pakurz@students.uni-mainz.de, paul.kaufmann@uni-mainz.de
[2]Department of Computer Science
TU Dortmund University, Germany
Email: roman.kalkreuth@tu-dortmund.de
[3]Department of Biosystems Science and Engineering
ETH Zürich, Zürich, Switzerland
Email: jborn@ethz.ch
[4]Department of Diagnostic and Interventional Radiology
University Medical Center of the Johannes Gutenberg-University Mainz, Germany
Email: roman.kloeckner@unimedizin-mainz.de, fhahn@uni-mainz.de
[5]Department of Radiology
Charité Universitätsmedizin Berlin, Germany
Email: felix.doellinger@charite.de, timo-alexander.auer@charite.de

Abstract

SARS-CoV-2 is a highly contagious virus that can induce pulmonary complications like viral pneumonia and acute respiratory distress syndrome (ARDS). In order to support RT-PCR testing, chest X-rays are used to identify the presence of COVID-19 in lungs. Radiologists proficient in chest X-Ray (CXR) interpretation are scarce, motivating our work of examining the performance of convolutional neural networks (CNNs) on this task. CNNs are the state-of-the-art image classification method. In this work we classify X-rays into four classes (COVID-19, other lung opacity, other diseases and normal). We find, that MobileNetV1 is the best CNN for this task and achieve an overall accuracy of 70% and a COVID-19 accuracy of 83% on a test data set. By increasing the number of images of the unrestricted classes normal, lung opacity and other, the COVID-19 accuracy can be increased to 95% and the overall accuracy to 78%. We leverage model interpretability techniques and provide attention heatmaps that can assist in validating the model's decision process.

1 Introduction

The novel coronavirus disease 2019 (COVID-19) has resulted in an ongoing outbreak of viral pneumonia all over the world. By now more than 12.9 million people have been infected of which more than 570 thousand have died [1]. The containment of the disease is based on the identification of infected people, backtracing of infections, and isolation of contagious people. Due to the fact, that COVID-19 is easily confused with influenza, specific tests are necessary. There are different methods to test for the existence of SARS-CoV-2 such as an RNA-based assay using nasopharyngeal swabs (RT-PCR testing). Unfortunately, there were already shortages of swabs in the past and testing capacities reached their limits in various regions of the world [2]. Therefore, supporting methods for COVID-19 identification are needed. One alternative way of testing COVID-19 is thorax X-Ray imaging which can give immediate diagnostic information. Additionally, the stage of the disease and the risk of a severe course can be seen [3]. While thoracic X-Ray can be conducted

rapidly, the scarce availability of analyzing radiologists depict a key bottleneck toward accelerating automatic differential diagnosis. Therefore, the study of automatic image classification approaches for the identification of COVID-19 is an emerging research topic. This work focuses on convolutional neural networks (CNNs). Our contributions can be summarized as follows: Using transfer learning, different CNN base models for image classification tasks such as Xception, ResNet50, InceptionV3, MobilenetV1, and MobilenetV2 are compared with well-known classification performance metrics. The analysis finds, that MobileNetV1 is the best CNN for this task and achieved an overall accuracy of 70% and a COVID-19 accuracy of 83% on a test data set. These results are based on a relatively small training data set (each class n=1000, except COVID-19 n=538). By increasing the number of images of the unrestricted classes normal, lung opacity and other, the COVID-19 accuracy can be increased to 95% and the overall accuracy to 78%. This work also indicates that there may be unique features in thorax X-Ray images of COVID-19 infected people which differ from other forms of pneumonia. This raises new questions for further research such as why some cases of COVID-19 are relatively easy to identify and what are the key characteristics on which a CNN determines the presence of COVID-19. Future work on these research questions might be helpful to obtain more understanding of COVID-19 which could contribute to the development of further treatments.

This work is structured as follows: We give an explanation of convolutional neural networks and transfer learning in Chapter 2. In Chapter 3 we survey relevant work in the field of medical image recognition with focus on the detection of COVID-19. Chapter 4 presents an overview of the data which we used for our experiments. Chapter 6 revolves around the structure and results of our experiments 5. We discuss our results and findings in Chapter 6. Finally, Chapter 7 summarizes the outcome of this work, and future research is proposed.

2 Related Work

2.1 Convolutional Neural Networks

Convolutional neural networks (CNN) are a state-of-the-art method for image classification. The advantage of CNNs to standard (deep) feed-forward networks is that they can work with two-dimensional input data. This allows CNNs to capture spatial dependencies of an image and reduce parameters as well as improve the reusability of weights [4]. CNNs receive matrices as input. Different layer types process the information. The most important ones are the convolutional layer, the fully connected layer, and the pooling layer.

The convolutional layer is the primary building block of CNNs. In the convolutional step, an input image is multiplied (dot product) with a weight matrix (filter) to generate a feature map representing one feature of the input image. Subsequent pooling steps reduce the size of feature maps while keeping the essential information. Pooling is a filter that strides over the given feature map and returns only a single value.

The last layers of a CNN are usually fully connected. Within these layers, every neuron is connected to every other neuron. Because fully connected layers can only process vectors, the last feature maps are flattened. The structure of a typical CNN is shown in Figure 2.

2.2 Transfer Learning

Many medical image classification tasks suffer from a low number of cases. Transfer learning can be used in such situations. Transfer learning describes the process of not training CNNs from scratch but to employ pre-trained models instead. The training time reduces dramatically as the weights of the most layers are rendered immutable, and only the last layers are re-trained. The results are excellent since most image features (e.g., edge detection) can be reused on related problems without modifications [5, 6, 5].

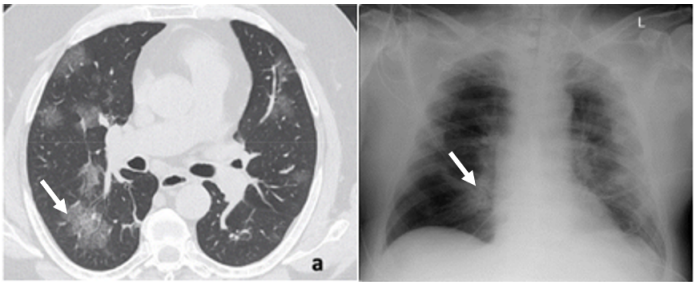


Figure 1: Comparison of COVID-19 glass-opacity in CT scan (left) and X-ray (right). Left image from Caruso et al. [16] right image from actual data set

3 CNN for Medical Image Recognition

The past few years have witnessed a rise of deep learning methods in medical image analysis [7]. CNNs now match or surpass the performance of humans in disease detection across a rich set of tasks [8] such as mammography detection [9], diagnosing pulmonary conditions from X-Ray [10], and computed tomography (CT) data [11].
COVID-19 induces patterns of viral pneumonia that can be identified through imaging techniques such as chest X-rays, CT, lung ultrasound. The most prevalent patterns are ground-glass opacity (GGO) and patchy consolidations in CT and X-Ray, and B-lines and pleural line abnormalities in ultrasound (for a detailed review see [12]). Throughout 2020, several hundred publications attempting to automatically detect COVID-19 from imaging data appeared [13]; overviews about individual methods can be found in [14, 15]). In the following, results of all three methods are briefly summarized.

3.1 COVID-19 Detection in X-rays

In current research there are two publications outstanding to the others. One was written by Narin et al. [18], who achieved the overall best score of 97% accuracy (COVID-19 sensitivity = 96%) with a pretrained ResNet50 model on a binary classification task (COVID-19 / healthy). As data set they used 50 X-rays of healthy and 50 of COVID-19 infected people [18]. The other

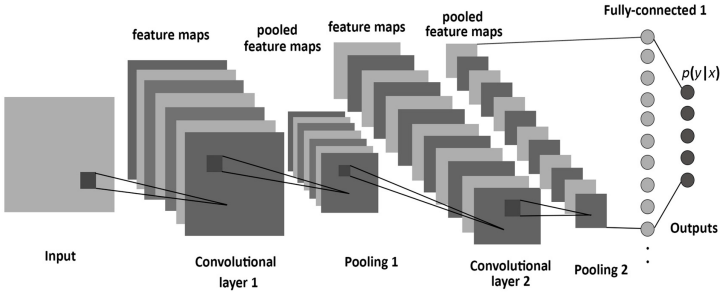


Figure 2: A typical convolutional network (Source: Albelwi and Mahmood [17], p. 5)

is related to this seminar. Apostolopoulos and Mpesiana [19] conducted a comparison of multiple CNNs for classification into three classes (common pneumonia, COVID-19, and normal incidents). They compared pretrained VGG19, MobileNetV1, Inception, Xception and Inception ResNetV2 on a data set containing 224 COVID-19, 700 common pneumonia and 500 healthy X-rays. They found VGG19 with the best overall accuracy of 92.85% (COVID-19 sensitivity = 92.85%) and MobileNetV1 with the best COVID-19 sensitivity of 99.10% (accuracy = 92.85%) [19].

3.2 COVID-19 Detection in CT Scans

With CT Scans it is possible to create cross-sectional images of the human body and thereby create a more holistic view of a specific body region. To achieve this, multiple X-rays are conducted from different angles, where detectors compute the image. Due to this procedural, more images from one patient can be generated. On the downside the amount of radiation exposure is much higher. In recent publications the biggest data set was generated by Chen et al. [20], who used 46.096 CT images from just 106 patients (51 COVID-19 positive). With that data, a U-Net++ model was trained to determine the presence of COVID-19. The model achieved an overall accuracy of 95.24% (COVID-19 sensitivity = 100%) on the test set. This model was also tested in combination with radiologists and improved their reading time of a CT scan by 65% (Chen et al. 2020). Another publication worth mentioning is

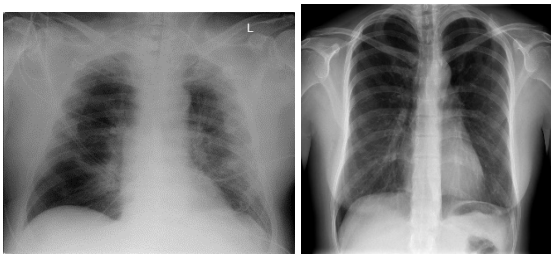
written by Butt et al. [21]. The authors achieved the best accuracy with 86.7% (COVID-19 sensitivity = 86.7%) on a three classes classification task (COVID-19, Influenza-A, irrelevant-to-infection). This accuracy was achieved by using a ResNet23-based model on a data set with 618 images (CT samples from 110 patients with COVID-19, 224 with Influenza-A and 175 healthy people) [21]. We show that in case of binary and tertiary classification an accuracy and sensitivity of around 90% on both image types is possible.

3.3 COVID-19 Detection in Lung Ultrasound (LUS)

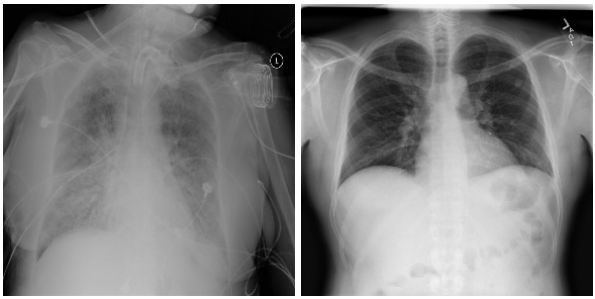
LUS has been repeatedly recommended by clinical authorities [22] but has been neglected by the ML community [13], partially due to data heterogeneity resulting from higher operator dependency. However, public databases with LUS recordings of different pathologies are emerging [23] and the authors achieve a specificty of 0.91 and sensitivity of 0.98 on COVID-19 detection in a 3-class classification incorporating also bacterial pneumonia and healthy controls [24]. Others focused on severity assessment of COVID-19 infection and achieve recall of 0.6 and positive predictive value of 0.7 [25].

4 Preparation of data

The different classification models need a common data set to make the results comparable. Therefore, a training set and a test set are built. Both sets consist of front view chest X-ray images divided in four different classes: COVID-19, other lung opacities, other diseases, and healthy lungs (Figure 3). All but the COVID-19 images are randomly drawn from the Kaggle RSNA Pneumonia Detection Challenge (Kaggle 2018). The COVID-19 data set was built by Kalkreuth and Kaufmann [26] which contains multiple sources (162 images). This data set is expanded with images provided by radiologists of the Johannes Gutenberg-University Mainz (73 images), by the Charité – Universitätsmedizin Berlin (9 images) and by the Cohen database [27] (389 images). The structure of the training set (85% of the whole data set) and test set (15% of the whole dataset) is shown in Table 1. As visible the number of COVID-19 images is



(a) COVID-19 (b) Healthy Lung



(c) Lung Opacity (d) Other Lung Disease

Figure 3: Example images from the data set

much smaller than the number of images in the other classes. On the given set the CNNs are trained and compared to identify the best one which is then optimized. On the one hand this produces an overview of the performance of different CNN and on the other hand it shows which model is the most promising for further fine tuning.

5 Experiments

5.1 Overview of Experiments

Figure 4 gives an overview of all experiments carried out. The first experiment was conducted to identify suitable hyperparameters. MobileNetV2 was chosen as the reference model due to its relatively small size. This introduces bias to the experiments because a good learning rate might differ between different

Table 1: Structure of the training and test dataset

Class	Training Set	Validation Set
COVID-19	538	95
Other lung opacity	1000	176
Other lung disease	1000	176
Healthy lung	1000	176

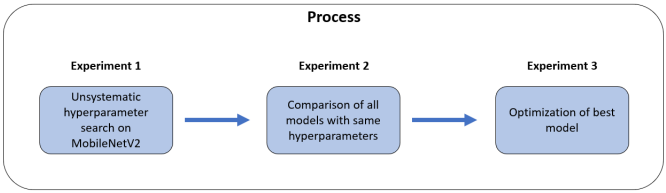


Figure 4: Process of experiments

models. In the second experiment the different models are compared with the given hyperparameters of the first experiment. The last experiment revolves around the optimization of the best model from the second experiment. In the following, each chapter describes the experiment in detail as well as its results.

5.2 Identification of Hyperparameter (Experiment 1)

The first experiment was used to identify the hyperparameter which work good for initial comparison. For this experiment the pretrained MobileNetV2 was used as the base model expanded by some custom layers which is shown in Figure 5.

Each set of hyperparameter settings (as well as the experiments later) should been conducted with a cross validation of at least five. Unfortunately, at the time of this work there was not enough computational power available. Thereby, the experiments should be re-evaluated with cross validation when possible. The results have been compared on the best validation error through

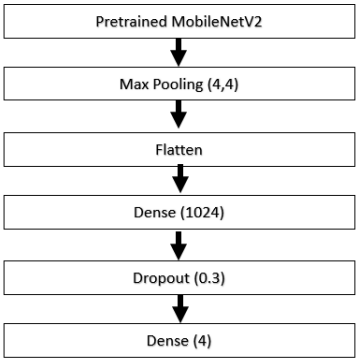


Figure 5: Additional layers on top of the pretrained model

training. Following hyperparameter have been sighted: the Adam learning rate, number of epochs, data augmentation, dropout rate and unfreezing of layers. The outcome indicates that a learning rate of 0.001, 60 epochs with a completely frozen pretrained model and nearly no data augmentation are good starting values to compare the different CNNs. Even though just little optimization has been conducted, the MobileNetV2 already has an accuracy of 64.8% on the validation set.

5.3 Benchmark of CNNs for COVID-19 identification (Experiment 2)

As a next step the five CNNs (Xception, ResNet50, InceptionV3, MobilenetV1, MobilenetV2) are compared with following Hyperparameters:

Just as in the experiments before the validation accuracy was used to compare the performance of the different base models. Not until experiment three, we have a look at the actual performance on the test set. All experiments were conducted with a cross validation of five and the mean is plotted. Figure 6 - Figure 10 show the training and validation accuracy for each model with Figure 11 as a summary. The three best performing models, regarding maximum validation accuracy over all epochs are MobileNetV1 (68.46%), InceptionV3

Table 2: Hyperparameter for Comparison

Hyperparameter	Value
Optimizer	Adam
learning rate	0.001
dropout rate	0.3
epochs	60
pretrained	ImageNet
freeze	Completely frozen
data augmentation	Rotation range = 10

(66.17%), and Xception (60.54%). Despite the max validation accuracy, we decided to optimize the MobileNetV2 due to its low training accuracy. It seems like an increase in the amount of epochs will also increase the overall accuracy which makes it much easier to optimize. Another side effect is the small size of the network which might be able to run on mobile x-ray devices. Thereby, MobileNetV2 is selected as the most promising CNN to optimize its hyperparameter systematically in experiment three.

5.4 Optimization of MobileNetV2 (Experiment 3)

In the last round of experiments, we try to optimize the MobileNetV2 as far as possible regarding the overall accuracy and then have a look at the performance on the test set. Equally to the previous experiment, cross validation of five was used. All hyperparameter were examined one after each other, consequently it can be possible, that other combinations might even increase the accuracy further, but this topic remains as a subject for following research. First, the learning rate of the adam optimizer was optimized, followed by the dropout rate and the number of frozen layers. The results are summarized in the following table, with best parameters in bold (Table 3 - Table 5):

These hyperparameter where used to train the final model and test it on the never seen test set. Due to the best performance in the second experiment, we also test the MobileNetV1 on the test set. For testing the overall best

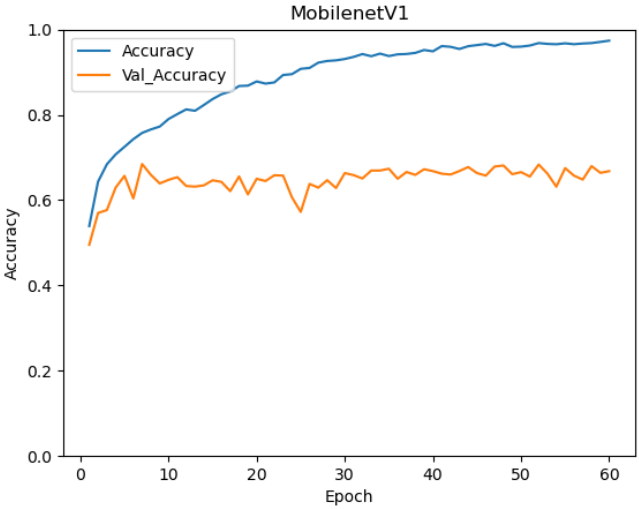


Figure 6: Performance result of MobilenetV1

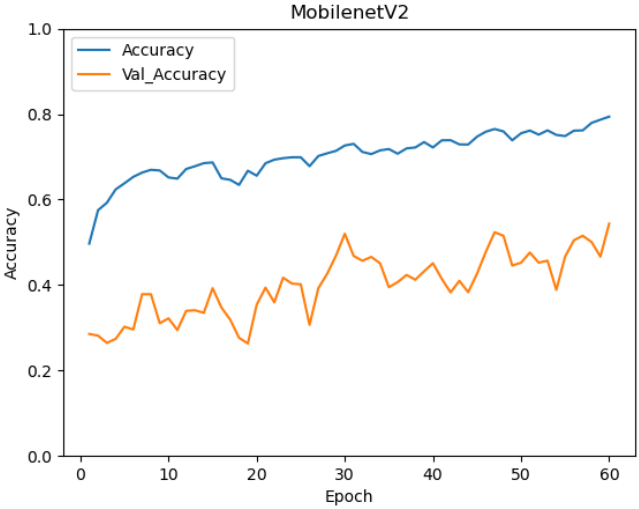


Figure 7: Performance result of MobilenetV2

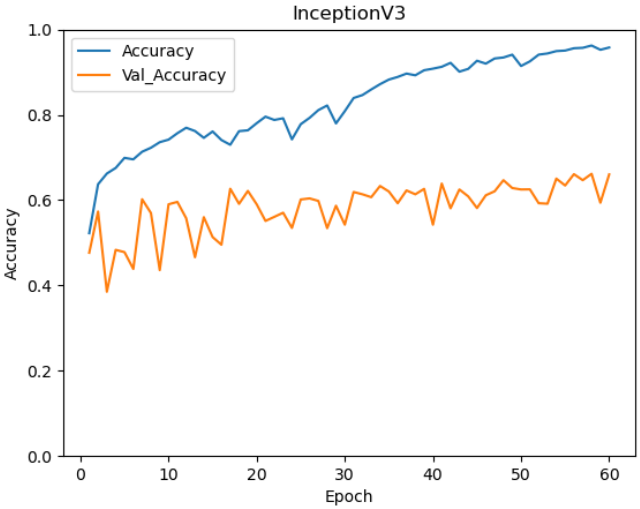


Figure 8: Performance result of InceptionV3

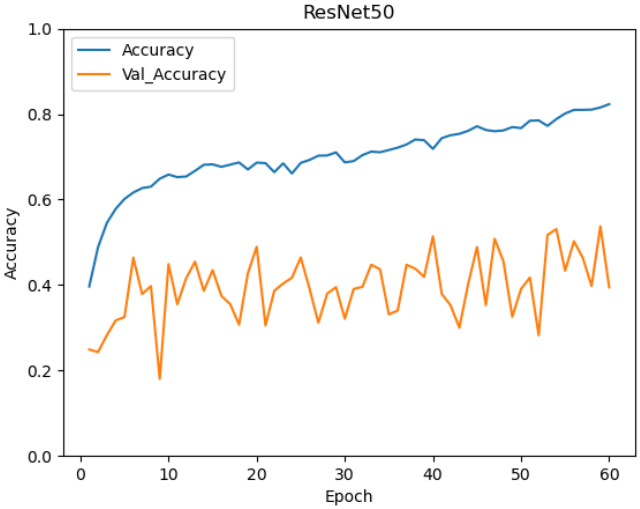


Figure 9: Performance result of ResNet50

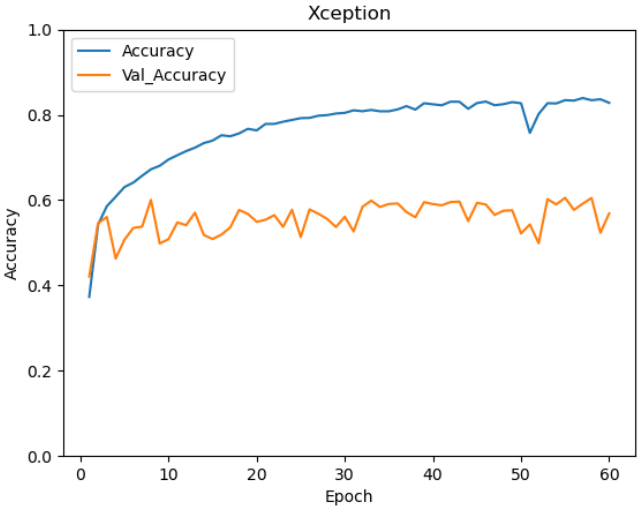


Figure 10: Performance result of Xception

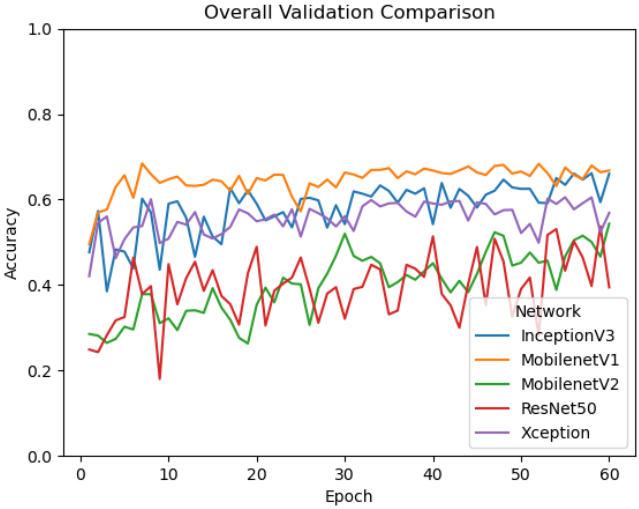


Figure 11: Overall performance of models

Table 3: Results from the optimization of the learning rate

Learning Rate	Val. Accuracy
0.001	0.5848
0.0025	0.3946
0.005	0.3206
0.0075	0.2784
0.00075	**0.606**
0.0005	0.582

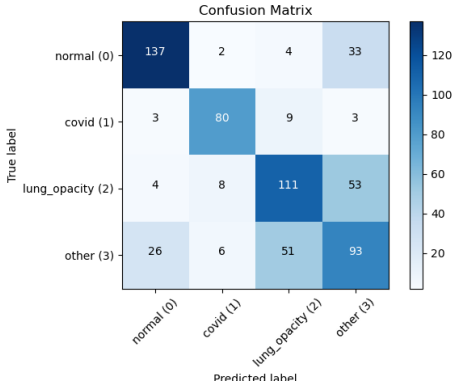
Table 4: Results from the optimization of the dropout rate

Dropout Rate	Val. Accuracy
0.25	0.5467
0.2	0.5416

epoch was used. Figure 12 and Figure 13 show that even though we optimized MobileNetV2 it performs slightly worse than MobileNetV1. Both CNN were able to distinguish COVID-19 and normal lungs from the other classes peaking in a COVID-19 accuracy of 83% (MobileNetV2 78%). Both networks had major problems to differentiate lung opacities from other lung problems. Which leads to lower average accuracy of 70% for the MobileNetV1 and 64% for the MobileNetV2.

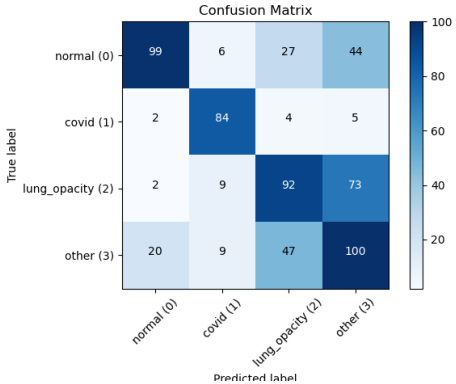
On the upside, the problematic classes are not restricted in terms of images therefore the same models were trained again on a bigger training data set. For each class except COVID-19, 2000 images instead of 1000 images were used. After training the new models were tested on the same test set. The results are summarizes in Figure 14 and Figure 15.

The increase in all classes except COVID-19 leads to an increase in all KPIs and raises the accuracy for COVID-19 detection up to 95% (MobileNetV2 91%).



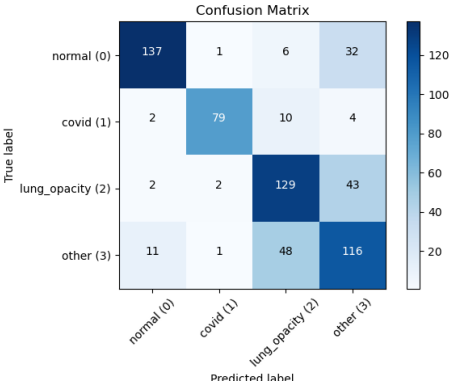
	precision	recall	f1-score	support
normal (0)	0.81	0.78	0.79	176
covid (1)	0.83	0.84	0.84	95
lung_opacity (2)	0.63	0.63	0.63	176
other (3)	0.51	0.53	0.52	176
accuracy			0.68	623
macro avg	0.70	0.69	0.70	623
weighted avg	0.68	0.68	0.68	623

Figure 12: Classification result of MobileNetV1 using 1000 images in each Non-COVID-19 class



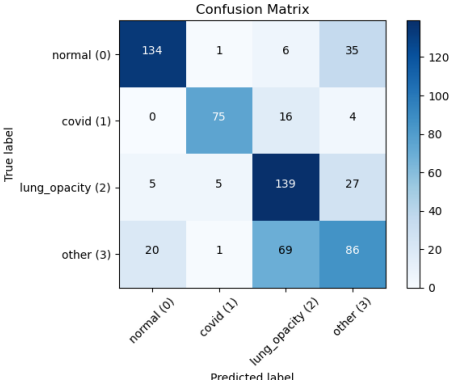
	precision	recall	f1-score	support
normal (0)	0.80	0.56	0.66	176
covid (1)	0.78	0.88	0.83	95
lung_opacity (2)	0.54	0.52	0.53	176
other (3)	0.45	0.57	0.50	176
accuracy			0.60	623
macro avg	0.64	0.63	0.63	623
weighted avg	0.63	0.60	0.61	623

Figure 13: Classification result of MobileNetV2 using 1000 images in each Non-COVID-19 class



	precision	recall	f1-score	support
normal (0)	0.90	0.78	0.84	176
covid (1)	0.95	0.83	0.89	95
lung_opacity (2)	0.67	0.73	0.70	176
other (3)	0.59	0.66	0.63	176
accuracy			0.74	623
macro avg	0.78	0.75	0.76	623
weighted avg	0.76	0.74	0.75	623

Figure 14: Classification result of MobileNetV1 using 2000 images in each Non-COVID-19 class



	precision	recall	f1-score	support
normal (0)	0.84	0.76	0.80	176
covid (1)	0.91	0.79	0.85	95
lung_opacity (2)	0.60	0.79	0.68	176
other (3)	0.57	0.49	0.52	176
accuracy			0.70	623
macro avg	0.73	0.71	0.71	623
weighted avg	0.71	0.70	0.70	623

Figure 15: Classification result of MobileNetV2 using 2000 images in each Non-COVID-19 class

Table 5: Results from the optimization of the unfreeze level

Unfreeze	Val. Accuracy
1	0.4087
2	0.4008
3	0.3847

5.5 Interpretability analysis

Model interpretability is paramount in healthcare applications, because in standard medical treatment, an explanation, accompanying the diagnosis, is required from a physicians. As an instance of post-hoc, gradient-based interpretability methods, GradCAM [28] is a technique that uses the gradients of any target layer (usually the last convolutional layer) to compute a localization heatmap highlighting the most informative parts of an input image given a class label.

Following the training of the MobileNetV2 model, we computed CAMs of all images in the test dataset and provide a few exemplary maps in Figure 16. While it can be seen that the model successfully learned to highlight pulmonary markers like GGOs in, e.g., the COVID-19 example, it is evident from the negative examples that the model is prone to artifcats and occasionally focuses on irrelevant regions such as borders.

6 Discussion

Previous experiments show that the chosen CNNs can identify COVID-19 infected lungs rather good on the test set, despite the small training set. This chapter is dedicated to point out different possible reasons for the examined behaviour. First, COVID-19 has some unique features to other infections and thereby can be identified easier than other classes. This theses would be supported by the fact that also normal lungs have a quite good accuracy compared to the classes lung opacitiy and other where the images are much harder to distinguish. Another reason for the good identification of COVID-19 might be

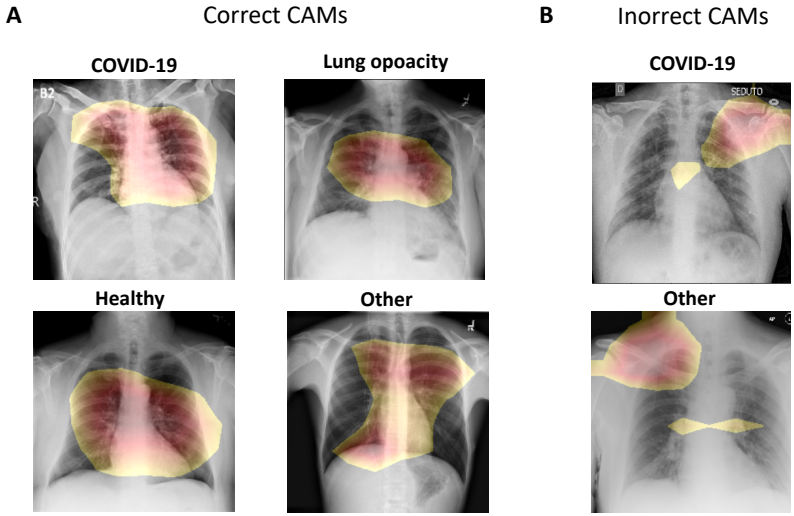


Figure 16: Visualizations of class activation maps with GradCAM [28]. **Subfigure A** shows four samples from the test dataset with CAMs focused on the chest. For example, the heatmap in the COVID-19 is highlighting GGOs. **Subfigure B** depicts two cases where the CAMs are not aligned to pathologically relevant areas.

the underlying data. Due to its novelty COVID-19 images are rare and available data is more homogeneous than the data in the other classes. This can lead to a bias in terms of accuracy where the CNN predicts COVID-19 always when the pictures differ from the homogeneous data from all other classes combined. Then it might not learn the features of COVID-19 but other differences such as the quality of the image to classify. The presence of this problem at least to some degree is supported by the last experiment with increased samples for all classes but COVID-19. Even though, the amount of COVID-19 samples stayed the same, its accuracy increased by 12% (MobileNetV1). Since this finding has been a major problem for our work, we started to analyze the predictions with the help of GradCA and found that our model can, to some extent, learn to focus on the relevant regions in the image. Although we also found a significant amount of samples with a mislocated attentional focus, it is encouraging to see the model extracting spatial biomarkers in a self-supervised way and without any segmentation masks. However, in a next step, lung segmentation with a U-

Figure 17: Lung segmentation

Net segmentation model [29] will be used to exclude identification by features other than the lung. An example is shown is Figure 17.

7 Conclusion and future work

In this work different CNN's were compared and the most promising (MobileNetV2) was optimized. Unexpectedly, MobileNetV1 still performed better and peaked in an overall accuracy of 78% and a COVID-19 accuracy of 95% the model was able to detect COVID-19 infected lungs. These scores are relatively high especially for COVID-19. This increases the risk of the probability that the underlying data mainly generated from two sources introduced some bias. This work raises multiple questions, which can be examined in further research. One topic might be why the cases of COVID-19 are relatively easy to identify and in combination with that what are the key characteristics on which a CNN determines the presence of COVID-19. This research could help to further understand the virus and ultimately might help to find a cure. Another topic might be to compare the performance of CNNs with support vector machines or other image classification methods.

References

[1] World Health Organization et al. Coronavirus disease 2019 (covid-19): situation report, 176. 2020.

[2] Zhengtu Li, Yongxiang Yi, Xiaomei Luo, Nian Xiong, Yang Liu, Shaoqiang Li, Ruilin Sun, Yanqun Wang, Bicheng Hu, Wei Chen, Yongchen Zhang, Jing Wang, Baofu Huang, Ye Lin, Jiasheng Yang, Wensheng Cai, Xuefeng Wang, Jing Cheng, Zhiqiang Chen, Kangjun Sun, Weimin Pan, Zhifei Zhan, Liyan Chen, and Feng Ye. Development and clinical application of a rapid igm-igg combined antibody test for sars-cov-2 infection diagnosis. *Journal of Medical Virology*.

[3] Mahmud Mossa-Basha, Carolyn C. Meltzer, Danny C Kim, Michael J Tuite, K. Pallav Kolli, and Bien Soo Tan. Radiology department preparedness for covid-19: Radiology scientific expert panel. *Radiology*, 0(0):200988, 0. PMID: 32175814.

[4] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences, 2014.

[5] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: Transfer learning from unlabeled data. ICML '07, page 759–766, New York, NY, USA, 2007. Association for Computing Machinery.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, page 580–587, USA, 2014. IEEE Computer Society.

[7] Ahmed Hosny, Chintan Parmar, John Quackenbush, Lawrence H Schwartz, and Hugo JWL Aerts. Artificial intelligence in radiology. *Nature Reviews Cancer*, 18(8):500–510, 2018.

[8] Xiaoxuan Liu, Livia Faes, Aditya U Kale, Siegfried K Wagner, Dun Jack Fu, Alice Bruynseels, Thushika Mahendiran, Gabriella Moraes, Mohith

Shamdas, Christoph Kern, et al. A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis. *The lancet digital health*, 1(6):e271–e297, 2019.

[9] Ayelet Akselrod-Ballin, Michal Chorev, Yoel Shoshan, Adam Spiro, Alon Hazan, Roie Melamed, Ella Barkan, Esma Herzel, Shaked Naor, Ehud Karavani, et al. Predicting breast cancer by applying deep learning to linked health records and mammograms. *Radiology*, 292(2):331–342, 2019.

[10] Jeremy Irvin, Pranav Rajpurkar, Michael Ko, Yifan Yu, Silviana Ciurea-Ilcus, Chris Chute, Henrik Marklund, Behzad Haghgoo, Robyn Ball, Katie Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[11] Diego Ardila, Atilla P Kiraly, Sujeeth Bharadwaj, Bokyung Choi, Joshua J Reicher, Lily Peng, Daniel Tse, Mozziyar Etemadi, Wenxing Ye, Greg Corrado, et al. End-to-end lung cancer screening with three-dimensional deep learning on low-dose chest computed tomography. *Nature medicine*, 25(6):954–961, 2019.

[12] Di Dong, Zhenchao Tang, Shuo Wang, Hui Hui, Lixin Gong, Yao Lu, Zhong Xue, Hongen Liao, Fang Chen, Fan Yang, et al. The role of imaging in the detection and management of covid-19: a review. *IEEE reviews in biomedical engineering*, 2020.

[13] Jannis Born, David Beymer, Deepta Rajan, Adam Coy, Vandana V Mukherjee, Matteo Manica, Prasanth Prasanna, Deddeh Ballah, Pallav L Shah, Emmanouil Karteris, et al. On the role of artificial intelligence in medical imaging of covid-19. *medRxiv*, 2020.

[14] Feng Shi, Jun Wang, Jun Shi, Ziyan Wu, Qian Wang, Zhenyu Tang, Kelei He, Yinghuan Shi, and Dinggang Shen. Review of artificial intelligence techniques in imaging data acquisition, segmentation and diagnosis for covid-19. *IEEE reviews in biomedical engineering*, 2020.

[15] Anwaar Ulhaq, Asim Khan, Douglas Gomes, and Manoranjan Pau. Computer vision for covid-19 control: A survey. *arXiv preprint arXiv:2004.09420*, 2020.

[16] Damiano Caruso, Marta Zerunian, Michela Polici, Francesco Pucciarelli, Tiziano Polidori, Carlotta Rucci, Gisella Guido, Benedetta Bracci, Chiara de Dominicis, and Andrea Laghi. Chest ct features of covid-19 in rome, italy. *Radiology*, page 201237, 2020.

[17] Saleh Albelwi and Ausif Mahmood. A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6):242, 2017.

[18] Ali Narin, Ceren Kaya, and Ziynet Pamuk. Automatic detection of coronavirus disease (covid-19) using x-ray images and deep convolutional neural networks, 2020.

[19] Ioannis D. Apostolopoulos and Tzani Bessiana. Covid-19: Automatic detection from x-ray images utilizing transfer learning with convolutional neural networks, 2020.

[20] Jun Chen, Lianlian Wu, Jun Zhang, Liang Zhang, Dexin Gong, Yilin Zhao, Shan Hu, Yonggui Wang, Xiao Hu, Biqing Zheng, Kuo Zhang, Huiling Wu, Zehua Dong, Youming Xu, Yijie Zhu, Xi Chen, Lilei Yu, and Honggang Yu. Deep learning-based model for detecting 2019 novel coronavirus pneumonia on high-resolution computed tomography: a prospective study. *medRxiv*, 2020.

[21] Charmaine Butt, Jagpal Gill, David Chun, and Benson A Babu. Deep learning system to screen coronavirus disease 2019 pneumonia. *Applied Intelligence*, page 1, 2020.

[22] Danilo Buonsenso, Davide Pata, and Antonio Chiaretti. Covid-19 outbreak: less stethoscope, more ultrasound. *The Lancet Respiratory Medicine*, 8(5):e27, 2020.

