Challenges in Deep Learning Based Forecasting of Time Series with Calendar-driven Periodicities

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Benedikt Maximilian Heidrich

Tag der mündlichen Prüfung: 04.12.2023

1. Referent: Prof. Dr. Veit Hagenmeyer
2. Referent: Prof. Dr. Tao Hong
3. Referent: Prof. Dr. Ralf Mikut

Benedikt Heidrich
Challenges in Deep Learning Based Forecasting of Time Series with Calendar-driven Periodicities
Dissertation, 2020 - 2023
Reviewers: Prof. Dr. Veit Hagenmeyer Prof. Dr. Tao Hong, and Prof. Dr. Ralf Mikut
Supervisors: Veit Hagenmeyer and Ralf Mikut

Karlsruhe Institute of Technology
Institute for Automation and Applied Informatics
Department of Informatics

Acknowledgement

I am grateful for the various encounters with people and the experiences during my PhD.

First, I thank my supervisors, Veit Hagenmeyer and Ralf Mikut. I thank Veit Hagenmeyer for the opportunity to do my doctorate at the IAI. I thank Ralf Mikut for his guidance and help in structuring my research, for the discussions that were very valuable and for the extensive and detailed feedback. In addition, I would also like to thank Tao Hong for agreeing to be a reviewer of my thesis and Achim Streit for being a committee member. Furthermore, I would like to give special thanks to Franz Kiraly for his valuable feedback on the draft of my thesis.

Second, I thank the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI for its financial support.

Third, I would like to thank my colleagues in the OC and ML4Time Group: Adrian, Angelo, Dorina, Frederik, Henrieke, Kaleb, Katharina, Luca, Marian, Matthias, Max, Nicole, Nils, Oli, Rebecca, Stefan, Tillmann, Xinjang. Also, thank you to all my other colleagues at IAI, especially Andreas, Bernadette, and Moritz. Furthermore, I am thankful to all my co-authors for our collaborative projects. I also want to thank the students who supported me as working students or during their Seminar, Bachelor's, or Master's thesis.

Finally, I also would like to say thank you to my friends and my family. Specifically, I want to thank my parents for always supporting me on my journey and for providing me with all I needed for this journey. Furthermore, I thank my brother and sister for their support and friendship. I would also like to thank Anna-Lena for listening to me, giving me hugs, and motivating me in difficult times. Finally, I would also like to thank all of my friends. Without you, this wouldn't be possible.

Abstract

Throughout history, forecasting has been crucial across various civilizations, influencing imagination and critical decisions. In contemporary times, it has become indispensable across diverse domains, where accurate predictions drive decision-making — e.g., in the operation of the electrical grid or traffic planning. However, forecasting is still associated with various challenges. Thus, the present thesis delves into neural network-based time series forecasting challenges, focusing on four aspects: missing data, missing scenarios, concept drifts, and periodicities. This thesis aims to solve these challenges using conditional generative models or profile-based methods. Conditional generative models are the main component of the proposed approaches to face the missing scenario challenge and the challenge of a small training data size. Moreover, a conditional generative model is also the main component of the proposed forecaster used for energy peak load forecasting in the BigDEAL challenge. The proposed profile-based methods are promising in handling concept drift and support neural networks in time series forecasting. In particular, they can be supportive since they cover specific aspects of the time series that need not be captured by the machine learning model anymore if the profiles are integrated with the machine learning models. Consequently, the proposed Profile Neural Network (PNN) and Probabilistic Profile Neural Network (ProbPNN) can beat state-of-the-art forecasting models on two datasets regarding the forecast quality and the computational time.

Publications

Main Publications

- B. Heidrich *et al.*, “Forecasting Energy Time Series with Profile Neural Networks,” in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, Association for Computing Machinery, 2020, pp. 220–230
- B. Heidrich *et al.*, “Adaptively coping with concept drifts in energy time series forecasting using profiles,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 459–470
- B. Heidrich *et al.*, “Boost short-term load forecasts with synthetic data from transferred latent space information,” *Energy Informatics*, vol. 5, 2022
- B. Heidrich *et al.*, “Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks,” *Applied Intelligence*, vol. 53, no. 8, pp. 8826–8843, 2023
- B. Heidrich *et al.*, “ProbPNN: Enhancing Deep Probabilistic Forecasting with Statistical Information,” *arXiv preprint arXiv:2302.02597*, 2023
- B. Heidrich *et al.*, “Using conditional Invertible Neural Networks to Perform Mid-Term Peak Load Forecasting,” *IET Smart Grid*, 2023

Further Publications

- B. Heidrich *et al.*, “Delay-robust Estimation of the Reproduction Number and Comparative Evaluation on Generated Synthetic Data,” *medRxiv*, 2020
- B. Heidrich *et al.*, “pyWATTS: Python Workflow Automation Tool for Time Series,” *arXiv preprint arxiv:2106.10157*, 2021
- O. Neumann *et al.*, “Smart Data Representations: Impact on the Accuracy of Deep Neural Networks,” in *Proceedings 31. Workshop Computational Intelligence*, 2021

- M. Turowski *et al.*, “Enhancing anomaly detection methods for energy time series using latent space data representations,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 208–227

- M. Turowski *et al.*, “Modeling and generating synthetic anomalies for energy and power time series,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 471–484

- D. Werling *et al.*, “Towards line-restricted dispatchable feeders using probabilistic forecasts for PV-dominated low-voltage distribution grids,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 395–400

- D. Werling *et al.*, “The Impact of Forecast Characteristics on the Forecast Value for the Dispatchable Feeder,” in *Proceedings of the Fourteenth ACM International Conference on Future Energy Systems (2023)*, 2023, pp. 59–71

- S. Meisenbacher *et al.*, “AutoPV: Automated photovoltaic forecasts with limited information using an ensemble of pre-trained models,” in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 386–414

- S. Meisenbacher *et al.*, “Automating day-ahead forecasting of photovoltaic power generation: Model design, monitoring, and adaption,” in *ETG Congress 2023*, 2023

- K. Phipps *et al.*, “Creating Probabilistic Forecasts from Arbitrary Deterministic Forecasts using Conditional Invertible Neural Networks,” *arXiv preprint arxiv:2302.01800v1*, 2023

- K. Phipps *et al.*, “Loss-Customised Probabilistic Energy Time Series Forecasts Using Automated Hyperparameter Optimisation,” in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 271–286

- M. Hertel *et al.*, “Transformer Training Strategies for Forecasting Multiple Load Time Series,” in *12th DACH+ Conference on Energy Informatics*, 2023

Talks on conferences without published papers

- F. Kiraly, B. Heidrich, M. Parker, M. Walter.[1], "sktime - python toolbox for time series: pipelines and transformers," pyDATA Global 2022, 1.3. Dezember 2022.

[1]Underlined names are the speakers.