[23] Jannis Born, Gabriel Brändle, Manuel Cossio, Marion Disdier, Julie Goulet, Jérémie Roulin, and Nina Wiedemann. Pocovid-net: Automatic

detection of covid-19 from a new lung ultrasound imaging dataset (pocus). *arXiv preprint arXiv:2004.12084*, 2020.

[24] Jannis Born, Nina Wiedemann, Gabriel Brändle, Charlotte Buhre, Bastian Rieck, and Karsten Borgwardt. Accelerating covid-19 differential diagnosis with explainable ultrasound image analysis. *arXiv preprint arXiv:2009.06116*, 2020.

[25] Subhankar Roy, Willi Menapace, Sebastiaan Oei, Ben Luijten, Enrico Fini, Cristiano Saltori, Iris Huijben, Nishith Chennakeshava, Federico Mento, Alessandro Sentelli, et al. Deep learning for classification and localization of covid-19 markers in point-of-care lung ultrasound. *IEEE Transactions on Medical Imaging*, 2020.

[26] Roman Kalkreuth and Paul Kaufmann. Covid-19: A survey on public medical imaging data resources, 2020.

[27] Joseph Paul Cohen, Paul Morrison, and Lan Dao. Covid-19 image data collection. *arXiv 2003.11597*, 2020.

[28] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

Reinforcement Learning Approaches for the Swing-Up and Stabilization of the Cart Pole

Antonius Hohenhövel[1], Steffen Borchers-Tigasson[1]

[1] Hochschule für Technik und Wirtschaft Berlin
Wilhelminenhofstraße 75A, 12459 Berlin
E-Mail: steffen.borchers@htw-berlin.de

1 Introduction

In many applications, from architecture to robotics, reinforcement learning approaches for stabilizing, controlling, and optimizing systems have been considered as an alternative to classic control methods. Advantageously, reinforcement learning is a quite flexible framework for numerous control problems, allows including optimality conditions, and prior knowledge if available.

Treating the control problem in the formalism of Markov decision processes (MDPs), strong convergence results on optimality are available for reinforcement learning algorithms (see e.g.[11, 4]) thus making reinforcement learning algorithms attractive.

Even if significant progress has been achieved in the field of reinforcement learning over the past decade (see e.g. [7, 9]), solving real world control problems for nonlinear systems using reinforcement learning is still a very challenging and difficult problem. It is well known that reinforcement learning approaches are subjected to the curse of dimensionality, see e.g. [8]. Thus, for practical solutions of possibly nonlinear systems, it is required to find reasonable approximations instead of the exact solution of the underlying Hamilton-Jacobi-Bellmann equation.

In this contribution we apply and compare reinforcement learning approaches for the swing-up of the nonlinear cart-pole problem considering disturbances.

To this end, we use a classic agent-environment scheme (see e.g. [7]). The system, and in particular the swing-up process, is available as nonlinear ODE model derived from Lagrange formalism, which allows for efficient simulation. Particularly, we compare dynamic programming and temporal difference learning for the swing-up of the cart-pole system. We evaluate the influence of the most important metaparameters (e.g. learning rate), as well es the granularity of the discretization on the learning process.

We show applicability of reinforcement learning to the studied problem and discuss practical issues of considered approaches. In addition, we propose a strategy based on reward scheduling for solving advanced control problems, and discuss adaptive discretization as a tool to trade-off computational efforts and accuracy. Although we show that a simulation model speeds up the learning process significantly, the approaches studied here can be applied to a hardware only setup.

The paper is structured as follows: In Section 2, the studied system and swing-up process is described, and in Section 3, the reinforcement learning approach is motivated. In Section 4, the results are presented. The paper concludes with a summary and discussion in Section 5.

2 Studied system: Swing-up of the Cart-Pole

2.1 Model

As model, we consider the classic cart-pole inverted pendulum, derived from Langrangian mechanics. A scheme of the system is depicted in Figure 1, and is decribed by the ODE system:

$$\dot{x}_1 = x_2 \tag{1}$$

$$\dot{x}_2 = \frac{-m_2 l \sin x_3 x_4^2 + u + m_2 g \cos x_3 \sin x_3}{m_1 + m_2 - m\cos(x_3)^2} \tag{2}$$

$$\dot{x}_3 = x_4 \tag{3}$$

$$\dot{x}_4 = \frac{-m_2 l \cos x_3 \sin x_3 x_3^2 + u\cos x_3 + m_2 g \sin x_3 + m_1 g \sin x_3}{l(m_1 + m_2 - m\cos(x_3)^2)} \tag{4}$$

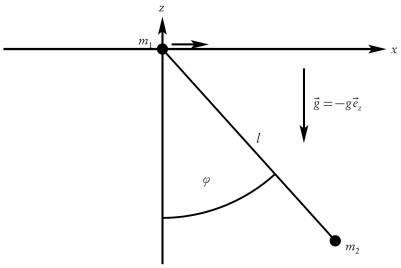


Figure 1: Scheme of the considered Cart-Pole system.

Hereby, x_1 denotes the carts position, x_2 the carts velocity, x_3 the angle φ, and x_4 the angular velocity $\dot{\varphi}$, and finally u denote the systems input in terms of a force applied to the cart in positive x-direction. The input is chosen for simplicity from the binary input set $u \in \mathbb{U} : \{-1, 1\}$.

2.2 System Simulation

In order to apply the reinforcement learning algorithm successfully, we need to simulate the system repeatedly. More precisely, we need to compute the future systems states (fixed time horizon $h = 0.2s$) given varying initial conditions and inputs. This is achieved here using the Python-based OpenAI gym engine [3], considering the inverted cart-pole as described in [1], see Figure 1. The OpenAI gym provides an efficient numerical solution as well as a visualisation of the systems dynamics.

2.3 Goal/Reward

Classically, in terms of optimal control, for the described non-linear cart-pole system we aim to find an (optimal) input sequence so as to transfer the system from its the lower, stable equilibrium point ($s^* := \{x_1 = 0, x_2 = 0, x_3 = 0, x_4 = 0\}$) toward the upper equilibrium point (i.e. $x_1 = \text{const.},\ x_2 = 0,\ x_3 = 180^\circ,\ x_4 = 0$). Hereby, the horizontal position is constrained throughout the process by $-l \leq x_1 \leq l$.

For the reinforcement framework, it is crucial to define a meaningful reward function, which is granted after each episode (try-out). Generally, the reward will depend on achievements made in an episode (and thus on the decisions made for the input), and is subsequently propagated backwards so as to learn from the current episode.

Here, a successful run (swing-up) is simplified to the condition $175^\circ \leq x_3 \leq 185^\circ$. Note that it is straightforward to add further (and more rigorous) conditions defining success, however for simplicity of presentation this is omitted here. Only in case of success as defined above, the reward $R = 0$ is granted, otherwise $R = -1$.

3 Reinforcement Learning Framework

The reinforcement learning approach considered here is treated as episodic agent-environment interaction scheme as described by [7]. As key algorithms we consider the temporal difference learning, particularly TD(0) 1-step-Q-learning, see e.g. [9]. Figure 2 depicts how the agent (i.e. the AI choosing the current input) interacts with its environment (cart-pole system).

3.1 initialization()

At the beginning of each learning episode we initialize the algorithm by defining the metaparameters (learning rate α, exploitation rate ε, discount factor λ) as well as the simulation constraints (maximum time steps, epsiode number, discretization, ...). Note that we consider a finite uniform state space discretization rather the infinite continuous state space. Particularly, we consider 4800 discrete states.

3.2 run()

With the function run() we start the interaction of the agent with the environment. First, the environment is reset (reset()), and the initial state s^* together

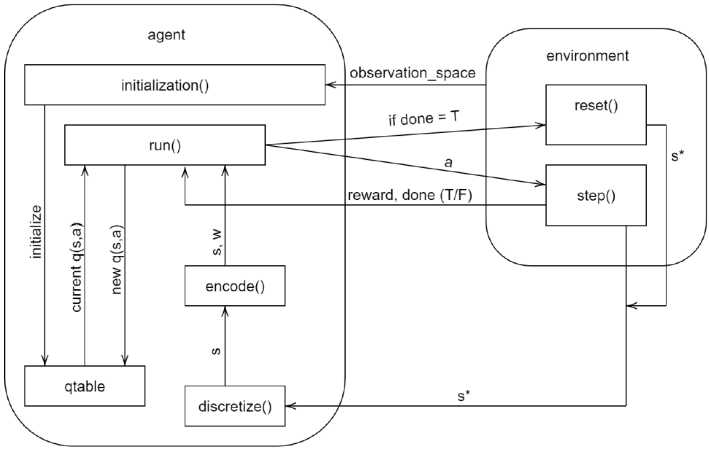


Figure 2: Reinforcement structure scheme.

with an initial input chosen by the agent is propagated to the environment. A simulation step is performed (step()), whereas the time-discretization of (1)–(4) with fixed step size $h = 0.2s$ is considered (discretize()). The function encode() is optional and used in a modified algorithm not further considered here. Next, the current state and the current input is updated, another step is performed and a successor state is reached. The process is repeated until either success, a constraint violation, or the maximum number of time steps is reached.

Of particular interest is the final or terminal state $\mathbf{s}_T$ at the end of an episode, which decides upon the reward granted. Only in case of success a positive reward is granted, in all other cases (e.g. violation of constraints, time's up,...) the reward is negative. Depending on the RL algorithm, at the end of each episode, the reward is back-propagated (new q(s,a)) so as to update the Q-table. Hence it is required to store the visited states within an episode. The here considered TD(0) algorithm however allows to update the Q-table at each step considering bootstrapping.

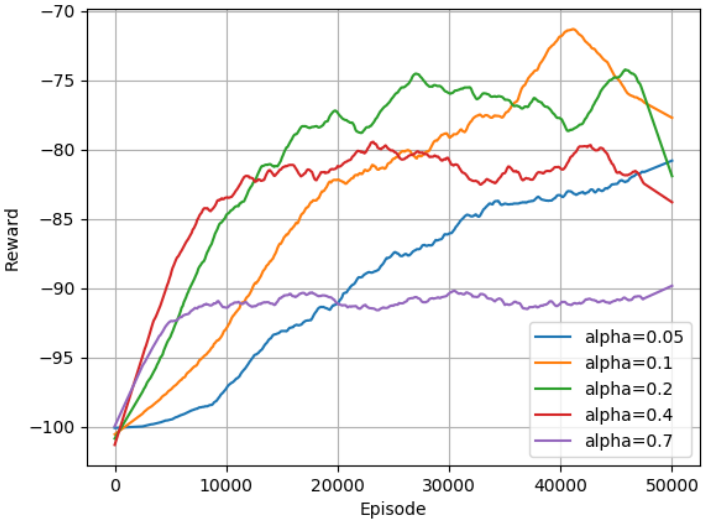


Figure 3: Influence of the learning rate α. Exploration rate $\varepsilon = 0.1$ is kept constant.

4 Results

We successfully applied the TD(0) algorithm to the cart-pole. To evaluate the influence of the learning rate, we evaluated the RL-approach considering different learning rates and compared the results see Fig. 3. Figure 3 shows the influence of the learning rate α onto the average reward given obtained by averaging the reward given five independent runs.

4.1 Influence of the learning rate

As general conclusion, high learning rate is advantageous only at the beginning, in the long run however a lower learning rate leads to better performance. A plausible cause here is the stochastic nature of the process due to the state space discretization. This is because the same action from a certain state may cause different successor states. Starting the process with high learning rate,

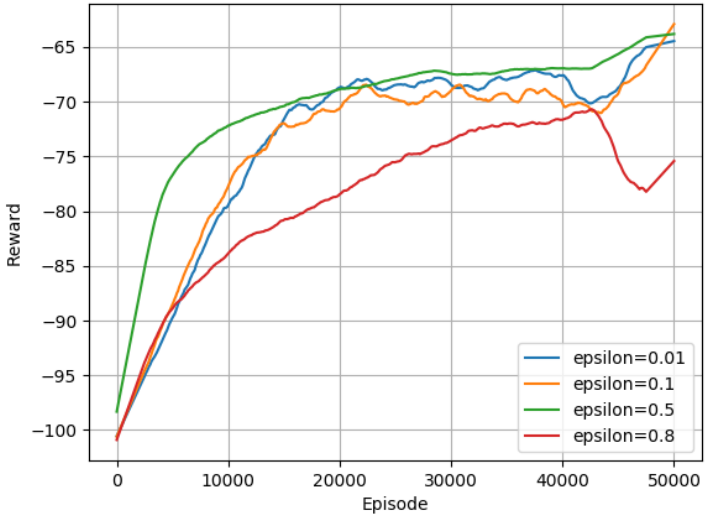


Figure 4: Influence of the exploration rate ε. Learning rate $\alpha = 0.2$ is kept constant.

the Q-table is quickly updated, however gets stuck in local optima. Given a smaller learning rate, the Q-Value of that action at that state changes more slowly thus averaging all possible transitions. This in turn leads, though more slowly, to a more complete overall picture of the swing-up process.

This in turn motivates a modification of the RL algorithm by realizing the learning rate α as a linearly decaying function over time. In the remainder, α decays from 0.5 to 0.01 in the first 25000 episodes.

4.2 Exploration vs exploitation

It is well known that learning following RL strategies depend crucially on the balancing of exploration of the state action space as well as exploiting the results already obtained. Fig. 4 depicts the influence of the exploration rate ε on the learning process. In general terms, while high exploration rates are theoretically advantageous since each state-action pair is visited often, and

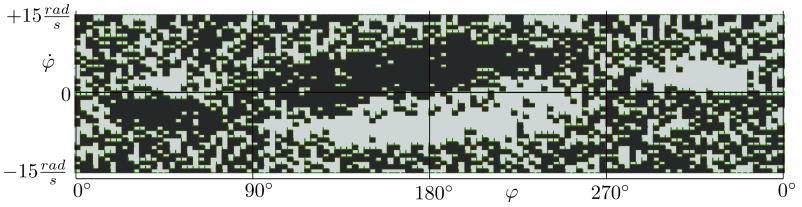


Figure 5: Visualisation of the Q-Table. The carts horizontal position and velocity are omitted. Dark grey corresponds to the action $u = -1$, light grey to $u = +1$.

thus convergence to the optimal policy is guaranteed, in practice however high exploration rates are numerically prohibitive.

Since learning not only depends on rigorous exploration of the state-action space, rather it is significantly boosted when a first success is achieved. So, exploitation is relevant so as to focus on most promising trajectories leading to an early success which can be exploited thereafter. This relation is in general very difficult to translate in reasonable ε parameter values. In our case, a exploration rate of 0.5 is well balanced.

4.3 Visualisation of the policy

The resulting Q-table with the greedy action is shown in Fig. 5. Clearly visible are the expected symmetrical properties. Each pixel corresponds to a discrete state. Omitted are the horizontal position and velocity of the cart for simplicity.

5 Discussion

The here presented RL algorithm is easily applied to four dimensional non-linear dynamic system. We have shown applicability of the algorithm for swing-up process of this cart-pole. The approach can be easily adapted so as to treat a variety of stabilisation goals such as keeping the cart-pole at the upper, instable stationary point. The presented temporal difference algorithm can be applied for this purpose directly, however a functional approach (i.e. learning

the control parameters of a state feedback using reinforcement learning) may be computationally more advantageous.

The presented approach is directly applicable to the hardware only case. However, in this naive setting with not prior knowledge, a rigorous exploration of the state space is mandatory. Thus, a simulation model is an advantage to speed up the training significantly. Prior knowledge can be encoded e.g. within the reward function or by adding additional constraints.

Regarding the computational tractability, discretizing the state space has to be considered carefully. While the available computational power limits the overall number of discrete states, it is advantageous to distribute the discrete states by taking the studied process into consideration. It is advantageous to constrain the state space to an meaningful observable area, so as to subsume large angular and horizontal velocities into a few (forbidden) states and thus reducing complexity.

Furthermore, it is possible to consider an adaptive dicretization scheme, starting with only few discrete states and subsequently splitting those states given an appropriate measure based on the transition probabilities as obtained from the reinforcement learning algorithm. Further research will apply the RL framework to different stabilization problems occurring in production, e.g. adaptive picking of objects with a robot arm, or placing objects in optimal fashion.

References

[1] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[2] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, UK, 2004.

[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAai Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[4] Peter Dayan and Terrence J Sejnowski. Td (λ) converges with probability 1. *Machine Learning*, 14(3):295–301, 1994.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Santanu Pattanayak, Pattanayak, and Suresh John. *Pro Deep Learning with TensorFlow*. Springer, 2017.

[7] Stuart Russell, Peter Norvig. Artificial intelligence: a modern approach, 2002

[8] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Reinforcement learning with soft state aggregation. In *Advances in neural information processing systems*, pages 361–368, 1995.

[9] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 2017.

[10] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[11] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. *Machine learning*, 16(3):185–202, 1994.

MATLAB-Toolbox zum Offline-Testsignalentwurf mit prozessmodellfreien und prozessmodellbasierten Methoden

Matthias Himmelsbach (geb. Gringard), Andreas Kroll

FG Mess- und Regelungstechnik, FB Maschinenbau, Universität Kassel
Mönchebergstr. 7, 34125 Kassel, DE
E-Mail: {matthias.himmelsbach,andreas.kroll}@mrt.uni-kassel.de

Kurzfassung

In diesem Kurzbeitrag wird eine MATLAB-Toolbox zum Testsignalentwurf für Standardtestsignale für die lineare und nichtlineare Systemidentifikation vorgestellt. Für die Identifikation dynamischer Systeme sind nicht nur die Parameterschätzverfahren und die ausgewählten Modellansätze relevant für die Qualität der geschätzten Modelle, sondern auch die Eigenschaften der zur Identifikation verwendeten Daten, weshalb zielgerichtete Methoden zur Zielsystemanregung von Interesse sind. Im Beitrag werden die Funktionen beschrieben und an Beispielen demonstriert.

1 Einführung

Neben Schätzverfahren und Modellansätzen spielen auch die Eigenschaften der zur Identifikation verwendeten Daten eine große Rolle. Eine wichtige Unterscheidung beim Testsignalentwurf ist in prozessmodellfreie und prozessmodellbasierte Ansätze. Prozessmodellfreie Verfahren verwenden Signalkennwerte und Vorwissen, um Testsignale zu entwerfen. Sie setzen wenig Vorwissen über das Zielsystem oder geeignete Modellansätze voraus. Sie dienen eben-

falls als Vorstufe für modellbasierte Entwürfe, die strukturelles Modellwissen und ggf. eine Initialparametrierung voraussetzen. Hierbei spielen insbesondere die Homgenisierungsverfahren eine große Rolle.

Prozessmodellbasierte Ansätze verwenden wenigstens Informationen über die Modellstruktur und im nichtlinearen Fall auch die zu schätzenden Modellparameter selbst. Die Entwurfsmethoden sind für lokal-affine Takagi-Sugeno-Modelle implementiert, können aber prinzipiell für weitere Modellklassen verwendet werden. Die Wahl einer Modellklasse lässt Spezifikationen der Verfahrens zu. Zwei Typen modellbasierter Verfahren sind in der Toolbox implementiert: Reduktion der Parameterunsicherheit basierend auf der Fisher-Informationsmatrix (FIM) und die Ausgangshomogenisierung mittels Vorsteuerung. Für alle Entwürfe muss eine Abtastzeit T_A vorgegeben werden, welche sich aus der Anwendung ergibt. Abbruchkriterien für die einzelnen Verfahren sind in den entsprechenden Veröffentlichungen beschrieben. Die Toolbox liefert zeitdiskrete Signale für die Identifikation von Single-Input-Single-Output-(SISO-)Systemen. Sie wird frei zugänglich auf der Homepage (http://www.uni-kassel.de/fb15/mrt) des Fachgebietes Mess- und Regelungstechnik (MRT) bereitgestellt werden.

Die Toolbox wurde in der MATLAB Version: 9.5.0.944444 (R2018b) implementiert und getestet. Die Partikelschwarmoptimierung stammt aus der Global Optimization Toolbox Version: 4.0 (R2018b). Die Toolbox wird frei zum Download zur Verfügung gestellt.

2 Prozessmodellfreie Entwürfe

Für den prozessmodellfreien Entwurf sind Standardtestsignale implementiert: Multisinus-, Multistufen- und Chirp-Signale. Auf Rauschsignale wurde verzichtet, da bei der Parametrierung kaum Vorwissen eingebracht werden kann, wie dem Frequenzraster oder sinnvollen Stufenlängen. Diese typischen Anregungssignale wurden auch in [1] vergleichend gegenübergestellt. Dieser Teil der Toolbox basiert darauf, dass ein *cell-array* mit den notwendigen Rahmenbedingungen an den Testsignalgenerator übergeben wird. Diese Bedingungen

Tabelle 1: Einstellparameter/Entwurfsoptionen Multisinussignal

Einstellung	Bedeutung/Optionen
$f_{\min}, f_{\max}$	Frequenzband
$T_{\max}$	Maximale Experimentdauer
Frequenzauswahl	Alle, Primzahlen, Jede n-te
Phasenauswahl	Zufallsphasen, Schroeder-Phasen
Homogenisierung	mit hom/normal

werden in den folgenden Unterabschnitten vorgestellt. Die prozessmodellfreien Entwurfsverfahren zielen auf die Einstellung der Signalwerteverteilung und die Kompaktheit der Signale ab. Daher können alle prozessmodellfrei entworfenen Signale zum Abschluss des Entwurfs auf den zulässigen Wertebereich skaliert werden

2.1 Multisinussignale

Bei Multisinussignalen sind die Parameter zur Einstellung der Signaleigenschaften die Amplituden und Phasen der im Signal enthaltenen Frequenzanteile, welche zuvor ausgewählt werden. Bevor Verfahren verwendet werden können, um über die Einstellparameter gezielt Signaleigenschaften zu erreichen, müssen noch Rahmenbedingungen (Tabelle 1) festgelegt werden, welche die Einstellparameter beeinflussen. Bis auf die Option zur Homogenisierung dient alles der Erstellung eines Initialsignals. Hierbei ist zu berücksichtigen, dass das Frequenzband mittels Vorwissen ausgewählt werden sollte. Häufig ist die Abtastzeit aus der vorgesehenen Anwendung vorgegeben. Durch die Wahl der maximalen Experimentdauer wird die Grundfrequenz und damit das Frequenzraster festgelegt. Bei der Auswahl der aktiven Frequenzen sollte darauf geachtet werden, dass in einem Frequenzband nur eine begrenzte Anzahl aktiver Frequenzen vorhanden ist, da sonst die Signalenergie je Frequenzanteil reduziert wird. Vertiefende Erläuterungen finden sich in [2].

Tabelle 2: Einstellparameter/Entwurfsoptionen Multistufensignal

Einstellung	Bedeutung/Optionen
n_{step}	Stufenanzahl
$l_{\min}, l_{\max}$	Wertebereich der Stufenlängen
Homogenisierung	mit hom/normal

Das Homogenisierungsverfahren passt die Phasen so an, dass der Wertebereich des Eingangssignals möglichst gleichförmig abgedeckt wird. Für eine genaue Beschreibung des Homogenisierungsverfahrens wird auf [3] verwiesen.

2.2 Multistufensignale

Bei Multistufensignalen sind die direkten Einstellparameter die Höhen und Längen der einzelnen Stufen, nachdem die Rahmenbedingungen aus Tabelle 2 festgelegt wurden. Die exakte Signallänge ist kein Entwurfsparameter. Sie wird indirekt über die Anzahl der Stufen und die Länge der Stufen eingestellt. Die Länge der Stufen sollte so gewählt werden, dass die langsamste Zeitkonstante, die zu finden ist, durch eine Stufe abbildbar ist. Erfahrungsgemäß ist bei einer hohen Stufenanzahl die maximale Experimentdauer in etwa $T_{\max} = n_{\text{step}} \cdot \frac{1}{2}(l_{\min} + l_{\max})$. Die absoluten Stufenhöhen werden an dieser Stelle nicht explizit eingestellt, sondern ergeben sich durch die Skalierung des Testsignals auf den zulässigen Signalwertebreich, welche im Anschluss durchgeführt werden kann. Die Entwurfsverfahren zielen auf die Signalverteilung selbst ab. Für die Grundlagen der Erstellung von Multistufensignalen aus Pseudozufallsbinärsequenzen wird auf [4] verwiesen. Für die nicht-homogenisierte Version werden gleichverteilte Zufallszahlen für die Stufenhöhen und -längen verwendet, da sich keine Vorteile bei der Verwendung von bspw. Sobol-Sequenzen gezeigt haben. Beim homogenisierten Multistufensignal wird eine Scheitelfaktoroptimierung eingesetzt. Dabei ist der Scheitelfaktor des Signals $\mathbf{u}$ definiert als:

$$c_{\text{r}} = \frac{\max|\mathbf{u}|}{\text{rmse}(\mathbf{u})}, \quad \text{rmse}(\mathbf{u}) = \sqrt{\frac{\mathbf{u}^{\top}\mathbf{u}}{\text{length}(\mathbf{u})}} \tag{1}$$

Tabelle 3: Einstellparameter/Entwurfsoptionen Chirp-Signal

Einstellung	Bedeutung/Optionen
$T_{\max}$	Maximale Experimentdauer
Homogenisierung	mit hom/normal

Da ohne weitere Nebenbedingungen ein Binärsignal entsteht, wurden Verteilungsmaße über Nebenbedingungen in das Optimierungsproblem integriert. Für eine detailliertere Beschreibung wird auf [3] verwiesen.

2.3 Chirp-Signale

Chirp-Signale sind eine schnelle Möglichkeit ein Signal zu entwerfen, welches einen Frequenzbereich abdeckt und dabei annähernd eine Gleichverteilung der Signalwerte aufweist. Tabelle 3 zeigt die Einstellmöglichkeiten dieses Signaltyps. Die Eigenschaften eines solchen Signals werden in [3] beschrieben.

2.4 Beispiel

Exemplarisch wird der Entwurf eines Multisinustestsignals vorgestellt. Zunächst muss ein cell-array mit den Rahmenbedingungen erstellt werden:
`info={'msine',[0.1 2], 50, 0.01, 'primes', 'random', 'normal'}`.
So wird ein zeitdiskretes Multisinussingal erzeugt, welches den Frequenzbereich von $0{,}1-2\,\mathrm{Hz}$ abdeckt, eine maximale Experimentdauer von $T_{\max}=50\,\mathrm{s}$ bei einer Abtastzeit von $T_{\mathrm{A}}=0{,}01\,\mathrm{s}$. Es werden nur Primzahlenvielfache der Grundfrequenz verwendet. Die Phasenfestlegung ist zufällig und es wird keine anschließende Homogenisierung durchgeführt. Durch den Aufruf der Funktion für den prozessmodellfreien Testsignalentwurf entsteht das Signal mit den folgenden Angaben gemäß Bild 1. Letzteres zeigt die Ausgabe der integrierten Anzeigefunktion, welche neben dem Scheitelfaktor auch einen Kennwert für

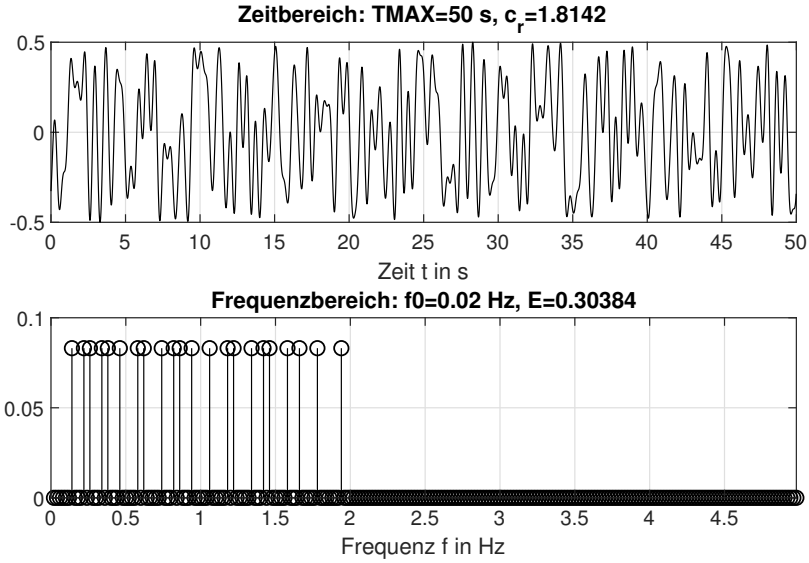


Bild 1: Toolboxanzeige für exemplarischen prozessmodellfreien Multisinusteststsignalentwurf

die Signalenergie

$$E = \mathbf{U}^{\top}\mathbf{U}^{*} \tag{2}$$

angibt, welcher im Frequenzbereich mit der diskreten Fouriertransformierten $\mathbf{U} = \mathrm{DFT}(\mathbf{u})$ bestimmt wird. Hierbei gibt * die komplexe Konjugation an.

3 Prozessmodellbasierte Entwürfe

Prozessmodellbasierte Entwürfe benötigen vollständige Initialmodelle, bestehend aus Strukturinformationen und Modellparametern, um durchgeführt werden zu können. Sowohl der FIM-basierte Entwurf in Abschnitt 3.1 und die Ausgangshomogenisierung in Abschnitt 3.2 sind für lokal-affine Takagi-Sugeno-Modelle implementiert, welche Zugehörigkeitsfunktionen aus dem Fuzzy-c-means-(FCM-)Clusteralgorithmus mit euklidischer Abstandsnorm verwenden.

Tabelle 4: Übergabe eines TS-Modells an die Toolbox

Übergabegröße	Bedeutung
	Strukturinformationen
c	Anzahl der Teilmodelle
n, m	Verzögerungen der Ein- und Ausgangsgröße
$\mathbf{T}_z$	Abbildung des Regressionsvektors auf die Schedulingvariable
ν	Unschärfeparameter
	Modellparameter
$\boldsymbol{\Theta}^\top$	Gesamtparametervektor
$\boldsymbol{\Theta}_{\mathrm{LM}}$	lokale Modellparameter
$\boldsymbol{\Theta}_{\mathrm{MF}}$	Partitionsparameter (Prototypen $\mathbf{v}_i$)

Für die Verwendung dieses Teils der Toolbox müssen Informationen über das TS-Modell übergeben werden. Zum einen wird ein `MATLAB`-*struct*, welches die Strukturinformationen (Anzahl der Teilmodelle c, maximale Lags der Ein- und Ausgänge n und m, Unschärfeparameter ν und ggf. eine Funktionsvorschrift zur Berechnung der Schedulingvariablen aus dem Regressionsvektor, welche hier als lineare Abbildung über die Matrix $\mathbf{T}_z$ implementiert ist) enthält und zum anderen der Gesamtmodellparametervektor übergeben. Da zur Bestimmung der FIM die analytischen Ableitungen der FBF nach den Modellparametern notwendig sind, muss dies bei einem Austausch der FBF berücksichtigt werden. Tabelle 4 zeigt die Übergabegrößen an den prozessmodellbasierten Teil der Toolbox.

3.1 Testsignalentwurf zur Reduktion der Unsicherheit

Der Testsignalentwurf zur Unsicherheitsreduktion basiert auf der Minimierung skalarer Maße auf der Fisher-Informationsmatrix (FIM), was z.B. in [5] schon beschrieben wurde. Unter der Annahme normalverteilten Rauschens am Aus-

gang, dessen Realisierungen unabhängig voneinander sind, kann die FIM wie folgt angegeben werden:

$$\mathbf{I} = \frac{1}{\sigma^2} \sum_{k=1}^{N} \left[\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right] \left[\frac{\partial \hat{y}(k)}{\partial \boldsymbol{\Theta}} \right]^{\top} \tag{3}$$

Hierbei ist σ^2 die Varianz des Rauschens, N die Anzahl der Beobachtungen, $\hat{y}(k)$ der Modellausgabewert im k-ten Zeitschritt und $\boldsymbol{\Theta}$ sind die Prozessmodellparameter. In der Literatur wird die FIM häufig in Abhängigkeit der Modellparameter angegeben, dies ist jedoch für die Realisierung für ein dynamisches Problem irreführend, denn die FIM hängt neben dem Testsignal auch von der Modellausgabe selbst ab, da vergangene Werte der Modellausgabe Teil der Regressionsvariablen sind. Dies macht die Verwendung eines internen Simulationsmodells nötig, welches in jedem Iterationsschritt ausgewertet werden kann. In der Toolbox wird das folgende Optimierungsproblem gelöst:

$$\boldsymbol{\beta}_{\text{opt}} = \operatorname{argmin}_{\boldsymbol{\beta}} J\left(\mathbf{I}\left(\hat{\mathbf{y}}\left(\boldsymbol{\Theta}, \mathbf{u}\left(\boldsymbol{\beta}\right)\right), \mathbf{u}\left(\boldsymbol{\beta}\right), \boldsymbol{\Theta}\right)\right) \tag{4}$$

Der Fettdruck von $\mathbf{u}$ und $\mathbf{y}$ bezeichnet hier, dass es sich um das gesamte Signal handelt. $\boldsymbol{\beta}$ kennzeichnet hier die Signalmodellparameter, da das Problem für eine individuelle Optimierung aller Einzelwerte von $u(k)$ nicht praktikabel ist. Die Verwendung von Signalmodellen beim optimalen Testsignalentwurf hat ebenfalls implizite Nebenbedingungen zur Folge. Die FIM-basierte Optimierung von Signalmodellparametern wird in [6, 7] für die verwendeten Takagi-Sugeno-Modelle beschrieben. Die Lösung des Problems wird mit dem in `MATLAB` implementierten Partikelschwarmalgorithmus [9] gelöst, welcher über die entsprechenden Übergabeparameter angepasst werden kann. Tabelle 5 zeigt die Entwurfsoptionen beim FIM-basierten Testsignalentwurf. Die Kombinationen beim Testsignaltyp meint eine Addition eines Multisinussignals mit einem Multistufensignal. Da sich die Modellparameter der TS-Modelle in lokale Modellparameter und Partitionsparameter aufteilen lassen, sind in der Toolbox auch getrennte Optimierungen möglich, um Testsignale zu entwerfen, die eine präzise Schätzung der entsprechenden Parametergruppe ermöglicht. Bild 2 zeigt exemplarisch ein FIM-basiert entworfenes Stufensignal.

Tabelle 5: Entwurfsoptionen FIM-basierter Testsignalentwurf

Einstellung	Optionen
Testsignaltyp	Multisinus, Multistufen, Überlagerung
Zielparameter	Alle, lokale Modellparameter, Partitionsparameter
FIM-Maß	Determinante, Spur, skalierte Spur, Maximieren des kleinsten Eigenwerts, Empfindlichkeitssumme

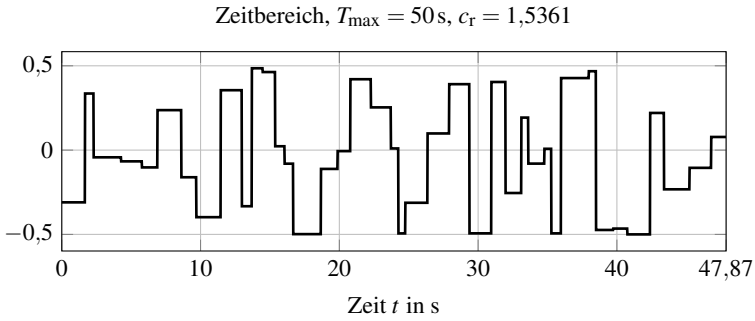


Bild 2: Multistufensignal durch FIM-basierten Testsignalentwurf

3.2 Ausgangshomogenisierung durch Vorsteuerung

Bei der Identifikation nichtlinearer dynamischer Systeme ist davon auszugehen, dass das nichtlineare Verhalten hauptsächlich durch die vergangenen Werte der Ausgangsgröße bestimmt wird. Bei statischen Systemen führen raumfüllende Ansätze zu einer besseren Rasterung bei einem möglichst sparsamen Umgang mit verfügbaren Messpunkten der Ausgangsgröße. Wird eine Homogenisierung der Eingangsgröße durchgeführt, wird außer Acht gelassen, dass dies nicht zwangsläufig zu einer Homogenisierung der Ausgangsgröße und damit zu einer besseren Rasterung des nichtlinearen Verhaltens führt. Aus diesem Grund enthält die Toolbox Methoden, welche eine flachheitsbasierte Vorsteuerung basierend auf Initialmodellen verwendet. Dazu wird zunächst ein

Ausgangssignal erzeugt, welches den Schedulingraum gleichmäßig abdeckt. Wird nur die Ausgangsgröße homogenisiert, können die prozessmodellfreien Testsignalentwurfsfunktionen verwendet werden. Wird der gesamte Schedulingraum betrachtet, müssen Abdeckungsmaße individuell für diesen angepasst werden, um eine geeignete Referenzausgangsgröße zu erhalten, welche vom System durch die entworfene Vorsteuerung erreicht werden soll. Ein n_z-dimensionalen Schedulingraum wird dabei in Hyperquader zerlegt, in welchem die Datenpunkte gezählt werden. Wie sich die Verteilung im Schedulingraum auf die Ausgangsgröße auswirkt, hängt von der individuellen Wahl der Schedulingvariable ab. Die Ausgangsgröße wird dann iterativ mit der `MATLAB`-PSO ermittelt, was aufgrund der geringen Komplexität keine hohen Rechenkosten verursacht. Für eine geeignete Wahl der Referenzausgangsgröße müssen Rahmenbedingungen wie der zulässige Wertebereich und das relevante Frequenzband des Systems ermittelt werden.

Das Verfahren basiert darauf, bei einem gegebenen Referenzsignal mit den gewünschten Eigenschaften mittels des TS-Initialmodells eine Vorsteuerung so zu entwerfen, dass das zu identifizierende System die Referenzausgangsgröße bestmöglich liefert. Bei der Umsetzung wird die lokale Modellstruktur der TS-Modelle ausgenutzt, indem lokale Steuerfunktionen $u_j(k)$ zu den lokalen Teilmodellen entworfen werden. Die $u_j(k)$ werden dann mit den Fuzzy-Basisfunktio-nen $\phi_j(k)$ des TS-Initialmodells zwecks simulativer Bewertung und Optimierung überlagert.

$$u(k) = \sum_{j=1}^{c} \phi_j(k) \cdot u_j(k) \tag{5}$$

Bei einem steuerbaren System kann der Entwurf mit einem virtuellen flachen Ausgang stets durchgeführt werden. Für weitere Informationen zu der Realisierung wird auf [8] verwiesen. Bild 3 zeigt die Auswirkung des Entwurfs. Im oberen Plot ist das entstandene Testsignal zu sehen. Im unteren Plot zum einen das Referenzsignal (gepunktet), die Antwort des Fuzzy-Initialmodells (schwarz), welches für den Vorsteuerungsentwurf verwendet wurde, sowie die tatsächliche Systemantwort (grau). Das Referenzsignal wurde so entworfen, dass der zugehörige Schedulingraum möglichst gleichmäßig abgedeckt ist. Auch wenn die Vorsteuerung nicht exakt ist, was durch die Überlagerung der

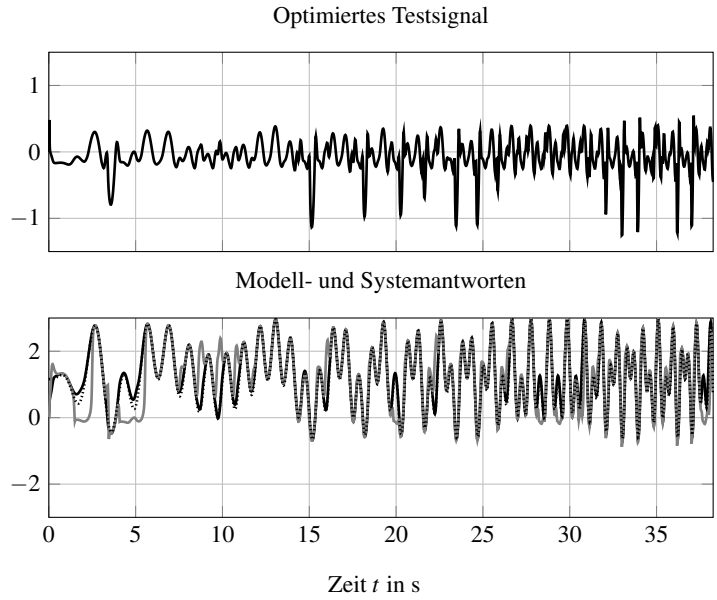


Bild 3: Testsignal und Ausgangsgrößen, Referenzsignal (gepunktet), Antwort Fuzzy-Modell (schwarz), Systemantwort (grau)

lokalen Steuerfunktionen bedingt ist, sind sich die Signale dennoch sehr ähnlich, so dass am System Daten erhoben werden konnten, die eine gleichmäßige Abdeckung des Schedulingraums aufweisen.