- B. Heidrich, K. Phipps, S. Meisenbacher, M. Turowski, O. Neumann, R. Mikut, V. Hagenmeyer. "Non-Sequential Machine Learning Pipelines with pyWATTS;" deRSE 2023, 20.-21. February 2023.
- B. Heidrich, O. Neumann, M. Hertel, V. Hagenmeyer, R. Mikut "Using Conditional Invertible Neural Networks To Perform Mid-Term Peak Load Forecasting," International Symposium on Forecasting 2023, 24.-28. June 2023.
- F. Kiraly, B. Heidrich. "sktime python toolbox for time series: pipelines and benchmarking," PyCon CZ 23, 15.17. September 2023.

Core Developer of Open Source Software

- sktime
- pyWATTS
- pyWATTS Pipeline

Contributing to Open Source Software

- Sktime:
  - **Issues:** 3159, 3177, 3219, 3291, 3348, 4650, 4651, 4709, 4766, 4814, 4898, 4923, 4924, 4993, 5067, 5177, 5182, 5200, 5205, 5217, 5218, 5225, 5236, 5237, 5238, 5248, 5254, 5280.
  - **Merged Pull Requests:** 3167, 3255, 4652, 4781, 4998, 5044, 5201, 5208, 5209, 5210, 5235, 5242, 5246.
- Further libraries and Frameworks (pandas, xarray, keras)
  - **Issues:** Keras: 230, Pandas: 42442, 43223, xarray 5492
  - **Merged Pull Requests:** Pandas: 42459

Contents

List of Figures

List of Tables

Acronyms

ADWIN	Adaptive Windowing
AR	Autoregression
ARMA	Autoregressive Moving Average
biGRU	bidirectional Gated Recurrent Unit
CD	Concept Drift
cINN	conditional Invertible Neural Network
CNN	Convolutional Neural Network
COTGAN	Conditional Optimal Transport GAN
CR	Coverage Rate
CRPS	Continuous Ranked Probability Score
cTimeGAN	conditional TimeGAN
cVAE	conditional Variational Autoencoder
DA	Data Augmentation
DeepAR	DeepAR network
DICR	Distance to Ideal Coverage Rate
EIA	Error Intersection Approach
ELM	Extreme Learning Machine
ES	Exponential Smoothing
ES-RNN	Exponential Smoothing Recurrent Neural Network
EWM	Exponential Weighted Moving
EWMA	Exponential Weighted Moving Average
EWMV	Exponential Weighted Moving Variance
FCN	Fully-Connected Neural network
FEDD	Feature Extraction for Explicit Concept Drift Detection
GAN	Generative Adversarial Network
gru	GRUGated Recurrent Unit
INN	Invertible Neural Network
LOTUS	Law Of The Unconscious Statistician
LR	Linear Regression
LSFE	Latent Space-based Forecast Enhancer
LSTM	Long-Short Term Memory
MA	Moving Average
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MLP	Multi-Layer Perceptron
MPE	Mean Percentage Error
N-BEATS	Neural Basis Expansion Analysis for interpretable Time Series forecasting network
N-BEATSx	Neural Basis Expansion Analysis for interpretable Time Series forecasting with exogenous variables network
nCRPS	normalised Continuous Ranked Probability Score
N-HiTS	Neural Hierarchical Interpolation for Time Series Forecasting network
NN	Neural Network
nMAE	normalised Mean Absolute Error
NNQF	Nearest Neighbour Quantile Filter
nPL	normalised Pinball Loss
nRMSE	normalised Root Mean Squared Error
OARNN	Online Adaptive Recurrent Neural Network
OS-ELM	Online Sequential Extreme Learning Machine
PDH	Profile-based Framework for Concept Drift Handling
PH	Page-Hinkley
PNN	Profile Neural Network
ProbPNN	Probabilistic Profile Neural Network
PSF	Profile Standard-Deviation Forecast
PSO	Particle Swarm Optimisation
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
SVR	Support Vector Regression
TFT	Temporal Fusion Transformer
TL	Transfer Learning
t-SNE	t-distributed Stochastic Neighbourhood Embedding
TSTR	Train-Synthetic Test-Real
VAE	Variational Autoencoder
wMAE	Weighted and capped Mean Absolute Error

Part I

Motivation

Introduction

1

Across all civilisation epochs, forecasting was important for the people and fascinated them. The old Egyptians forecasted the yearly flood in the lower reaches of the Nile to increase their agricultural yields [121]. These forecasts of the floods were so important that the priests seem to have been responsible for them [121]. While these forecasts were strongly related to religion, forecasts became more secular over time. For example, Halley proposed a method to predict the orbit of Halley's Comet using Newton's law of gravitation, thereby proving this law experimentally [42].

In contemporary times, for various applications in many domains, time series forecasting is essential since they need forecasts as input. For example, supply and demand must always be met in the energy domain. However, this is not easy due to the fluctuating behaviour of the increasing share of renewable energy sources. Thus, downstream applications such as dispatchable feeders [142] exist to stabilise the grid using forecasts of rooftop photovoltaic power generation and the residential load of the corresponding building. Another domain where time series forecasts are important is the traffic domain since they can support and improve traffic control or mitigate congestion [84].

As already indicated, sufficient accuracy of the forecast is essential. However, due to the different properties of time series, it may be challenging to achieve good forecasting accuracy. The literature describes various of these challenges. From these challenges, this thesis focuses on a subset of four challenges that are relevant for neural network-based time series forecasting and provides solutions. These four challenges are missing scenarios, small training data size, concept drifts, and periodicities. The missing scenario challenge describes that different recorded scenarios are often missing for the robustness evaluation [67]. Such scenarios might be scenarios with specific, rarely observed changes in the time series. The challenge of a small training data size is strongly related to the missing scenarios challenge. However, in contrast to the missing scenarios, which state that certain scenarios are unavailable in the data, this challenge describes that the amount of available training data is too small for the needs of data-driven methods [64], [81]. The concept drift challenge is associated with the problem that changes in the time series

– so-called concept drifts– may harm the forecaster's performance [17], [21], [43], [85]. Last, the periodicities challenge is related to machine learning models, which often struggle to learn periodic behaviour [91], [124], [155].

To contextualise and fully comprehend the research questions and contributions, the remainder of the first part provides the foundation for time series forecasting and generation in Chapter 2. Additionally, the first part introduces the four challenges in more detail and provides related work concerning these challenges in Chapter 3.
Afterwards, the second part describes how generative models can solve the challenge of missing scenarios in Chapter 4 and small training data size in Chapter 6. Furthermore, this thesis shows that mid-term and long-term forecasts, which are related to generating scenarios, can be solved with generative models in Chapter 5.
Part three focuses on how profiles can solve time series forecasting challenges. In particular, Chapter 8 shows how profiles can support machine learning models in handling concept drifts, and Chapter 9 shows how profiles can support neural networks with periodicities.
Finally, the last part discusses the overall results of this thesis in Chapter 10, wraps up this thesis and provides an outlook on further related research questions in Chapter 11.