4 Zusammenfassung und Ausblick

In diesem Beitrag wurden kurz die Funktionen einer MATLAB-Testsignalentwurfstoolbox vorgestellt. Hierbei ist der prozessmodellfreie Teil für jeden Modelltyp verwendbar und nicht auf Takagi-Sugeno-Modelle beschränkt.

Die prozessmodellbasierten Entwürfe sind für TS-Modelle implementiert. Die Toolbox ist frei verfügbar. Sie enthält für alle Kategorien Demonstrationsbeispiele, sowie eine detaillierte Anleitung ihrer einzelnen Elemente. Bei der Ausgangshomogenisierung kann der Effekt direkt am Signal abgelesen werden.

Die Wirkung der Methoden wurde an verschiedenen Fallbeispielen untersucht und in [3, 6, 8] veröffentlicht.

Danksagung

Die Forschungsarbeit wurde durch die Deutsche Forschungsgemeinschaft (DFG) unterstützt, Projektnummer KR 3795/7-1.

Literatur

[1] T. O. Heinz und O. Nelles, „Vergleich von Anregungssignalen für nichtlineare Identifikationsaufgaben“, In: *Proc. 26. Workshop Computational Intelligence*, S. 139–158, Dortmund, Deutschland, 2016.

[2] R. Pintelon und J. Schoukens, „System identification: a frequency domain approach“, 2. Ausgabe, Wiley, 2012.

[3] M. Gringard und A. Kroll, „Zur Homogenisierung von Testsignalen für die nichtlineare Systemidentifikation“, In: *at - Automatisierungstechnik* Band 67, Nr. 10, S. 820-832, 2019.

[4] K. Godfrey. „Perturbation signals for system identification“, Prentice Hall, 1993.

[5] A. Atkinson und A. Donev, „Optimum experimental designs“. Oxford: Clarendon Press, 1992.

[6] M. Gringard und A. Kroll, „Optimal experiment design for identifying dynamical Takagi-Sugeno models with minimal parameter uncertainty“, In: *Proc. Symposium on System Identification 2018*, Stockholm, Schweden, 2018.

[7] C. Hametner, M. Stadlbauer, M. Deregnaucourt, S. Jakubek und T. Winsel, „Optimal experiment design based on local model networks and multiplayer perceptron networks“. In: *Engineering Applications of Artificial Intelligence* Band 25, Nr. 1, S. 251-261, 2013.

[8] M. Gringard und A. Kroll, „On considering the output in space-filling test signal designs for the identification of dynamic Takagi-Sugeno models“, In: *Proc. IFAC World Congress 2020*, Berlin, Deutschland, 2020.

[9] Mathworks, „Optimization Toolbox User's Guide“, 2019.

Identifikation eines nichtlinearen dynamischen Mehrgrößensystems mit rekurrenten neuronalen Netzen im Vergleich zu lokal-affinen Zustandsraummodellen

Stephan Godt, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Schulstraße 10, 33330 Gütersloh
E-Mail: {stephan.godt, martin.kohlhase}@fh-bielefeld.de

1 Einführung

Die Systemidentifikation versucht anhand von Beobachtungsdaten Prozesse mit Hilfe von Modellen bestmöglich zu beschreiben. Diese Modelle ermöglichen es nicht nur das Verhalten eines komplexen Systems abzubilden, sondern dieses auch in gewissem Maße vorherzusagen. Diese Fähigkeit findet besonders in Bereichen der Automatisierungs- und Regelungstechnik ihren Nutzen, vor allem in Teilgebieten wie beispielsweise der modellbasierten Prozesssteuerung und Prozessoptimierung. Der Schlüssel bei jeder Modellierungsaufgabe ist es, eine geeignete Modellstruktur zur Abbildung des untersuchten Prozesses zu wählen. Hier finden bei komplexen Prozessen besonders nichtlineare dynamische Ansätze ihre Anwendung. Die Verfahren zur nichtlinearen dynamischen Modellbildung können allgemein in zwei Kategorien unterteilt werden. Der erste Ansatz verwendet eine externe Dynamik zur Schätzung der Modelle, wobei der aktuelle Modellausgang $\hat{y}$ auf Basis verzögerter Ein- und Ausgangsgrößen geschätzt wird. In der Systemidentifikation finden in diesem Bereich häufig Modelle mit einer NARX-Struktur (Nichtlinear autoregressiv mit exogenem Eingang (Nonlinear ARX)) ihre Anwendung. Verfahren mit einer internen Dynamik, die der zweiten Kategorie zugeordnet werden können, haben keine externe Kopplung verzögerter Ein- und Ausgänge. Hier werden die

internen Zustände innerhalb des Modells lokal zurückgegeben. Häufige Vertreter dieser Kategorie sind NLSS (*general NonLinear State-Space*)-Modelle. Auch der Einsatz von rekurrenten neuronalen Netzen in diesem Bereich ist ein viel diskutiertes Thema in der heutigen Forschung. Es wurden bereits erste Verbindungen und Gemeinsamkeiten dieser Verfahren aufgezeigt und eine gewisse Vergleichbarkeit in der Modellgenauigkeit nachgewiesen [1], [2].

In dieser Arbeit werden die Untersuchungen hinsichtlich der Vergleichbarkeit der beiden Ansätze aufgegriffen und zwei Methoden zur Systemidentifikation bezüglich ausgewählter Modellcharakteristika miteinander verglichen. Zum einen kommt ein lokal-affines Zustandsraummodell zum Einsatz, welches das Systemverhalten mit Hilfe von lokal-linearen Modellen abbildet, die auf Basis aufgenommener Prozessdaten geschätzt werden. Dies erfolgt mit dem Programmpaket DYLAMOT (DYnamic Local Affine MOdeling Toolbox). Für die lokal linearen Modelle wird dabei eine NARX-Struktur gewählt, wobei die Parameter anhand einer Ausgangsfehlerschätzung bestimmt werden. Zum anderen werden für die Systemidentifikation rekurrente neuronale Netze, genauer „Long short-term memory"-Netze (LSTM), verwendet. Diese besitzen Rückkopplungen zwischen Neuronen verschiedener Schichten, sodass Prozessinformationen über einen Zeitraum erhalten bleiben. Bei einem möglichen Einsatz der Verfahren an realen Prozessen ergeben sich verschiedene Herausforderungen. Einerseits besteht die Möglichkeit, dass Prozesse außerhalb ihres üblichen Einsatzbereichs betrieben werden, was ein gewisses Maß an Extrapolationsfähigkeit in den verwendeten Modellen voraussetzt. Andererseits wirken auf reale Prozesse häufig Störeinflüsse, sodass es sinnvoll ist, die Empfindlichkeit der eingesetzten Modelle beispielsweise gegenüber dem Einfluss von Rauschen zu untersuchen.

Nach einer kurzen Vorstellung der zwei Methoden muss in einem ersten Schritt eine ausreichende Modellgüte mit beiden Verfahren erzielt werden. Daraufhin werden in einem nächsten Schritt die verwendeten Modelle hinsichtlich ihrer Extrapolationsfähigkeit und ihrer Rauschempfindlichkeit näher betrachtet. Die Untersuchungen werden zunächst an einem zuvor definierten Testprozess vorgenommen. Folgend werden ausgewählte Aspekte zusätzlich an einem realen nichtlinearen Mehrgrößensystem untersucht.

2 Modellbildung

In diesem Beitrag sollen dynamische Prozesse, die mit Hilfe nichtlineare Differentialgleichungen in der Form $\dot{x}(t) = f(t,x(t),u(t))$ und $y(t) = h(x(t),u(t))$ abgebildet werden können, durch datenbasierte Ansätze dargestellt werden. Dabei sollen die funktionalen Zusammenhänge zwischen f und h untersucht werden. Zur Modellierung der nichtlinearen dynamischen Systeme werden lokal - affine Zustandsraummodelle sowie tiefe rekurrente neuronale Netze eingesetzt. Beide Verfahren werden im Folgenden kurz vorgestellt.

2.1 Tiefe rekurrente neuronale Netze

In einem *Feedforward*-Netzwerk werden die Informationen auf einem direkten Weg durch das Netz geleitet, d.h. von der Eingabeschicht durch die einzelnen verdeckten Netzwerkschichten (*hidden layers*) zur Ausgabeschicht. Dabei berücksichtigen die Neuronen (*hidden units*) in den jeweiligen Schichten ausschließlich den aktuellen Netzeingang. Ein klassisches einschichtiges *Feedforward*-Netzwerk stellt eine Funktionsabbildung von einem Eingang x zu einem Ausgang $\hat{y}$ (skalare Größen) dar und wird für die Netzwerkschicht j wie folgt beschrieben:

$$\hat{y} = \sum_{j=1}^{d_1} \gamma_j \sigma(\alpha_j \cdot (x + \beta_j)). \tag{1}$$

Hierbei sind die Parameter α und β an die Dimension von x angepasste Größen. Die Variablen α, β und γ stellen die Modellparameter des Netzes dar mit $\theta = \{\alpha_j, \beta_j, \gamma_j\}_{j=1}^{d_1}$. Als Aktivierungsfunktion wird eine Sigmoidfunktion σ herangezogen [3]:

$$y = \sigma(x) = \frac{1}{1 + \exp(-x)}. \tag{2}$$

Der Ausdruck $\sigma(\alpha_j \cdot (x + \beta_j)$ beschreibt in diesem Fall die *hidden units*, die zusammen die verdeckte Netzwerkschicht (*hidden layer*) repräsentieren [1].

Feedforward-Netzwerke können Informationen nur von einem Zeitschritt in den nächsten übertragen. Um Informationen über einen längeren Zeitraum beizubehalten, bieten sich rekurrente neuronale Netze (RNNs) an. Eine effizi-

ente Art von RNNs sind Long-Short Term Memory (LSTM)-Netze, die aufgrund ihrer inneren Struktur Informationen auch über einen längeren Zeitraum speichern können [4]. Ein vollständiges LSTM-Netzwerk besteht aus einer oder mehreren verketteten LSTM-Schichten und einer abschließenden vollständig verketteten linearen Netzwerkschicht. Die Eigenschaft, Informationen über einen längeren Zeitraum zu speichern, wird durch zwei Arten von internen Zuständen bestärkt sowie der Möglichkeit, den Informationsfluss zu und aus diesen Zuständen zu steuern. Als innere Zustände einer LSTM-Schicht zum Zeitpunkt t dienen der verdeckte Zustand $\mathbf{h}_t$, auch *hidden state* genannt, und der Zellstatus $\mathbf{c}_t$, der für das Langzeitgedächtnis des Netzwerks verantwortlich ist. Der *hidden state* sowie der Zellstatus repräsentieren dabei sowohl den Ausgang einer LSTM-Schicht zum Zeitpunkt $t-1$ als auch den Eingang der folgenden LSTM-Schicht zum Zeitpunkt t. Auf Basis der Eingangsgrößen $\mathbf{x}_t$ und des *hidden states* $\mathbf{h}_{t-1}$ aus der vorherigen Netzwerkschicht wird der Informationsfluss in und aus dem Zellstatus über sogenannte *gates* gesteuert. Hierbei handelt es sich um ein *forget gate* $\mathbf{f}_t$, um ein *input gate* $\mathbf{i}_t$, einem *cell gate* $\mathbf{g}_t$ und einem *output gate* $\mathbf{o}_t$. Das *forget gate* $\mathbf{f}_t$ entscheidet, welche Informationen aus dem Zellstatus entfernt werden sollen. Hierfür wird eine Sigmoidfunktion σ (Gleichung (2)) herangezogen, deren Ausgabe zwischen 0 (vollständiges Vergessen) und 1 (Information vollständig beibehalten) liegt. Das *input gate* $\mathbf{i}_t$ entscheidet, ebenfalls auf Basis einer Sigmoidfunktion, welche Werte dem Zellstatus hinzugefügt werden sollen. Dafür wird durch das *cell gate* $\mathbf{g}_t$ mithilfe einer tanh-Funktion eine Kandidatenliste erstellt. Die Kombination aus *input gate* und *cell gate* führt zu einer Aktualisierung des Zellstatus $\mathbf{c}_t$. Zum Schluss gibt das *output gate* $\mathbf{o}_t$ auf Basis einer Sigmoidfunktion an, welche Informationen aus dem aktuellen Zellstatus $\mathbf{c}_t$ in den Ausgang der Netzwerkschicht fließen.

Die *gates* lassen sich wie folgt definieren:

$$\mathbf{f}_t = \sigma(\mathbf{W}_{\mathrm{xf}}\mathbf{x}_t + \mathbf{b}_{\mathrm{xf}} + \mathbf{W}_{\mathrm{hf}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{hf}}) \tag{3}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{\mathrm{xi}}\mathbf{x}_t + \mathbf{b}_{\mathrm{xi}} + \mathbf{W}_{\mathrm{hi}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{hi}}) \tag{4}$$

$$\mathbf{g}_t = \tanh(\mathbf{W}_{\mathrm{xg}}\mathbf{x}_t + \mathbf{b}_{\mathrm{xg}} + \mathbf{W}_{\mathrm{hg}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{hg}}) \tag{5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{\mathrm{xo}}\mathbf{x}_t + \mathbf{b}_{\mathrm{xo}} + \mathbf{W}_{\mathrm{ho}}\mathbf{h}_{t-1} + \mathbf{b}_{\mathrm{ho}}). \tag{6}$$

Hierbei repräsentieren $\mathbf{W}_{\mathrm{xf}}$, $\mathbf{W}_{\mathrm{xi}}$, $\mathbf{W}_{\mathrm{xg}}$ und $\mathbf{W}_{\mathrm{xo}}$ die Gewichtsmatrizen des Eingangs $\mathbf{x}_t$ und die Matrizen $\mathbf{W}_{\mathrm{hf}}$, $\mathbf{W}_{\mathrm{hi}}$, $\mathbf{W}_{\mathrm{hg}}$, $\mathbf{W}_{\mathrm{ho}}$ die Gewichtsmatrizen des vorherigen *hidden states* für das jeweilige *gate*. Darüber hinaus spiegeln die Vektoren $\mathbf{b}_{\mathrm{xf}}$, $\mathbf{b}_{\mathrm{xi}}$, $\mathbf{b}_{\mathrm{xg}}$ und $\mathbf{b}_{\mathrm{xo}}$ den Bias am Eingang und $\mathbf{b}_{\mathrm{hf}}$, $\mathbf{b}_{\mathrm{hi}}$, $\mathbf{b}_{\mathrm{hg}}$, $\mathbf{b}_{\mathrm{ho}}$ den Bias im *hidden state* des jeweiligen *gates* wider. Die Gewichtsmatrizen und Bias-Vektoren enthalten dabei die Parameter der LSTM-Schichten. Zur Berechnung des Zellstatus $\mathbf{c}_t$ und des aktuelle *hidden states* $\mathbf{h}_t$ werden die *gates* wie folgt verwendet:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \mathbf{g}_t \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t). \tag{8}$$

Zur Aktualisierung des Zellstatus $\mathbf{c}_t$ erfolgt eine Addition der Hadamard - Produkte ($\circ$) aus dem *forget gate* $\mathbf{f}_t$ und dem vorherigen Zellstatus $\mathbf{c}_{t-1}$ sowie aus dem *input gate* $\mathbf{i}_t$ und dem *cell gate* $\mathbf{g}_t$. Der *hidden state* $\mathbf{h}_t$ kann folgend auf Basis von $\mathbf{o}_t$ und $\mathbf{c}_t$ berechnet werden. Abschließend bildet der Ausgang der letzten LSTM-Schicht den Eingang für eine vollständig verkettete lineare Netzwerkschicht, die den finalen Netzausgang y_t ausgibt. Eine Übersicht über den Aufbau von LSTM-Schichten sowie deren Verkettung kann [5] entnommen werden.

Ein weiterer herauszustellender Fakt im Kontext von LSTM-Netzwerken ist deren strukturelle Ähnlichkeit zu den in der Systemidentifikation etablierten NLSS-Modellen. Beide Strukturen können unter gewissen Annahmen in die jeweilig andere überführt werden. Detaillierte Untersuchungen zur Vergleichbarkeit von NLSS-Modellen und LSTM-Netzen können [1] und [2] entnommen werden.

2.2 LOLIMOT

Eine weitere Möglichkeit der datenbasierten Modellbildung stellt der LOLIMOT (LOcal LInear MOdel Tree) - Algorithmus dar. Dieser gehört zu den inkrementellen heuristischen Konstruktionsalgorithmen und bildet nichtlineare Prozesse durch Überlagerung von mehreren lokal-affinen Teilmodellen ab, was den Vorteil einer leichten Interpretierbarkeit bietet [9].

Der Modellausgang eines lokal-affinen Modells wird wie folgt definiert:

$$\hat{y} = \sum_{i=1}^{M} (w_{i0} + w_{i1}x_1 + \ldots + w_{in}x_n)\Phi_i(\mathbf{z}). \tag{9}$$

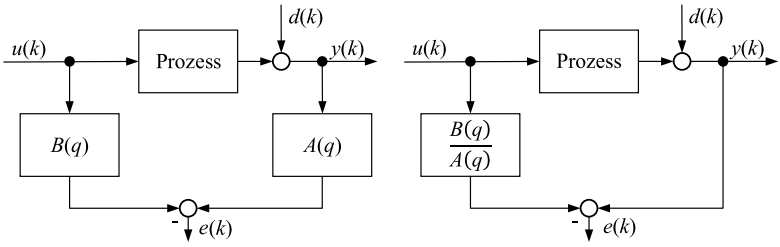
Der Modellausgang mit M affinen Teilmodellen ergibt sich aus der Summe der einzelnen Teilmodelle, gewichtet mit einer Aktivierungsfunktion $\Phi_i(\mathbf{z})$. Dabei wird jedes Teilmodell i durch n Eingangsgrößen $\mathbf{x} = [x_1\ x_2\ \ldots\ x_n]^T$ und n+1 Parametern $\mathbf{w}_i = [w_{i0}\ w_{i1}\ \ldots\ w_{in}]^T$ bestimmt. Die Aktivierung hängt von dem Eingangsvektor der Aktivierungsfunktionen $\mathbf{z} = [z_1\ z_2\ \ldots\ z_n]^T$ ab, welcher dem Arbeitspunkt des lokal-affinen Modelles entspricht. Wie $\mathbf{x}$ wird auch $\mathbf{z}$ aus den insgesamt zur Verfügung stehenden Eingangsgrößen bestimmt [7]. Der Gültigkeitsbereich der Teilmodelle wird durch eine normierte Aktivierungsfunktion beschrieben, welche den Ausgang eines jeden Teilmodells in Abhängigkeit des z-Regressors gewichtet und wie folgt definiert ist:

$$\Phi_i(\mathbf{z}) = \frac{\mu_i(\mathbf{z})}{\sum_{m=1}^{M} \mu_m(\mathbf{z})}. \tag{10}$$

Hier spiegelt $\mu_i(\mathbf{z})$ den Grad der Zugehörigkeit zu den Gültigkeitsbereichen wider, wobei die Summe der Aktivierungsfunktionen für eine sinnvolle Interpretation gleich Eins sein muss: $\sum_{i=1}^{M} \Phi_i(\mathbf{z}) = 1$. Um eine einfach zu interpretierende Modellstruktur sowie einen glatten Übergang zwischen den Teilmodellen zu erzeugen, werden als Zugehörigkeitsfunktion Gaußglocken verwendet

$$\mu_i(\mathbf{z}) = \exp\left(-\frac{1}{2}\frac{(z_1 - c_{i1})^2}{\sigma_{i1}^2}\right) \cdot \ldots \cdot \exp\left(-\frac{1}{2}\frac{(z_n - c_{in})^2}{\sigma_{in}^2}\right), \tag{11}$$

wobei c_{ij} das Zentrum und σ_{ij} die Standardabweichung der Gaußglocke des i-ten Teilmodells in der j-ten Dimension des z-Regressors sind. Die Größen c_{ij} und σ_{ij} ergeben sich dabei aus der Position und Größe der jeweiligen Gültigkeitsbereiche der Teilmodelle im Eingangsraum [7]. LOLIMOT zeichnet sich durch eine achsen-orthogonale Partitionierung des Eingangsraums aus, was eine einfache, schnelle und vor allem transparente Strukturoptimierung ermöglicht. Dabei wird schrittweise das lokal schlechteste Teilmodell durch eine achsen-orthogonale, mittige Teilung in zwei neue Teilmodelle aufgeteilt.



(a) Gleichungsfehler-Anordnung (b) Ausgangsfehler-Anordnung

Bild 1: Darstellung der Gleichungsfehler-Anordnung (NARX) für die Parameterschätzung in a) und der Ausgangsfehler-Anordnung für die Strukturoptimierung in b).

Das lokal schlechteste Teilmodell kann beispielsweise anhand der lokalen Verlustfunktion V:

$$V_{i,\text{lokal}} = \sum_{k=1}^{N} (y(k) - \hat{y}(k))^2 \Phi_i(\mathbf{z}(k)) \tag{12}$$

beurteilt werden, welche den quadrierten Modellfehler, gewichtet mit der jeweiligen Aktivierungsfunktion, aufsummiert. Dabei wird aus allen möglichen Teilungen diejenige ausgewählt, die zu dem geringsten globalen Fehler führt. Ein großer Vorteil der in Gleichung (9) beschriebenen Struktur kommt bei der Parameterschätzung zum Tragen. Da der Modellausgang linear in seinen Parametern ist, können lineare Verfahren, wie die Methode der gewichteten kleinsten Fehlerquadrate, eingesetzt werden.

LOLIMOT wird im Folgenden zur Modellierung eines nichtlinearen dynamischen Systemmodells verwendet, sodass die bisher betrachtete Modellstruktur um eine Dynamik erweitert werden muss. Dazu wird eine NARX-Struktur zur Schätzung der Parameter verwendet, wobei ein Ein-Schritt-Prädiktionsfehler basierend auf einer Gleichungsfehler-Anordnung (Bild 1a) minimiert wird. Hierbei stellt $B(q)$ das Zählerpolynom und $A(q)$ das Nennerpolynom der Übertragungsfunktion des Modells dar, wobei q mit $q^{-1}x(k) = x(k-1)$ den Zeitverzögerungsoperator darstellt. Des Weiteren repräsentiert $d(k)$ eine mögliche Störung [7]. Der Modellausgang des dynamischen lokal-affinen Gesamtmo-

dells kann wie folgt beschrieben werden:

$$\begin{aligned}\hat{y}(k) = \sum_{i=1}^{M} (&b_{i1}u(k-1) + \ldots + b_{in}u(k-n) \\ &- a_{i1}y(k-1) - \ldots - a_{in}y(k-n) + \xi_i)\Phi_i(\mathbf{z}(k)).\end{aligned} \tag{13}$$

Dabei wird exemplarisch ein SISO-System (*Single-Input-Single-Output*) n-ter Ordnung zugrunde gelegt. Es werden verzögerte Eingangsgrößen mit den Parametern a_i und verzögerte Ausgangsgrößen mit den Parametern b_i sowie ein Offset ξ als Regressoren verwendet. Die Minimierung des Gleichungsfehlers hinsichtlich der Parameterschätzung hat den Vorteil, dass auch bei dynamischen Systemen lineare Schätzverfahren, wie die Methode der kleinsten Fehlerquadrate, anwendbar sind, da Linearität in den Parametern gegeben ist. Die Strukturoptimierung basiert dagegen auf dem Ausgangsfehler (NOE (Nichtlinearer Ausgangsfehler)-Struktur), Bild 1b. Dies hat den Vorteil, dass die durch die Rückkopplung bedingte Fehlerfortpflanzung, aufgrund der Parameteroptimierung in der Gleichungsfehler-Anordnung, in der Strukturoptimierung vermieden werden kann [8]. Die Gleichung (13) kann auch in einer parameterveränderlichen Form dargestellt werden:

$$\hat{y}(k) = \sum_{i=1}^{n} b_i(\mathbf{z}(k))u(k-i) - \sum_{i=1}^{n} a_i(\mathbf{z}(k))\hat{y}(k-i) + \xi(\mathbf{z}(k)). \tag{14}$$

Das Software-Paket DYLAMOT (DYnamic Local Affine MOdeling Toolbox) beinhaltet Erweiterungen des LOLIMOT-Verfahrens zur Identifikation nichtlinearer dynamischer Prozesse mit Hilfe von lokal-affinen Modellen, wodurch eine bessere Modellgüte erzielt werden kann. Dabei verwendet DYLAMOT zur Partitionierung des Eingangsraums den heuristischen LOLIMOT - Konstruktionsalgorithmus. Eine der implementierten Erweiterungen ist die Verzögerung des arbeitspunktbeschreibenden z-Regressors entsprechend der jeweiligen Modelleingangsgrößen. Hierbei erfolgt die Änderung der Modellparameter in Abhängigkeit von $\mathbf{z}(k)$ zum Zeitpunkt k nicht gleichzeitig, sondern jeweils verzögert mit der Verzögerung i der dazugehörigen Eingangsgrößen $u(k-i)$ oder $\hat{y}(k-i)$ [7]. Ein großer Vorteil der Verzögerung der z-Regressoren ist ein stark verbessertes Eingangs- /Ausgangsverhalten im Vergleich zu einem

Modell mit gleichzeitiger Parameteränderung. Während die verzögerte Parameteränderung das Eingangs- /Ausgangsverhalten verbessert, wird der Offset mit einer unabhängigen Zähler-Dynamik ausgestattet, wodurch besonders die Interpretierbarkeit der Modelle erhöht wird. Durch die Erweiterung werden für jedes Teilmodell Offset-Parameter mit verzögerter Parameteränderung geschätzt. Der Modellausgang mit verzögerter Parameteränderung und Offset-Zähler-Dynamik ergibt sich dann zu:

$$\begin{aligned}\hat{y}(k) = &\sum_{i=1}^{n} b_i(\mathbf{z}(k-i))u(k-i) - \sum_{i=1}^{n} a_i(\mathbf{z}(k-i))\hat{y}(k-i) \\ &+ \sum_{i=1}^{n} \xi_i(\mathbf{z}(k-i)).\end{aligned} \tag{15}$$

Durch die eingeführte Erweiterung ist der Offset im Modell abhängig von unterschiedlichen Zeitpunkten, wodurch die Flexibilität des Modells gesteigert wird. Die verzögerte Parameteränderung ist dabei für die Verwendung mehrerer Offset-Parameter zwingend, da sonst alle Offset-Parameter zu einem zusammengefasst werden können [7]. Durch die vorgenommenen Erweiterungen der Modellstruktur ergibt sich, dass jedes Teilmodell in der Ausgangsfehler-Anordnung (Bild 1b) geschätzt wird. Hierdurch fließen die Parameter nichtlinear in den Ausgangsfehler ein, sodass nichtlineare Optimierungsverfahren zur Schätzung der Parameter herangezogen werden müssen [9].

Durch die getroffenen Erweiterungen der Modellstruktur lässt sich eine minimale Zustandsraumrealisierung erzeugen, die in Beobachternormalform dargestellt werden kann [7]. Dies ermöglicht einen Vergleich der vorgestellten Modellstrukturen gemäß [2].

3 Anwendungsbeispiel - Testprozess

Bei dem Vergleich der vorgestellten Methoden liegt der Fokus auf der zu erreichenden Modellgüte. Des Weiteren werden Aspekte wie die Extrapolationsfähigkeit sowie die Rauschempfindlichkeit der verwendeten Modelle untersucht. Der Vergleich findet zunächst an einem ausgewählten Testprozess statt. Dies

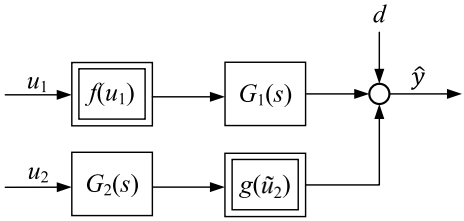


Bild 2: Strukturbild des verwendeten Testprozesses.

erfolgt zum einen mit dem Programmpaket DYLAMOT, zum anderen unter Verwendung der *Deep Learning Toolbox* in MATLAB [10].

3.1 Systembeschreibung - Testprozess

Die Struktur des Testprozesses kann Bild 2 entnommen werden. Bei dem Testprozess handelt es sich um ein nichtlineares dynamisches MISO (*Multiple Input, Single Output*)-System, basierend auf einem Hammerstein- und einem Wiener Modell. Zudem wird der Ausgang $\hat{y}$ mit einem weißen Rauschen d beaufschlagt. Zur Anregung des Prozesses werden APRB (Amplituden- moduliertes Pseudo Rausch Binär)-Signale verwendet und auf beide Eingänge u_1 und u_2 aufgeschaltet. Um eine hohe Modellgüte zu erzielen, müssen dabei die entscheidenden Frequenz- und Amplitudenbereiche des Systems hinreichend angeregt werden. Hierfür werden APRB-Signale mit einer Länge von 2046 Sekunden und einer Taktzeit von 2 Sekunden gewählt. Die Amplituden basieren dabei auf einer Gleichverteilung. Die statischen Nichtlinearitäten

$$f(u_1) = \frac{1}{1+\exp(-u_1)} = \frac{1}{2}(1+\tanh(\frac{u_1}{2})) \tag{16}$$

$$g(\tilde{u}_2) = \arctan(\tilde{u}_2)+\tilde{u}_2 \tag{17}$$

werden für $\{(u_1,u_2) \in \mathbb{R}^2 \mid 0,1 \leq u_j \leq 0,9 \; \forall \; j \in \{1,2\}\}$ betrachtet. Die Übertragungsfunktionen $G_1(s)$ und $G_2(s)$ werden wie folgt definiert:

$$G_1(s) = \frac{1}{5s^3+66s^2+73s+12} \tag{18}$$

$$G_2(s) = \frac{1}{s^3+2s^2+6s+5}. \tag{19}$$

Bei den Untersuchungen am Testprozess werden $N = 204600$ Datenpunkte aufgenommen, von denen 70% zum Trainieren der Modelle und 30 % als Testdatensatz verwendet werden. Zur Beurteilung der jeweiligen Modellgüte wird die Abweichung zwischen dem gemessenen Wert y und dem vorhergesagten Wert $\hat{y}$ bestimmt: $e = y - \hat{y}$. Zudem wird der RMSE auf den Testdaten berechnet:

$$\text{RMSE} = \sqrt{\frac{1}{N_\text{T}} \sum_{t=1}^{N_\text{T}} (y_t - \hat{y}_t)}, \tag{20}$$

wobei N_T die Anzahl an Testdatenpunkten beschreibt. Für die Untersuchungen muss eine geeignete Auswahl der verschiedenen Hyperparameter gefunden werden. Für eine ausreichende Modellgüte wird in DYLAMOT zur Modellbildung die Anzahl der lokal linearen Modelle zwischen 4 und 8 variiert. Des Weiteren wird die Ordnungszahl der lokalen Modelle zwischen 2 und 4 angepasst. Für das LSTM-Netzwerk wird die Anzahl der Neuronen, die Anzahl der verdeckten Schichten (*hidden layers*) und die Anzahl der Epochen für das Training so variiert, dass sich eine vergleichbare Modellgüte einstellt.

3.2 Modellvergleich - Testprozess

Mit beiden Verfahren kann bei den Untersuchungen eine hohe Modellgüte erreicht werden. Das beste Ergebnis wird in DYLAMOT mit einem lokallinearen Modell, bestehend aus 7 Teilmodellen vierter Ordnung, erzielt. Bei dem LSTM-Netzwerk wird das beste Ergebnis unter der Verwendung einer LSTM-Schicht mit 128 Neuronen erzielt. Dabei wird beim Training eine maximale Anzahl von 500 Epochen gewählt und zum Anlernen der ADAM (*adaptive moment estimation*)-Algorithmus [11] mit einer konstanten Lernrate von 0.01 verwendet. In Bild 3a ist ein Ausschnitt der zur Anregung verwendeten

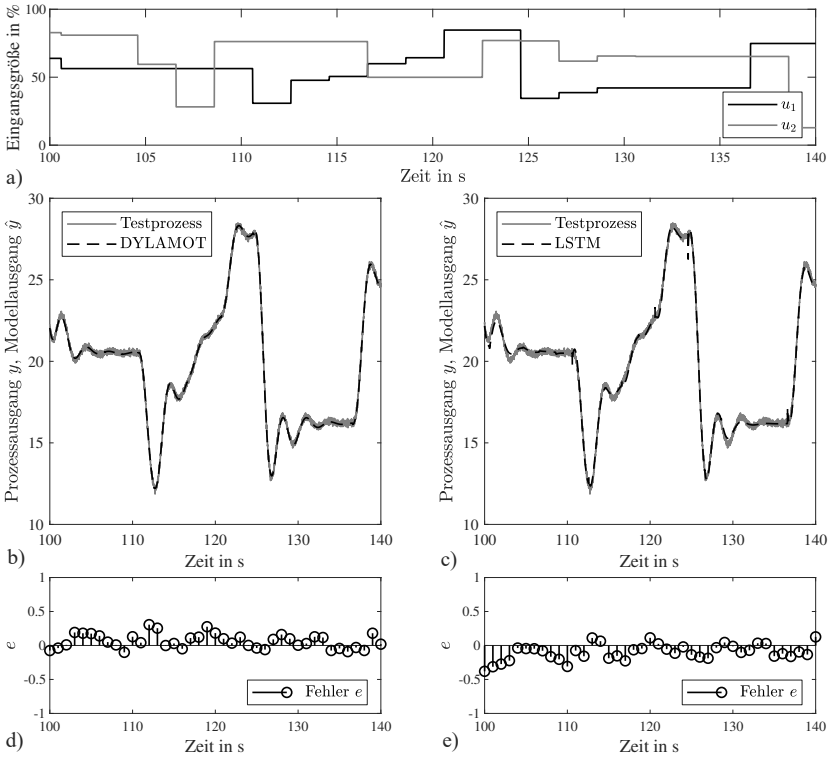


Bild 3: Vergleich der Modelle am Testprozess: a) verwendete Eingangsgrößen; b) Testprozessausgang und Modellausgang DYLAMOT; c) Testprozessausgang und Modellausgang LSTM; d) Fehler *e* Modellausgang DYLAMOT und e) Fehler *e* Modellausgang LSTM.

APRB-Signale für die beiden Eingangsgrößen u_1 und u_2 dargestellt. Aufgrund der Vergleichbarkeit zur Heizstrecke werden hierbei prozentuale Größen angegeben. In den Bildern 3b und c sind für beide Modelle die gemessenen und geschätzten Werte (y bzw. $\hat{y}$) der Ausgangsgröße zu sehen, basierend auf einem separaten Testdatensatz. Ebenfalls dargestellt ist der Fehler e bzw. die Abweichung zwischen gemessenem und geschätztem Wert Bilder 3d und e. Die hohe Modellgüte wird anhand des Vergleichs zwischen den gemessenen und geschätzten Werten in den Bildern 3b und c deutlich. Hierbei werden bei beiden Verfahren keine großen Abweichungen in den Verläufen festgestellt. DYLAMOT erreicht dabei eine geringfügig bessere Modellgüte als das

LSTM-Netz. Dies spiegelt sich auch beim RMSE wider. Dieser nimmt für den gesamten Testdatensatz bei DYLAMOT einen Wert von $0,132$ an und unter Verwendung des LSTM-Netzes einen Wert von $0,298$.

Für die Untersuchung der Extrapolationsfähigkeit werden die Amplituden der verwendeten APRB-Signale bei der Modellbildung verringert. Die Modellbildung findet mit APRB-Signalen statt, deren Amplituden aus dem Intervall [0.3,0.7] stammen. Daraufhin erfolgt die Generalisierung dieser Modelle auf Testdaten, deren Ursprung eine Anregung mit Amplituden aus dem Intervall [0.1,0.9] ist. Die trainierten Modelle werden folglich auf Daten getestet, deren minimalen und maximalen Werte die min. bzw. max. antrainierten Modellausgänge unter- bzw. überschreiten. In den Darstellungen 4a und b sind für beide Methoden die Eingangsgrößen aufgetragen sowie die Grenzen des antrainierten Bereichs. In den Bildern 4c und d sind die Modelle auf den amplitudenmodifizierten Testdaten generalisiert, was durch das Überschreiten der min. und max. Modellausgänge aus dem Training deutlich wird. Bei einem Vergleich der Eingangsdaten mit den jeweiligen Prozess- und Modellausgängen werden die Extrapolationsbereiche sichtbar. Bei dem Vergleich der Verläufe in den Bildern 4c und d wird deutlich, dass in diesen Bereichen unter der Verwendung von DYLAMOT die Abweichungen zwischen gemessenem und geschätztem Wert deutlich kleiner sind als beim LSTM-Netzwerk. Dieses Ergebnis wird auch vom RMSE widergespiegelt, der in diesem Fall für DYLAMOT $0,7554$ und für das LSTM-Netzwerk $1,5957$ beträgt. Ein Grund für dieses Ergebnis ist im Aufbau von LSTM-Netzen zu finden. Die einzelnen LSTM-Schichten basieren auf Sigmoid- und Tanh-Funktionen, die für die Aktualisierung der Zellzustände verantwortlich sind. Diese weisen entsprechend ihrer Verläufe einen unteren und einen oberen Sättigungsbereich auf, in denen das Wachstum gegen Null strebt. Diese Bereiche werden bei der Verwendung von Eingangsdaten, die außerhalb des antrainierten Modellbereichs liegen, erreicht. Diese Einschränkungen bestehen unter der Verwendung von DYLAMOT nicht. DYLAMOT weist ein lineares Extrapolationsverhalten auf, sodass keine unteren oder oberen Sättigungsbereiche erreicht werden können.