2 Forecasting and Generation of Time Series with Calendar-driven Periodicities

This chapter provides the necessary foundations to follow this thesis. First, it introduces the considered time series class with calendar-driven periodicities. Second, the foundations of forecasting and, third, generating time series are provided. Finally, the last section of this chapter describes the general experimental setup used in this thesis.

2.1 Time Series with Calendar-Driven Periodicities

As mentioned, the considered time series class is the class of regular time series with calendar-driven periodicities. This section first presents the used notation of time series to introduce the class of time series with calendar-driven periodicities afterwards.

2.1.1 Time Series Notation

A time series describes measurement values ordered in time. E.g., the number of passengers in an aeroplane for each year is a time series, where the number of passengers is the measured value (for more examples, see [63]).
To formally describe time series, this thesis uses the following time series notation

$$\mathcal{X}_t = (x_{t_1}, x_{t_1}, ..., x_{t_n}), \tag{2.1}$$

where x_{t_i} are the time series values, t_i is a time index from the ordered set of time indexes $\mathcal{T} = (t_1, t_1, ..., t_n)$ with $t_i - t_j = (i - j) \cdot \Delta$ with $i > j$, Δ the sample time of the time series as time difference[1], and n is the length of the time series.

[1]This means that the distance between two measured consecutive values is equidistant, i.e., the considered time series are regular with a fixed sample time Δ.

This thesis requires that each time index is associated with a time stamp. This requirement enables a simple definition of calendar extraction functions based on these time stamps. Calendar extraction functions are, for example, $h_{\text{hour}}(t)$ or $h_{\text{cal}}(t)$ that provide the hour of the day of the time index t or an arbitrary but fixed set of calendar information of the time stamp t.

The introduced time series definition (Equation (2.1)) allows all time series to have an arbitrary length. However, many machine learning models can not cope with arbitrary lengths. E.g., non-recurrent neural networks have a fixed input and output size. Thus, to use such methods, this thesis transforms the time series into samples, which is a fixed-length cutout of a time series containing consecutive values. Note the used samples overlap. The sample starting at time index t_i is defined as

$$\mathbf{x}_{t_i,h} = (x_{t_i}, x_{t_{i+1}}, ..., x_{t_{i+h-1}}), \tag{2.2}$$

where x_t is a time series measurement at time index t_i with $i \in [1, n-h]$, n is the length of the time series, and h is the sample size. To specify the j-th entry of the sample $\mathbf{x}_{t_i,h}$, this thesis uses a superscript $\mathbf{x}_{t_i,h}{}^{\{j\}}$.
The sampling can be reversed by using a merger. Consider the time series samples $\mathbf{x}_{t_i}$, $i \in [1, n]$, which can be aggregated together

$$x_{t_k} = \text{Median}(\{\mathbf{x}_{t_i,h}{}^{\{j\}} \mid i + j = k\}), \tag{2.3}$$

where $i + j = k$ ensures that only entries with the same time are aggregated[2].

2.1.2 Probablistic Interpretation of Time Series

In general, time series $\mathcal{X}_t$ can not only be considered as a sequence of measurement data, but they can also be considered as a realisation of a stochastic process $\{X_t\}$, with X_t being a random variable for time index t. Thus, a time series measurement x_t can also be considered as a realisation of the random variable X_t.

2.1.3 Time Series with Calendar-Driven Periodicities

Time series are ubiquitous, and thus, a wide variety of time series exists. For example, there exists time series describing stock prices, gross domestic products of countries, the pandemic development, the residential electricity demand, or the

[2]Instead of the median, the mean or some quantiles are applicable too.

traffic occupancy. These different time series may have differing characteristics, e.g., different trends, periodicities, or drifting behaviours. Based on these different characteristics, time series can be categorised into different time series classes. This thesis focuses on time series with calendar-driven periodicities. I.e., the periodicities of the time series depend, for example, on the hour of the day, the day of the week, and the month of the year. Furthermore, these periodicities also interact and overlay. From the exemplary mentioned time series, only the residential electrical load and the traffic occupancy time series contain calendar-driven periodicities and are part of this time series class. Considering this specific class of time series with calendar-driven periodicities enables specialised algorithms that use calendar information as prior. For example, in the electricity domain, a widely used approach for estimating the required amount of energy is the usage of so-called type days, which describe the load curves of exemplary users. Thus, in the following, this thesis formally describes calendar-driven periodicities.

To formally define periodicities in time series, trend-free[3] time series with random components are considered. The random component causes the time series to have constant unpredictable fluctuations. Thus, $x_t = x_{t+\rho},\ \forall t \in [1, L]$, as definition for periodicities with $\rho = i_1 \cdot \eta_1 \wedge i_2 \cdot \eta_2 \wedge ... \wedge i_k \cdot \eta_k,\ i_1, i_2, ..., i_k \in \mathbb{N}$ being a multiplicative of all period lengths η_i is to strict. Thus, in the case of trend-free time series, it is only required that measurements separated by the multiplicative of the k periods (ρ) are similar to each other, i.e.,

$$x_t \approx x_{t+\rho} \approx x_{t+2\cdot\rho} \approx ... \approx x_{t+\mathcal{P}\cdot\rho},\ \mathcal{P} \in \mathbb{N}. \tag{2.4}$$

To check if a time series contains periodicities, the autocovariance of the time series can be evaluated since the autocovariance is higher at the multiples of the periods' multiplicative (ρ) [63].

Suppose such periodicities occur in time series, and their periods correspond to calendar information, such as the day of the week. In that case, this thesis refers to them as time series with calendar-driven periodicities. Exemplary time series that do not belong to this class are stock time series, gross domestic product time series, or pandemic time series, such as the number of infections of COVID, since all of these time series do not show clear daily, weekly, or yearly periodicities.

[3]With detrending algorithms, trend-free time series can be obtained [63].

2.2 Foundations of Time Series Forecasting

Time series forecasting is essential for a variety of applications in different domains. For example, it is important to balance energy system's load and supply. Therefore, downstream applications such as dispatchable feeders [142] require demand and supply forecasts as input. Another domain in which time series forecasts are important is the traffic domain [84]. In this domain, accurate traffic forecasts can enable traffic control mechanisms that mitigate congestion or increase air quality.
Thus, this section first formally introduces the task of time series forecasting and the corresponding notation used in this thesis. Afterwards, it provides an overview of different forecasting approaches that are important for the present thesis.