Ein weiterer Untersuchungspunkt in diesem Beitrag zielt auf die Empfindlichkeit der erstellten Modelle gegenüber möglichen Rauscheinflüssen ab. Für die Untersuchung werden die erstellten Modelle herangezogen und mit Rauschen

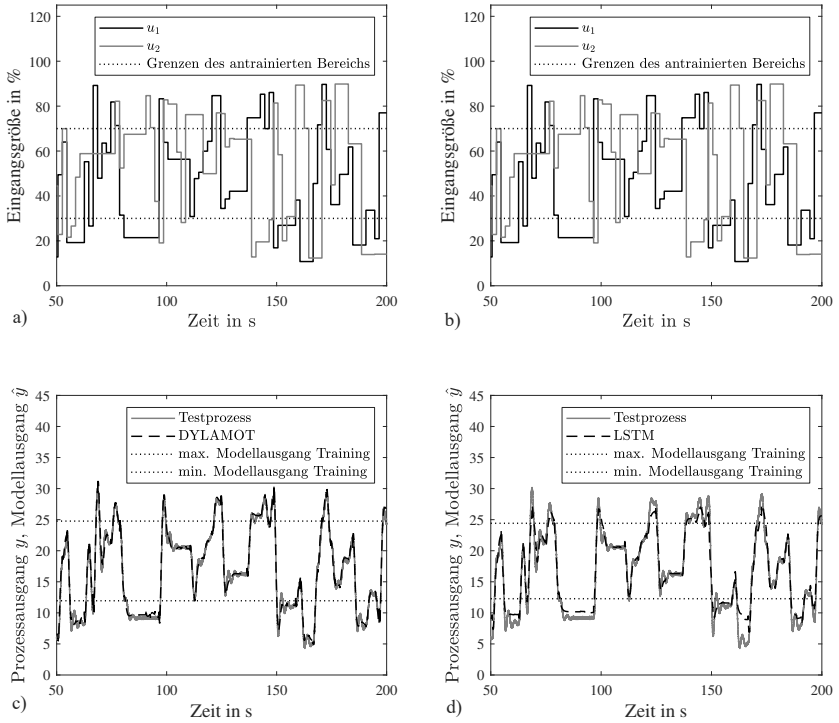


Bild 4: Extrapolationsverhalten am Testprozess: a),b) verwendete Eingangsgrößen; c) Modellausgang DYLAMOT bei Generalisierung auf amplitudenmodifizierten Testdatensatz; d) Modellausgang DYLAMOT bei Generalisierung auf amplitudenmodifizierten Testdatensatz.

beaufschlagt. Dafür wird der Testprozess mit weißem Rauschen angeregt. Der Prozessausgang y wird dann zur Generalisierung der Modelle herangezogen. Zur Anregung wird ein weißes Rauschen mit einer Abtastzeit von $0,01$ Sekunden auf beide Eingänge aufgeschaltet. Für u_1 wird zusätzlich eine spektrale Leistungsdichte von $8\mathrm{W/Hz}$ gewählt und für u_2 ein Wert von $4\mathrm{W/Hz}$. In Darstellung 5 sind für beide Methoden die gemessenen und geschätzten Werte (y bzw. $\hat{y}$) der Ausgangsgröße zu sehen. Beim Vergleich der Abbildungen ist deutlich zu erkennen, dass bei den gewählten Modell- und Rauschparametern das DYLAMOT-Modell (Bild 5a) stärker auf den verrauschten Eingang reagiert als das LSTM-Netz (Bild 5b). Dies spiegeln ebenfalls die mittels FFT (*fast Fourier transform*) ermittelten Grenzfrequenzen des Prozessausgangs und

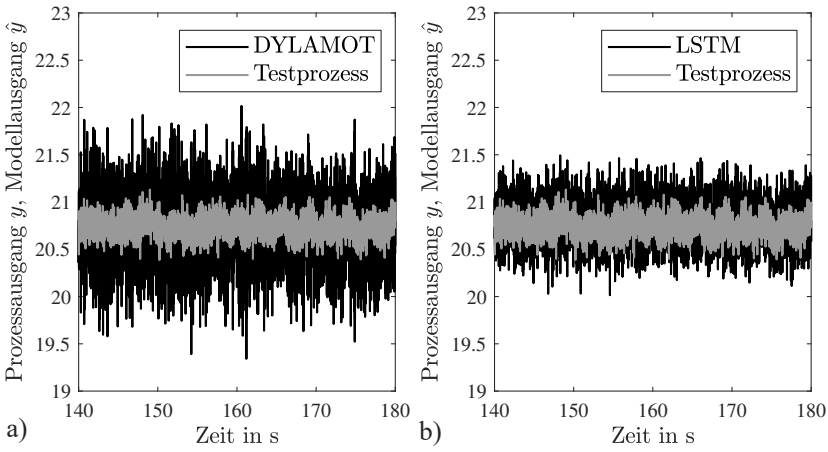


Bild 5: Empfindlichkeit der erstellten Modelle gegen weißem Rauschen: a) DYLAMOT; b) LSTM.

Tabelle 1: Grenzfrequenzen des Testprozesses und der erstellten Modelle.

Ausgang	Grenzfrequenz rad/s
Testprozess (Bode)	0,5000
Testprozess (FFT)	0,4947
DYLAMOT (FFT)	0,7727
LSTM (FFT)	0,5283

der jeweiligen Modellausgänge wider, die der Tabelle 1 entnommen werden können. Zur Kontrolle wird ebenfalls die Grenzfrequenz des Testprozesses aus dem Bode-Diagramm ermittelt. Die ermittelten Grenzfrequenzen aus dem Bode-Diagramm und der FFT sind wie erwartet beinahe identisch. Die größte Grenzfrequenz bei der Untersuchung weist DYLAMOT auf, was den Ergebnissen in Darstellung 5 entspricht. In dem unter der Verwendung von DYLAMOT erstellten Modell werden höhere Frequenzen deutlich weniger stark gedämpft als durch den eigentlichen Prozess und das erstellte LSTM-Modell. Eine Erklärung dafür bietet die Modellordnung, die für DYLAMOT gewählt wurde, um eine akzeptable und vergleichbare Modellgüte für den gewählten Testprozess zu erhalten. Man kann sagen, dass eine höhere Modellordnung zwar zu einer

besseren Prozessabbildung auf den Testdaten führt, das Modell aber anfälliger für Störeinflüsse macht.

4 Anwendungsbeispiel - Heizstrecke

Während im vorherigen Abschnitt die Untersuchungen an einem ausgewählten Testprozess stattfanden, wird nun ein realer nichtlinearer dynamischer Prozess herangezogen. Hierbei wird wie zuvor die jeweilige Modellgüte verglichen und die Extrapolationsfähigkeit untersucht. Auf eine Untersuchung der Rauschempfindlichkeit wird am realen Prozess verzichtet.

4.1 Systembeschreibung - Heizstrecke

Für die Untersuchungen wird eine Heizstrecke der Firma „ELWE Technik“ verwendet. Es handelt sich dabei um ein Labormodell, das über ein xPC Target-System unter Matlab-Simulink angesteuert werden kann. In der Heizstrecke verbaut sind neben einer Heizung und einem Lüfter auch Sensoren, mit denen die Lufttemperatur und der Luftmassenstrom erfasst werden können. Des Weiteren befindet sich eine manuell verstellbare Drosselklappe am Lufteinlass. Das System mit zwei Eingangs-, zwei Ausgangs- und einer Störgröße kann somit als Mehrgrößensystem angesehen werden. Bei den Eingangsgrößen des Systems handelt es sich um die Lüfterleistung P_{L} und die Heizleistung P_{H}. Bei der für die Untersuchung betrachteten Ausgangsgröße handelt es sich um die gemessene Temperatur T. Zur Anregung des Prozesses werden ebenfalls APRB-Signale verwendet und auf beide Eingänge u_1 und u_2 aufgeschaltet. Dabei stammen die Amplituden aus dem Intervall [10,90]% der jeweils maximal möglichen Leistungen. Sonstige Einstellungen zur Systemanregung gleichen den Angaben, die für die Untersuchungen am Testprozess getroffen worden sind, und können dem vorherigen Kapitel entnommen werden.

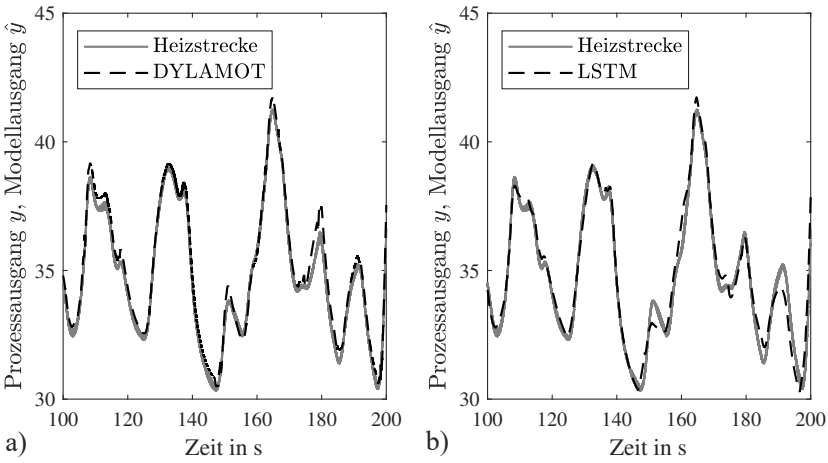


Bild 6: Vergleich der Modelle an der Heizstrecke: a) DYLAMOT; b) LSTM.

4.2 Modellvergleich - Heizstrecke

Mit beiden Verfahren kann eine vergleichbar hohe Modellgüte erreicht werden. Dabei findet die jeweilige Modellbildung mit denselben Hyperparametern statt, die auch für die Untersuchungen am Testprozess gewählt worden sind. Die Modellgüte wird anhand des Vergleichs zwischen den gemessenen und geschätzten Werten (Bilder 6a und b) deutlich. Hierbei kommt es zwar an einigen Stellen in den Verläufen zu geringen Abweichungen, für den gesamten Verlauf kann aber von einer ausreichend hohen Modellgüte gesprochen werden. DYLAMOT erreicht dabei ein geringfügig besseres Ergebnis als das LSTM-Netz. Dies spiegelt sich auch beim RMSE wider. Dieser nimmt für den gesamten Testdatensatz bei DYLAMOT einen Wert von $0,3884$ an und unter Verwendung des LSTM-Netzes einen Wert von $0,5121$.

Für die Untersuchung der Extrapolationsfähigkeit werden wie am Testprozess die Amplituden der zur Anregung verwendeten APRB-Signale bei der Modellbildung verringert. Die Modellbildung findet mit APRB-Signalen statt, deren Amplituden aus dem Intervall [30,70]% der jeweils maximal möglichen Leistungen stammen. Daraufhin erfolgt die Generalisierung dieser Modelle auf Testdaten, deren Ursprung eine Anregung mit Amplituden aus dem Intervall

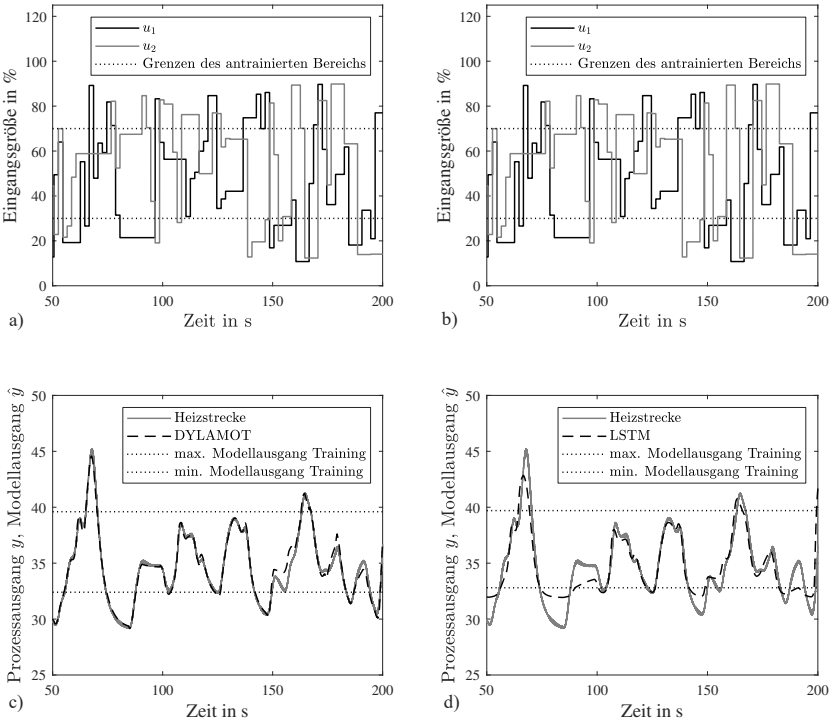


Bild 7: Extrapolation an der Heizstrecke: a), b) verwendete Eingangsgrößen; c) Modellausgang DYLAMOT bei Generalisierung auf amplitudenmodifizierten Testdatensatz; d) Modellausgang DYLAMOT bei Generalisierung auf amplitudenmodifizierten Testdatensatz.

[10,90]% ist. In den Darstellungen 7a und b sind für beide Methoden die Eingangsgrößen aufgetragen sowie die Grenzen des antrainierten Bereichs. In den Bildern 7c und d werden wie am Testprozess die Modelle auf den amplitudenmodifizierten Testdaten generalisiert, was durch das Überschreiten der min. und max. Modellausgänge aus dem Training deutlich wird. Durch den Vergleich der Eingangsdaten mit den jeweiligen Prozess- und Modellausgängen werden die Extrapolationsbereiche sichtbar. Hierbei werden die Ergebnisse aus den Untersuchungen am Testprozess bestätigt. Auch an der Heizstrecke kann mit DYLAMOT eine bessere Modellgüte erzielt werden als mit dem LSTM-Netzwerk. Beim Vergleich der Verläufe in den Bildern 7c und d wird deutlich, dass unter der Verwendung von DYLAMOT die Abweichungen zwi-

schen gemessenem und geschätztem Wert unter- und oberhalb des antrainierten Bereichs deutlich kleiner sind als beim LSTM-Netzwerk. Dieses Ergebnis wird auch vom RMSE widergespiegelt, der in diesem Fall für DYLAMOT $0,9848$ und für das LSTM-Netzwerk $1,8220$ beträgt.

5 Zusammenfassung und Ausblick

In diesem Beitrag werden tiefe rekurrente neuronale Netze (LSTM-Netze) bzw. lokal-affine Zustandsraummodelle (DYLAMOT) zur Abbildung nichtlinearer dynamischer Systeme verwendet. Dabei wird die jeweils mit den Verfahren erzielte Modellgüte verglichen. Zusätzlich wird ein Vergleich hinsichtlich der Extrapolationsfähigkeit und der Rauschempfindlichkeit gezogen. Es wird gezeigt, dass unter den gewählten Rahmenbedingungen in beiden Anwendungen eine hohe Modellgüte erzielt werden kann, DYLAMOT jedoch am Testprozess sowie an der Heizstrecke ein geringfügig besseres Ergebnis erzielt. Des Weiteren wird dargestellt, dass bei der Verwendung von DYLAMOT die Abweichungen zwischen gemessenem und geschätztem Wert außerhalb der antrainierten Bereiche deutlich kleiner sind als beim LSTM-Netzwerk. Zudem wird gezeigt, dass bei diesen Rahmenbedingungen das DYLAMOT-Modell stärker auf den verrauschten Eingang reagiert als das LSTM-Netz. Als weiterer Schritt in dem Methodenvergleich kann der Speicherbedarf der Modelle untersucht werden. Dies ist besonders bei einer Verwendung der Modelle im industriellen Umfeld von Bedeutung, wo Rechen- und Speichermöglichkeiten häufig begrenzt sind. Ein weiterer zu untersuchender Aspekt ist die zu erzielende Modellgüte bei der Verwendung verschiedener Anregungssignale. APRB-Signale können häufig in der Praxis, aufgrund der hohen Dynamik, nicht eingesetzt werden. Es werden stetige Anregungssignale gesucht, die in der Praxis sowohl einsetzbar sind als auch eine ausreichende Eingangsraumabdeckung vorweisen, um eine ausreichende Modellgüte zu erzielen.

Förderung

Dieses Vorhaben wurde aus Mitteln des Europäischen Fonds für regionale Entwicklung (EFRE) gefördert (Förderkennzeichen 34.EFRE-0300119).

Literatur

[1] L. Ljung, C. Andersson, K. Tiels und T. B. Schön. „Deep Learning and System Identification“. In: *Proc., IFAC Congress* Berlin. 2020.

[2] M. Schüssler, T. Münker und O. Nelles. „Deep Recurrent Neural Networks for Nonlinear System Identification“. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, S. 448-454. IEEE, 2019.

[3] J. Sjöberg, H. Hjalmarsson und L. Ljung. „Neural Networks in System Identification“. Linköping University. 1994.

[4] S. Hochreiter und J. Schmidhuber. „Long Short - Term Memory“. In: *Neural computation* 9.8. S. 1735-1780. Princeton University Press. 2019.

[5] J. Gonzalez und W. Yu. „Non-linear system modeling using LSTM neural networks“. In: *IFAC-PapersOnLine* 51.13. S. 485-489. 2018.

[6] O. Nelles. „Nonlinear System Identification“. Berlin Heidelberg: Springer. 2001.

[7] R. Zimmerschied. „Identifikation nichtlinearer Prozesse mit dynamischen lokal-affinen Modellen: Maßnahmen zur Reduktion von Bias und Varianz“. Dissertation, In: *Fortschritt-Bericht VDI Nr. 1150, Reihe 8*. Düsseldorf: VDI-Verlag. 2008.

[8] D. Schröder. „Intelligente Verfahren: Identifikation und Regelung nichtlinearer Systeme“. Berlin Heidelberg: Springer. 2010.

[9] L. Ljung. „System Identification: Theory for the User“. Prentice Hall. 1999.

[10] M. H. Beale, M. T. Hagan und H. B. Demuth. „Deep Learning Toolbox: User's Guide“. www.mathworks.com. 2020.

[11] D. P. Kingma und J. L. Ba. „Adam: A method for stochastic optimization“. In: *International Conference on Learning Representations*,12. 2014.

Takagi-Sugeno Observer for Tower Crane System

Lobna Tarek Aboserre[1], Horst Schulte[2], Ayman El-Badawy[1]

[1]Mechatronics Engineering Department, German University in Cairo, Egypt
E-Mail: lobna.aboserre@guc.edu.eg, ayman.elbadawy@guc.edu.eg

[2]School of Engineering I, University of Applied Sciences HTW Berlin
E-Mail: schulte@htw-berlin.de

Abstract

In this paper, a new representation of the nonlinear dynamics of the tower crane system in Takagi-Sugeno (TS) fuzzy form is proposed and used for observer design. The TS fuzzy nonlinear observer is utilized to estimate unmeasurable states with guaranteed global asymptotic stability. The stability analysis is formulated as linear matrix inequalities (LMIs). The TS fuzzy model is equivalent to a reduced-order nonlinear model of the tower crane system with a varying cable length. For verification, simulation results of the reduced-order model, TS nonlinear fuzzy model and the estimated observer states are compared to the results of a tower crane on a laboratory scale.

1 Introduction

Cranes are widely used for heavy load transportation and are typecilly classified into (1) tower cranes, (2) rotary cranes and (3) overhead cranes. Due to their wide range of applications on the construction site, tower cranes are the subject of investigations in automation and control engineering. It must

be taken into account that this type of crane system has a non-linear under-actuated complicated dynamics. Therefore, controlling of tower crane systems is a challenge.

Various crane control techniques have been proposed to achieve precise positioning and oscillation suppression of the payloads. Model-based fuzzy control has been recognized as an alternative approach to conventional techniques for overhead cranes. Adaptive fuzzy sliding-mode control is designed to guarantee asymptotic stability for payload oscillations [1]. A discrete-time TS fuzzy observer and controller is designed by [3]. In addition, the Mamdani-type fuzzy approach was used in [2] to design an active anti-swing controller.

Few of these techniques have been extended to the application of tower cranes [4]. Mainly conventional methods were used such as command shaping for oscillation reduction [5] and an optimal iterative method is presented in [6]. Just a fuzzy anti-swing controller for tower cranes based on the Mamdani type with consideration of friction and time delay was proposed in [7].

Regarding robust controllers, sliding mode control based on the nonlinear model is proposed in [8] and extended to an adaptive scheme [9]. Optimal control has been proposed with the path-following method [10]. The parametric uncertainties are handled using adaptive control in [11], adaptive backstepping for 2D system is proposed in [12] and adaptive nonlinear integral sliding mode was investigated in the work [13].

The existing methods are developed based on simplified control-oriented models using approximations and assumptions of the original tower crane dynamics. The difference between the original dynamics and the simplified model affects the controllers' performance and might lead to instability [11]. On the other hand, the consideration of the full nonlinear model increases the complexity of the control design and the closed-loop stability assessment. This complexity can be eliminated by reaching a proper reduced-order system. The main aim of this paper is to derive a suitable Takagi-Sugeno (TS) fuzzy model equivalent to a reduced-order nonlinear model for actual reflection of the system's dynamics. The Takagi-Sugeno framework is chosen for the advantage of an exact representation of the original nonlinear model and facilitation of the stability analysis and observer design of nonlinear systems. [14].

Moreover, the estimation convergence of state observer depends on the mathematical model [3]. Therefore, TS fuzzy observer is designed based on the aforementioned model. The observer is used to estimate the unmeasurable system states found in practical applications, such as the velocities that are required to be known for the controller. The previous closed-loop controllers' feedback depended on differentiating the measured positions [10] and filtering the results [11], which cause a time delay and reduction in the accuracy of the feedback.

This paper is organized as follows: Section II presents the nonlinear reduced order model of the tower crane and the state space representation. The equivalent TS fuzzy model is presented in Section III. In Section IV, the TS fuzzy observer is designed and its stability is analyzed. Section V presents a comparison study of the models, estimated observer states with the experimental results. Section VI presents the conclusion.

2 Continuous-time Nonlinear Dynamical Model

The model is derived based on the Euler Lagrange method [15] resulting in highly nonlinear under-actuated MIMO equations:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = F_{q_i}, \qquad i = 1,\ldots,5 \quad , \tag{1}$$

where q_i is a vector of the tower crane's five degrees of freedom shown in Figure 1. The vector $q_i = [x_t, \theta, \alpha, \beta, l]^T$ contains the trolley position, jib rotation, alpha, beta oscillation and the cable length respectively. The joint torque vector is given by $F_{q_i} = [F_x, F_\theta, F_l, 0, 0]^T$, where F_x denotes the trolley driving force, F_θ denotes the tower rotating torque and F_l denotes the cable driving force.

2.1 Continuous-time Nonlinear Model

The equations can be reformulated by dividing the tower crane's DOF into (1) actuated states q_1 and (2) un-actuated states q_2. The actuated states are the trol-

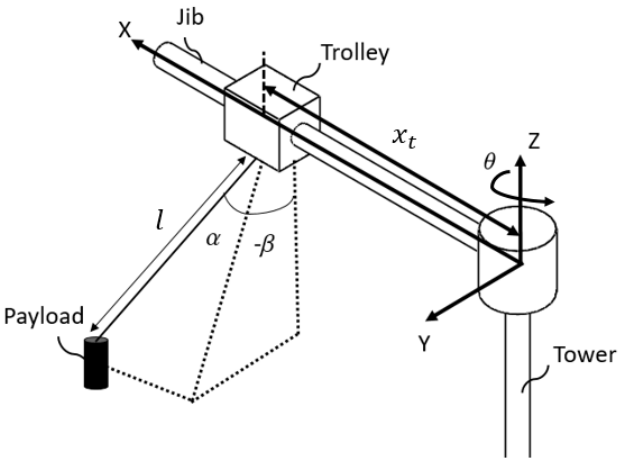


Figure 1: Schematic Representation of Tower Crane

ley position, the tower rotation and cable length denoted by $q_1 = [\ x_t \quad \theta \quad l\ ]^T$. The un-actuated states are the swing angles of the payload where $q_2 = [\ \alpha \quad \beta\ ]^T$. Using the method of order analysis [1], the complete nonlinear model is reduced for each term [16]. Small swing angles are assumed [17], the un-actuated states' equations are substituted into the actuated states' equations, and the actuated states' equations are substituted into the un-actuated states' equations, resulting in the following equations:

$$\ddot{x}_t + \left(\frac{B_x}{m_t} + K_2 \frac{1}{r_x^2 m_t}\right) \dot{x}_t - g\,\frac{m_p}{m_t}\alpha = \frac{1}{m_t} K_1 u_x\ , \tag{2}$$

$$(1 + \frac{m_t}{J_\theta} x_t^2)\ddot{\theta} + (B_\theta + K_2)\,\frac{\dot{\theta}}{J_\theta} - g\frac{m_p}{J_\theta} x_t \beta = \frac{1}{J_\theta} K_1 u_\theta\ , \tag{3}$$

$$\ddot{l} + \left(\frac{B_l}{m_p} + K_2 \frac{1}{r_x^2 m_p}\right) \dot{l} = \frac{1}{r_x m_p} K_1 u_l\ , \tag{4}$$

$$(J_p + m_p l^2)\ddot{\alpha} + B_\alpha \dot{\alpha} + g\, m_p\, l\, \alpha + m_p\, l\, \ddot{x}_t - m_p x_t l\, \dot{\theta}^2 - 2\, m_p\, l^2\, \dot{l}\, \dot{\theta} = 0\,,$$

$$m_p\, x_t\, l\, \ddot{\theta} + (J_p + m_p\, l^2)\ddot{\beta} + B_\beta\, \dot{\beta} + g\, m_p\, l\, \beta = 0\,, \tag{5}$$

where

$$F_{q_i} = K_1\, u_{(q_i)} - K_2\, \dot{q}_i\,, \tag{6}$$
$$K_1 = \frac{\eta_{(q_i)} K_{g(q_i)} K_{m(q_i)}}{R_{a(q_i)}} G_{a(q_i)}\,, \quad K_2 = \frac{\eta_{(q_i)} k^2_{g(q_i)} k^2_{m(q_i)}}{R_{a(q_i)}}\,.$$

m_t is the mass of the trolley, m_p is the mass of the payload, $B_{(q_i)}$ is the viscous friction coefficient, g is the gravitational constant, J_θ and J_p are the moment of inertia for the jib and load respectively. The motor parameters are: $K_{g(q_i)}$ is the gear ratio, $\eta_{(q_i)}$ is the motor gearbox and motor efficiency, $k_{m(q_i)}$ is the torque constant, r_x is the radius of pulley, $R_{a(q_i)}$ is the armature resistance and $G_{a(q_i)}$ is the amplifier gain. The equations of motion of the tower crane is in the form of:

$$M(q)\ddot{q} + B(q,\dot{q}) + G(q) = F\,, \tag{7}$$

where $M(q) \in R^{n\times n}$ is the inertia matrix which is a positive definite matrix for $l > 0$ and its inverse exists, $B(q,\dot{q}) \in R^{n\times 1}$ is the Coriolis, centripetal and friction matrix and $G(q) \in R^{n\times 1}$ denotes the gravitational force vector.

2.2 Continuous-time Nonlinear State Space Representation

A representation of a given nonlinear system in Takagi-Sugeno form is obtained in a compact set if the state space of a nonlinear system can be expressed as follows:

$$\begin{aligned} \dot{x} &= f(x,u)x + g(x,u)u, \\ y &= h(x,u)x. \end{aligned} \tag{8}$$

f, g and h are smooth nonlinear matrix functions and assumed to be bounded. $x = [x_t, \dot{x}_t, \theta, \dot{\theta}, l, \dot{l}, \alpha, \dot{\alpha}, \beta, \dot{\beta}]^T$ is the state vector of (8) and $u = [u_x, u_\theta, u_l]^T$ is the input vector. In detail with (2) - (5) this results in

$$\begin{aligned}
\dot{x}_1 &= x_2 \\
\dot{x}_2 &= A_1 \\
\dot{x}_3 &= x_4 \\
\dot{x}_4 &= A_2 \\
\dot{x}_5 &= x_6 \\
\dot{x}_6 &= \frac{1}{m_p}[K_1\, u_l - (B_l + K_2)\, x_6] \\
\dot{x}_7 &= x_8 \\
\dot{x}_8 &= \frac{m_t\, x_5(-A_1 + x_1\, x_4^2 + 2\, x_{10}\, x_4\, x_5 - x_7\, g) - B_\alpha\, x_8}{m_p\, x_5^2 + J_p} \\
\dot{x}_9 &= x_{10} \\
\dot{x}_{10} &= \frac{-B_\beta\, x_{10} - g\, m_p\, x_5\, x_9 - m_p\, A_2\, x_1\, x_5}{(m_p\, x_5^2 + J_p)}
\end{aligned} \tag{9}$$

where

$$\begin{aligned}
A_1 &= \frac{1}{m_t}[K_1\, u_x - x_2(B_x + K_2) + x_7\, g\, m_p], \\
A_2 &= \frac{K_1 u_t - x_4(B_\theta + K_2) + g\, m_p\, x_1\, x_9}{m_t x_1^2 + J_p}
\end{aligned} \tag{10}$$

3 Nonlinear Dynamic TS Fuzzy Model

In this paper the TS model of the tower crane is constructed analytically based on the previously presented state space model (9).

3.1 Sector Nonlinearity Approach

Takagi-Sugeno fuzzy representation of the crane model is derived using sector non-linearity approach [18] to be used in the design process of the continuous-time observer. The scheduling variables are chosen as $z_j \in [\underline{z}_j, \bar{z}_j]$, $j = 1, 2, \ldots, p$ where $\underline{z}_j$ and $\bar{z}_j$ are the minimum and maximum values in the considered operating range respectively. The six premise variables are chosen as:

$$z = [x_t, \frac{1}{m_t\, x_t^2 + J_\theta}, l, \dot{\theta}\, l, \beta, \frac{1}{m_p\, l^2 + J_p}]^T \tag{11}$$

The system's states are bounded and the bounds are based on the physical constraints of the real system to be investigated [21]: $z_1 \in [0.22, 0.52]$, $z_2 \in [0.386, 0.41]$, $z_3 \in [0.15, 1.2]$, $z_4 \in [-0.15, 1.2]$, $z_5 \in [-\pi/2, \pi/2]$ and $z_6 \in [2.1, 45]$. The rules of the TS system are constructed:

$$\begin{aligned} &\text{if } z_1 \text{ is } Z_1^i \text{ and } \ldots \text{ and } z_p \text{ is } Z_p^i \text{ then} \\ &\dot{x} = A_i\, x + B_i\, u, y = C_i\, x\,, \end{aligned} \tag{12}$$

where $Z_1^i, \ldots, Z_p^i$ are the corresponding sets of the premise variables with the number of rules $i = 1, 2, \ldots, m$ equal to $m = 2^p = 64$, where $p = 6$ is the number of premise variables and 2 is the number of weighting functions per premise variable. The nonlinear system is represented as TS fuzzy model in the form of

$$\begin{aligned} \dot{x} &= \sum_{i=1}^{m} h_i(z)(A_i\, x + B_i\, u), \\ y &= \sum_{i=1}^{m} h_i(z)\, C_i\, x \end{aligned} \tag{13}$$

$h_i(z(t)) \geq 0$ are the normalized membership function with convex sum property $\sum_{i=1}^{m} h_i(\mathbf{z}(t)) = 1$ [14] and is calculated as the product of the weighting functions:

$$h_i(z) = \prod_{j=1}^{p} w_{i_j}^j(z_j) \tag{14}$$

where $i_j \in \{0,1\}$. For each z_j, two weighting functions are constructed:

$$w_0^j = \frac{\bar{z}_j - z_j}{\bar{z}_j - \underline{z}_j}, \qquad w_1^j = 1 - w_0^j, \qquad j = 1,2,\ldots,p\,. \tag{15}$$

4 TS Fuzzy Observer

In this section, the nonlinear observer is designed to estimate the unmeasured states relying on the system model (13):

$$\begin{aligned} \dot{\hat{x}} &= \sum_{i=1}^{m} h_i(z)[]A_i\,\hat{x} + B_i\,u + L_i(y-\hat{y})]\,, \\ \hat{y} &= C\,\hat{x}\,, \end{aligned} \tag{16}$$

where

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{17}$$

that refers to the five measured states, and $\hat{x}$ is the estimated state vector, $\hat{y}$ is the estimated measurement and L_i are the observer gains.

4.1 Stability analysis via Lyapunov Approach

The stability analysis is reduced to linear matrix inequality (LMI) problem, which is equivalent to finding solutions to original problems. The estimation error is defined as $e = x - \hat{x}$, while the error dynamics are [19]:

$$\begin{aligned} \dot{e} = \dot{x} - \dot{\hat{x}} &= \sum_{i=1}^{m} h_i(z)\,\left[A_i\,(x-\hat{x}) - L_i(y-\hat{y})\right] \\ &= \sum_{i=1}^{m} h_i(z)\,(A_i - L_i\,C)\,e \end{aligned} \tag{18}$$

Theorem 4.1 [14](page 64): The estimation error dynamics with common measurement matrix C in (18) is asymptotically stable, if there exist $P = P^T$ and L_i, so that

$$\mathscr{H}(P(A_i - L_i C)) < 0 \tag{19}$$

for all $i = 1, 2, \ldots, m$, where $\mathscr{H}(X) = X^T + X$. The following LMI problem is feasible using the variable $M_i = P L_i$. The performance measure is satisfied by adding a convergence rate of the observer, such that:

$$\mathscr{H}(P A_i - M_i C_i) + 2\alpha P < 0\,, \tag{20}$$

where α is the decay rate of the estimation error e.

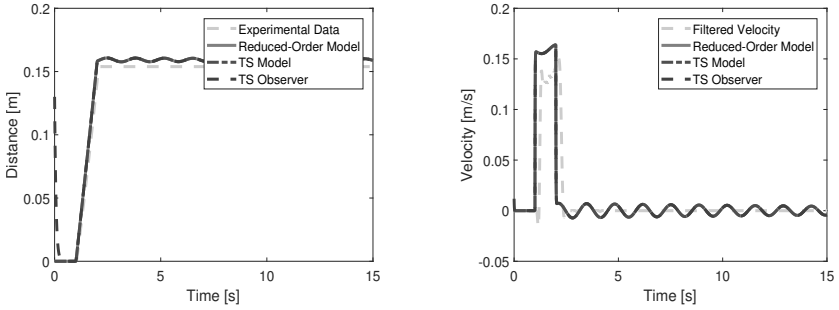
The stability condition of Theorem 4.1 is derived using the quadratic function $V(e) = e^T P e$. The derivative of the Lyapunov function is:

$$\begin{aligned}
\dot{V}(e) &= \dot{e}^T P e + e P \dot{e} \\
&= \sum_{i=1}^{m} h_i(z)\, ((A_i - L_i C)e)^T P e + e^T P (A_i - L_i C)\, e \\
&= \sum_{i=1}^{m} h_i(z)\, e^T ((A_i - L_i C)^T P + P(A_i - L_i C)) e < 0
\end{aligned} \tag{21}$$

If a common positive definite matrix $P = P^T > 0$ exists for all $m = 64$ fuzzy models and the Lyapunov function is decreasing, therefore the system is globally asymptotically stable. The observer by design guarantees, that the estimation error converges asymptotically to zero. The system equation are used to obtain M_i for observer gains L_i with $L_i = P^{-1} M_i$. The solution of the observer is computed using YALMIP toolbox [20] and SEDUMI solver in MATLAB

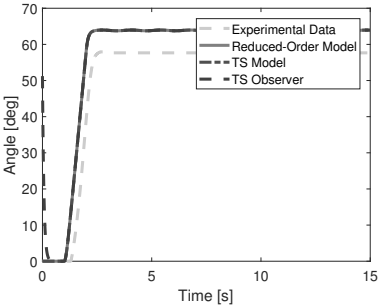
Table 1: The values of the estimated parameters

Crane parameter mechanics	Crane parameter drives
$m_t = 0.7$ Kg	$K_{g(x_t)} = K_{g(l)} = 76.84$
$m_p = 0.32$ Kg	$K_{g(\theta)} = 275$
$B_x = 28$ Nm/s	$\eta_{(x_t)} = \eta_{(l)} = 0.36$
$B_\theta = 14$ Nm/s	$\eta_{(\theta)} = 0.24$
$B_l = 19.5$ Nm/s	$k_{m(x_t)} = k_{m(l)} = 0.032$ Nm/A
$B_\alpha = 0.001$ Nm/s	$k_{m(\theta)} = 0.0195$ Nm/A
$B_\beta = 0.001$ Nm/s	$r_x = 0.0375$ m
$J_\theta = 1.7$ Kgm^2	$R_{a(x_t)} = R_{a(l)} = 25$ V/A
$J_p = 0.023$ Kgm^2	$R_{a(\theta)} = 0.5$ V/A
$g = 9.81$ m/s^2	$G_{ax} = 15$
	$G_{a(\theta)} = G_{a(l)} = 12$

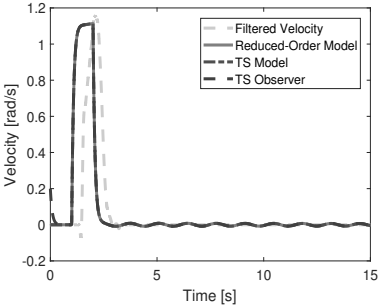


(a) Trolley position x_t (b) Trolley velocity $\dot{x}_t$

Figure 2: Trolley motion

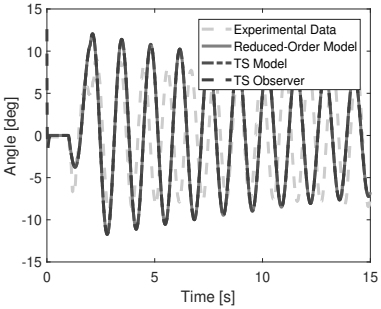


(a) Jib angle θ

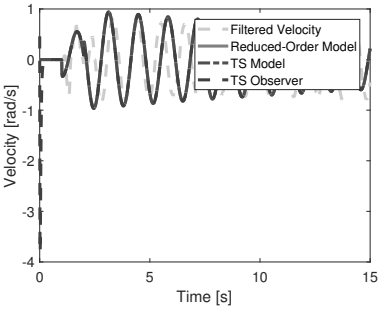


(b) Jib angular velocity $\dot{\theta}$

Figure 3: Jib motion

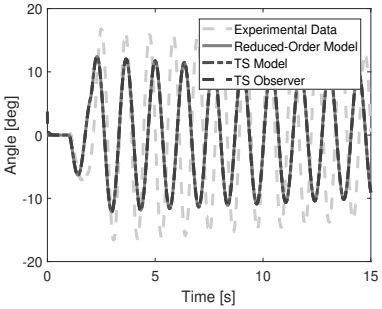


(a) Payload angle α

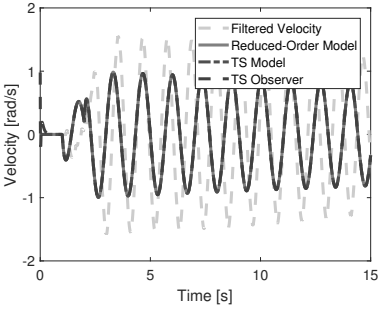


(b) Payload angular velocity $\dot{\alpha}$

Figure 4: Payload motion: α coordinate

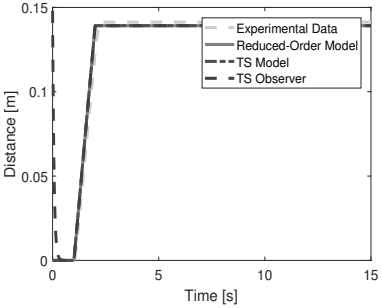


(a) Payload angle β

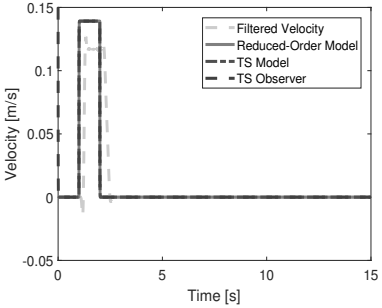


(b) Payload angular velocity $\dot{\beta}$

Figure 5: Payload motion: β coordinate



(a) Cable length l



(b) Payload velocity: l-coordinate

Figure 6: Payload motion: cable coordinate l

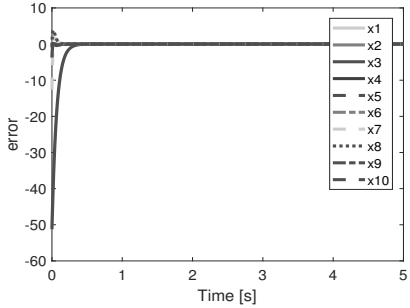


Figure 7: Estimation error $e = x - \hat{x}$

5 Experiments

For the experimental investigations, a real-time data acquisition (RT-DAC /USB2 board) is used as an interface between the personal computer and the tower crane in laboratory scale [21]. The crane's parameters used in the model equations were estimated by [22] using the prediction error method. Other parameters such as friction coefficients and the inertia are estimated using an off-line identification parameter estimation tool in MATLAB via sum square method based on the structure of the model. The motor parameters are presented by [17] and the parameter values are given in Table 1. The decay rate used is $\alpha = 10$, hence the observer dynamics is faster than the dynamics of the closed loop system. The controller needs a fast reconstructed signal, giving an advantage over existing methods. The initial conditions used for the experimental setup and the model are equal to the home position. The home point of the mechanical system is equal to 0.22 m for the trolley position and zero for the rest of the states. The initial conditions for the estimated states are $\hat{x}_0 = [0.35, 0.0118, 0.894, 0.1991, 0.298, 0.15, 0.22, 0.46, 0.065, 0.98]^T$. The input used for moving the system is a pulse signal for 1 second and the corresponding experimental data is measured. This data is plotted against the results of the TS model and TS observer using the same input. The comparison between the three results is carried out for a typical crane maneuver which is a superposition of three motions: the trolley translation, jib rotation and cable motion.