2.2.1 Definition

In time series forecasting, the aim is to estimate yet unobserved future time series measurements of a time series $\mathcal{X}_t$. More formally, given a time series $\mathcal{X}_t$ with observed values until time t_i, a time series' forecast with a horizon h is $\mathbf{\hat{x}_{t_i,h}} = (\hat{x}_{t_i}^{\{1\}}, \hat{x}_{t_i}^{\{2\}}, ..., \hat{x}_{t_i}^{\{h\}})$, where $\hat{x}_{t_i}^{\{j\}}$ is the estimated value at time t_{i+j} based on the information available at t_i. To distinguish forecasts from actual time series values, this thesis indicates them using a hat, e.g., $\hat{x}$. A forecasting algorithm can use the observed time series values up to time index t_i and other exogenous time series $(\mathcal{X}_{t'}^{\text{exog}})$[4] on which the time series $\mathcal{X}_t$ depends[5]. Formally, a forecasting method is a parametrised function f_Θ

$$\mathbf{\hat{x}_{t_i,h}} = (\hat{x}_{t_i}^{\{1\}}, \hat{x}_{t_i}^{\{2\}}, ..., \hat{x}_{t_i}^{\{h\}}) = f_\Theta(\mathcal{X}_t, \mathcal{X}_{t'}^{\text{exog}}), \tag{2.5}$$

with parameters Θ that are determined using historical data. Note, for simplicity, this definition uses all past data of $\mathcal{X}_t$ and all available data of $\mathcal{X}_{t'}^{\text{exog}}$. This can also include forecasts of exogenous time series such as forecasted weather time series. However, in general, the forecasting algorithm only receives and considers small parts of $\mathcal{X}_t$ and $\mathcal{X}_{t'}^{\text{exog}}$. Furthermore, also note that Equation (2.5) introduces a point forecasting method. However, often, it is also important to consider the uncertainty. Thus, instead of forecasting points, we can consider the future values as random variables $X_{t_{i+1}}$, $X_{t_{i+2}}$, ..., $X_{t_{i+h}}$ for which the forecasting method estimates probability distributions $\hat{F}_{t_i}^{\{1\}}$, $\hat{F}_{t_i}^{\{2\}}$, ..., $\hat{F}_{t_i}^{\{h\}}$. For more information on probabilistic forecasting, see Gneiting and Katzfuss [39].

[4] i has not to be equal to i' to allow the algorithm to use future values of $\mathcal{X}_{t'}^{\text{exog}}$.
[5] Static features might be transformed to a static time series and thus be part of $\mathcal{X}_{t'}^{\text{exog}}$.

2.2.2 Methods

Concerning the forecasting methods, there exist different methods in the literature. Thus, this subsection shortly explains the time-series model-based, machine learning, and hybrid approaches. Note that further forecasting approaches exist, for example, physical model-based forecasting, which is often applied in weather forecasting [8], photovoltaic power forecasting [119], or wind power forecasting [76]. However, since this thesis focuses on models that are applicable to time series with calendar-driven periodicities in general, regardless of the underlying system, this thesis does not cover physical model-based forecasting in the following. In addition, note forecasting models are often embedded in pipelines with additional preprocessing and postprocessing steps, resulting in many hyperparameters that need to be tuned [99]. However, this thesis focuses on the forecasting models. Thus, automated tuning approaches for forecasting pipelines are also not covered in the following. Instead, the focus is on providing as much background information on time series models, machine learning, and hybrid forecasting approaches as required to understand this thesis instead of providing an extensive overview of the different approaches. For a comprehensive overview, see, for example, [63], [112].

Time Series Model-based Forecasting In time series forecasting, a popular approach is time series modelling. A resulting model for a specific time series can be used to perform forecasts. Time series models try to incorporate specific knowledge of the time series. For example, Autoregression (AR) methods weight past time series values to predict the next values [63]. Other exemplary methods are the Moving Average (MA) and Exponential Smoothing (ES) methods. MA methods model the time series using its mean plus a linear regression of the past error terms [63], and ES by an exponentially weighted moving average [63].
These basic methods are often combined to provide more advanced time series model-based forecasting methods. For example, the Autoregressive Moving Average (ARMA) model consists of an autoregressive and a moving average part [63].
Besides that, another important concept of time series model-based forecasts is that the time series can be composed of different components. An important decomposition is

$$\mathcal{X}_t = \mathcal{X}_{t_i}^{\text{Season}} + \mathcal{X}_{t_i}^{\text{Trend}} + \mathcal{X}_{t_i}^{\text{Remainder}}, \tag{2.6}$$

with $\mathcal{X}_{t_i}^{\text{Season}}$ being the seasonality part of the time series, $\mathcal{X}_{t_i}^{\text{Trend}}$ the trend part, and $\mathcal{X}_{t_i}^{\text{Remainder}}$ the remainder of the time series [63][6]. With this decomposition, different

[6]This thesis uses the additive decomposition. However, there exists also the multiplicative decomposition.

algorithms are applicable for the different components, and afterwards, by adding them together, the final forecast can be made.

Machine Learning Forecasting Methods In contrast to the time series modelling approach, there are also purely data-driven approaches. I.e., instead of providing a model structure that consists of different integration and moving average components, the data-driven approach tries to learn to forecast by just using data. From this approach, important methods for this thesis are linear regression, XGBoost [16], and different neural network architectures as fully connected [126], convolution [73], and recurrent neural networks [56]. Furthermore, there also exists more advanced neural network architecture, e.g. GRUs [18] and LSTMs [57], or the more recently proposed transformers such as Temporal Fusion Transformer (TFT) [80], Informers [154], and Autoformers [145]. These purely data-driven methods are often enhanced with prior knowledge to model and consider specific aspects of the context to which the models are applied. E.g. for Short-Term Electricity Load forecasting, calendar, temperature information and their interaction are important additional features that improve the forecasts [59], [60]. Even more specific knowledge, such as if *Demand Side Management is active*, can be useful for forecasting electricity time series [59]. Another example is the electricity spot price forecasting. In this context, additional information based on that context can be considered to improve the forecasts [68]

Hybrid Forecasting Methods Besides the purely data-driven and the time series model-based approach, hybrid forecasting methods also exist.

These forecast methods aim to integrate methods and ideas from the time series model-based approaches into machine learning models. For example, Exponential Smoothing Recurrent Neural Network (ES-RNN) incorporates ES into a Recurrent Neural Network (RNN). A combination of using AR terms in RNN is used by the DeepAR network (DeepAR) network [123]. Furthermore, another kind of hybrid neural network architecture for time series forecasting integrates the idea of time series decomposition into neural networks, e.g., N-BEATS [105], N-BEATSx [102], or N-HiTS [15].

2.3 Foundations of Time Series Generation

Time series generation is related to time series forecasting. Both tasks are about creating new data points. However, they differ in the sense that in time series forecasting, the new points are forecasts of an existing time series, while in time series generation, the new data form a new time series. This similarity of the tasks makes generation methods interesting for time series forecasting. Additionally, time series generation has the potential to support time series forecasting by providing additional data.
Based on these considerations, this thesis uses and investigates the potential benefit of time series generation for forecasting. Thus, in the following, this section first introduces time series generation formally and the three different time series generation methods used in this thesis afterwards.