From Figure 2 to 6 it can be seen that the error between the experimental data and the reduced-order model is equal to 0.62 cm in the trolley position x_t, 6 degrees in the jib position θ, 3 degrees in the α oscillation and 4 degrees for the β oscillation. The velocities are calculated using the conventional method of differentiating the position and filtering the results. However, the controller will be based on the observer's result to avoid using inaccurate feedback. The estimated velocities have the same profile as the filtered velocities while the differences are due to the differentiation error and the time delay found in the filtered data. Therefore, the results show that the reduced-order model captures the actual dynamics accurately with small magnitude of error.

Moreover, in Figure 2 to Figure 6, the overlapping of the data shows that the obtained TS model is same as the original in the considered limits. The estimation error converges to zero in less than 1 second for all states as shown in Figure 7.

References

[1] M.-S. Park, D. Chwa, and S.-K. Hong, "Antisway tracking control of overhead cranes with system uncertainty and actuator nonlinearity using an adaptive fuzzy sliding-mode control". In: *IEEE Transactions on Industrial Electronics*, vol. 55, no. 11, pp. 3972–3984, 2008.

[2] D. Antic, Z. Jovanovic, S. Peric, S. Nikolic, M. Milojkovic, and M. Milosevic, "Anti-swing fuzzy controller applied in a 3D crane system". In: *Engineering, Technology & Applied Science Research*, vol. 2, no. 2, pp. pp–196, 2012.

[3] P. Petrehuş, Z. Lendek, and P. Raica, "Fuzzy modeling and design for a 3D crane". In: *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 479–484, 2013.

[4] L. Ramli, Z. Mohamed, A. M. Abdullahi, H. Jaafar, and I. M. Lazim, "Control strategies for crane systems: A comprehensive review". In: *Mechanical Systems and Signal Processing*, vol. 95, pp. 1–23, 2017.

[5] G. G. Parker, B. Petterson, C. Dohrmann, and R. D. Robinett, “Command shaping for residual vibration free crane maneuvers”. In: *Proceedings of 1995 American Control Conference - ACC’95*, vol. 1, 1995, pp. 934–938 vol.1.

[6] A. R. Golafshani, “Modeling and optimal control of tower crane motions”. Ph.D. dissertation, University of Waterloo, 1999.

[7] H. M. Omar and A. H. Nayfeh, “Anti-swing control of gantry and tower cranes using fuzzy and time-delayed feedback with friction compensation”. In: *Shock and Vibration*, vol. 12, no. 2, pp. 73–89, 2005.

[8] T. A. Le, V.-H. Dang, D. H. Ko, T. N. An, and S.-G. Lee, “Nonlinear controls of a rotating tower crane in conjunction with trolley motion”. In: *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 227, no. 5, pp. 451–460, 2013.

[9] A. T. Le and S.-G. Lee, “3D cooperative control of tower cranes using robust adaptive techniques”. In: *Journal of the Franklin Institute*, vol. 354, no. 18, pp. 8333–8357, 2017.

[10] M. Böck and A. Kugi, “Real-time nonlinear model predictive path-following control of a laboratory tower crane”. In: *IEEE Transactions on Control Systems Technology*, vol. 22, no. 4, pp. 1461–1473, 2014.

[11] N. Sun, Y. Fang, H. Chen, B. Lu, and Y. Fu, “Slew/translation positioning and swing suppression for 4-DOF tower cranes with parametric uncertainties: Design and hardware experimentation”. In: *IEEE Transactions on Industrial Electronics*, vol. 63, no. 10, pp. 6407–6418, 2016.

[12] W. W. Bai and H.-P. Ren, “Horizontal positioning and anti-swinging control tower crane using adaptive sliding mode control”. In: *The 30th Chinese Control And Decision Conference (2018 CCDC)*. IEEE, 2018, pp. 4013–4018.

[13] M. Zhang, Y. Zhang, H. Ouyang, C. Ma, and X. Cheng, “Adaptive integral sliding mode control with payload sway reduction for 4-DOF tower crane systems”. In: *Nonlinear Dynamics*, pp. 1–15, 2020.

[14] Z. Lendek, T. M. Guerra, R. Babuska, and B. De Schutter, *Stability Analysis and Nonlinear Observer Design Using Takagi-Sugeno Fuzzy Models*. Springer, 2011.

[15] M. W. Spong, S. Hutchinson, M. Vidyasagar *et al.*, *Robot Modeling and Control*. wiley New York, 2006, vol. 3.

[16] A. H. Nayfeh, *Introduction to Perturbation Techniques*. John Wiley & Sons, 2011.

[17] P. Breuning, "Linear model predictive control of a 3D tower crane for educational use". Ph.D. dissertation, University of Stuttgart, 2015.

[18] K. Tanaka and H. Ohtake, "Fuzzy modeling via sector nonlinearity concept". In: *Transactions of the Society of Instrument and Control Engineers*, vol. 37, no. 4, pp. 372–378, 2001.

[19] H. O. Wang, K. Tanaka, and M. F. Griffin, "An approach to fuzzy control of nonlinear systems: Stability and design issues". In: *IEEE Transactions on Fuzzy Systems*, vol. 4, no. 1, pp. 14–23, 1996.

[20] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab". In: *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*. IEEE, 2004, pp. 284–289.

[21] L. Inteco, "Inteco ltd, The laboratory tower crane system controlled from PC, User Manual". 2006.

[22] F. Altaf, "Modeling and event-triggered control of multiple 3D tower cranes over WSNs". Master Thesis, Automatic Control Laboratory, School of Electrical Engineering, KTH Stockholm, Sweden, October 2010.

Vergleich datengetriebener dynamischer Modelle des Wärme- und Feuchteübertragungsvorgangs in einer Wand

Alessio Cavaterra, Markus Östreich, Steven Lambeck

Fachbereich Elektrotechnik und Informationstechnik, Hochschule Fulda
Leipziger Str. 123
E-Mail: {alessio.cavaterra,markus.oestreich,steven.lambeck}@et.hs-fulda.de

1 Einführung

Im Rahmen des „HumFlow"–Projekts[1] am Fachbereich Elektrotechnik und Informationstechnik der Hochschule Fulda versuchen die Autoren das Problem des Eingriffs in denkmalgeschützte Bausubstanz im Bereich der Messdatenerfassung durch Einsatz von Funksensoren im Außen- und Innenbereich eines Raums zu lösen. Diese Sensoren messen die Temperaturen und relativen Luftfeuchtigkeiten der Wandoberfläche und sind gegenüberliegend an der Außenwand und an der Innenwand „minimal-invasiv" angebracht, d. h., dass zur Befestigung der Sensoren an der Wand keine signifikanten Beschädigungen entstehen. Dies ist insbesondere für denkmalgeschützte Gebäude interessant. Mit Hilfe der genannten Messgrößen und einem hygrothermischen Modell der Wand ist beispielsweise eine bessere Prädiktion des Innenraumklimas möglich. Die Vorhersagen können z. B. in einem modellprädiktiven Regelungskonzept weiterverarbeitet werden, was die Einhaltung der Raumklimaanforderungen gemäß der Präventiven Konservierung [1] sicherstellen kann. Weitere Anwendungsbereiche sind denkbar.

[1] gefördert durch das Förderprogramm „Forschung für die Praxis" des HMWK

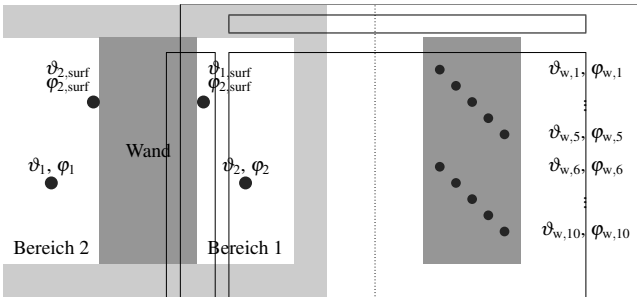


Bild 1: Links: Querschnitt des „HumFlow“-Versuchsstandes. Das Gehäuse ist hellgrau dargestellt. Die Sensoren stellen dunkelgraue, gefüllte Kreise dar. Rechts: Anordnung der Kombi-Sensoren (dunkelgrau) in unterschiedlichen Tiefen im Wandelement (grau).

Im vorliegenden Beitrag wird die datengetriebene Modellbildung der Wärme- und Feuchteübertragungsmechanismen in einem Wandsegment behandelt. Der Beitrag führt grundlegende Arbeiten aus [2] fort und erweitert diese um nichtlineare Modellierungsansätze mit Hilfe künstlicher neuronaler Netze (KNN) sowie Takagi-Sugeno-Fuzzy-Systeme (TS) [3]. Der Versuchsaufbau ist ausführlich in [2] beschrieben, wird im zweiten Abschnitt jedoch nochmals kurz skizziert. Nach einer kurzen Beschreibung der Methodik im dritten Abschnitt werden die Ergebnisse anschließend im vierten Abschnitt bewertet.

2 Der Versuchstand

Bild 1 skizziert den Versuchsaufbau. Er besteht aus einem einseitig offenen Gehäuse aus mit Phenolharz beschichtetem Sperrholz. In das Gehäuse kann ein herausnehmbares Wandsegment eingelassen werden, was das Gehäuseinnere in zwei Bereiche aufteilt. Die offene Seite des Gehäuses wird als „Bereich 2“ bezeichnet und steht in direktem Austausch mit der Umgebungsluft des Labors. Der durch das Wandsegment abgetrennte „Bereich 1“ beinhaltet ein elektrisches ansteuerbares Heizelement sowie ein kompaktes Konstantfeuchtegerät. Hierdurch können verschiedene Temperatur- und Luftfeuchtebedingungen erzeugt werden.

Zur Messung der Temperatur und der relativen Luftfeuchtigkeit in den Bereichen 1 und 2 werden mehrere kombinierte Sensoren eingesetzt. Indizes 1 und 2 zeigen den Messort an. Die Oberflächentemperaturen und -feuchten der Wand werden ebenfalls beidseitig mit jeweils einem Sensorpaar (Index surf) erfasst. Auch innerhalb der Wand sind zehn weitere kombinierte Sensoren, angeordnet in zwei Reihen, zur Erfassung der genannten Messgrößen in Bohrungen unterschiedlicher Tiefe eingefasst (s. Bild 1 rechts). Insgesamt werden so zwanzig Messgrößen innerhalb des Wandsegments gemessen. Die obere und die untere Sensor-Reihe sind hierbei in derselben Tiefe angeordnet.

3 Systemidentifikation

Das hygrothermische Verhalten des Wandsegments (Lehm) wird mit Hilfe eines linearen Zustandsraummodells (ZR), eines TS-Modells und eines KNN abgebildet. Die beiden letztgenannten Modellansätze sind in der Lage, nichtlineare dynamische Prozesse zu beschreiben [4]. Das ZR-Modell dient als Referenz, um die Güte des TS- bzw. KNN-Modells besser einordnen zu können. Zum Einsatz kommen die in der Software MATLAB integrierte System Identification Toolbox sowie die Deep Learning Toolbox. Letztere beinhaltet auch alle notwendigen Methoden zum Training flacher Netze. Zur Schätzung des TS-Modells wird der LOLIMOT-Algorithmus aus der LMN-Toolbox Version 1.5.2 [5] verwendet. Die Modellstrukturen der nichtlinearen Modellansätze sind als nichtlineare autoregressive Modelle mit zusätzlichen Eingängen (NARX) interpretiertbar [4], siehe Gleichungen (1) und (2).

$$\hat{y}_{\mathrm{ARX}}(k) = \sum_{i=1}^{n_b} b_i \cdot u(k-i) - \sum_{i=1}^{n_a} a_i \cdot y(k-i)$$

$$\hat{y}_{\mathrm{ARX}}(k) = \underline{\theta x} = \left(b_1, \ldots, b_{n_b}, -a_1, \ldots, -a_{n_a}\right) \begin{pmatrix} u(k-1) \\ \vdots \\ u(k-n_b) \\ y(k-1) \\ \vdots \\ y(k-n_a) \end{pmatrix} \tag{1}$$

In obiger Gleichung ist der einfache Fall eines linearen Eingrößensystems dargestellt. Eine Erweiterung auf Mehrgrößensysteme gestaltet sich sehr einfach und wird deshalb hier nicht nochmals aufgeführt. Mit einer nichtlinearen Funktion $f(\cdot)$ entsteht ein NARX-Modell

$$\hat{y}_{\mathrm{NARX}}(k) = f(\underline{\theta x}) \,. \tag{2}$$

Der Regressorvektor $\underline{x} \in \mathbb{R}^{(36\times 1)}$ setzt sich aus Messgrößen des letzten und vorletzten Abtastschrittes zusammen. Hierbei werden alle acht Messgrößen (Temperaturen und relativen Luftfeuchtigkeiten) außerhalb des Wandsegments als Eingänge aufgefasst. Die insgesamt zehn Messgrößen innerhalb der Wand (fünf Temperatur- und fünf Feuchtemesswerte) stellen die Ausgänge des Systems dar. Damit gestaltet sich $\underline{x}$ wie folgt

$$\underline{x}^T = \left(\underline{x}_1^T \,, \underline{x}_2^T \,, 1\right) \,, \tag{3}$$

wobei $\underline{x}_1$ und $\underline{x}_2$ mit $\gamma = 1,2$ darstellbar sind als

$$\begin{aligned} \underline{x}_\gamma^T = \Big(& \vartheta_1(k-\gamma), \varphi_1(k-\gamma), \vartheta_2(k-\gamma), \varphi_2(k-\gamma), \\ & \vartheta_{1,\mathrm{surf}}(k-\gamma), \varphi_{1,\mathrm{surf}}(k-\gamma), \vartheta_{2,\mathrm{surf}}(k-\gamma), \varphi_{2,\mathrm{surf}}(k-\gamma), \\ & \vartheta_{\mathrm{w},1}(k-\gamma), \ldots, \vartheta_{\mathrm{w},5}(k-\gamma), \varphi_{\mathrm{w},1}(k-\gamma), \ldots, \varphi_{\mathrm{w},5}(k-\gamma) \Big) \,. \end{aligned} \tag{4}$$

Der „1“-Regressor in $\underline{x}$ ist für die Schätzung eines Offsetparameters sinnvoll. Damit ergibt sich eine Modellstruktur für das lineare ZR-Modell mit acht Eingängen und $p = 10$ Ausgängen. Zustände und Ausgänge sind equivalent. Alle Elemente der Systemmatrix und der Steuermatrix werden als frei zu schätzende Parameter festgelegt.
Die Modellstrukturen für das TS- und KNN-Modell sind hingegen komplexer. Das TS-Modell bildet aus den $M = 10$ lokal-linearen ARX-Modellen und den $i = 1, \ldots, M$ Parametermatrizen $\theta_{\mathrm{TS},i} \in \mathbb{R}^{(p\times 36)}$ mit Hilfe der Fuzzy-Basisfunktionen $\phi_i(\underline{z})$ eine gewichtete Summe für alle $\underline{\hat{y}}_{\mathrm{TS,w}}(k) \in \mathbb{R}^{(p\times 1)}$ Ausgänge

$$\underline{\hat{y}}_{\mathrm{TS,w}}(k) = \sum_{i=1}^{M} \phi_i(\underline{z}) \underline{\theta}_{\mathrm{TS},i} \underline{x} \,. \tag{5}$$

Die Fuzzy-Basisfunktionen $\phi_i(\underline{z})$ haben ihren Wertebereich zwischen null und eins. Berechnet werden sie aus dem Mittel der Zugehörigkeitsgrade

$$\phi_i(\underline{z}) = \frac{\mu_i(\underline{z})}{\sum_{i=1}^{M} \mu_i(\underline{z})} \quad , \quad \sum_{i=1}^{M} \phi_i(\underline{z}) = 1 \, . \tag{6}$$

Alle $\mu_i(\underline{z})$ Zugehörigkeitsfunktionen (ZF) werden als Gaußglocken festgelegt. Mit dem Zentrum v und der Standardabweichung σ gilt für eine Gauss'sche ZF:

$$\mu_{\text{Gauss}}(x) = \exp\left(\frac{-(x-v)^2}{2\sigma^2}\right) \, . \tag{7}$$

Der Schedulingvektor bzw. Prämissenvektor $\underline{z}$ dient als Regelwerk, nach dem die einzelnen lokal-linearen Modelle aktiviert werden. $\underline{z}$ enthält alle Regressoren des letzten Abtastschrittes ($\underline{x}_1$, vgl. Gleichung (4)). Die Parameterschätzung des TS-Modells in Gleichung (5) geschieht auf Basis einer linearen, lokalen Kleinste-Quadrate-Schätzung.
Für das KNN werden $N_{\text{in}} = p = 10$ Neuronen für die Zwischenschicht bzw. die Ausgangsschicht festgelegt. Die Neuronen der Zwischenschicht werden jeweils über eine $f_{\text{tansig}}(\cdot)$-Aktivierungsfunktion gezündet, wobei die Neuronen der Ausgangsschicht jeweils eine lineare Abbildung darstellen. Zusätzlich wird ein Offsetvektor $\beta_{\text{out}} \in \mathbb{R}^{(p\times 1)}$ auf die Modellausgänge addiert:

$$\underline{\hat{y}}_{\text{KNN,w}}(k) = \underline{W}_{\text{out}} f_{\text{tansig}}\left(\underline{\theta}_{\text{KNN}} \underline{x}\right) + \underline{\beta}_{\text{out}} \, , \tag{8}$$

wobei die Parametermatrix $\theta_{\text{KNN}} \in \mathbb{R}^{(N_{\text{in}}\times 36)}$ die Regressoren entsprechend gewichtet und $\underline{W}_{\text{out}} \in \mathbb{R}^{(p\times N_{\text{in}})}$ die Ausgänge der Zwischenschicht nochmals gewichtet. Das Training des KNN-Modells benötigt einen nichtlinearen Parameteroptimierer. Hier wird der Levenberg-Marquardt-Algorithmus eingesetzt. Alle Modelle werden auf einen knapp 30-wöchigen Trainingsdatensatz trainiert und auf einen ungefähr einmonatigen Testdatensatz evaluiert. Beide Datensätze sind zehnminütig abgetastet. Nach dem erfolgreichen Training der Modelle werden diese auf dem Testdatensatz simuliert und deren Performanz ausgewertet. Eine Simulation heißt hier, dass nicht die realen Prozessausgänge - sprich die Messwerte der Temperaturen und relativen Luftfeuchtigkeiten in der Wand - in jedem Abtastschritt als Regressoren übergeben werden, sondern

die Modellausgänge selbst zurückgekoppelt werden. Die Bewertung der Perfomanz eines auf unbekannten Daten simulierten Modells ist aufschlussreich hinsichtlich der Generalisierungsfähigkeit. Soll ein Modell als Grundlage für eine Mehrschrittprädiktion dienen, kann die Simulation des Modells ebenfalls von Vorteil sein, wenn der Prädiktionshorizont genügend groß ist.
Die Rückkopplung der Modellausgänge hat zur Folge, dass keine ARX-Modellstruktur mehr vorliegt. Es handelt sich hierbei um eine Output-Error-Struktur (OE). Weil OE-Modelle allerdings weitaus schwieriger zu trainieren sind bzw. einen höheren Zeitaufwand zur Parameterschätzung benötigen, werden in der Systemidentifikation i. d. R. ARX-Modellstrukturen für das Training herangezogen und inkonsistente Parameterschätzungen hingenommen. Gleichwohl werden häufig gute Resultate mit dieser Methodik erzielt [4].

4 Ergebnisse

Die grafische Gegenüberstellung sämtlicher Prozessausgänge mit den entsprechenden Modellausgängen würde den Rahmen des Kurzbeitrags sprengen. Im Folgenden werden deshalb nur die relativen Feuchtigkeitsmesswerte des dritten Sensors ($\vartheta_{w,3}, \varphi_{w,3}$) im Wandsegment mit den simulierten Modellausgängen verglichen. Ein Vergleich der Modellgüten findet nur auf den Testdaten statt.

In Bild 2 ist der Zeitreihenverlauf der relativen Luftfeuchtigkeit $\varphi_{w,3}$ mitsamt der Modellausgänge dargestellt. Wie auch die Güte- bzw. Fehlermaße in der Tabelle 1 zeigen, schneidet das KNN am besten ab, gefolgt vom TS-Fuzzy-System. Das lineare Zustandsraummodell liefert nur eine dürftige Modellgüte, weil es die Nichtlinearitäten des Prozesses nicht abbilden kann. Allerdings ist deutlich zu erkennen, dass das ZR-Modell die Temperaturverläufe in einem ausreichenden Maße abbilden kann. Wesentlich schwieriger gestalten sich die Schätzungen der relativen Luftfeuchtigkeiten in der Wand. Die zugrundeliegenden nichtlinearen Feuchtespeicher- und Feuchtetransporteffekte (Sorptionsisotherme, Diffusionsvorgänge, etc.) fallen hier stark ins Gewicht.

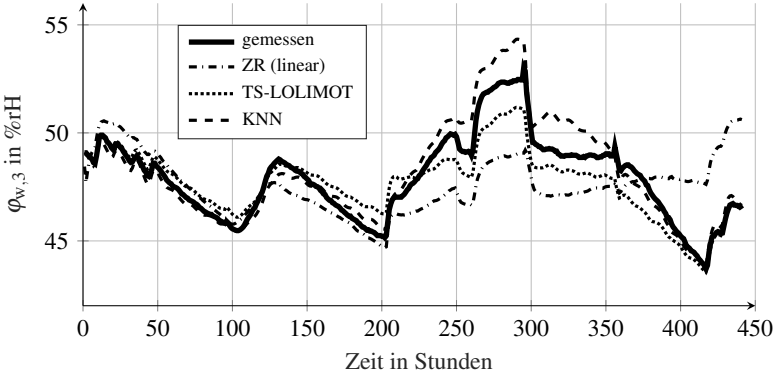


Bild 2: Gemessene relative Luftfeuchtigkeit $\varphi_{w,3}$ innerhalb der Wand im Vergleich mit den simulierten Modellausgängen.

5 Fazit und Ausblick

Das bessere Abschneiden des KNN kann auf ein mehrmaliges erneutes Trainieren zurückgeführt werden, was notwendig gewesen ist, weil die Güte des KNN nicht zufriedenstellend war. Dieses „Re-Training“ hat allerdings neben einem hohen Rechenaufwand auch eine hohe Rechendauer beansprucht. Das mit dem LOLIMOT-Algorithmus synthetisierte TS-Modell erforderte nur einen einzigen Trainingsdurchlauf und liefert zufriedenstellende Güte- und Fehlermaße (s. Tabelle 1). Aus Sicht der Autoren ist das TS-Modell auch aufgrund der lokalen Interpretierbarkeit vorzuziehen. Das lineare Zustandsraummodell reicht aufgrund fehlender Flexibilität und unzureichender Genauigkeit zur Modellierung des Wärme- und Feuchtetransports durch ein Wandsegment nicht aus.

Der Beitrag bietet einen Einblick in die laufenden Arbeiten am „HumFlow“-Projekt und zeigt, dass mit sehr einfach synthetisierten nichtlinearen Modellen bereits die Schätzung von Temperatur- und Feuchtigkeitsverteilungen in Wandsegmenten mit zufriedenstellender Genauigkeit ermöglicht werden kann. Aus bauphysikalischer Sicht wäre somit eine günstige und dauerhafte Schätzung des Wärme- und Feuchtigkeitseintrags in eine Wand vorstellbar, ohne die Bausubstanz zu beschädigen. Diese Anwendung ist insbesondere im Bereich denkmalgeschützter Altbauten interessant. Nach jetzigem Kenntnisstand der

Tabelle 1: Normalized Mean Squared Error (NMSE) sowie Best Fit Rate (BFR) nach [6] der Modelle für jede Messgröße. Die Fehler- und Gütemaße für $\varphi_{w,3}$ aus Bild 2 sind hervorgehoben.

	NMSE			BFR in %		
Größe	ZR	TS	KNN	ZR	TS	KNN
$\varphi_{w,1}$	0.3305	0.2047	0.0373	42.51	54.75	80.67
$\vartheta_{w,1}$	0.0318	0.0105	0.0020	82.18	89.77	95.56
$\varphi_{w,2}$	0.6350	0.1919	0.0569	20.31	56.19	76.14
$\vartheta_{w,2}$	0.0567	0.0076	0.0013	76.19	91.28	96.33
$\varphi_{w,3}$	**0.9620**	**0.1835**	**0.1454**	**1.92**	**57.16**	**61.87**
$\vartheta_{w,3}$	0.0155	0.0092	0.0020	87.54	90.40	95.51
$\varphi_{w,4}$	0.5557	0.1435	0.1908	25.46	62.12	56.32
$\vartheta_{w,4}$	0.2853	0.0130	0.0069	46.59	88.61	91.70
$\varphi_{w,5}$	0.1291	0.0576	0.0791	64.07	76.01	71.87
$\vartheta_{w,5}$	0.0596	0.0203	0.0104	75.59	85.74	89.78

Autoren ist ein vergleichbares Messverfahren nicht am Markt erhältlich. Aus einer regelungstechnischen Perspektive kann der vorgestellte Ansatz wichtige Vorhersagewerte für Raumlufttemperatur- und Raumluftfeuchtigkeitsregelungen bereitstellen und somit bestehende Raumklima-Systeme erweitern. In künftigen Vorhaben soll der TS-Fuzzy-Ansatz weiter verfolgt, aber der Prämissenraum mit Hilfe von Clustering-Algorithmen partitioniert werden [7].

Literatur

[1] A. Burmester. „Was ist Präventive Konservierung? Eine Einführung.“ *Grundlagen der Meßtechnik in der Präventiven Konservierung, Tagungsband.* S. 8–11. 2007

[2] A. Cavaterra, A. Böttcher und S. Lambeck. „The „HumFlow“ Project – Developing a minimal invasive measurement system for estimating energy and humidity transfer processes through building walls“. *13th REHVA World Congress CLIMA 2019*. Bucharest, Romania. 2019.

[3] T. Takagi und M. Sugeno. „Fuzzy identification of Systems and Its Applications to Modeling and Control“. *IEEE Transactions on Systems, Man and Cybernetic*. SMC-15, S. 116–132. 1985

[4] O. Nelles. „Nonlinear System Identification“. *Springer Berlin Heidelberg*. Zweite Auflage. 2020

[5] B. Hartmann, T. Ebert, T. Fischer, T.Belz, J. Kampmann und O. Nelles. „LMNtool – Toolbox zum automatischen Trainieren lokaler Modellnetze“. *22. Workshop Computational Intelligence, KIT Scientific Publishing*. 45, S. 341–355. 2012.

[6] A. Kroll und H. Schulte. „Benchmark problems for nonlinear system identification and control using Soft Computing methods: Need and overview“. *Applied Soft Computing*. 25, S. 496–513. 2014.

[7] A. Kroll „Computational Intelligence. Probleme, Methoden und technische Anwendungen“. *De Gruyter Studium*. Zweite Auflage. 2016

Least-Squares-Based Construction Algorithm for Oblique and Mixed Regression Trees

Marvin Schöne, Martin Kohlhase

Center for Applied Data Science Gütersloh, FH Bielefeld
Interaktion 1, 33619 Bielefeld
E-Mail: {marvin.schoene, martin.kohlhase}@fh-bielefeld.de

1 Introduction

As part of Smart Factories, industrial processes must be optimized in terms of efficiency, flexibility and process reliability. This is primarily achieved by *Advanced Analytics*, where data-driven models are used to analyze, describe and predict process behavior [1]. In this way, new process knowledge is gained and used, for instance, to adjust the operation mode of the process or reduce defects and quality problems [2]. These models need to fulfill certain requirements to be applicable in an industrial environment. In order to ensure a reliable operation of the plant and to enable optimization, they must be highly accurate. Furthermore, to gain process knowledge and confidence towards the operators and to fix model uncertainties more easily, they must be interpretable.

Decision Trees are a model class that can fulfill these requirements [3]. They are algorithmic constructed and represented as a top-down directional acyclic graph, consisting of decision nodes and terminal leaves. This graph is specified as a tree, which starts with a single decision node, the root, and ends up in multiple terminal leaves. To predict an output variable $\hat{y}$ (e.g. process behavior), an unlabeled sample $\mathbf{x} = [x_1 \ \dots \ x_M]$, which consists of M input variables x_m with $m \in \mathbb{N} \mid 1 \leq m \leq M$, must pass through the tree until a terminal leaf is reached. Each decision node contains a test function, that is applied at $\mathbf{x}$ and effects the path that $\mathbf{x}$ passes through the tree. The test functions are usually formulated as univariate splitting criteria $x_m \leq c$ for $c \in \mathbb{R}$

or $x_m \in \mathscr{B}$ for $\mathscr{B} \subset \mathscr{A}$ with a numerical threshold value c or a subset $\mathscr{B}$ of a merge of categorical attributes $\mathscr{A}$. Each terminal leaf contains a local model for prediction, that is only valid in a certain partition of the input space defined by trees' structure. Because of the rule-based structure, trees are human readable and easy to interpret. They can predict both numerical (*Regression Tree*) and categorical (*Classification Tree*) output variables. In addition, numerical and categorical input variables as well as missing input values can be handled and the importance of input variables can be measured [3, 4, 5].

However, especially for *Regression Trees* with univariate splitting criteria, there are limitations which can result in lower model accuracy and interpretability [4, 6]. Univariate splitting criteria depend on a single input variable, resulting in axis-orthogonal splits that limit model flexibility. Depending on the process function, this leads to lower model accuracy and, if simple local models are used, to a larger tree, which reduces interpretability [7]. To overcome these issues, multivariate splitting criteria $\sum_{m=1}^{M} \beta_m x_m \leq c$ with M coefficients β_m can be used to construct axis-oblique splits. The resulting tree is called *Oblique Regression Tree* or, if uni- and multivariate splitting criteria are used, *Mixed Regression Tree* [8]. The direction of an axis-oblique split has to adapt to the curvature of the function and is given by its coefficients β_m [8, 9, 10]. Furthermore, to maintain interpretability, to avoid overfitting and to overcome the curse of dimensionality, an efficient and generalized approach is necessary.

In this paper, a novel algorithm to construct *Mixed* and *Oblique Regression Trees* is presented. To determine an axis-oblique split direction adapted to the curvature in a partition, a first-order *Least Squares Regression* (LSR) model is used. This model is limited to significant input variables to describe this curvature, which maintains interpretability and generalization. The input variables are selected by analyzing the residuals of the resulting splitting model, which additionally weakens the curse of dimensionality. To construct the local models for prediction, stepwise regression is used. In Section 2, common algorithms for the construction of *Regression Trees* are explained. The proposed algorithm is presented in Section 3 and tested in Section 4 in an extensive experimental analysis with both synthetic and real-world data. Moreover, the results are compared with a state-of-the-art construction algorithm. At the end, Section 5 summarizes the paper and gives an overview on further research.

2 Construction Algorithms for Regression Trees

In this Section, the functionality of algorithms to construct *Regression Trees* is explained. The functionality is described in more detail for the common algorithms SUPPORT [11], CART [12], GUIDE [6] and PHDRT [10], which generate the eponymous trees.

Regression Trees are constructed by a *divide-and-conquer* strategy, which splits a set of N labeled samples $\mathscr{D} = \{\mathbf{X}, \mathbf{y}\}$ with $\mathbf{X} = [\mathbf{x}_1 \ \dots \ \mathbf{x}_N]^{\mathrm{T}}$ and the corresponding labels $\mathbf{y} = [y_1 \ \dots \ y_N]^{\mathrm{T}}$ recursively into smaller subsets $\mathscr{D}_k$ until a stopping criterion is reached. Each set of labeled samples $\mathscr{D}_k$ is represented as a node t_k with $k \in \{1, 2, \dots, K\}$ within the tree T, that consists of $|T| = K$ nodes [5].

The recursive splitting process to construct a tree is shown in Figure 1 and starts with the entire data set $\mathscr{D}_1 = \mathscr{D}$, represented by the root t_1. At first, in step a) a stopping rule for the node t_k is check. The stopping rule ensures that only meaningful splits of $\mathscr{D}_k$ are performed and the size of the tree is limited. A common stopping rule is a lower bound of the number of samples in a node, which is used in all four algorithms [5].

In step b) of Figure 1, the node t_k becomes a terminal leaf $\tilde{t}_k$ when splitting is stopped. Each terminal leaf represents a certain partition of the input space and contains a local model $\hat{y}_{\tilde{t}_k}(\mathbf{x}) \in \mathbb{R}$, that approximates the function within that partition. The local models of GUIDE and PHDRT are first-order multiple regression models and those of SUPPORT are third-order polynomial regression models [11, 6, 10]. Furthermore, SUPPORT combines all local models by a weighted average to create a continuous model output. For the local models of SUPPORT and PHDRT, all input variables are used. In contrast, GUIDE limits the local models to significant input variables using stepwise regression. The local models of CART are constant values, which are determined by the mean value of $\mathbf{y}$ in that partition [12].

The node t_k will be further split if the stopping rule is not fulfilled. For this purpose, in step c) of Figure 1 the input variable(s) x_m and the threshold value c or subset $\mathscr{B}$ to construct an uni- or multivariate splitting criterion are selected. The components x_m and c or $\mathscr{B}$ are selected in a way that the impurity is

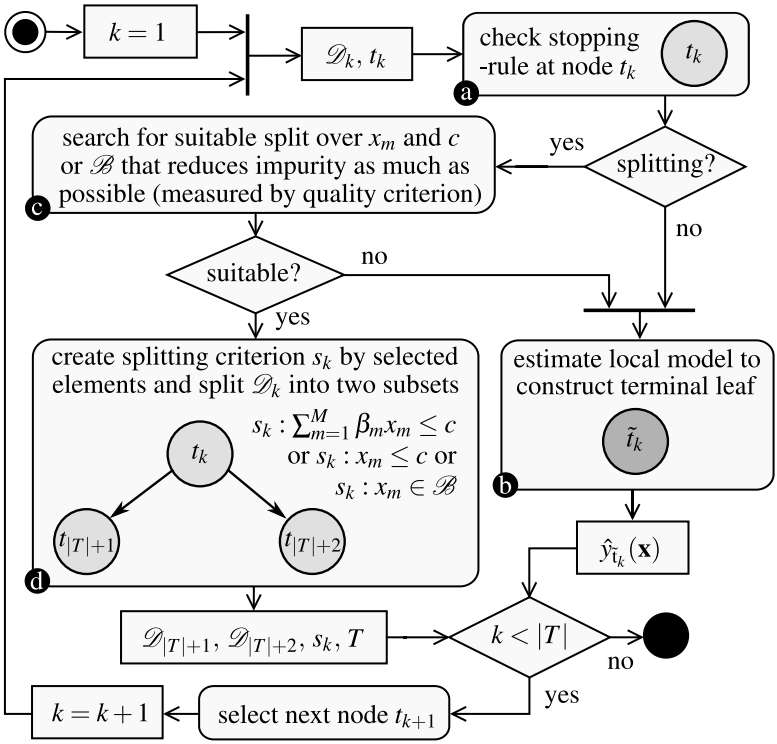


Figure 1: Recursive splitting process of a construction algorithm to create an unpruned tree T.

reduced as much as possible by the resulting splitting criterion [5, 13]. In *Regression Trees* the impurity is described e.g. by the degree of non-linearity in a partition, which is measured by a specific quality criterion. To measure the non-linearity in a partition, the error of a linear approximation can be used. To split a node, CART uses both uni- and multivariate criteria, which are either selected by a brute force method (univariate) or by a heuristic-based selection method (multivariate) called *Linear Combination Search Algorithm* [12]. In contrast, SUPPORT is limited to univariate splitting criteria and numerical input variables. For split selection, the samples within a node are approximated by multiple linear regression and divided into two sample groups with positive and negative signs of residuals. Due to statistical tests for differences in mean and variance between these two sample groups, dependencies between input

variables and residuals are analyzed to select the most significant x_m. The threshold value c is determined by averaging the means of the two groups of x_m [11]. The splitting method of GUIDE is similar to SUPPORTs' and differs in the use of a χ^2-independence tests, an interaction test between input variables and the ability to handle categorical input variables. Furthermore, due to a bootstrap-based bias correction, the significance of input variables is more comparable. Threshold values c are either determined by the median or mean of x_m and subsets $\mathscr{B}$ for a categorical input variable by a heuristic [6]. The splitting of PHDRT is limited to numerical input variables and multivariate criteria, which are determined by the first component of *principal Hessian Directions* (PHD). This component describes the direction in which the function to be approximated has the greatest curvature. To select a threshold value, the residuals of a multiple linear regression model are split into two partitions and approximated by one linear regression model each, using the first component of PHD as an input variable. The balance of the partitions is adjusted so that both linear models approximate the residuals with a similar standard deviation. Finally, the point of intersection is taken as the threshold value [10].