2.3.1 Definition

The time series generation task aims to create time series that are realistic and indistinguishable from real ones. As mentioned previously, time series can be considered as realisations of stochastic processes with a random variable for each time step Knowing the probability distributions for all random variables X_t allows the creation of time series. However, this requires knowing infinitely many probability distributions. This issue can be solved by recurrent generative methods that handle such families of random variables directly by updating the RNNs state. However, this thesis uses non-recurrent generative methods with a fixed input and output size. Thus, this thesis mitigates the problem by not trying to generate a time series as a realisation of a stochastic process $\{X_t\}$. Instead, it creates samples from the time series and assumes that these samples are realisations of a random variable X that follows the unknown distribution P_X. To be able to create time series samples from this unknown distribution, this thesis uses deep generative methods. Therefore, different approaches to deep generative methods exist. However, in the end, each of the considered deep generative methods maps the latent space random variable Z that follows a known distribution P_Z to the random variable X with an unknown distribution P_X

$$X = g_\theta(Z). \tag{2.7}$$

Note, to be able to create a time series out of the generated time series samples, this thesis also provides calendar information to the mapping g_θ, the resulting function is $g_\theta(Z, \mathbf{x}^{\text{cal}}_{t_i,h})$, with $\mathbf{x}^{\text{cal}}_{t_i,h}$ being a sample of the calendar information. A resulting assumption is that this information guides g_θ to create time series samples with the correct temporal structure that enables the usage of Equation (2.3).

2.3.2 Methods

This section presents the three general types of deep generative methods used in this thesis.

Generative Adversarial Network (GAN) The GAN introduced in 2014 by [41] consists of two neural networks, namely the generator and the discriminator. The underlying idea of the GAN is adversarial learning, i.e., the generator and discriminator compete against each other and thus improve their respective performances. Thereby, the generator generates data that looks as realistic as possible, while the discriminator aims to classify correctly if the data is generated by the generator or is real data.
During joint training, the discriminator aims to maximise its accuracy in classification while the generator aims to minimise the accuracy of the discriminator by generating time series that are hard to distinguish from real time series.
During the inference or the time series generation, only the generator is needed. The generator takes noise as input to generate a time series similar to the time series in the training data.

Variational Autoencoder (VAE) Similar to the GAN, a VAE also consists of two neural networks, namely the encoder and the decoder [72]. However, their tasks differ from the generator's and discriminator's tasks. The underlying idea of the VAE is that the time series information is encoded in a normally distributed latent space. Thereby, the task of the encoder is to capture the underlying information of the time series data by encoding it in the latent space, while the decoder's task is to reconstruct the time series based on the data in the latent space.
During joint training, the VAE aims to minimise the reconstruction error and the Kullback-Leibler divergence between the latent space and a normal distribution. Minimising the reconstruction error ensures that the data look realistic, while minimising the Kullback-Leibler divergence ensures that the latent space is normally distributed.
For generating the data, only the decoder is required. As the latent space is normally distributed, normal distributed random noise can be fed as input to the decoder, which uses this noise to generate synthetic time series.

Invertible Neural Network (INN) In contrast to the GAN and VAE, the INN consists only of one network. The underlying idea of an INN is to learn a bijective mapping between a normal distributed latent space and the real data space to realise a normalising flow. The bijectivity is ensured by affine coupling layers such as GLOW [71] or RealNVP [27].
During the training, the INN aims to minimise the Kullback-Leibler divergence between the data mapped to the latent space and the normal distribution by applying the Change-Of-Variable formula [4].
For generating synthetic time series, the bijectivity of the INN is leveraged by using the reverse mapping from the latent space to the real data space. More specifically, as the latent space is normally distributed, the normally distributed random noise can be used as input and mapped to the real data space, resulting in time series samples.

2.4 Experimental Setup for Time Series

This section provides the basic experimental setup configurations that are used in this thesis for evaluating the proposed methods. The first subsection introduces the used metrics. The second presents the datasets. Finally, the last subsection presents the hardware and software setup.

2.4.1 Used Evaluation Criteria

This subsection describes how the method that this thesis proposes is assessed. Thereby, this subsection is structured as follows. First, this thesis describes how forecasts are evaluated. Afterwards, the evaluation strategies used for synthetic time series are described. Finally, this thesis briefly describes how the computational effort is measured.

Evaluating Forecasts

For evaluating time series forecasts, this thesis uses the train test split backtesting strategy. I.e., the time series is split temporally in a training and testing time series at time index t_i, resulting in $(x_{t_1}, x_{t_2}, ..., x_{t_i})$ as training and $(x_{t_{i+1}}, x_{t_{i+2}}, ..., x_{t_n})$ as testing time series. After training a model on the training time series, the model performs for the time steps $t_i, t_{i+1}, ...t_{n-h}$ forecasts for the next h time steps

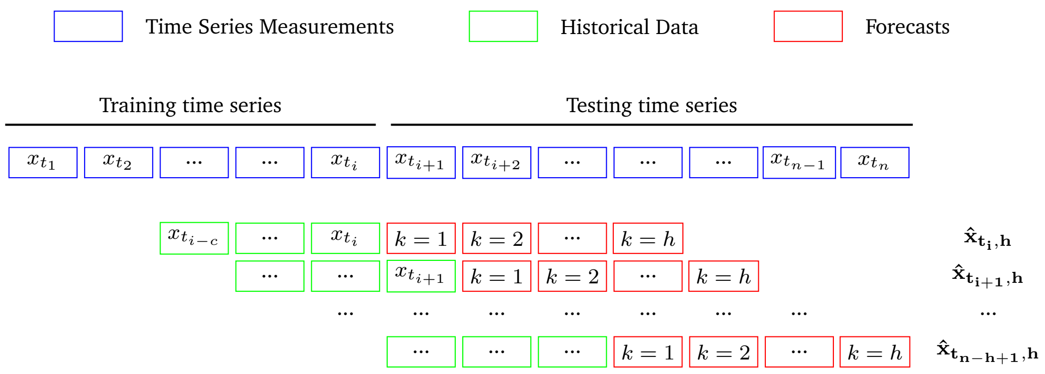


Figure 2.1.: To evaluate the time series forecasts $\hat{\mathbf{x}}_{\mathbf{t_i,h}}$, this thesis uses the backtesting approach on the testing dataset.

$\hat{\mathbf{x}} = (\hat{\mathbf{x}}_{\mathbf{t_i,h}}\hat{\mathbf{x}}_{\mathbf{t_{i+1},h}}, ...\hat{\mathbf{x}}_{\mathbf{t_{n-h},h}})$. Thereby, for providing the forecast for the next h values at time index t_i, the forecaster uses the $c+1$ most recent values at time index t_i, i.e., $x_{t_{i-c}}, ..., x_{t_i}$. For the notation of $\hat{\mathbf{x}}_{\mathbf{t_i,h}}$ compare Equation (2.5). Using these forecasts and the testing time series, metrics are calculated to assess the forecast quality. Note, for the probabilistic forecasts, not a point is forecasted but a probability distribution. Thus, we use $\hat{F}_{t_i}$ instead of $\hat{x}_{t_i}$ $\hat{F}_{t_i}^{\{k\}}$ instead of $\hat{x}_{t_i}^{\{k\}}$, and $\hat{\mathbf{F}}$ instead of $\hat{\mathbf{x}}$

In the following, this thesis presents metrics for point and probabilistic forecasts. Furthermore, also insight-related metrics are described.

Metrics for point forecasts This thesis uses four different metrics for point forecasts. Note, to simplify the definitions, the definitions consider only one forecast. For considering all forecasts of the test time series the resulting scores of the metrics could be averaged.
The first metric is the **Mean Absolute Error (MAE)**, which is defined as

$$\mathrm{MAE}(\hat{\mathbf{x}}, \mathcal{X}_t) = \frac{\sum_{j=i}^{n-h} \sum_{k=1}^{h} (|\hat{x}_{t_j,h}^{\{k\}} - x_{t_{j+k}}|)}{(n-i-h+1) \cdot h}, \tag{2.8}$$

where n is the length of the time series, $\hat{x}_{t_j,h}^{\{k\}}$ is the specific entry k of the forecast $\hat{\mathbf{x}}_{\mathbf{t_j,h}}$ at t_j from the set of forecasts $\hat{\mathbf{x}}$, $x_{t_{j+k}}$ is the time series value of the time series $\mathcal{X}_t$ at time index t_{j+k}, and h the forecasting horizon. In some scenarios, different levels of deviations from the ground truth cause different costs. Thus, the **Weighted**

and capped Mean Absolute Error (wMAE) exists to reflect this. The wMAE is defined as

$$\mathrm{wMAE}(\hat{\mathbf{x}}, \mathcal{X}_t) = \frac{\sum_{j=i}^{n-h} \sum_{k=1}^{h} \mathrm{weighted}(|\hat{x}_{t_j,h}^{\{k\}} - x_{t_{j+k}}|)}{(n-i-h+1) \cdot h}, \tag{2.9}$$

where $\mathrm{weighted}$ is a weighting function and the remaining symbols are defined as for the MAE.

Besides MAE based metric, this thesis also uses the **Root Mean Squared Error (RMSE)**, which punishes huge deviations stronger than the MAE. It is defined as

$$\mathrm{RMSE}(\hat{\mathbf{x}}, \mathcal{X}_t) = \sqrt{\frac{\sum_{j=i}^{n-h} \sum_{k=1}^{h} (\hat{x}_{t_j,h}^{\{k\}} - x_{t_{j+k}})^2}{(n-i-h+1) \cdot h}}, \tag{2.10}$$

where the symbols are defined as for the MAE.

A disadvantage of the MAE and RMSE is that they are not data scale-invariant and, thus, not comparable across different time series. To overcome this issue, this thesis uses the nMAE and normalised Root Mean Squared Error (nRMSE), whose generalized definition is

$$\mathrm{nMetric}(\hat{\mathbf{x}}, \mathcal{X}_t) = \frac{\mathrm{Metric}(\hat{\mathbf{x}}, \mathcal{X}_t)}{\sigma_{\mathcal{X}_t}}, \tag{2.11}$$

where $\mathrm{Metric}(\hat{\mathbf{x}}, \mathcal{X}_t)$ is either the RMSE or MAE, $\sigma_{\mathcal{X}_t}$ is the standard deviation of the time series $\mathcal{X}_t$, and the remaining symbols are defined as for the MAE.

Similar to the normalised scores, the Mean Absolute Percentage Error (MAPE) can also be used to compare the prediction performance across different time series. The **MAPE** is calculated as

$$\mathrm{MAPE}(\hat{\mathbf{x}}, \mathcal{X}_t) = \frac{100}{(n-i-h+1) \cdot h} \sum_{j=i}^{n-h} \sum_{k=1}^{h} \left| \frac{\hat{x}_{t_j,h}^{\{k\}} - x_{t_{j+k}}}{x_{t_{j+k}}} \right|, \tag{2.12}$$

where the symbols are defined as for the MAE. Note the MAPE is only suited for time series without values near zero. Note that the values of the time series on which this metric is applied are always greater zero.

Probabilistic Prediction Metrics As metrics for probabilistic forecasts, this thesis uses the normalised Continuous Ranked Probability Score (nCRPS) and the normalised Pinball Loss (nPL). The first probabilistic metric used is the **nCRPS** [95]. It is defined as

$$\mathrm{nCRPS}(\hat{\mathbf{F}}, \mathcal{X}_t) = \frac{\sum_{j=i}^{n-h} \sum_{k=1}^{h} \int_{\mathcal{R}} (\hat{F}_{t_j}^{\{k\}}(y) - \mathbb{1}(y \geq x_{t_{j+k}}))^2 dy}{\sigma_{\mathcal{X}_t} \cdot (n - i - h + 1) \cdot h}, \tag{2.13}$$

where n is the length of the time series, $\hat{F}_{t_j}^{\{k\}}$ is the cumulative distribution function of the k-th entry of the probabilistic forecast $\hat{F}_{t_j,h}$ at t_j from the set of forecasts $\hat{\mathbf{F}}$, $x_{t_{j+k}}$ is the time series value of the time series $\mathcal{X}_t$ at time index t_{j+k}, h the forecasting horizon, and $\sigma_{\mathcal{X}_t}$ is the standard deviation of the time series $\mathcal{X}_t$. Note, according to Trinh *et al.* [130] normalising the Continuous Ranked Probability Score (CRPS) by using the standard deviation preserves its properties. The second probabilistic metric used is the **normalised Pinball Loss (nPL)**. This metric assesses the quality of a quantile forecast by normalising and averaging the quality of all considered quantiles, i.e.,

$$\mathrm{nPL}(\hat{\mathbf{F}}, \mathcal{X}_t) = \frac{1}{\sigma_{\mathcal{X}_t} \cdot (n - i - h + 1) \cdot h \cdot \mid Q \mid} \sum_{\alpha \in Q} \sum_{j=i}^{n-h} \sum_{k=1}^{h} \begin{cases} (x_{t_{j+k}} - \hat{F}_{t_j}^{\{k\}}(\alpha)) \cdot \alpha & \text{for } x_{t_{j+k}} \geq \hat{F}_{t_j}\{k\}(\alpha) \\ (\hat{F}_{t_j}^{\{k\}}(\alpha) - x_{t_{j+k}}) \cdot (1 - \alpha) & \text{for } \hat{F}_{t_j}\{k\}(\alpha) > x_{t_{j+k}} \end{cases}, \tag{2.14}$$

where $\hat{F}_{t_i}^{\{k\}}(\alpha)$ is the quantile forecast for the quantile α, Q is the set of all considered quantiles, $\mid Q \mid$ is the cardinality of Q, and the remaining symbols are defined as for the nCRPS.