If a suitable split is determined, in d) of Figure 1 the splitting criterion s_k is constructed based on the selected components. In addition, the node is split into two nodes $t_{|T|+1}$ and $t_{|T|+2}$ containing the subsets $\mathscr{D}_{|T|+1}$ and $\mathscr{D}_{|T|+2}$. Recursive splitting is completed when no more nodes can be split [5, 13].

Further approaches to limit the size of the tree are pre- and postpruning techniques. Prepruning is closely related to the stopping rule and limits the size during the construction. In contrast, postpruning is applied after the construction and prunes an oversized tree backwards to a more generalized one. For this purpose, PHDRT stops splitting if the first component of PHD is insignificant, which is more similar to a stopping rule than to a prepruning technique [10]. The complexity of SUPPORT is limited by a prepruning technique, which uses cross validation to check whether a subtree can be created from t_k that significantly improves model quality [11]. Both CART and GUIDE limit the size by a postpruning technique called *Minimal Cost Complexity Pruning*, which uses cross validation to evaluate the generalization capabilities of different subtrees during the pruning [12, 6]. In the following, the proposed least-squares-based tree construction algorithm is explained in more detailed.

3 Least-Squares-Based Construction Algorithm

Model quality of *Regression Trees* can be improved by an extension to *Oblique* or *Mixed Regression Trees* using multivariate splitting criteria. To obtain the advantages of *Regression Trees*, a trade-off must be found between an increase in model accuracy and a loss of interpretability. This is a challenging task. In order to achieve this, the axis-oblique split direction of a multivariate splitting criterion must adapt to the function gradient $\nabla f(\mathbf{x})$ in a curvature area. Furthermore, the complexity of this criterion must be limited in such a way that interpretability is maintained. To determine this criterion with an appropriate computational effort, the curse of dimensionality must be weakened [6, 4, 8].

To construct the splitting criterion, the proposed algorithm uses the coefficients of a first-order LSR model $\hat{y}_{s_k}(\mathbf{x})$. The input variables of $\hat{y}_{s_k}(\mathbf{x})$ are selected in a way that $\hat{y}_{s_k}(\mathbf{x})$ adapts to the gradient $\nabla f(\mathbf{x})$ in the area of curvature in that partition. This is performed by a forward selection method (FSM) and depending on the number of selected input variables either an uni- or multivariate splitting criterion is constructed. A model with a single input variable constructs an univariate splitting criterion and a model with multiple input variables a multivariate splitting criterion. The split direction is defined by the contour lines, contour planes or contour hyperplanes (depending on M) of $\hat{y}_{s_k}(\mathbf{x})$ and by selecting a suitable output value of $\hat{y}_{s_k}(\mathbf{x})$ as a threshold, the position of the split is adjusted, which is explained in Subsection 3.1. The quality of the resulting split is measured by a criterion which analyzes the residuals of $\hat{y}_{s_k}(\mathbf{x})$ using a hinge function $h(\hat{y}_{s_k})$. Due to a reduction of the search space to the one-dimensional space of residuals, the FSM overcomes the curse of dimensionality. Furthermore, by limiting the number of significant input variables to a maximum of λ, interpretability and generalization is maintained. In Subsection 3.2 the FSM is presented in more detail.

If the sample size is too small or an insufficient improvement in model quality is achieved, splitting is stopped and a local model for prediction is determined. The local models are also determined by LSR and a FSM, which is explained in Subsection 3.3. To control the size of tree called *Least Squares Regression Tree* (LSRT), the technique *Minimal Cost Complexity Pruning* is used [12]. Finally, Subsection 3.4 shows the structure of LSRT using a practical example.

3.1 Axis-Oblique Split Direction

To get an axis-oblique split direction that is adapted to the gradient $\nabla f(\mathbf{x})$ in the area of curvature within a partition, a direction orthogonal to $\nabla f(\mathbf{x})$ has to be determined. This is achieved using a first-order LSR model

$$\hat{y}_{s_k}(\mathbf{x}) = \beta_0 + \sum_{m=1}^{M} \beta_m x_m \ . \tag{1}$$

If the non-linearity in a local area is not excessive, a direction orthogonal to $\nabla f(\mathbf{x})$ is obtained in this way. The coefficients of the model are determined by

$$\boldsymbol{\beta} = [\beta_0 \ \ldots \ \beta_M]^{\mathrm{T}} = (\mathrm{X}^{\mathrm{T}}\mathrm{X})^{-1}\mathrm{X}^{\mathrm{T}}\mathbf{y} \tag{2}$$

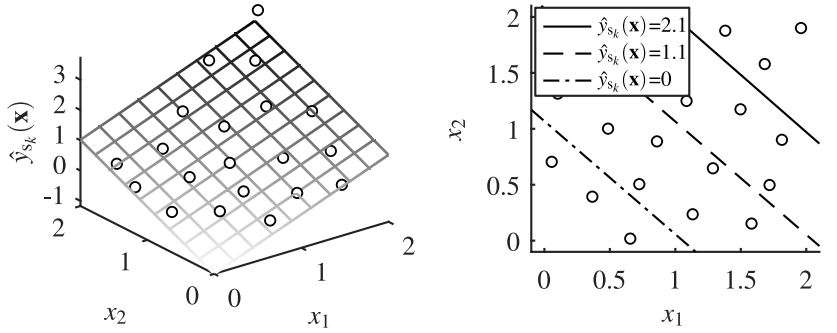
using the expanded $N \times (1+M)$ predictor matrix

$$\mathrm{X} = \begin{pmatrix} 1 & \mathbf{x}_1 \\ 1 & \mathbf{x}_2 \\ \vdots & \vdots \\ 1 & \mathbf{x}_N \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,M} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,M} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & x_{N,2} & \cdots & x_{N,M} \end{pmatrix}, \tag{3}$$

that consists of N samples $\mathbf{x}_n$ and an additional column of ones to determine the constant part β_0 of the LSR model [9].

Figure 2a shows a first-order LSR model $\hat{y}_{s_k}(\mathbf{x})$, that was trained on the 20 samples generated by a test function $f(\mathbf{x}) = x_1 x_2$. A contour line for the constant model output $\hat{y}_{s_k}(\mathbf{x}) = \alpha$ is formed by the various input combinations which result in α and runs as an axis-oblique border through the input space. The direction of the contour line results from the coefficients $[\beta_1 \ \ldots \ \beta_M]^{\mathrm{T}}$ and is orthogonal to $\nabla\hat{y}_{s_k}(\mathbf{x})$, which is presented in Figure 2b. This Figure shows three possible contour lines resulting from $\alpha \in \{0, 1.1, 2.1\}$ and $\hat{y}_{s_k}(\mathbf{x})$ in Figure 2a. The contour lines are splitting the input space into two partitions and by varying α, they are parallel shifted. This allows to determine a suitable threshold value for splitting. Finally, to construct the multivariate splitting criterion

$$\sum_{m=1}^{M} \beta_m x_m \leq \alpha - \beta_0 \ , \tag{4}$$



(a) Model output $\hat{y}_{s_k}(\mathbf{x})$ of a first-order LSR model (grid), trained on 20 samples generated from $f(\mathbf{x}) = x_1 x_2$.

(b) Three oblique splits that result from the contour lines of the model and are orthogonal to $\nabla\hat{y}_{s_k}(\mathbf{x})$.

Figure 2: Construction of axis-oblique splits using contour lines of a LSR model.

the constant part β_0 of the LSR model is subtracted. The threshold value $c = \alpha - \beta_0$ as well as the input variables to construct the LSR model are determined by a FSM, which is explained in the following Subsection.

3.2 Split Selection

In order to construct an uni- or multivariate splitting criterion with regard to the requirements of approximation capability, interpretability and generalization, suitable input variables for the LSR model and a suitable threshold value c must be selected. This is achieved by the FSM presented in Figure 3.

At first, the local optimal quality value $\gamma^* \in \mathbb{R}$, a maximum number of input variables $\lambda \in \mathbb{N} \mid 1 \leq \lambda \leq M$ to limit the complexity of the splitting criterion and the index m are initialized. Furthermore, the selected input variables $\mathbf{x}^* \in \mathbb{R}^i$ to construct the final splitting criterion are initialized with $\mathbf{x}^* = [1]$ for the constant part β_0. Each forward iteration performs a greedy search over all unselected input variables to extend an existing splitting criterion (resulting from $\mathbf{x}^*$) by a local optimal candidate input variable x_m. To identify the local optimal candidate during the greedy search, the quality of the splitting crite-

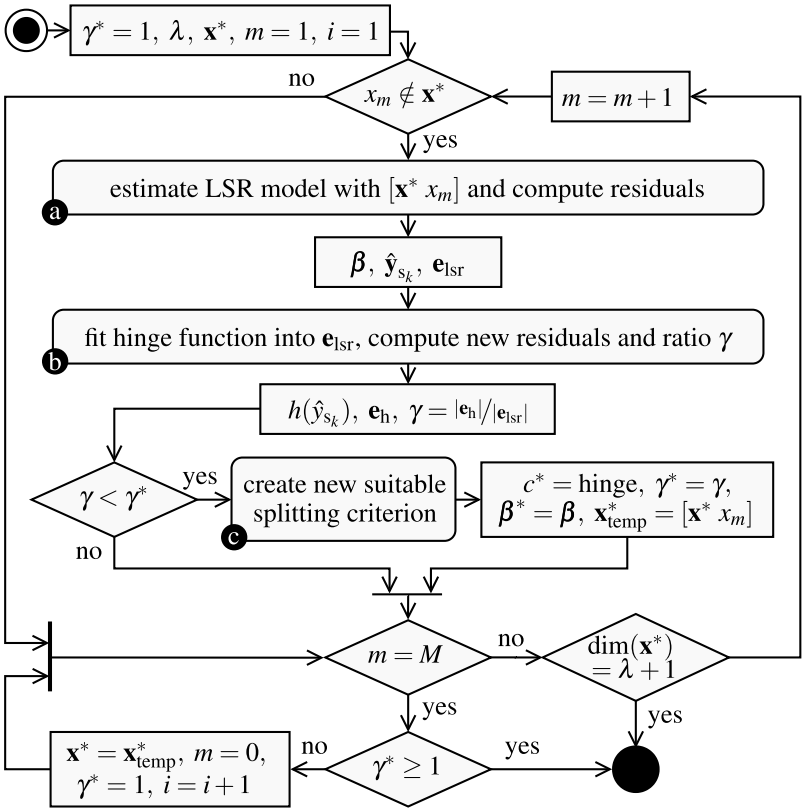


Figure 3: Activity diagram of the FSM to construct a suitable uni- or multivariate splitting criterion.

rion resulting from an extension with x_m is measured using a specific quality criterion.

To measure the quality which results from the extension, in step a) a LSR model $\hat{y}_{s_k}(\mathbf{x}^*, x_m)$ for splitting is determined. This model is constructed based on the previously selected input variables $\mathbf{x}^*$ and the candidate x_m. The residuals [9]

$$\mathbf{e}_{\text{lsr}} = [e_1\ \ldots\ e_N]^{\text{T}} = \mathbf{y} - \hat{\mathbf{y}}_{s_k} = [y_1 - \hat{y}_{s_k,1}\ \ldots\ y_N - \hat{y}_{s_k,N}]^{\text{T}} \tag{5}$$

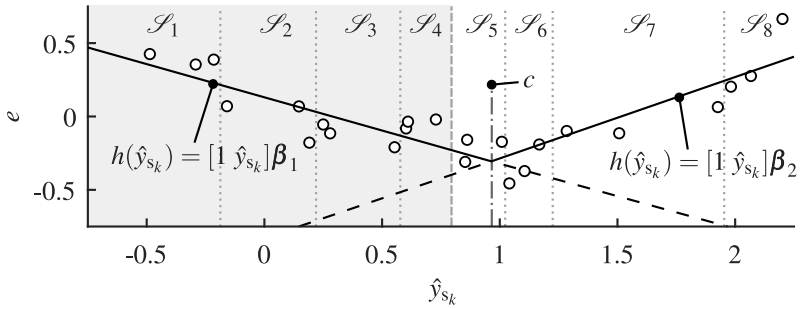


Figure 4: Example of the process to measure the quality of a splitting criterion. The residuals e of a LSR model $\hat{y}_{s_k}(\mathbf{x}^*)$ are approximated by a hinge function $h(\hat{y}_{s_k})$ (solid line), which consists of two LSR models trained by different subsets $\mathscr{S}_k$. The quality is measured by the improvements in approximation capability by $h(\hat{y}_{s_k})$.

of the model output $\hat{y}_{s_k,n} = \hat{y}_{s_k}(\mathbf{x}_n^*, x_{n,m})$ with $n \in \{1,\dots N\}$ are computed and in step b) approximated by a hinge function

$$h(\hat{y}_{s_k}) = \min([1\ \hat{y}_{s_k}]\boldsymbol{\beta}_1, [1\ \hat{y}_{s_k}]\boldsymbol{\beta}_2) \quad \text{or} \quad h(\hat{y}_{s_k}) = \max([1\ \hat{y}_{s_k}]\boldsymbol{\beta}_1, [1\ \hat{y}_{s_k}]\boldsymbol{\beta}_2)\ . \quad (6)$$

The hinge function consists of two local linear LSR models $\boldsymbol{\beta}_i = [\beta_0\ \beta_1]^{\mathrm{T}}\ \forall\ i \in \{1,2\}$, which are joined together by a hinge point [9, 14].

Figure 4 shows a hinge function (solid line), that was determined by $N = 24$ samples $\mathscr{E} = \{\hat{\mathbf{y}}_{s_k}, \mathbf{e}_{\mathrm{lsr}}\}$. To construct $h(\hat{y}_{s_k})$, the samples are ordered and segmented into K subsets $\mathscr{S}_k$, containing an equal number of samples. This segmentation helps to overcome the challenging effects of skewed data. The subsets resulting from the segmentation are grouped into $\mathscr{E}_{\mathrm{left}}$ and $\mathscr{E}_{\mathrm{right}}$ to determine $\boldsymbol{\beta}_1$ with $\mathscr{E}_{\mathrm{left}}$ and $\boldsymbol{\beta}_2$ with $\mathscr{E}_{\mathrm{right}}$. This is done iterative by changing the proportion of the groups until a stopping criterion is reached. In Figure 4, $K = 8$ subsets are used. First, the models are determined by two balanced groups $\mathscr{E}_{\mathrm{left}} \supset \{\mathscr{S}_1, \mathscr{S}_2, \mathscr{S}_3, \mathscr{S}_4\}$ and $\mathscr{E}_{\mathrm{right}} \supset \{\mathscr{S}_5, \mathscr{S}_6, \mathscr{S}_7, \mathscr{S}_8\}$, which is illustrated in Figure 4 by the gray and white area. If the resulting hinge point is out of a predefined area, e.g. outside the subgroups $\{\mathscr{S}_3, \dots, \mathscr{S}_{K-2}\}$, the balance of the groups is adjusted and two new LSR models are determined by the adjusted groups. Otherwise, the stopping criterion is fulfilled and the quality of the

splitting criterion, resulting from the candidate x_m and the hinge point as a threshold value [10], is measured.

The quality is measured by a specific criterion, which results in the quality value

$$\gamma = \frac{|\mathbf{e}_\mathrm{h}|}{|\mathbf{e}_\mathrm{lsr}|} \tag{7}$$

with the residuals $\mathbf{e}_\mathrm{h} = [e_1 - h(\hat{y}_{\mathrm{s}_k,1}) \ \ldots \ e_N - h(\hat{y}_{\mathrm{s}_k,N})]^\mathrm{T}$ of $h(\hat{y}_{\mathrm{s}_k})$. The quality criterion describes how much the non-linearity in a partition can be reduced along a certain direction by the splitting criterion. A decrease of non-linearity is indicated by $\gamma < 1$ and to fulfill the quality criterion within a forward iteration, the candidate x_m has to be selected in a way that γ is minimized to γ^*. Due to this minimization, the direction of non-linearity in a partition is identified which can most likely be approximated by two local linear models. Furthermore, this minimization effects that

- the orientation of $\hat{y}_{\mathrm{s}_k}(\mathbf{x}^*, x_m)$ becomes more similar to $\nabla f(\mathbf{x})$ in the area of curvature in that partition.
- the partition is split in an area near the curvature due to the hinge point.

Apart from these improvements, the computational effort to select a suitable splitting criterion increases linearly by $M \cdot \lambda$, whereby the curse of dimensionality is weakened.

If a new local optimal quality value is measured ($\gamma < \gamma^*$), in step c) of Figure 3 a new suitable splitting criterion is created. After the greedy search was applied ($m = M$) and no suitable candidate was identified ($\gamma \geq \gamma^*$), the whole FSM is stopped. Otherwise, $\mathbf{x}^*$ is extended by x_m that minimizes γ^* and, if complexity limitation isn't reached ($\dim(\mathbf{x}^*) < \lambda + 1$), the FSM is continued. The FSM is successfully completed if at least one suitable candidate has been identified. In this case $\boldsymbol{\beta}^*$ and c^* construct either an uni- or multivariate splitting criterion. If no input variables are selected by the FSM or if a minimal number of samples is reached, a local model for prediction is determined, which is described next.

3.3 Local Models

Similar to the requirements on the multivariate splitting criterion, the local models must be accurate, as interpretable as possible and well generalized. In order to achieve this, stepwise regression with another FSM is performed. Due to the bias-corrected *Akaike's Information Criterion*

$$\mathrm{AIC_C} = N \log \frac{\mathrm{RSS}}{N} + 2\tilde{M} + N + N\log(2\pi) + \frac{2(\tilde{M}+2)(\tilde{M}+3)}{N-(\tilde{M}+2)-1} \quad , \tag{8}$$

which is embedded into the FSM, both the model accuracy and complexity are taken into account during the forward selection [15]. The first term of (8) considers the model accuracy using the residual sum of squares

$$\mathrm{RSS} = \sum_{n=1}^{N} e_n^2 \tag{9}$$

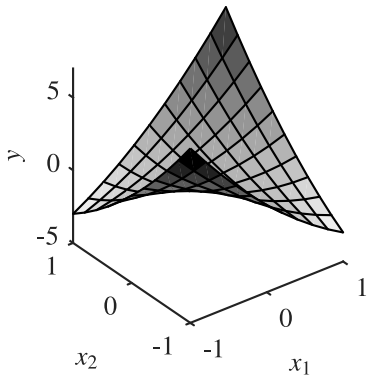
and the remaining terms are considering model complexity using the dimension of selected input variables $\tilde{M}$ and the number of samples N. $\mathrm{AIC_C}$ differs from the uncorrected criterion through the additional bias-correction term resulting from the last fraction, which leads to an improvement in accuracy for small data sets or high dimensional input spaces [15]. In this paper, the method is limited to a first-order LSR model

$$\hat{y}_{\tilde{\mathrm{I}}_k}(\tilde{\mathbf{x}}) = \beta_0 + \sum_{m \in \mathbb{N} \mid x_m \in \tilde{\mathbf{x}}} \beta_m x_m \quad , \tag{10}$$

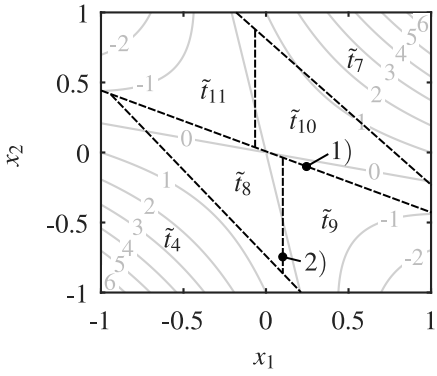
which is constructed by the selected input variables $\tilde{\mathbf{x}}$ with $\dim(\tilde{\mathbf{x}}) = \tilde{M}$. This is illustrated next using a practical example.

3.4 Tree Structure

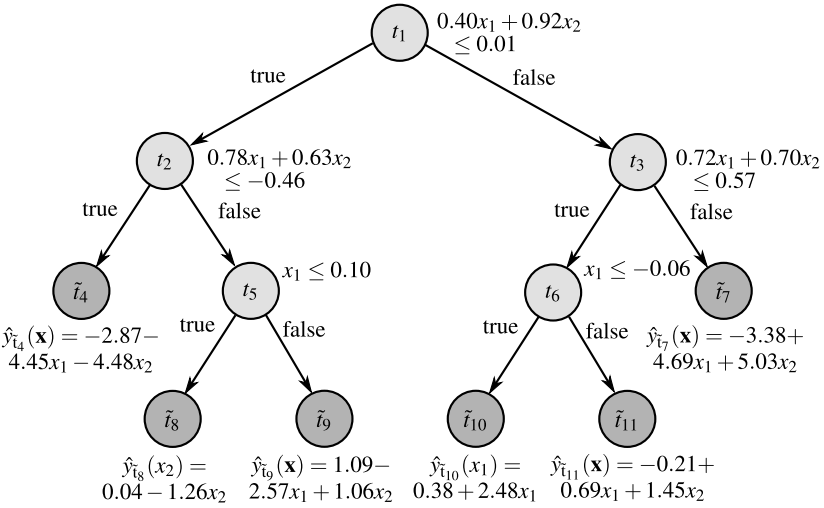
The proposed algorithm constructs a binary tree, called LSRT, using both uni- and multivariate splitting criteria. Figure 5c shows a *Mixed Regression Tree* that approximates the function $f(\mathbf{x}) = 5x_1x_2 + x_1^2 + x_2^2$ which is presented in Figure 5a.



(a) Test function $f(\mathbf{x}) = 5x_1x_2 + x_1^2 + x_2^2$ on which LSRT was trained using 50 samples generated from $f(\mathbf{x})$.



(b) Partitions resulting from LSRT and represented by $\tilde{t}_k$. The splits (black dashed lines) are adapted to functions' gray contour lines.



(c) LSRT consisting of uni- and multivariate splitting criteria (right to t_k) and local linear models $\hat{y}_{\tilde{t}_k}$ (below $\tilde{t}_k$).

Figure 5: *Mixed Regression Tree* which is called LSRT and constructed by the proposed algorithm.

The tree was trained on 50 samples and to cover the whole input space, these samples and generated from an optimized *Latin Hypercube Design* [16]. The tree consists of five decision nodes $t_k \; \forall \; k \in \{1,2,3,5,6\}$ with an uni- or multivariate splitting criterion displayed to the right of the node and six terminal leaves $\tilde{t}_k \; \forall \; k \in \{4,8,9,10,11,7\}$ with a local model $\hat{y}_{\tilde{t}_k}(\tilde{\mathbf{x}})$.

Figure 5b presents the partitions of the input space resulting from the splitting criteria, which are drawn by black dashed lines. For instance, the axis-oblique split 1) results from the multivariate splitting criterion of the root t_1 and the axis-orthogonal split 2) results from the univariate splitting criterion of node t_5. Each partition contains a first-order LSR model $\hat{y}_{\tilde{t}_k}(\tilde{\mathbf{x}})$, which is only valid in its partition. It can be recognized that the direction of the splits are adapted to the gray contour lines of $f(\mathbf{x})$, which indicates that the algorithm determines suitable splits. To investigate the performance of the algorithm in more detail, in the following an extensive experimental analysis is performed.

4 Experimental Analysis

In order to analyze the proposed algorithm with regard to accuracy and model complexity, the algorithm is tested in Subsection 4.1 on synthetic data and in Subsection 4.2 on real-world data. Furthermore, to compare the performance to state-of-the-art construction algorithms for *Regression Trees*, LSRT is compared to GUIDE, which is determined by a toolbox [17].

To obtain comparable results among LSRT and GUIDE, the trees are constructed based on similar hyperparamters. Both trees are pruned by the same method using the same hyperparameters and splitting is stopped by a lower bound of six samples. In addition, the local models are both determined using a FSM. To ensure the interpretability of LSRT, multivariate splitting criteria are limited to $\lambda = 2$. Although both trees are constructed on similar hyperparameters, the size of the pruned tree can vary between LSRT and GUIDE. For a comparison without the restriction of the tree size, a third tree LSRT_{adj} is considered that is pruned to the same size as GUIDE.

Test results are evaluated by the root mean squared error $\bar{E}$ and the tree size $|\bar{T}|$, measured by the number of nodes within the tree T. To generate meaningful results, $\bar{E}$ and $|\bar{T}|$ are averaged over 150 runs.

4.1 Synthetic Data

To generate synthetic data, a common test function from [18] is extended to

$$f(\mathbf{x}) = \sum_{i=1}^{I} \frac{10}{i^2} \sin(\pi x_{5i-4} x_{5i-3}) + \frac{20}{i^2}(x_{5i-2} - 0.5)^2 + \frac{10}{i^2} x_{5i-1} + \frac{5}{i^2} x_{5i} \ , \quad (11)$$

so that the dimensionality of $\mathbf{x} \in \mathbb{R}^M \mid M = I \cdot 5$ can be varied in discrete steps $I \in \mathbb{N}$ of five. In this way, the influence of dimensionality can be analyzed. The input space is limited to $0 \leq x_m \leq 1$ and the N samples for training are generated from an optimized *Latin Hypercube Design* [16] to fill the whole input space. To analyze the influence of noise, white Gaussian noise ε with mean $\bar{\varepsilon} = 0$ and variance $\sigma^2 = 0.5$ is added to $f(\mathbf{x})$. Furthermore, M_n noisy input variables without a dependence on $f(\mathbf{x})$ are constructed using ε with $\bar{\varepsilon} = 0.5$ and $\sigma^2 = 0.1$. In each of the 150 runs, 1500 samples are randomly generated from (11) for testing. Table 1 shows the experimental results on eight synthetic data sets.

For all data sets, LSRT and LSRT_{adj} are more accurate than GUIDE. Compared to GUIDE, the error of LSRT is reduced by 18.8% and the error of LSRT_{adj}, which has the same complexity as GUIDE, is reduced by 13.6%. Furthermore, it can be recognized that the difference in error between GUIDE and LSRT_{adj} is slight for data set $\{50,5,0,0\}$, whereas the difference for $\{300,5,0,0\}$ is significant (20.2%). These differences result from the mixed tree structure of LSRT. Due to the properties of $f(\mathbf{x})$, axis-oblique splits occur in deeper layers of LSRT. With an average tree size of $|\bar{T}| = 2.2$ in data set $\{50,5,0,0\}$, LSRT_{adj} consists only of univariate splits. In contrast, LSRT_{adj} with an average size of $|\bar{T}| = 15.6$ consists of several axis-oblique splits, which demonstrates the improvements resulting from the oblique splits.

Table 1 shows that an influence of noise can be handled well by the proposed algorithm. The results of LSRT for the noisy data sets $\{300,5,0,0.5\}$ and

Table 1: Experimental results on synthetic data for two different trees LSRT and GUIDE. LSRT_{abj} is an complexity adjusted version of LSRT with the same size as GUIDE. The elements in the brackets (left column) indicate the properties of the eight data sets. The best results for model complexity $|\bar{T}|$ and test error $\bar{E}$ are in bold print.

Settings	LSRT		GUIDE		LSRT_{adj}
$\{N, M, M_{\text{n}}, \sigma_\varepsilon^2\}$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\bar{E} \pm \sigma$
$\{50,5,0,0\}$	3.9	2.13±0.19	2.2	2.35±0.20	2.31±0.30
$\{100,5,0,0\}$	7.1	1.73±0.27	4.1	2.07±0.16	1.94±0.20
$\{200,5,0,0\}$	11.0	1.23±0.14	11.6	1.52±0.15	1.23±0.17
$\{300,5,0,0\}$	11.4	1.09±0.26	15.6	1.19±0.26	0.95±0.14
$\{300,10,0,0\}$	11.0	1.42±0.17	8.2	1.92±0.32	1.68±0.32
$\{300,15,0,0\}$	9.5	1.67±0.19	4.1	2.17±0.13	2.08±0.21
$\{300,5,5,0\}$	11.6	1.11±0.18	13.1	1.64±0.41	1.24±0.36
$\{300,5,0,0.5\}$	11.9	1.10±0.23	13.8	1.30±0.22	1.03±0.13

$\{300,5,5,0\}$ are similar to the results of $\{300,5,0,0\}$. In addition, the error of LSRT resulting from $\{300,5,5,0\}$ is 32.3% lower than the error of GUIDE. An influence of dimensionality cannot be evaluated clearly. Due to an increase of the function values by the addition of further terms, $\bar{E}$ is equally increased. Based on the error reduction (23.0%) between LSRT and GUIDE for $\{300,15,0,0\}$, it can be expected that the curse of dimensions is weakened.

In four out of eight data sets, both LSRT and GUIDE have the lowest complexity, which means that both trees achieve comparable results in interpretability. A comparison between LSRT and LSRT_{adj} shows that for $\{300,5,0,0\}$ and $\{300,5,0,0.5\}$ tree size was penalized too much by the pruning method. This can be recognized by the lower test error of LSRT_{adj}. In the following, LSRT is tested in a more challenging task using real-world data.

4.2 Real-World Data

In contrast to synthetic data, real-world data provides more challenging tasks for data driven-models due to incomplete samples, outliers and skewed data. To analyze and compare the proposed algorithm with regard to a more challenging task, four different real-world data sets Baseball, Tecator, CPU and Redwine

Table 2: Experimental results on real-word data for two different trees LSRT and GUIDE. LSRT_{abj} is an complexity adjusted version of LSRT with the same size as GUIDE. The real-world data sets Baseball and CPU are scaled by 10^{-2} and 10^{-1}.

	LSRT		GUIDE		LSRT_{adj}
Data sets	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\lvert\bar{T}\rvert$	$\bar{E} \pm \sigma$	$\bar{E} \pm \sigma$
Baseball	3.1	2.273±0.118	3.0	2.346±0.088	2.255±0.101
Tecator	3.6	0.903±0.057	1.6	0.926±0.065	0.858±0.040
CPU	5.2	5.214±0.800	4.3	5.129±0.514	5.471±1.157
Redwine	3.3	0.645±0.007	1.8	0.653±0.007	0.652±0.007

with a dimension from 6 to 24 input variables and a size from 209 to 1599 samples are used [19, 20, 21, 22]. Because of LSRTs' limitation to numerical input variables, categorical input variables are excluded from the data sets. In addition, the Tecator data set is reduced by 100 input variables containing the absorbance spectrum. Because of the small sample size of CPU with $N = 209$ and Tecator with $N = 240$ the analysis is performed by a k-fold cross validation. Within each run and each data set, k trees of each type are trained by $k-1$ varying data subsets, which predict the remaining data subset [9]. For this purpose, CPU and Tecator are analyzed by $k = 10$, Baseball by $k = 5$ and Redwine by $k = 2$. The results are shown in Table 2.

Compared to the results on synthetic data, the improvements by the proposed algorithm are less significant. On average, the error of LSRT is 1.3% and the error of LSRT_{abj} is 1.2% less than that of GUIDE. Furthermore, GUIDE is less complex for each data set. Nevertheless, due to a maximum size of $|\bar{T}| = 5.2$ there are no limitations in interpretability.

Improvements of an axis-oblique structure are only apparent at the Baseball data set. The root of LSRT is split by a multivariate criterion, which reduces the error versus GUIDE by 3.9%. For Tecator, the error of LSRT_{abj} is 7.3% less than the error of GUIDE. Because of the small tree size ($|\bar{T}| = 1.6$), the error reduction only result from the FSM to determine the local model. This can be explained by distinct linear dependencies, which can be well fitted with a single multiple regression model. A multivariate split of the root, which is performed by LSRT ($|\bar{T}| = 3.6$), results in an increase of error. Due to Tecators'

dimension of 24, this could be caused either by a wrong split selection or by a limitation $\lambda = 2$ of two input variables for splitting. Compared to GUIDE, LSRT performs slightly worse on the CPU data set. Five out of six input variables of CPU are discrete, so the segmentation process of the split selection (compare Subsection 3.2) does not work correctly anymore. The Redwine data set contains much noise and functional dependencies are low. Therefore, an increase in accuracy of LSRT may result from an increase in complexity.

5 Conclusion

To solve the issues of *Regression Trees* with respect to model accuracy, their structure can be extended to *Mixed* or *Oblique Regression Trees* using axis-oblique splits. In order to obtain the advantages of *Regression Trees* when using axis-oblique splits, a trade-off between an increase in model accuracy and a loss of interpretability must be found. In this paper, a novel construction algorithm for *Mixed* and *Oblique Regression Trees* was presented. The direction for splitting within a partition is determined by a first-order LSR model $\hat{y}_{s_k}(\mathbf{x}^*)$, which is limited to a maximum number of significant input variables $\mathbf{x}^*$ due to a forward selection method. Depending on the number of selected input variables, this direction can be either axis-orthogonal or axis-oblique. The selection of $\hat{y}_{s_k}(\mathbf{x}^*)$ is based on a quality criterion, which is determined by an approximation of candidate models' residuals using a hinge function. In this way, a split direction adapted to functions' curvature within the partition is obtained and the resulting one-dimensional search space for the selection weakens the curse of dimensionality. By the limitation to significant input variables, interpretability and generalization is maintained. The proposed algorithm was tested in an extensive experimental analysis using synthetic and real-world data and compared with a state-of-the-art algorithm for *Regression Trees*. Especially for synthetic data, significant improvements in model accuracy are achieved, resulting in lower test error compared to the state-of-the-art algorithm. The improvements for real-world data were less significant due to effects like discrete input values and partially unsuitable data sets. To obtain meaningful results on real-world data, further experiments are necessary.

For further improvements on real-world data, statistical test and an approach to consider categorical input variables and discrete values can be included into the split selection process. To improve the over-all performance of the proposed algorithm, the hinge function should be determined by a common algorithm in combination with a bootstrapping process. Stepwise selection combined with a complexity penalty for $\hat{y}_{s_k}(\mathbf{x}^*)$ could also provide further improvements in split selection. Furthermore, to increase model flexibility by curved splits, $\hat{y}_{s_k}(\mathbf{x}^*)$ could be extended to a higher order. Additionally, trees' structure can be extended to a neuro-fuzzy structure and to decrease the computational effort of the proposed algorithm, an efficient technique for prepruning is necessary.

Acknowledgements

This work was supported by the EFRE-NRW funding programme "Forschungsinfrastrukturen" (grant no. 34.EFRE-0300180).

References

[1] Brenno Menezes, Jeffrey Kelly, Adriano Leal and Galo Carrillo Le Roux. "Predictive, Prescriptive and Detective Analytics for Smart Manufacturing in the Information Age". In: *IFAC-PapersOnLine*, 52:568–573. 2019.

[2] Chao Shang and Fengqi You. "Data Analytics and Machine Learning for Smart Process Manufacturing: Recent Advances and Perspectives in the Big Data Era". In: *Engineering*, 5(6):1010 – 1016. 2019.

[3] Puran Tewari, Kapil Mittal and Dinesh Khanduja. "An Insight into Decision Tree Analysis". In: *World Wide Journal of Multidisciplinary Research and Development*, 3:111–115. 2017.

[4] Wei-Yin Loh. "Fifty Years of Classification and Regression Trees". In: *International Statistical Review*, 82. 2014.

[5] Lior Rokach and Oded Maimon. “Data Mining With Decision Trees: Theory and Applications”. World Scientific Publishing Co., Inc., USA, 2nd edition. 2014.

[6] W.-Y. Loh. “Regression Trees With Unbiased Variable Selection and Interaction Detection”. In: *Statistica Sinica*, 12:361–386. 2002.

[7] Carla E. Brodley and Paul E. Utgoff. “Multivariate Decision Trees”. In: *Machine Learning*, 19(1):45–77. 1995.

[8] Marek Kretowski. “Evolutionary Decision Trees in Large-Scale Data Mining”. Springer International Publishing. 2019.

[9] Oliver Nelles. “Nonlinear System Identification”. Springer Berlin Heidelberg. 2001.

[10] Ker-Chau Li, Heng-Hui Lue and Chun-houh Chen. “Interactive Tree-Structured Regression via Principal Hessian Directions”. In: *Journal of the American Statistical Association*, 95:547–560. 2000.

[11] P. Chaudhuri, M.-C. Huang, W.-Y. Loh and R. Yao. “Piecewise-Polynomial Regression Trees”. In: *Statistica Sinica*, 4:143–167. 1994.

[12] Leo Breiman, Jerome Friedman, Charles J. Stone and R.A. Olshen. “Classification and Regression Trees”. Chapman and Hall/CRC, New York. 1984.

[13] Ethem Alpaydin. “Introduction to Machine Learning”. The MIT Press, 2nd edition. 2010.

[14] Tamás Kenesei and János Abonyi. “Hinging hyperplane based regression tree identified by fuzzy clustering and its application”. In: *Applied Soft Computing*, 13(2):782–792. 2013.

[15] Charles Lindsey and Simon Sheather. “Variable Selection in Linear Regression”. In: *The Stata Journal*, 10(4):650–669. 2010.

[16] Tobias Ebert, Torsten Fischer, Julian Belz, Tim Oliver Heinz, Geritt Kampmann and Oliver Nelles. “Extended Deterministic Local Search Algorithm for Maximin Latin Hypercube Designs”. In: *IEEE Symposium Series on Computational Intelligence*. 2015.

[17] Wei-Yin Loh. “GUIDE Classification and Regression Trees and Forests (version 35.2)”, http://pages.stat.wisc.edu/ loh/guide.html, Last accessed 25 September 2020.

[18] Jerome H. Friedman, Eric Grosse and Werner Stuetzle. “Multidimensional Additive Spline Approximation”. In: *SIAM Journal on Scientific and Statistical Computing*, 4(2):291–301. 1983.

[19] Joaquin Vanschoren, “OpenML baseball-hitter”, https://www.openml.org/d/525, Last accessed 25 September 2020.

[20] Joaquin Vanschoren, “OpenML tecator”, https://www.openml.org/d/505, Last accessed 25 September 2020.

[21] Jan van Rijn. “OpenML machine_cpu”, https://www.openml.org/d/230, Last accessed 25 September 2020.

[22] Pieter Gijsbers, “OpenML wine-quality-red”, https://www.openml.org/d/40691, Last accessed 25 September 2020.

A new criterion for Latin hypercube optimization

Timm J. Peter, Oliver Nelles

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

Abstract

In this contribution, a new approach for optimizing LH designs based on the estimation and evaluation of pdfs is presented. The proposed algorithm minimizes the mean absolute error between the estimated pdf of the LH design, evaluated solely on its data points, and the uniform distribution. To validate the functionality of the new approach, it is compared to other state-of-the-art methods to create space-filling designs. The methods are compared using the KL divergence of the resulting datasets and the uniform distribution, as well as the resulting computation times for various dimensions and number of data points. Overall, the KL divergence performance of the new approach is outstanding, but expensive in terms of the computational demand. An additional benefit of the proposed approach is that it allows higher flexibility for DoE desgins. For example, it can be extended to approach any arbitrary point distribution, not just uniform, and may be suitable for the integration of constrains.

1 Introduction

The training data point distribution in the input space, also called the experimental design, is an important influencing factor regarding the quality of data-driven models. For this reason, an assessment of the quality of the design of

experiments (DoE) is important before measurements take place. If no prior knowledge about a system exists, uniformly distributed input data should be used [2]. Furthermore, besides being space-filling in the original input space, two other properties are advantageous and concern the projection of the data onto the individual input axes: a) one-dimensional uniform distribution on each axis (1Duni) and b) non-collapsing design which means that the projected data points stay distinct in their 1D projections.