Insight Related Metrics Regarding gaining insights, this thesis uses four metrics. The first is the **Mean Percentage Error (MPE)**, which measures the forecast bias and is defined as

$$\mathrm{MPE}(\hat{\mathbf{x}}, \mathcal{X}_t) = \frac{100}{(n - i - h + 1) \cdot h} \cdot \sum_{j=i}^{n-h} \sum_{k=1}^{h} \frac{x_{t_{j+k}} - \hat{x}_{t_j,h}^{\{k\}}}{x_{t_{j+k}}}, \tag{2.15}$$

where the symbols are defined as for the MAE. Note that $x_{t_{j+k}}$ has always to be greater than zero. The MPE can be interpreted as follows. A negative MPE means that the forecast overestimates the time series, while a positive means that the forecast underestimates the time series. Furthermore, the distance of the MPE to zero provides information about the severity of the bias. The larger the distance to zero, the bigger the bias.

For gaining insights into the probabilistic forecast, this thesis uses the Coverage Rate (CR). It measures the share of ground truth that lies in the interval between two quantiles, i.e.,

$$\mathrm{CR}_{q_1,q_2}(\hat{\mathbf{F}}, \mathcal{X}_t) = \frac{\sum_{j=i}^{n-h} \sum_{k=1}^{h} \mathbb{1}(\hat{F}_{t_j}^{\{k\}}(q_1) < x_{t_{j+k}} < \hat{F}_{t_j}^{\{k\}}(q_2))}{(n-i-h+1) \cdot h}, \tag{2.16}$$

where q_1 and q_2 are the lower and upper considered quantiles, $\mathbb{1}(\hat{F}_{t_i}^{\{k\}}(q_1) < x_{t_{j+k}} < \hat{F}_{t_i}^{\{k\}}(q_2))$ is the indicator function that counts how many values are between the quantiles q_1 and q_2, and the remaining symbols are defined as for the CRPS. An ideal forecast would result in a CR of $q_2 - q_1$. Using this observation, this thesis introduces the **DICR**, which sums up all distances of forecaster's CRs and the ideal CRs, resulting in

$$\mathrm{DICR}(\hat{\mathbf{F}}, \mathcal{X}_t) = \sum_{q_i, q_j \in Q} \mid (CR_{q_i,q_j}(\hat{\mathbf{F}}, \mathcal{X}_t) - (q_j - q_i)) \mid, \tag{2.17}$$

where Q is the set of all considered quantile tuples and $q_j > q_i$.

Finally, to assess the improvement concerning a specific metric, this thesis applies skill scores. A skill score is defined as

$$\mathrm{Skill}_{\text{metric}}(\hat{\mathbf{x}}_{\text{method}}, \hat{\mathbf{x}}_{\text{base}}, \mathcal{X}_t) = 1 - \frac{\mathrm{metric}(\hat{\mathbf{x}}_{\text{method}}, \mathcal{X}_t)}{\mathrm{metric}(\hat{\mathbf{x}}_{\text{base}}, \mathcal{X}_t)}, \tag{2.18}$$

where metric is the selected base metric, $\hat{\mathbf{x}}_{\text{method}}$ the prediction of the method that is evaluated, $\hat{\mathbf{x}}_{\text{base}}$ the prediction of the baseline, and the remaining symbols are defined as for the MAE[7]. The skill score can then be interpreted as follows. If the score is greater than one, the method's prediction is worse than that of the base and vice versa if the skill score is smaller than one.

Evaluating Synthetic Time Series

In contrast to time series forecasting, metrics cannot be applied directly, since no ground truth exists. Thus, this thesis uses the same evaluation approach as Yoon *et*

[7]In general, this definition can be extended to probabilistic forecasts. However, this thesis applies this metric only on point forecasts.

al. [151] and examines the usefulness, fidelity, and diversity.

Usefulness The usefulness examines if a downstream application benefits from the generated time series. This thesis selects a forecaster as a downstream application and applies the Train-Synthetic Test-Real (TSTR) [31] evaluation approach. I.e., a forecaster is trained on the synthetic data and tested on the real data. The better the forecaster performs on the real data, the better the corresponding generation method. To assess the performance of the forecaster, the previously introduced forecasting metrics can be used.

Fidelity The fidelity examines the indistinguishability of the generated and the real time series samples. Therefore, this thesis merges the same number of generated and the real time series samples and splits them afterwards into a training and a test set. Note that in both datasets, there should be the same number of generated and real time series samples. Correspondingly, the labels of each generated time series sample are 'generated' and of each real time series sample 'real', resulting in a binary classification problem. Using these training data, a classifier is trained to distinguish the generated and the real time series samples. Afterwards, the trained classifier is applied on the test set. Based on the classifier's prediction, the accuracy is calculated as follows:

$$\text{accuracy}(\hat{y}, y) = \frac{n(\hat{y}_i = y_i)}{n}, \tag{2.19}$$

with n being the size of the test set, $\hat{y}$ being the classifier's prediction on the test set, y being the labels of the test set, and $n(\hat{y}_i = y_i)$ being the number of samples where $\hat{y}_i = y_i$. Finally, this accuracy is used to determine the discriminative score as $\mid \text{accuracy} - 0.5 \mid$ for the two-class problem of distinguishing synthetic from real time series[8].

Diversity The diversity assesses if the generated time series is similarly distributed as the original time series. Therefore, t-distributed Stochastic Neighbourhood Embedding (t-SNE) is applied to map the generated time series samples and the actual time series samples into a two-dimensional plane and visualise it. The more similar the points of the generated and the actual time series samples are, the better the corresponding generation method according to the diversity.

[8]This thesis subtracts 0.5 since the test data set contains the same number of generated and real time series samples.

Table 2.1.: Meta data of the used for time series data. Note, the row Number exogenous Time Series (Num. exogenous TS) compromises only the exogenous times that are part of the dataset and do not correspond to the time stamp.

	Electricity	Traffic	BigDEAL
Num. TS	370	963	3
Num. used TS	130	963	3
Num. exogenous TS	0	0	6
Time Span	01/2011 – 12/2014	01/2008 – 03/2009	01/2015 – 12/2018.
Resolution	Hourly	Hourly	Hourly
Domain	Energy	Traffic	Energy

Evaluating Computational Effort

Finally, this thesis also evaluates the computational effort by measuring the **training time** in seconds.

2.4.2 Used Time Series Datasets

During the evaluation, this thesis uses three different time series datasets. Besides the time series data itself, calendar information is also considered. Thus, this subsection first describes the datasets and, afterwards, the calendar information encodings.

Used Datasets

As mentioned, this thesis uses three datasets– the electricity, BigDEAL and traffic datasets. Their meta-data are provided in Table 2.1.

Electricity Dataset The first dataset is the "ElectricityLoadDiagrams20112014 Data Set"[9] from the UCI Machine Learning Repository [28].

This dataset contains real-world time series of 370 clients that have a quarter-hourly resolution and mostly cover the period from January 2011 to December 2014. This data set contains clients with different consumption behaviours, such as factories and hotels [120]. From the electricity dataset, all time series with less than eight hours of consecutive zeros or not almost constant values over several days are used since such values might relate to measurement errors or non-available data.