A well-known strategy used for the experimental design that places points on a grid while avoiding the curse of dimensionality is the Latin hypercube (LH) design. LH designs fulfill the 1Duni property, but do not inherently provide a uniform data distribution. Therefore, several loss functions were proposed, which try to rate uniformity of the data point distribution, such as maximin or ϕ_p [1]. These loss functions focus on the data point pair with the smallest distance which makes them suitable for optimization purposes, e.g. for the optimization of LH designs as in [3]. These approaches drive all points away from each other during optimization, thereby creating a uniform point distribution. On the downside, these local approaches are structurally not able to rate the overall distribution quality, but can be utilized for optimization. It can be carried out by local search, e.g., based on point exchanges [4, 3] and global optimization methods, e.g., particle swarm optimization [10], simulated annealing [11] or evolutionary algorithms [12].

This contribution proposes a new approach to optimize the data point distribution of an LH design based on the use of probability density function (pdf) estimation. Thus, contrary to the above-mentioned approaches, the proposed method can rate the overall distribution quality. The approach estimates the pdf for all data points. In this contribution a local search method that utilizes point exchanges is employed. In each iteration the algorithm exchanges a coordinate of a point pair, based on the difference of the estimated pdf and the uniform distribution. This procedure is compared to other state-of-the-art approaches for space-filling designs. The quality of the different approaches is evaluated using the Kullback-Leibler (KL) divergence.

The contribution is structured as follows. Section 2 introduces the most important methods to calculate space-filling designs, namely (i) Sobol sequences

and (ii) LH designs. Additionally, since LH designs are not necessarily space-filling, optimization strategies to achieve space-filling LH designs are introduced. Section 3 provides the concept of kernel density estimation and a special evaluation strategy for pdfs, before in Sec. 4 the new density-based LH design optimization strategy is presented. In Sec. 5 the performance of the algorithm is analyzed and compared to other state-of-the-art methods for space-filling designs. Finally, a conclusion is given.

2 Space-filling designs

The two main properties of a design of experiments (DoE), if no prior knowledge about the process is available, are the (i) 1Duni property and (ii) the space-filling property. This section gives a quick overview of two important methods to achieve space-filling designs.

2.1 Sobol sequences

The most commonly used strategy to create space-filling datasets is introduced in [9]. The so-called Sobol sequences are low-discrepency sequences, which are uniformly distributed for $N = 2^x$ data points with $x \in \mathbb{N}$. The foundation to create Sobol sequences is the successive subdivision of each dimension in halves and the reordering of the coordinates in each dimension. This procedure is computationally very cheap, even for high numbers of data points and high dimensions and thus used commonly. On the downside, Sobol sequences do not fulfill the 1Duni property.

2.2 Latin hypercubes

Originally being used in the field of computer experiments, LH designs fulfill the 1Duni property due to their structure, but they do not fulfill the space-filling property intrinsically. Therefore, a subsequent optimization of the LH design has to be performed, if a space-filling dataset is desired.

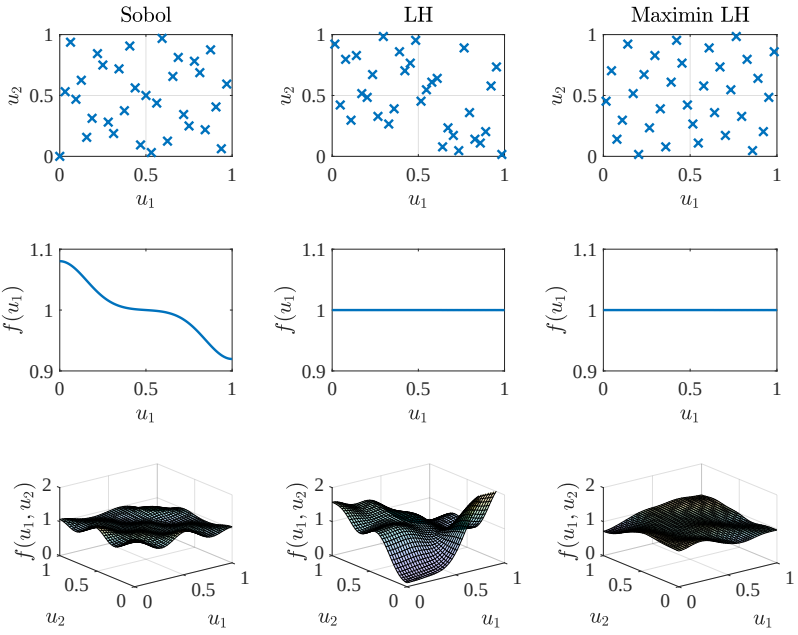


Figure 1: The first row of subplots visualizes the point distribution, the second row the 1Duni property and the last row the space-filling property for $N = 32$ data points and $n = 2$.

To create an LH design, the number of samples N and the dimension n have to be fixed by the user. Each input $u_i, i = 1, 2, ..., n$ is partitioned into N levels. Thus, for N samples and n input dimensions, N^n grid points are constructed. Out of these N^n grid points, N are occupied by data points. Here, each level of the N levels is occupied once, establishing the non-collapsing property of the LH design.

2.3 Optimization of Latin hypercubes using deterministic local seach

This subsection focuses on the optimization of LH designs using the deterministic local search (DLS) [4] and the extended deterministic local search (EDLS) [3] algorithms.

The procedure of the DLS Algorithm is as follows. A dataset, here an LH design, with N data points in n dimensions is given. The goal is to maximize the distance of the two nearest neighbors in the dataset. To achieve this, the first step is to calculate the nearest neighbor distance between all data points. The two data points with minimal distance are labeled as the critical points, the rest of the dataset is labeled as potential swap partners. Now the coordinates of a point pair are swapped in one dimension. This point pair consisting of one out of the two critical points and one swap partner point. If the coordinate swap reduces the minimum nearest neighbor distance, the procedure starts again with the first step. If not, the algorithm tries all possible dimensions and all potential swap partners. If no further improvements can be achieved, the algorithm is terminated. This procedure effectively drives all data points away from each other, thereby creating a space-filling design. At the same time, the 1Duni property is preserved.

The DLS algorithm has different positive properties. On the one hand, it preserves the LH design, if the initial set is also an LH design. On the other hand, the algorithm can optimize any collapsing design as well. It is deterministic, thus the results are reproducible and it ensures improvement in each iteration.

The idea of the DLS algorithm is amplified in [3] to the extended deterministic local search (EDLS). A small modification of the DLS leads to a vast change in the algorithm's behavior. Instead of considering only the *smallest* nearest neighbor distance, all neighbor distances are inspected and maximized. This has the effect of even more homogeneous data distributions in comparison to the DLS. On the downside, the computing time increases considerably.

3 Estimation and evaluation of probability density functions

For a n-dimensional dataset consisting of the data points $\underline{u}(i) = [u_1(i)\ \ u_2(i)\ \ \cdots\ \ u_n(i)]^T,\ \ i = 1,2,\ldots,N$ summarized in a matrix $\underline{U} = [\underline{u}(1)\ \underline{u}(2),...,\underline{u}(N)]$ a density estimator can be used to estimate its pdf $q(\underline{u})$. The pdf is based on placing a kernel on each data point. This procedure is known as kernel density or Parzen estimator. In areas, where two or more

kernels overlap, the corresponding values are added. In most cases, a Gaussian kernel is used. For other kernel types, see [6, 8].

The pdf is estimated by

$$\hat{p}(\underline{u}) = \frac{1}{N}\sum_{i=1}^{N} K(\underline{u},\underline{u}(i)),\ K(\underline{u},\underline{u}(i)) = \frac{\exp\left(-\frac{1}{2}[\underline{u}-\underline{u}(i)]^T\Sigma^{-1}[\underline{u}-\underline{u}(i)]\right)}{\sqrt{(2\pi)^n|\Sigma|}} \quad (1)$$

typrically with a diagonal covariance matrix $\Sigma = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ of n standard deviations for each dimension. Estimator (1) can be interpreted (i) as a sum of N kernels with height one normalized by their integral sum or (ii) as an average of N normal distributions.

One commonly used strategy to determine standard deviations is to use Silverman‘s rule-of-thumb [8]

$$\sigma_i = \sigma_{ui}\left(\frac{4}{n+2}\right)^{\frac{1}{n+4}} N^{-\frac{1}{n+4}} \quad (2)$$

with the standard deviation of the data σ_{ui} in dimension i. For alternative approaches to determine standard deviations, see [7].

Typically, Monte Carlo sampling is used to evaluate pdf estimates of datasets. This is a computationally demanding task since the number of sampling points has to be high “enough”. In a recent publication, a different, less time-consuming strategy to estimate and evaluate pdfs was proposed [5]. Here, a pdf is estimated using kernel density estimation and evaluated solely on the data points of the dataset itself, to select a subset of an original dataset. Compared to Monte Carlo sampling the computational effort decreases dramatically, without decreasing the subset selection performance.

In terms of calculation, this strategy can be visualized using a symmetric $N \times N$ matrix. Figure 2 shows the combinations of each point of a dataset $\underline{u}$ used for evaluation (rows) with the kernels of the estimated density (columns). The dataset’s pdf value evaluated at one data point can be calculated by taking the mean of the associated row of this calculation matrix.

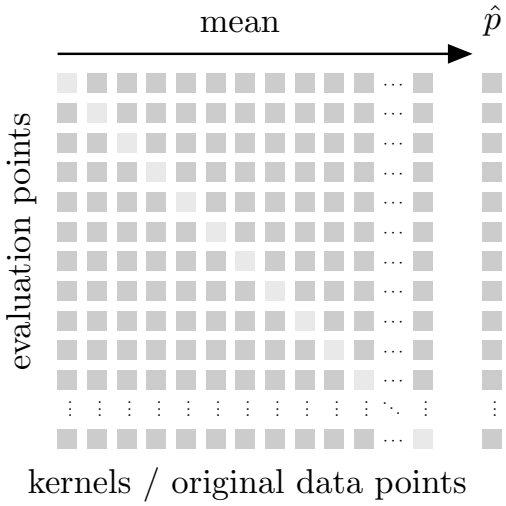


Figure 2: Pdf visualization in form of a matrix $\underline{X}$.

This calculation matrix additionally offers an interesting opportunity for the estimation of pdfs. If a matrix is build-up for a dataset and one data point is replaced by a new data point, the pdf can easily be adjusted, instead of being forced to recalculate the whole matrix from scratch. One simply has to calculate the effect of a kernel, placed on the new data point, on all other data points, and vice versa. The effect of the kernel on all data points equals the row of the particular data point in the calculation matrix, whereas the effect of all kernels on the data point equals the column. If the row and column of the new data point are updated, the matrix represents the pdf of the new dataset.

4 New algorithm

The proposed algorithm effectively combines elements from the DLS algorithm and the efficient pdf calculation strategy proposed in [5]. While the DLS algorithm maximizes the minimum nearest neighbor distance to find optimal LH designs, the new method has a more global approach for the loss function. It minimizes the mean absolute error of the difference between the uniform

distribution and the estimated pdf of a given LH design. For the flow chart of the new algorithm, see Fig. 3. The proposed algorithm proceeds as follows.

1. Initialize the algorithm with an arbitrary n-dimensional LH design $\underline{U} = [\underline{u}(1), \underline{u}(2), \ldots, \underline{u}(N)]$ consisting of the data points $\underline{u}(i) = [u_1(i)\ u_2(i)\ \cdots\ u_n(i)]^T$, $i = 1, 2, \ldots, N$.

2. In order to estimate the pdf $\hat{p}(\underline{u})$ of the LH design, kernel density estimation is used. The $N \times N$ calculation matrix $\underline{X}$ is set up for the LH design $\underline{U}$. The matrix $\underline{X}$ is symmetric, since data points and evaluation points are the same. Take the mean of all rows of $\underline{X}$ to receive $\hat{p}(\underline{u})$.

3. Select the data point with the maximum pdf value as the point to swap. This represents the point where data is most dense and therefore should be moved away by swapping with a partner point.

4. Go through all non-selected points $i = 1 : N-1$ in each dimension $ii = 1 : n$ and initialize a swap matrix $\underline{X}_s = \underline{X}$ for each point-dimension combination and an associated matrix $\underline{U}_s = \underline{U}$ to store the altered LH design. Swap the coordinates of the swap point and point i in dimension ii. Instead of recalculating the whole pdf, this merely equals a recalculation of two rows and two columns of $\underline{X}_s$ and the update of two points of $\underline{U}_s$. Calculate the pdf at all data points. Calculate the error $E(i, ii) = \frac{1}{N} \sum_{k=1}^{N} (|1 - \hat{p}(\underline{u}(k))|)$. For $N-1 \times n$ swaps, this results in the $N-1 \times n$ error matrix $\underline{E}$.

5. Execute the swap with the minimal value of the error matrix $\underline{E}$. Update the corresponding matrices $\underline{X} = \underline{X}_s$ and $\underline{U} = \underline{U}_s$ as well as $\hat{p}(\underline{u}) = \hat{p}(\underline{u}_s)$. Terminate the algorithm if no improvement is possible, otherwise continue at step 3.

5 Performance analysis

In this section, the performance of the new density-based LH design optimization approach is analyzed and compared to other state-of-the-art methods. For this purpose, space-filling designs for different dimensions ($n = 2, ..., 5$)

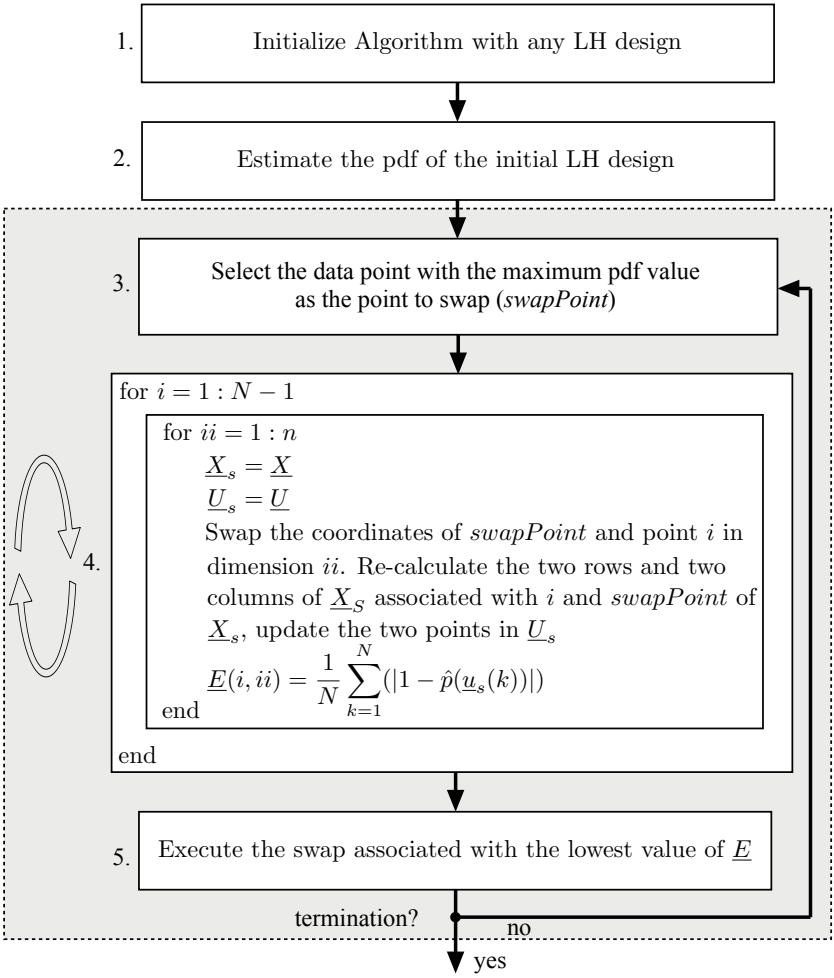


Figure 3: Procedure of the new algorithm.

and numbers of data points ($N = [20, 40, ..., 200]$) are compared. To assess the space-filling qualities of the methods, the Kullback-Leibler (KL) divergence between the estimated pdf of the datasets and the uniform distribution using Monte Carlo sampling with 10000 random points is calculated. Lower KL divergence values indicate a higher similarity of two pdfs. For a KL divergence value of zero two pdfs are identical. Furthermore, the computation time is investigated.

5.1 KL divergence evaluation

Figure 4 shows the result of the evaluation of the KL divergence over the number of data points for $n = 2, ..., 5$. For each investigated dimension, the proposed density-based LH design optimization consistently produces the best results. In most parts of the evaluation, the EDLS achieves the second-best results, followed by the DLS algorithm. The Sobol sequences exhibit the critical overall KL divergence scores and thus the critical space-filling datasets.

The difference between the diverse approaches can be attributed to the employed optimization criteria. The (extended) deterministic local search optimizes a nearest-neighbor-based criterion, thus only distances between pairs of data points are considered. On the contrary, the density-based LH design optimization directly optimizes the pdf of the LH design. Because of this global optimization approach, the density-based LH design optimization is more powerful and therefore able to achieve better KL divergence performances. Sobol sequences have the most simplistic structure out of the four analyzed methods. Furthermore, Sobol sequences work best if $N = 2^x$ with $x \in \mathbb{N}$. Thus, in a separate case not shown in this contribution, the four methods were analyzed for $N = [16, 32, 64, 128, 256]$. The performance differences were similar to the presented case.

5.2 Calculation time evaluation

Figure 5 shows the evaluation of the calculation time in dependence on the number of data points for different dimensions. Since the calculation time of

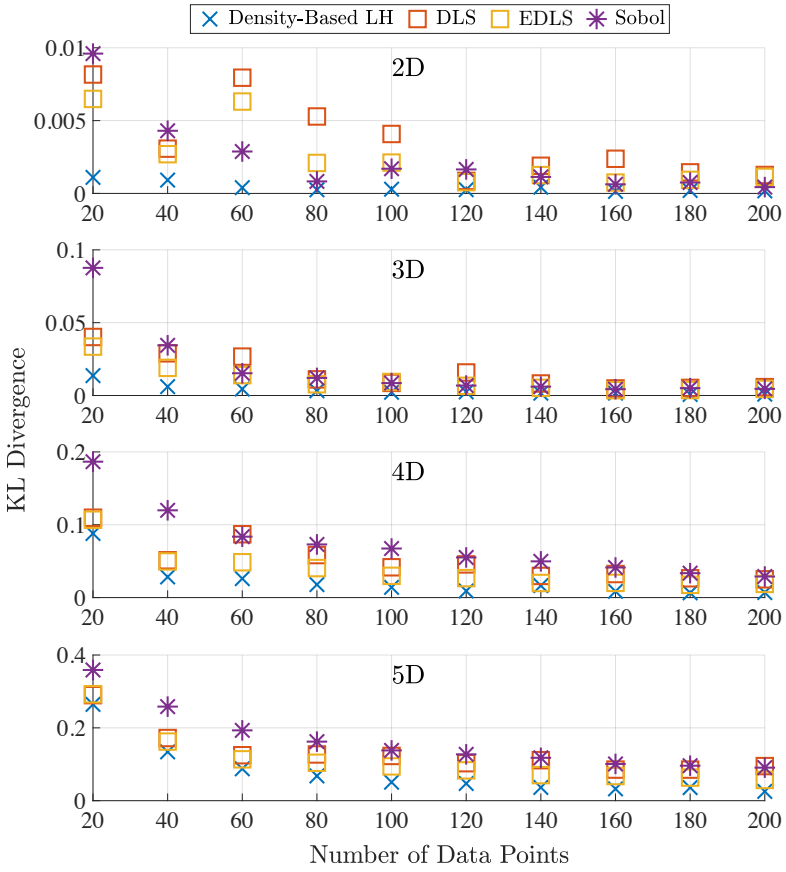


Figure 4: Comparison of KL divergences over different dimensions and number of data points.

Sobol sequences is negligibly small, it is left out of this particular evaluation. The density-based LH design optimization shows the highest calculation times, increasing with both the number of data points and the number of dimensions, followed by the EDLS and, with a big difference due to the simpler structure, the DLS. Furthermore as a reference several higher dimensional cases have been investigated. In this test case the calculation time for the new algorithm was 32 minutes in 7D and 69 minutes in 10D, both with $N = 100$ data points.

The computational effort of the density-based LH design optimization can be attributed to the time-consuming calculations of the pdfs for every possible point switch in each iteration. An alteration of the algorithm in steps 4 and 5, compare Fig. 3, could make it more efficient. In the proposed version, i runs over all $N-1$ data points and executes the swap with the lowest overall error. Alternatively, rather than performing the best swap, the algorithm could run until the *first* swap improves the error. This alteration will be subject to further research.

5.3 Extension to a stochastic algorithm

The density-based LH design optimization always performs the best possible point switch. Therefore, it represents a local search method. A simple, straightforward stochastic extension is presented to circumvent this shortcoming to possibly increase the algorithm's performance even more. On the downside, the convergence of the algorithm is not strictly decreasing.

The only necessary change takes place in step 5, compare Fig. 3. Instead of executing the *best* swap resulting in the best possible quality, a swap is executed with a probability proportional to the size of a quality measure. Here, the inverse mean absolute error is used as the measure of quality. This stochastic element helps the algorithm to escape bad local optima, see Fig. 6. Here, the density-based LH approach stops after around 100 iterations, because no swap leads to further improvement of the quality. In comparison, the density-based LH with stochastic extension is terminated after around 400 iterations through an artificial criterion, which terminates the algorithm if no improvement is achieved for 20 iterations.

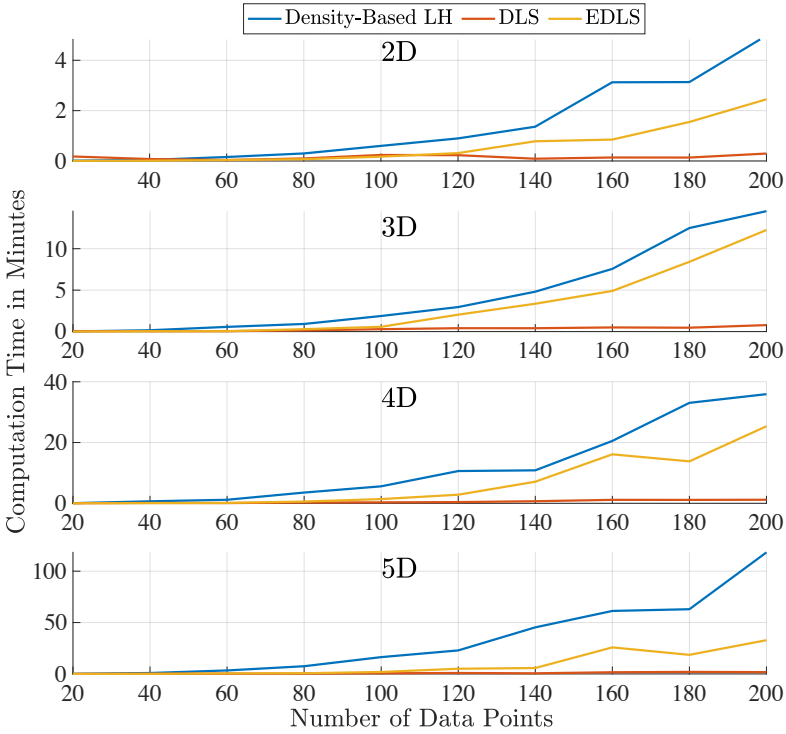


Figure 5: Comparison of computation times over different dimensions and number of data points.

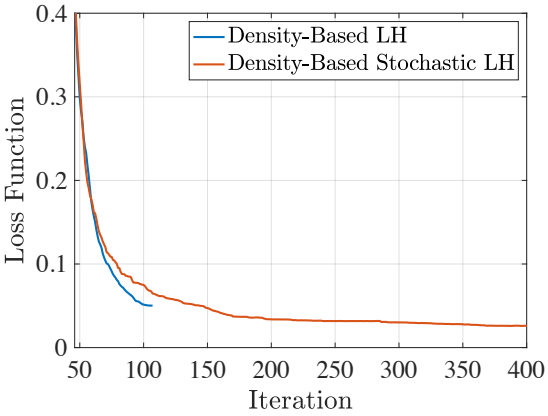


Figure 6: Exemplary comparison of the loss function progress over the iterations for $N = 120$ and $n = 3$.

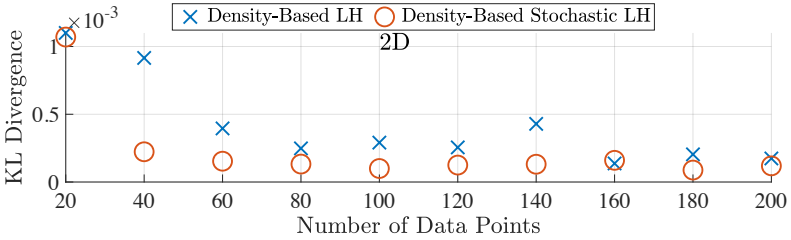


Figure 7: Comparison of KL divergences over the number of data points.

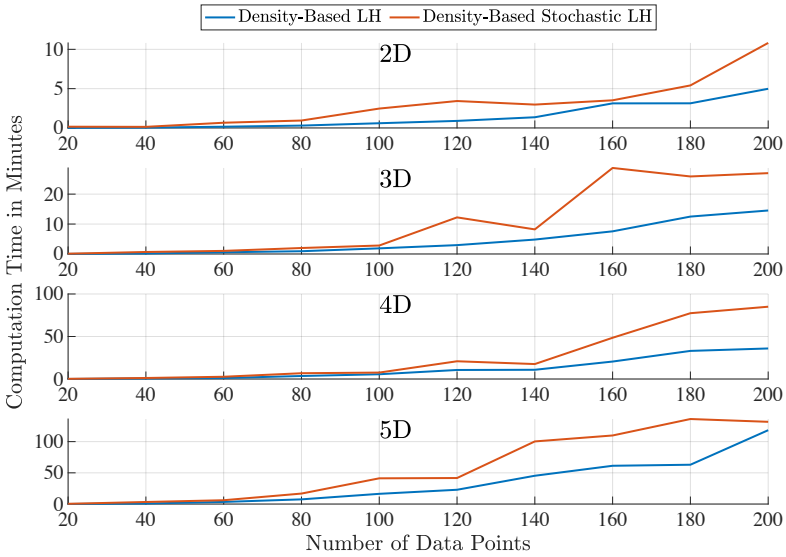


Figure 8: Comparison of calculation times over different dimensions and number of data points.

As Fig. 7 shows, the impact of the stochastic extension is visible in 2D. Since the influence in dimensions 3 to 5 is negligible, the plots are left out. While the improvement is moderate to non-existing in terms of performance, it is significant in terms of computational demand, see Fig. 8. In the perspective of the author, the almost non-existing advantage in terms of performance is not worth the impact on the computational demand.

6 Conclusion

In this contribution, a new approach to optimize LH designs, based on the estimation and evaluation of pdfs, was presented. The algorithm minimizes the mean absolute error or any other criterion between the estimated pdf of the LH design, evaluated solely on its data points, and the uniform distribution. With this structure, the algorithm is capable of swapping data points of an LH design

to achieve a uniform point distribution. Alternative optimization strategies in combination with the new proposed criterion are also possible.

In order to validate the functioning of the new approach, it was compared to other state-of-the-art methods to create space-filling designs. The methods were compared using the KL divergence of the resulting datasets and the uniform distribution, as well as the resulting calculation times. Overall, the KL divergence performance of the new approach was outstanding, the new algorithm outperformed the other methods significantly. However, in terms of computation time, the new approach is expensive.

A possible alteration of the algorithm is to reduce computation time by executing not the best but the first point swap which achieves an improvement of the loss function. Furthermore, the global loss function opens up different interesting research topics. A way to implement boundaries or constrains in the input space are promising subjects for future research, as is the possibility to optimize a design to any desired distribution, not just the uniform one.

References

[1] A. Rimmel and F. Teytaud: "A survey of meta-heuristics used for computing maximin Latin hypercube". In: *European Conference on Evolutionary Computation in Combinatorial Optimization* pp. 25–36. Springer. 2014.

[2] L. Pronzanto and W. G. Müller. "Design of computer experiments: space filling and beyond". In: *Statistics and Computing* vol. 22, no. 10, pp. 681–701. 2012.

[3] T. Ebert, T. Fischer, J. Belz, T. O. Heinz, G. Kampmann and O. Nelles. "Extended deterministic local search algorithm for maximin Latin hypercube designs". In: *Computational Intelligence*, 2015 IEEE Symposium Series pp. 375–382. 2015.

[4] A. Grosso, A. Jamali, and M. Locatelli. "Finding maximin latin hypercube designs by iterated local search heuristics". In: *European Journal of Operational Research* vol. 197, no. 2, pp. 541–547. 2009

[5] T. J. Peter and O. Nelles. “Fast and simple dataset selection for machine learning”. In: *at-Automatisierungstechnik* vol. 67, no. 10, pp. 833–842. 2019

[6] D. W. Scott. “Multivariate density estimation: theory, practice, and visualization”. John Wiley & sons. 2015.

[7] S. Sheater and M. Jones. “A reliable data-based bandwidth selection method for kernel density estimation”. In: *Journal of the Royal Statistical Society* vol. 53, no. 3, pp. 683–690. 1991

[8] B. W. Silverman. “Density estimation for statistics and data analysis” vol. 26, CRC press. 1986.

[9] I. M. Sobol. “On quasi-monte carlo integrations”. In: *Mathematics and Computers in Simulation* vol. 47, no. 2, pp. 103–112. 1998

[10] R. B. Chen, D. N. Hsieh, Y. Hung and W. Wang. “Optimizing latin Hypercube designs by particle swarm”. In: *Statistics and Computing* vol. 23, no. 5, pp. 663–676. 2013

[11] M. D. Morris and T. J. Mitchell. “Exploratory designs for computational experiments”. In: *Journal of Statistical Planning and Inference* vol. 43, no. 3, pp. 381–402. 1995

[12] R. Jin, W. Chen and A. Sudjianto. “An efficient algorithm for constructing optimal design of computer experiments”. In: *Journal of Statistical Planning and Inference* vol. 134, no. 1, pp. 268–287. 2005

Constrained Multi-Agent Optimization with Unbounded Information Delay

Stefan Heid, Arunselvan Ramaswamy, Eyke Hüllermeier

Heinz Nixdorf Institute and Department of Computer Science
Paderborn University
Warburger Straße 100, 33098 Paderborn
E-Mail: {stefan.heid, arunselvan.ramaswamy, eyke}@upb.de

Abstract

A multi-agent system (MAS) consists of a group of agents that solve a common task through cooperation. Many problems arising in this setting can be formulated as distributed constrained optimization. In recent work, we considered the unconstrained version of the problem. In particular, we developed a theory to understand distributed gradient-based optimization methods, wherein the local (state) information is communicated via a lossy wireless network. A key contribution of the theory is that the information delay could be unbounded, however, it does not consider constraints. In this work, we present preliminary experimental results aimed towards extending the aforementioned work to the *constrained* setting. First, the constrained optimization problem is transformed into an unconstrained one using the penalty-based method. Then, we employ the distributed gradient approach from our previous work to solve the unconstrained optimization in a decentralized manner. The illustrative experiments are based on autonomous pattern formation tasks for robotic swarms. The (simulated) robots cooperate to form a specified pattern (line, circle), with the constraint that the distances between neighboring robots equal a given constant.

1 Introduction

A multi-agent system (MAS) is typically large-scale in nature, and a wireless communication network is used to connect the various agents involved, due to its convenience and cost. Examples of MAS include wireless sensor networks and smart grids, see [3]. Many problems that arise in these systems can be cast as constrained optimization problems that need to be solved in a distributed decentralized manner [6]. For example, in smart grids, a group of controllers has a common objective to minimize the control errors in terms of AC frequency or to maintain voltage levels in the whole grid with time-variant loads or energy sources. The controllers cooperate to solve this problem under constraints on the system state.

The literature on distributed algorithms to solve constrained optimization problems is rich, see e.g. [1]. However, they typically assume that the delay associated with the transfer of information from one agent to the other is bounded. Failed transmissions and channel delay are two main factors that contribute to information delay. In this paper, we focus on *information delay due to failed transmissions*. We study the effect of unbounded information delay on distributed algorithms for constrained optimization. In the past, unbounded information (update) delays were studied within the setting of unconstrained optimization in [4, 5].

The global objective is formulated in terms of a differentiable function. The agents solve this objective, together, by searching appropriate local subspaces via gradient steps. The solution to the global problem is obtained by putting together the distributed solutions. For local gradient calculations, at every step, the agents require information from other agents. Furthermore, each agent has to optimize subject to some local constraints. To this end, we use the penalty method to transform the constrained problem into an unconstrained one. In other words, the distributed gradient updates of each agent is augmented by a penalty term that encodes the violation of local constraints. It may be noted that the associated penalty hyper-parameter is increased over time. Since the communication channel is lossy, the information from the peers may be delayed and the agent is therefore forced to carry out update steps using *outdated information*. In [4], mild requirements on the quality of the wireless network

are presented, which ensure that using outdated information does not hinder convergence. In this paper, we conjecture and present preliminary numerical results which suggest that *similar conclusions can be drawn even in the presence of constraints*.

To illustrate the ideas, we consider pattern forming tasks for robot swarms as an application. To this end, the specified pattern (line, circle) is expressed as an objective function. The objective is constructed, such that the minimum of the objective is reached when the robots arrange in the pattern. The objective function is evaluated using all robot positions, and the distances to neighboring agents constitute the local constraint set for every agent. At every time step, each robot moves in accordance to the local gradient update. To calculate this gradient, it uses the last known position of the other robots in the swarm. Since robot positions are communicated using lossy channels, the last known position may be outdated. In our experiments, we assume that the robots are ordered and communicate their knowledge of the swarm, only with direct neighbors in the chain.

2 Problem Definition

Broadly speaking, we have m agents that aim to minimize a given *global* objective function while satisfying *local* constraints. In other words, the agents cooperate to find:

$$\begin{aligned} \mathbf{z}^* &= \arg\min_{\mathbf{z}} \mathbb{E}_{\xi}\left[J(\mathbf{z},\xi)\right], \\ \text{s.t. } G_i &= \{g_{ik} \leq 0 \mid 1 \leq k \leq k_i\} \quad 1 \leq i \leq m, \end{aligned} \tag{1}$$

where $\mathbf{z}^* = (\mathbf{z}_1^*, \ldots, \mathbf{z}_m^*)$ such that $\mathbf{z}_i^*$ is the component of the minimum that is calculated by the i^{th} agent a_i, $G_i = \{g_{ik} \mid 1 \leq k \leq k_i\}$ is the local constraint set of a_i containing k_i inequality constraints. The stochastic objective function, $J : \mathbb{R}^n \times \mathbb{S} \to \mathbb{R}$, is such that $\mathbf{z} \doteq (\mathbf{z}_1, \ldots, \mathbf{z}_m)^T \in \mathbb{R}^n$ where $\mathbf{z}_i$ is the *local variable* associated with a_i, and ξ is an $\mathbb{S}$-valued random variable. In typical applications, $\mathbb{S}$ is some compact subset of $\mathbb{R}^k$, $k \geq 1$, or $\mathbb{R}^k$ itself. Please note that we *allow for general vector-valued* $\mathbf{z}_i$s.

The reader may note that the local constraint set G_i, of a_i, is *not visible* to a_j for $j \neq i$. In other words, the agents are only aware of their local constraints, not that of others. Since we use the penalty-based method to transform the constrained optimization problem into an unconstrained one, we may associate each G_i with the following penalty function:

$$P_i(\mathbf{z}) \doteq \sum_{k=1}^{k_i} \max\left(0, g_{ik}(\mathbf{z})\right)^2. \tag{2}$$

2.1 Communication Model

As stated earlier, the agents are connected using a wireless communication network. We model this using a weighted directed graph $G = (V,E)$. In this graph, each agent is represented as a node and a directed edge $e_{ij} = (a_i, a_j)$ exists if a_i can directly transmit messages to a_j, possibly using a dedicated *unidirectional channel*. The edge-weights ($\in [0,1]$) represent the probability of successful transmission along that edge. We assume that the transmissions along different edges are independent, i.e., there is *zero interference*. We allow for graph evolution, provided it is connected at all times. In particular, at any point in time, there exists a path connecting a_i to a_j such that the product of the edge-weights (success probabilities) is strictly greater than zero, $1 \leq i, j \leq m$. Hence, there is a chance that the message sent by a_i reaches a_j.

To find a solution, $\mathbf{z}^* = (\mathbf{z}_1^*, \ldots, \mathbf{z}_m^*)$, to the above described constrained optimization problem, a_i searches for $\mathbf{z}_i^*$ in its local search space $\mathbb{R}^{n_i}$ using the following gradient formula:

$$\frac{\partial J}{\partial \mathbf{z}_i} + \beta \frac{\partial P_i}{\partial \mathbf{z}_i}, \tag{3}$$

where $\frac{\partial}{\partial \mathbf{z}_i}$ is the partial derivative with respect to the variable $\mathbf{z}_i$, β is the *penalty parameter*, and $\sum_{i=1}^{m} n_i = n$. In order to calculate $\frac{\partial J}{\partial \mathbf{z}_i}$, a_i requires updates from a_j, $j \neq i$. This information is exchanged using the underlying wireless communication network, which causes delays. In this paper, we consider the following sources of *information delays*:

- packet losses;
- routing through other agents in the system, due to the lack of direct connection.

Note that we do not consider channel delays in this paper. However, we believe that our ideas may be readily extended to incorporate, possibly unbounded, channel delays. The delays directly affect the *age of the information* available to an agent. In this paper, we use the term *information delay* and *age of information*, interchangeably.

The gradient calculation in (3) *deals with information delays, by using the latest available updates from other agents in the system.*

2.1.1 On unbounded information delays

Let us suppose that there are no packet losses. The delay due to indirect routing grows linearly as a function of the distance between the nodes in the graph. We assume that the diameters (maximum distance between any pair of nodes) of the evolving graphs are bounded, independent of time. Hence the delay due to indirect routing is also bounded. If we now consider packet loss, then updates within any *bounded time-frame* cannot be guaranteed. Hence, *packet loss* is the major contributor to information delay. The probability that a_i successfully communicates with a_j within any d time-step interval is some $p > 0$, where d is the above mentioned bound on the graph diameter. Note that p may vary over time. Hence, the event of unsuccessful communication over successive d length intervals is geometrically distributed. In other words, there is no absolute bound on the information delay.

3 Algorithm

We are now ready to present an algorithm to solve (1). It may be noted that it is based on penalty-based gradient descent methods for the centralized version. In the setting considered here, agent a_i updates $\mathbf{z}_i$ in an iterative manner, through gradients calculated using the latest available $\mathbf{z}_j$, $j \neq i$. At any time t, a_i

Algorithm 1.: Distributed Optimization

1: Initialize a_i with $\mathbf{z}$
2: **for all** time-step **do**
3: process received $\hat{\mathbf{z}}_j^t$
4: update $\mathbf{z}_i^t$
5: **for all** $a_j : (a_i, a_j) \in E$ **do**
6: send $\hat{\mathbf{z}}_i^t$ to a_j

maintains a *local view*, $\hat{\mathbf{z}}_i^t$, of the global variable $\mathbf{z}^t$. Formally speaking, the local view of a_i at time t is given by $\hat{\mathbf{z}}_i^t \doteq \left(\mathbf{z}_1^{\tau_{i1}(t)}, \ldots, \mathbf{z}_i^t, \ldots, \mathbf{z}_m^{\tau_{im}(t)}\right)^T$, where $0 \leq \tau_{ij}(t) \leq t$, and $t - \tau_{ij}(t)$ is the *age of the information* from a_j available to a_i at time t. The local $\mathbf{z}_i$ is updated as follows:

$$\mathbf{z}_i^{t+1} \leftarrow \mathbf{z}_i^t - \eta(t) \left[\frac{\partial J(\hat{\mathbf{z}}^t, \xi^t)}{\partial \mathbf{z}_i} + \beta(t) \frac{\partial P_i(\hat{\mathbf{z}}^t)}{\partial \mathbf{z}_i} \right], \tag{4}$$

where $\eta(t)$ is the learning rate and $\beta(t)$ is the time-varying penalty parameter. ξ^t are statistically independent samples that have the same distribution as ξ.

3.1 Information exchange

At time t, a_i sends $\hat{z}_i^t$ to its neighbors. Simultaneously, it receives $\hat{z}_j^t$s from a subset of its neighbors (some may be lost due to packet drops). It uses the obtained information, and z_i^t, to update $\hat{z}_i^t$ to $\hat{z}_i^{t+1}$, such that it contains the latest variables associated with other agents. To facilitate consistent updates, we assume that each variable is associated with a *time stamp*. Hence, at time-step t, a_i receives $\hat{\mathbf{z}}_j^t$ along with the vector of time stamps $\hat{\mathbf{t}}_i \doteq (\tau_{j1}(t), \ldots, t, \ldots, \tau_{jm}(t))^T$, from a subset of its neighbors. Using the obtained information, a_i updates the entries of $\hat{\mathbf{z}}_i^t$ by comparing time stamps. In other words, a_i checks to see if $\tau_{jk}(t) > \tau_{ik}(t)$, if yes, then updates $\hat{\mathbf{z}}_i^t(k)$ to $\hat{\mathbf{z}}_j^t(k)$. It also updates the corresponding entry in $\hat{\mathbf{t}}_i$. Otherwise, old entries are retained. Note that $\hat{\mathbf{z}}_i^t(k)$ is used to represent the information that a_i has of a_k, at time t. Subsequently, a_i executes update step (4). Finally, the agent sends its updated $\hat{\mathbf{z}}_i^{t+1}$ along with the updated time stamps to all its neighbors. This allows, agent a_j similarly to discard outdated updates. The reader must note that no retransmissions are

triggered in case of failed transmissions since we do not assume that received packets are acknowledged.

The discussion in this section has been codified in Algorithm 1.

3.2 On the convergence of Algorithm 1

We believe that a proof of convergence of the algorithm will proceed along similar lines as the analysis in [4]. Here, we studied the unconstrained version of Algorithm 1. Suppose the random variables associated with the age of information, at every time-step, have bounded second moments, then it is shown, in [4], that the associated errors are asymptotically in the order of the learning rate. Since the learning rate diminishes to zero, the effect of information delays vanishes asymptotically. Also, that the distributed algorithm has the same asymptotic properties of a centralized one.

A sufficient condition on the wireless network to ensure the above mentioned bounded second moment requirement is stated as assumption (A6) in [4]. It is restated below for our setting:

- for each pair of agents, there is a non-zero probability of successful transmission of the routed package,
- and the transmission probabilities of all edges are statistically independent.

As discussed in Section 2.1, the communication graph is always connected. Hence the above statement conditions are readily satisfied.

Now, we discuss the influence of the penalty parameters on convergence. In our algorithm, we do not use a constant β, rather we take $\beta(t) \uparrow \infty$. This is done to avoid the scenario wherein the algorithm converges to $\mathbf{z}^\infty$ such that:

$$\begin{aligned} \nabla J(\mathbf{z}^\infty) + \beta \sum_{i=1}^{m} \nabla P_i(\mathbf{z}^\infty) &= 0 \text{ and} \\ \nabla J(\mathbf{z}^\infty) &= -\beta \sum_{i=1}^{m} \nabla P_i(\mathbf{z}^\infty) \neq 0. \end{aligned}$$

This phenomenon is explained in the literature of centralized penalty-based method. As stated earlier, we can use the arguments from [4] to conclude that Algorithm 1 has the same long-term behavior as its centralized counterpart. In other words, $\beta(t) \uparrow \infty$ is important to ensure that $\nabla J(\mathbf{z}^{\infty}) = \sum_{i=1}^{m} \nabla P_i(\mathbf{z}^{\infty}) = 0$, as desired.

However, the main issue with a time-varying penalty parameter that goes to infinity is the growth in the *variance of the descent directions*. Hence, a diminishing learning rate is required to counteract this. More precisely, $\eta(t)$ and $\beta(t)$ are chosen such that

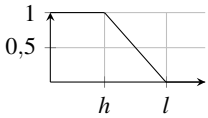
$$\sum_{t} \left(\eta(t) \cdot \beta(t)\right)^2 < \infty.$$

This condition is inspired by a similar assumption, A1, in [7]. Intuitively, it is clear that a condition such as $\eta(t)\beta(t) \to 0$ is required. However, it is shown in [7] that is not always sufficient.

4 Experiments

In this section, we present the results of two experimental studies, in which mobile robots are simulated as points in the two-dimensional Euclidean space[1]. The two scenarios optimize for a different objective function and have slightly different penalties, while the communication model is the same for all experiments. A simple packet reception probability model is used to calculate the success probability of transmission as a function of robot distance. To ensure connectedness of the communication graph, the minimal transmission probability is bounded as follows:

$$p_{h,l}(d) = \mathbf{clip}\left(\frac{h-d}{h-l}, 0.01, 1\right)$$



[1] https://github.com/stheid/DDSCO

The parameters h and l determine the thresholds of maximum and minimum transmission probability, respectively.

The general setting of the following scenarios are robot-swarm pattern formation tasks. Each agent is represented by a point $\mathbf{z}_i = (x_i, y_i)^T \in \mathbb{R}^2$. In the first scenario, the agent's objective is to form a line on which the agents evenly space out. In the second scenario, the agents should form a circle and also create a constant distance to each other. The inter-robot distance is modeled by the constraints, while the created structure is modeled in the objective function. Initially, the agents are placed uniformly at random in a quadratic region. The vector $\mathbf{z}$ is the concatenation of local position vectors $\mathbf{z}_i$ of all agents a_i. Additionally, we denote $\mathbf{x} = (x_1, \ldots, x_m)^T$ and $\mathbf{y} = (y_1, \ldots, y_m)^T$. In our experiments, we simulate a swarm of $m = 10$ robots.

4.1 Scenario 1: Forming a Line

The objective function for forming a line consists of two parts. The first component is the residual error of the ordinary least squares (OLS) regression over all points. The second part is the distance between the first and last point. Maximizing the distance along with minimizing the residual error encourages the robots to unravel if they form a folded line.

The x-term of each position is augmented by a constant for fitting the bias term and a random value to make the objective more challenging: $\phi(x_i) = (1, \gamma_i, x_i)$ with $\gamma_i \sim \mathcal{N}(\mu = 0, \sigma = 0.1)$. For ease of notation we define $\Phi = (\phi(x_1)^T, \ldots, \phi(x_m)^T)^T$ as the transformation of $\mathbf{x}$. Formally the objective is defined as follows:

$$
\begin{aligned}
J(\mathbf{x}, \mathbf{y}) &= \frac{\mathbf{e}^T \mathbf{e}}{n^2} - \frac{\|\mathbf{z}_1 - \mathbf{z}_m\|^2}{n} \\
\mathbf{e} &= \mathbf{y} - \Phi \mathbf{b} \\
\mathbf{b} &= (\Phi^T \Phi)^{-1} \Phi^T \mathbf{y}
\end{aligned}
$$

Here, $\mathbf{b}$ is the OLS regression line and $\mathbf{e}$ is the residual error of the regression estimate. Since the regression error is in the order of the magnitude of the squared number of nodes, it has been normalized accordingly. Similarly, the

distance between the first and last agent of a chain is normalized by the number of agents.

Each agent has up to two local constraints. The first and last agent have only one neighbor, therefore, they will only have one constraint. All other agents have two equality constraints to maintain a constant distance to their neighbors.

More precisely, agent i has the following constraints:

$$g_{i1}(\mathbf{z}) = |d_0 - \|\mathbf{z}_i - \mathbf{z}_{i-1}\|^2| = 0 \qquad \text{if } i > 1$$
$$g_{i2}(\mathbf{z}) = |d_0 - \|\mathbf{z}_i - \mathbf{z}_{i+1}\|^2| = 0 \qquad \text{if } i < m$$

with d_0 the demanded inter agent distance and $\|\cdot\|^2$ the Euclidean distance.

4.2 Scenario 2: Forming a Circle

Several objective functions could be chosen to form a circle. A quite obvious one would be to make the agents maximize the area of the polygon they span. Correctly calculating the area of arbitrary polygons is a non-trivial task, as self-intersecting polygons pose additional challenges [2]. Fortunately, the equation for calculating the area of simple polygons can be used as a lower bound for the actually covered area. Therefore, it is sufficient to use this equation for arbitrary polygons in our case, since we aim for maximization:

$$J(\mathbf{x},\mathbf{y}) = -\left|\sum_{i=0}^{n-1} x_i y_{i+1} - x_{i+1} y_i\right| \qquad \text{with } x_0 = x_m \text{ and } y_0 = y_m$$

For simplicity, the constant prefix has been omitted, as it will not influence the position of a minimum. Since the algorithm is purely guided by the gradient, it might converge to local optima.

The constraints are similar to the previous example, however, the first and last agent will now be considered as neighbours:

$$g_{i1}(\mathbf{z}) = \left| d_0 - \|\mathbf{z}_i - \mathbf{z}_{i-1}\|^2 \right| = 0$$
$$g_{i2}(\mathbf{z}) = \left| d_0 - \|\mathbf{z}_i - \mathbf{z}_{i+1}\|^2 \right| = 0 \qquad \text{with } \mathbf{z}_0 = \mathbf{z}_m$$

The hyperparameters for learning rate and penalty scale are chosen as follows:

$$\eta(t) = \frac{100}{t + 30000}$$
$$\beta(t) = 1 + \sqrt{\frac{t}{100}}$$

The learning rate is therefore decreasing in a inverse proportional manner, while the the penalty term goes to infinity in the square root of the timestep. The constants in the equation were tuned by hand to achieve fast convergence.

4.3 Results

In both of the above experimental scenarios, we observed convergence to at least a local minimum, while satisfying all local constraints. For the line scenario, the robots eventually arrange on a line, however, sometimes the line is folded into itself, which will not maximize the distance, but minimize the regression error. For the circle scenario, we similarly see the forming of perfect circles or shapes like spirally intersecting circles, which represent a local maximum of the covered area.

The rate of convergence was observed to strongly depend on the quality of the wireless network (transmission success probabilities). Figure 1 illustrates the evolution of the algorithm, tasked to form the circle, in the second scenario. In the beginning, the agents were severely penalized due to large inter-agent distances. This causes them to move towards each other, and form a closely-knit cluster. In the next phase, the agents try to form a larger circle, due to the design of the objective function. In the final phase, the increasing penalty

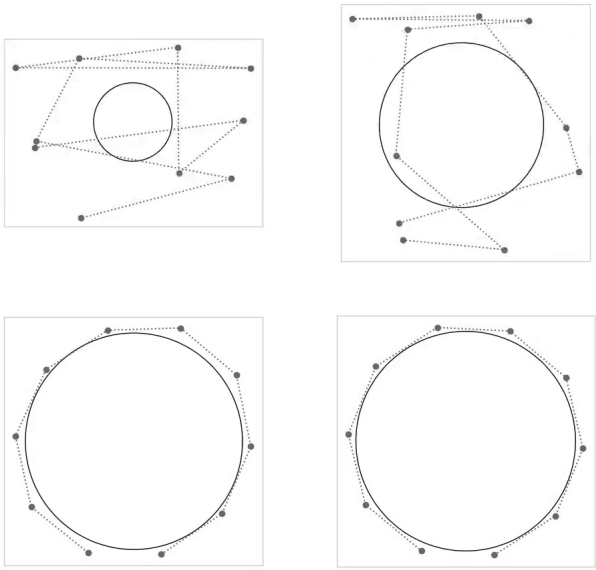


Figure 1: Snapshots of different phases of scenario 2. Each dot represents the position of a robot, the communication link is visualized as the blue line. The optimal solution is indicated with the black circle.

parameter forces the agents to *move closer to each other*, to fulfill the distance constraint.

4.3.1 Impact of Communication Quality on Rate of Convergence

The quality of the wireless network seems to affect the rate of convergence. To illustrate this, we experimented with different success probabilities (network qualities). Empirical results suggest that a strong correlation cannot be directly seen. This is because aged information about the peer's positions allow constraints to be violated more freely in some cases. In other words, bad communication may allow for a streamlined convergence to a wrong minimum (for e.g., does not satisfy constraints). Loosely speaking, old information facilitates

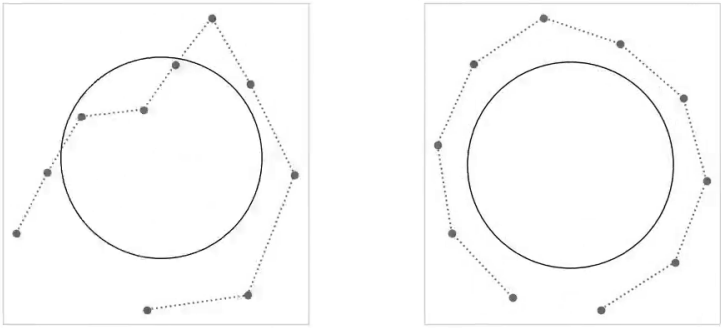


Figure 2: Convergence progression in scenario 2 after about 2000 update steps. Left picture shows convergence with less reliable communication channels and hence slower convergence.

a more localized optimization and constraint satisfaction, and updated information brings back a clearer picture of the global constrained optimization. Figure 2 shows the convergence of two identical configurations of scenario 2, which only vary in the communication channel quality. The optimal solution subject to the distance constraints is reached when all agents arrange on the black circle, in an equally spaced manner. The figure to the right illustrates the scenario with good communication, and the figure on the left the bad one. The algorithm has almost formed a circle, under good communication, only some constraint violations are left to be addressed. In the case of bad communication, the algorithm has not yet formed a circle and is hence lagging.

4.3.2 Constraint Satisfaction

Let us consider the first scenario, wherein the agents try to form a straight line while maximizing the spanned distance of the collection. However, the constraint requires that a given inter-agent distance be achieved. With small penalty scaling factors, the constraint is very loose and can easily be violated. Therefore, the penalty needs to be scaled continuously to allow for arbitrarily small constraint violations. The continuous increase of penalty allows a smooth transition from the unconstrained to the fully constrained scenario. Similarly,

in the second scenario, the agents aim to form a circle of area. Therefore, drifting outwards, however, again inter-agent distances must be maintained. Again, the penalty term needs to grow to infinity, to dominate the overall penalized objective when the solution does not satisfy constraints. In particular, growing the penalty parameter prevents convergence to a point that satisfies $\nabla f = -\beta \nabla \sum_i P_i \neq 0$.

The experimental results are in agreement with the general argument of convergence stated at the end of Section 3.

Videos showing the evolution of the convergence can be found in the repository https://github.com/stheid/DDSCO.

5 Conclusion

We considered the problem of distributed constrained optimization with stochastic objective and inequality-type constraints. To solve this problem, we presented a penalty-based distributed gradient algorithm. We presented preliminary empirical results to support the conjecture that results from [4] naturally extend to the inclusion of constraints. In particular, that the convergence, in the presence of local constraints, is unaffected by stochastic information delays with bounded second moments.

For visualization, we investigated two scenarios of pattern formation in robot swarms. The objective function was used to specify the pattern, subject to the inter-robot distance constraints. The agents collectively minimized the objective function by searching in appropriate subspaces. In the experiments, we observed that the convergence speed correlates with the quality of the communication channel, although a more thorough investigation is needed to paint a clearer picture. Also, we look forward to analyzing the setting in a more formal way to prove the convergence theoretically.

References

[1] Stephen Boyd. Distributed optimization and statistical learning via the alternating direction method of multipliers. 3(1):1–122.

[2] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[3] Chun-Hao Lo and Nirwan Ansari. Decentralized controls and communications for autonomous distribution networks in smart grid. 4(1):66–77. Conference Name: IEEE Transactions on Smart Grid.

[4] Arunselvan Ramaswamy, Adrian Redder, and Daniel E. Quevedo. Optimization over time-varying networks with unbounded delays.

[5] Benjamin Sirb and Xiaojing Ye. Decentralized consensus algorithm with delayed and stochastic gradients. 28(2):1232–1254.

[6] Wen-Zhan Song, Renjie Huang, Mingsen Xu, Andy Ma, Behrooz Shirazi, and R. Lahusen. Air-dropped sensor network for real-time high-fidelity volcano monitoring. pages 305–318, 01 2009.

[7] J. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37:332–341, 1992.

A Comparative Study on Subgraph Crossover in Cartesian Genetic Programming

Roman Kalkreuth

Department of Computer Science
TU Dortmund University, Germany
Email: roman.kalkreuth@tu-dortmund.de

1 Introduction

While tree-based Genetic Programming (GP) [1] is often used with crossover, Cartesian Genetic Programming (CGP) [2] is mostly used only with mutation as the sole genetic operator. In contrast to comprehensive and fundamental knowledge about crossover in tree-based GP, the state of knowledge in CGP appears to be still ambiguous and ambivalent. Two decades after CGP was officially introduced, the role of recombination in CGP is still considered to be an open and remaining question. The state of knowledge about crossover in CGP has been recently surveyed and the role of crossover is still considered to be an open and remaining question [3]. Even if some progress has been made in recent years, comprehensive and detailed knowledge about crossover in CGP is still missing [3]. A promising step forward was made by the introduction of the subgraph crossover [4] but this technique has not been comprehensively studied in the past. Therefore, this work follows up former work on the crosover question by investigating if the search performance of CGP algorithms that utilize the subgraph crossover can be more efficient as the commonly used mutation-only CGP on a set of well-known benchmark problems. This short paper provides an overview of the full paper version [5] of the presented work.

1.1 Cartesian Genetic Programming

In contrast to tree-based GP, CGP represents a genetic program via genotype-phenotype mapping as an indexed, acyclic, and directed graph. The CGP decoding procedure processes groups of genes and each group refers to a function node of the graph. The last genes of the genotype represent the outputs of the phenotype. Each node is represented by two types of genes which index the function number in the GP function set and the node inputs. These nodes are called *function nodes* and execute functions on the input values. The number of input genes depends on the maximum arity n_a of the function set. Given the number of outputs n_o, the n_o last genes in the genotype represent the indices of the nodes, which lead to the outputs. A backward search is used to decode the corresponding phenotype. An example of the backward search of the most popular one-row integer representation is shown in Figure 1. The backward search starts from the program output and processes all nodes which are linked in the genotype. In this way, only active nodes are processed during evaluation. The genotype in Figure 1 is grouped by the function nodes. The first (underlined) gene of each group refers to the function number in the corresponding function set in the figure. The integer-based representation of CGP phenotypes is mostly used with mutation only. The number of inputs n_i, outputs n_o, and the length of the genotype is fixed. Every candidate program is represented with $n_r * n_c * (n_a + 1) + n_o$ integers. CGP is traditionally used with a (1+λ) selection scheme of evolutionary algorithms. The new population in each generation consists of the best individual of the previous population and the λ created offspring. The breeding procedure is mostly done by a point mutation that swaps genes in the genotype of an individual in the valid range by chance.

2 The Subgraph Crossover Technique

The subgraph crossover technique for CGP is inspired by the subtree crossover found in tree-based GP. To recombine two directed acyclic graphs, the subgraph recombination is performed by respecting the CGP phenotype. The phenotype of each individual is represented by the active path of the graph

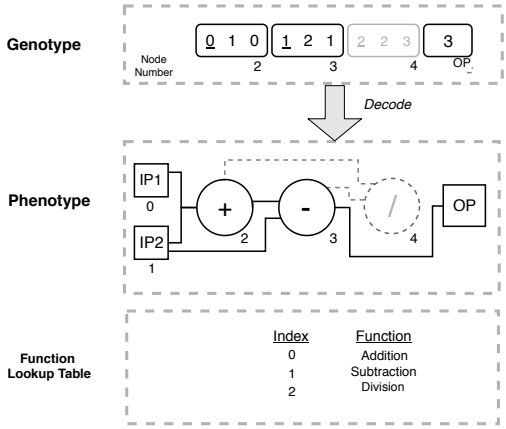


Figure 1: Example of the decoding procedure of a CGP genotype to its corresponding phenotype. The nodes are represented by two types of numbers which index the number in the function lookup table (underlined) and the inputs (non-underlined) for the node. Inactive function nodes are shown in gray color. The identifiers IP1 and IP2 stand for the two input nodes with node index 0 and 1. The identifier OP stands for the output node of the graph.

and is determined through the evaluation process. Furthermore, the active path of a graph leads to the semantic value of a certain individual in CGP. As a consequence, the subgraph crossover exclusively recombines the genetic material of the active paths. The idea of the subgraph crossover is that it should reduce the disruption which is caused by the genotypic single-point crossover in standard CGP and truly recombine subgraphs.

For the description of the subgraph crossover procedure, let n_i be the predefined number of input nodes and let n_f be the predefined number of function nodes. In CGP, the input nodes are indexed from n_i to $n_\mathrm{i}-1$ and the function nodes of each graph are indexed from 0 to $n_\mathrm{i}+n_\mathrm{f}-1$. The nodes which lie between the input and output nodes are denoted as function nodes. The crossover is done with two parents which are denoted as P_1 and P_2. For the crossover procedure, the node numbers of the active function nodes are necessary. The node numbers of the active nodes of P_1 and P_2 are stored in two arrays M_1 and M_2. The active nodes are determined by the backward search in the evaluation procedure.

To define one suitable crossover point, we define two possible crossover points $C_{\mathrm{P}1}$ and $C_{\mathrm{P}2}$ of the two parents. With information about the active nodes and the

length of the path, we can choose two possible crossover points. The possible crossover points C_{P1} and C_{P2} are chosen by chance in the range of the active function nodes which are stored in M_1 and M_2. The possible crossover points may not be input or output nodes. A general crossover point C_P is defined by choosing the smaller crossover point from C_{P1} and C_{P2}. The reason for this is that the subgraphs of the parents, which will be placed in front of or behind the crossover point of the offspring's genome should be balanced. The representation of CGP allows active paths of an individual, which can start in the middle or back of the graph. The subgraph which will be placed in front of the crossover point has to start at more leading active nodes. If C_P is defined as the possible point C_{P1}, the subgraph of P_1 in front of C_P will be placed in front of C_P in the offspring genome. The subgraph behind C_P of P_2 will be placed behind C_P in the offspring genome The crossover procedure produces a new genome that represents the offspring involving the phenotypes of both parents. In the case that two children should be produced, the crossover procedure is performed twice with two different general crossover points. Since the representation of CGP provides connections to any of the previous function nodes of the graph, performing only the *neighbourhood connect* could result in a monotone data flow of the resulting phenotype. An example of the crossover procedure is illustrated in Figure 2.

3 Experiments and Findings

We performed experiments in the problem domain of symbolic regression and Boolean function learning. To evaluate the search performance of the tested algorithms, we measured the number of fitness evaluations until the CGP algorithm terminated successfully (*fitness-evaluations-to-success*) and the best fitness value which was found after a predefined number of generations (*best-fitness-of-run*). We investigated a diverse set of popular GP benchmarks, including single and multiple output problems. The problems are listed in Table 2 and 3. We used a minimizing fitness function in all experiments. For the symbolic regression problems, the fitness of the individuals was represented by a cost function value. The cost function was defined by the sum of the absolute

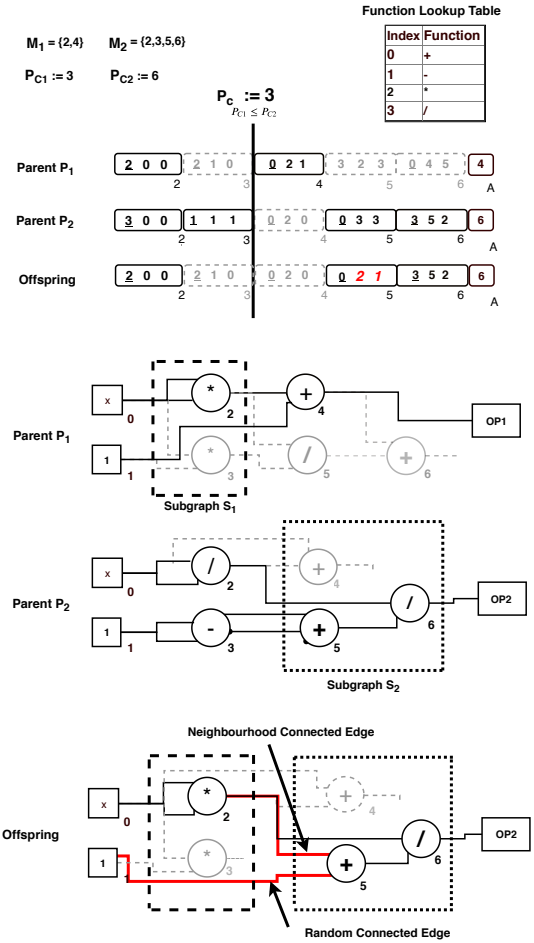


Figure 2: Example of the subgraph crossover technique. The subgraph crossover basically works similar to the single-point crossover except that the active nodes on both sides of the crossover point are preserved. The crossover point is chosen in a way that it is located between active function nodes. At the top of the figure, the arrays with the active nodes and crossover points are listed. Below this information, the genotypes and phenotypes of the parents and the offspring are shown, and the parts of the crossover are marked with dashed boxes.

difference between the real function values and the values of an evaluated individual. For the boolean parity even problems, the fitness was represented by the number of fitness cases for which the candidate solution failed to generate the correct value of the Even-Parity function. To evaluate the fitness on the multiple output problems, we defined the fitness value of an individual as the number of different bits to the corresponding truth table.
In addition to the mean values of the measurements, we calculated the standard deviation (SD) and the standard error of the mean (SEM). The algorithms which were used in our study are listed in Table 1. The best parameter configuration for each algorithm and problem has been determined with the help of meta-evolution. To classify the significance of our results, we used the Mann-Whitney-U-Test. The mean values are denoted $a^{\dagger}$ if the p-value is less than the significance level 0.05 and $a^{\ddagger}$ if the p-value is less than the significance level 0.01 compared to the $(1+4)$-CGP. Note that the mean values are **only** denoted with the significance level marker if the result of a certain algorithm is better than the result of the $(1+4)$-CGP. We performed 100 independent runs with different random seeds.
Table 4 and Table 5 show the results of the algorithm comparison in the Boolean domain. As visible, the Canonical-CGP and the $(\mu+\lambda)$-CGP perform better than the mutation-only CGP algorithms on various problems. The results of our experiments in the symbolic regression domain are shown in Table 6, Table 7 and Table 8. It is visible that the Canonical-CGP algorithm performs better than the mutation-only CGP algorithms on all tested problems.
The experiments demonstrate that the subgraph crossover can contribute to the search performance by using a canonical GA or $(\mu+\lambda)$-strategy. Moreover, the results of our experiments indicate that the predominance of the $(1+4)$-CGP and $(1+\lambda)$-CGP algorithms cannot be generalized in the Boolean domain. The experiments in the symbolic regression domain indicate that the use of the subgraph crossover is beneficial for the use of CGP and can contribute significantly to the search performance in this problem domain. Especially the search performance of the Canonical-CGP algorithm was superior to the $(1+4)$-CGP on all tested problems in the symbolic regression domain.

Table 1: List of the CGP algorithms

Identifier	Description
$(1+4)$-CGP	Traditional $(1+4)$-CGP algorithm
$(1+\lambda)$-CGP	Traditional $(1+\lambda)$-CGP algorithm
$(\mu+\lambda)$-CGP	$(\mu+\lambda)$-algorithm with subgraph crossover
Canonical-CGP	Canonical genetic algorithm (GA) with tournament selection and subgraph crossover

Table 2: Symbolic regression problems

Problem	Objective Function	Vars
Koza-1	$x^4+x^3+x^2+x$	1
Koza-2	x^5-2x^3+x	1
Koza-3	$x^6-2x^4+x^2$	1
Nguyen-4	$x^6+x^5+x^4+x^3+x^2+x$	1
Nguyen-5	$\sin(x^2)\cos(x)-1$	1
Nguyen-6	$\sin(x)+\sin(x+x^2)$	1
Nguyen-7	$\ln(x+1)+\ln(x^2+1)$	1
Keijzer-6	$\sum_i^x 1/i$	1
Pagie-1	$1/(1+x^{-4})+1/(1+y^{-4})$	2

Table 3: Boolean function problems

Problem	Number of Inputs	Number of Outputs
Parity-Even 3	3	1
Parity-Even 4	4	1
Parity-Even 5	5	1
Parity-Even 6	6	1
Parity-Even 7	7	1
Adder 1-Bit	3	2
Adder 2-Bit	5	3
Subtractor 2-Bit	4	3
Multiplier 2-Bit	4	4

Table 4: Results for the Boolean single-output problems evaluated by the number of fitness evaluations (FE) to termination

Problem	Algorithm	Mean FE	SD	SEM	1Q	Median	3Q
Parity-Even-3	$(1+4)$-CGP	3177	3417	±343	1246	2136	3760
	$(1+\lambda)$-CGP	2495	2919	±293	846	1534	2872
	Canonical-CGP	3107	3070	±307	1201	2104	3907
	$(\mu+\lambda)$-CGP	**1565**‡	**1517**	**±152**	**602**	**1168**	**1892**
Parity-Even-4	$(1+4)$-CGP	15420	14152	±1422	6292	10358	17726
	$(1+\lambda)$-CGP	16523	19168	±1926	6095	11276	18557
	Canonical-CGP	54967	47042	±4727	24813	40612	71851
	$(\mu+\lambda)$-CGP	**11135**‡	**8447**	**±845**	**5117**	**8527**	**14085**
Parity-Even-5	$(1+4)$-CGP	45542	33947	±3411	21524	36834	61222
	$(1+\lambda)$-CGP	**34375**‡	**28146**	**±2828**	**20685**	**27104**	**38941**
	Canonical-CGP	**28413**‡	**25538**	**±2566**	**23388**	**19640**	**34876**
	$(\mu+\lambda)$-CGP	43476	2055	±1022	23814	36188	57182
Parity-Even-6	$(1+4)$-CGP	199989	142915	±14291	107418	163234	242573
	$(1+\lambda)$-CGP	**118768**‡	**73682**	**±7368**	**65766**	**91577**	**156639**
	Canonical-CGP	242986	161762	±16257	134518	200196	309346
	$(\mu+\lambda)$-CGP	**110158**‡	**75163**	**±7516**	**63908**	**90676**	**135148**
Parity-Even-7	$(1+4)$-CGP	478055	301113	±30111	268210	393362	605372
	$(1+\lambda)$-CGP	441857	328539	±32853	226272	352254	545197
	Canonical-CGP	631568	548180	±54818	293613	453204	750792
	$(\mu+\lambda)$-CGP	**358420**‡	**246131**	**±24613**	**189278**	**303988**	**451667**

Table 5: Results for the Boolean multi-output problems evaluated by the number of fitness evaluations (FE) to termination

Problem	Algorithm	Mean FE	SD	SEM	1Q	Median	3Q
Adder-1Bit	$(1+4)$-CGP	1895	1856	$\pm$186	634	1252	2494
	$(1+\lambda)$-CGP	1415	1532	$\pm$154	508	1057	1640
	Canonical-CGP	2155	2018	$\pm$202	882	1521	2907
	$(\mu+\lambda)$-CGP	**1393**†	**1311**	**$\pm$132**	**496**	**954**	**1784**
Adder-2Bit	$(1+4)$-CGP	85667	84355	$\pm$8478	29506	58650	110794
	$(1+\lambda)$-CGP	73417	58589	$\pm$5888	33367	53654	94009
	Canonical-CGP	225652	200384	$\pm$20038	78247	158130	271717
	$(\mu+\lambda)$-CGP	68375	43229	$\pm$5361	32998	64006	98052
Multiplier-2Bit	$(1+4)$-CGP	11583	10469	$\pm$1046	5020	8524	14498
	$(1+\lambda)$-CGP	17664	21664	$\pm$2177	5400	9233	19262
	Canonical-CGP	30489	25700	$\pm$2582	13384	22696	22916
	$(\mu+\lambda)$-CGP	**11055**‡	**13281**	**$\pm$1334**	**3635**	**6693**	**13220**
Subtractor-2Bit	$(1+4)$-CGP	11029	13975	$\pm$13975	4642	6986	11878
	$(1+\lambda)$-CGP	**8377**‡	**9958**	**$\pm$1000**	**2989**	**6111**	**9579**
	Canonical-CGP	35829	41822	$\pm$4203	10346	20056	40698
	$(\mu+\lambda)$-CGP	15161	22388	$\pm$2250	5291	9671	16705

Table 6: Results for the algorithm comparison for the problems Koza 1, 2 & 3 evaluated by the number of fitness evaluations (FE) to termination

Problem	Algorithm	Mean FE	SD	SEM	1Q	Median	3Q
Koza-1	$(1+4)$-CGP	8675635	16681422	$\pm$1668142	441477	1814344	7045961
	$(1+\lambda)$-CGP	7370880‡	17384354	$\pm$1738435	204400	1050936	4294170
	Canonical-CGP	**663822**‡	**838546**	**$\pm$83854**	**135162**	**337950**	**710275**
	$(\mu+\lambda)$-CGP	7780751‡	15830735	$\pm$1583073	197284	1830312	6318740
Koza-2	$(1+4)$-CGP	8264426	19894512	$\pm$1989451	150140	888884	4378756
	$(1+\lambda)$-CGP	8191549	20275790	$\pm$2027579	94290	559028	4710848
	Canonical-CGP	**444118**‡	**95000**	**$\pm$286700**	**627550**	**29650**	**78800**
	$(\mu+\lambda)$-CGP	5729778	11021660	$\pm$1102166,	238156	1320880	5878696
Koza-3	$(1+4)$-CGP	600153	1214527	$\pm$121452	39076	177418	443038
	$(1+\lambda)$-CGP	753551	2535215	$\pm$253521	29528	120368	431318
	Canonical-CGP	**32870**‡	**57156**	**$\pm$10435**	**2488**	**6700**	**32713**
	$(\mu+\lambda)$-CGP	926857	3473467	$\pm$347347	28548	121040	362180

Table 7: Results of the algorithm comparison algorithm for the symbolic regression problems evaluated with the *best-fitness-of-run* method

Problem	Algorithm	Mean Best Fitness	SD	SEM	1Q	Median	3Q
Nguyen-4	$(1+4)$-CGP	0,68	0,55	±0,05	0,34	0,58	0,77
	$(1+\lambda)$-CGP	0,61	0,46	±0,04	0,35	0,54	0,74
	Canonical-CGP	**0,50**†	**0,28**	**±0,04**	**0,31**	**0,47**	**0,60**
	$(\mu+\lambda)$-CGP	**0,60**†	**0,40**	**±0,04**	**0,36**	**0,54**	**0,76**
Nguyen-5	$(1+4)$-CGP	0,45	0,42	±0,04	0,06	0,32	0,81
	$(1+\lambda)$-CGP	0,39	0,33	±0,03	0,08	0,27	0,63
	Canonical-CGP	**0,29**‡	**0,27**	**±0,03**	**0,05**	**0,20**	**0,40**
	$(\mu+\lambda)$-CGP	**0,28**‡	**0,25**	**±0,02**	**0,06**	**0,19**	**0,45**
Nguyen-6	$(1+4)$-CGP	0,54	0,66	±0,06	0,16	0,29	0,61
	$(1+\lambda)$-CGP	0,50	0,67	±0,06	0,15	0,22	0,50
	Canonical-CGP	**0,31**‡	**0,31**	**±0,03**	**0,15**	**0,24**	**0,40**
	$(\mu+\lambda)$-CGP	0,61	0,67	±0,06	0,16	0,35	0,67
Nguyen-7	$(1+4)$-CGP	0,79	0,48	±0,05	0,45	0,67	1,06
	$(1+\lambda)$-CGP	0,71	0,45	±0,04	0,44	0,67	0,76
	Canonical-CGP	**0,60**‡	**0,35**	**±0,03**	**0,36**	**0,60**	**0,68**
	$(\mu+\lambda)$-CGP	**0,62**‡	**0,40**	**±0,04**	**0,42**	**0,63**	**0,68**

Table 8: Results of the algorithm comparison algorithm for the symbolic regression problems evaluated with the *best-fitness-of-run* method

Problem	Algorithm	Mean Best Fitness	SD	SEM	1Q	Median	3Q
Keijzer-6	$(1+4)$-CGP	$3,78$	$2,61$	$\pm 0,26$	$2,16$	$3,24$	$4,59$
	$(1+\lambda)$-CGP	$3,38$	$2,52$	$\pm 0,25$	$2,41$	$3,03$	$3,158$
	Canonical-CGP	$\mathbf{2,81}^{\dagger}$	$\mathbf{1,13}$	$\mathbf{\pm 0,11}$	$\mathbf{1,78}$	$\mathbf{2,90}$	$\mathbf{3,75}$
	$(\mu+\lambda)$-CGP	$\mathbf{2,88}^{\dagger}$	$\mathbf{1,09}$	$\mathbf{\pm 0,1}$	$\mathbf{2,25}$	$\mathbf{3,14}$	$\mathbf{3,15}$
Pagie-1	$(1+4)$-CGP	$128,18$	$48,19$	$\pm 4,81$	$87,81$	$119,09$	$161,08$
	$(1+\lambda)$-CGP	$120,75$	$44,95$	$\pm 4,49$	$86,14$	$120,91$	$155,06$
	Canonical-CGP	$\mathbf{98,52}^{\ddagger}$	$\mathbf{50,57}$	$\mathbf{\pm 5,08}$	$\mathbf{59,04}$	$\mathbf{85,31}$	$\mathbf{130,04}$
	$(\mu+\lambda)$-CGP	$\mathbf{99,74}^{\ddagger}$	$\mathbf{41,246}$	$\mathbf{\pm 4,12}$	$\mathbf{65,32}$	$\mathbf{95,79}$	$\mathbf{131,76}$

References

[1] J. Koza. Genetic Programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Dept. of Computer Science, Stanford University, June 1990.

[2] Julian F. Miller. An empirical study of the efficiency of learning boolean functions using a cartesian genetic programming approach. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, Florida, USA, 1999.

[3] Julian Francis Miller. Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines*, 21(1):129–168, 2020.

[4] Roman Kalkreuth, Guenter Rudolph, and Andre Droschinsky. A new subgraph crossover for cartesian genetic programming. In *EuroGP 2017: Proceedings of the 20th European Conference on Genetic Programming*, volume 10196 of *LNCS*, pages 294–310, Amsterdam, 19-21 April 2017. Springer Verlag.

[5] Roman Kalkreuth. A comprehensive study on subgraph crossover in cartesian genetic programming. In *Proceedings of the 12th International Joint Conference on Computational Intelligence, IJCCI 2020, November 02-04, 2020*. ScitePress, 2020.

Dieser Tagungsband enthält die Beiträge des 30. 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 26. – 27.11.2020 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 978-3-7315-1051-2