[9]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

BigDEAL Dataset The data set used in the BigDEAL challenge is an electrical load time series dataset that spans a period from 1. January 2015 to the 31. December 2018 (this corresponds to 35064 time series values). It contains the electrical load time series of three substations located in the USA with an hourly resolution. Besides these three substations, the dataset also comprises exogenous temperature data from six weather stations located nearby the substations.

Traffic Dataset The fourth data set is the Traffic data set[10]. This data set contains time series representing the occupancy rate of car lanes in the San Francisco Bay Area. This thesis uses all time series from this data set for the evaluation. Regarding the value range of the time series, their range is between zero and one since they measure the occupancy rate.

Calendar Features

This thesis considers time series with calendar-driven periodicities. Thus, calendar information is an important feature. Thus, this thesis extracts them from the timestamp. Thereby, two different kinds of features are extracted. The first are cyclic features such as hour of the day, day of the week, or month of the year. This thesis encodes these features trigonometrically to map semantically similar values to numerically similar values. I.e., the hour of the day is encoded as a sine function $\sin(2 \cdot \pi \cdot \text{hour}/24)$ and cosine function $\cos(2 \cdot \pi \cdot \text{hour}/24)$ or the month of the year as sine function $\sin(2 \cdot \pi \cdot \text{month}/12)$ and cosine function $\cos(2 \cdot \pi \cdot \text{month}/12)$. The second type of feature is binary calendar information as is-weekend or not. For encoding this kind of feature, this thesis uses Boolean variables.

2.4.3 Used Hardware and Software

For all experiments, the present thesis uses the same hardware and software setup described in the following.

Used Hardware Due to the variety of experiments and the allocation of the servers, this thesis uses different hardware setups. The used hardware setups are described in Table 2.2.

[10] `https://archive.ics.uci.edu/ml/datasets/PEMS-SF`

Table 2.2.: The hardware setup used for the evaluations in this thesis.

	Setup 1	Setup 2
CPU cores	96	32
RAM	126GB	64GB
GPU	no GPU	Nvidia Titan RTX 24GB

Used Software As a programming language, this thesis uses Python[11] and different Python libraries for data handling, modelling, experiment automation, and metric calculation.

In machine learning, data-handling libraries are essential. The present thesis uses Numpy [44] for working with multi-dimensional datasets without index information, Pandas [96], [127] for two-dimensional data with time information, and xarray [62] for working with the automation framework pyWATTS [45].

As machine learning libraries, this thesis uses the following libraries. For generative deep learning models, this thesis uses PyTorch [109] and for generative models that are based on invertible neural networks or normalising flows, FreIA[12], which builds up on PyTorch. For non-generative neural networks, this thesis uses Keras [19] and Tensorflow [1]. For non-deep learning methods, this thesis uses SKLearn [110], which provides various machine learning methods that share the same interface, XGBoost [16], and pyoselm[13]. As a further non-deep learning library, we use River [100] for the Concept Drift (CD) detection algorithms.

For automated conduction of the experiments and an easier evaluation of the results, this thesis uses pyWATTS [45]. pyWATTS consists of two parts. The first is a library of different time series algorithms. The second is the graph pipeline, which enables the implementation of complex non-sequential machine learning pipelines.

While most of the used metrics are implemented by the author of this thesis, `properscoring` is used as existing implementation for the ensemble-based nCRPS[14].

[11] `https://www.python.org/`
[12] `https://github.com/VLL-HD/FrEIA`
[13] `https://github.com/leferrad/pyoselm`
[14] `https://github.com/properscoring/properscoring`

Challenges on Time Series Forecasting

3

In time series forecasting, multiple challenges exist. In the following, this thesis provides an overview of challenges in general before presenting the four considered in more detail.

3.1 Time Series Forecasting Challenges in Literature

This section provides an overview of challenges for time series forecasting identified in the literature. Furthermore, this section identifies the research gap that this thesis addresses.

3.1.1 Related Work

Several publications describe challenges in time series forecasting (e.g., [21], [43], [81], [85], [90]). Some of the described challenges are domain-agnostic, and others domain-specific. Regarding the domain agnostic challenges, this subsection provides an overview of three different groups of challenges: non-linearity, data availability and quality, and diversity of possible downstream applications-related challenges.

Regarding challenges related to the complexity and the non-linearity of the time series, different publications state that the complexity and non-linearity itself are challenging [17], [84], [90], [126] but also the interplay between the considered time series and exogenous time series [35], [43], [64] and the collinearity and the cofounding of time series [43], [81]. Another aspect of the non-linearity is non-stationarity and concept drift in time series, which can harm the forecasters' performance [17], [21], [43], [85]. Also related to the complexity of time series are seasonalities. If modelled correctly, they can support time series forecasters [93]. However, they can also be challenging. For example, difficulties can arise if multiple interacting seasonalities [35], [58] exist. Furthermore, these seasonalities and periodicities can also challenge pure machine learning models [91], [155]. The problems of pure machine learning models can correspond to their missing periodic bias [155] or an overfitting problem that arises if the complexity of such models is increased to capture periodicities [91]. The bad generalisation of such machine learning forecasting models is also observed in [81], [85].

Regarding challenges associated with data availability and quality, the quality may be harmed by anomalies, missing values or data (gaps in time series) [85], [129], [132], [139], which need to be handled, e.g., by applying imputation methods as the copy-paste imputation [139]. Regarding the data availability, few data might be challenging [64], [81], since machine learning approaches require a sufficient amount of data resulting in the challenge of a small training data size. However, too much data can also be challenging, especially in real-time environments. Thus, there is also the challenge of being computationally efficient [17], [53], [58], [65], [85], [90], [91]. Strongly related to the group of data-related challenges is also the missing scenarios challenge, which describes that not enough scenarios may exist to assess the robustness of forecasters or decision-making systems [67].

Due to the diversity of downstream applications, there exist different challenges related to that. Reported challenges are potentially varying forecast horizons [64], uncertainties if accurate prediction intervals or point forecasts are more important [35], and uncertainties about what characteristics of point forecasts are important for the downstream application [141]. Finally, also a challenge is the forecast feedback loop that may arise from the interplay of the downstream applications and forecast [43].

Regarding domain-specific challenges, this subsection mentions three exemplary domains in the following. In the traffic domain, it can be challenging that various spatial information needs to be combined [65]. Regarding the electricity domain, e.g. Gasparin *et al.* [37], Haben *et al.* [43], and Henze *et al.* [53] mention that forecasts are also required for low aggregated entities. However, such time series are often spiky and noisy. Finally, the last domain is the data centres. In this domain, [64] mentions that the non-linear interaction between the data centres metrics that needs to be forecasted is challenging.

3.1.2 Research Gap

As shown in the related work, various publications focus on challenges. While some of these publications are reviews or comments providing an overview of different challenges in time series forecasting, e.g., [43], [58], [81], [91] others aim to solve one particular challenge, such as the cold-start problem [97] or concept drift [5]. However, even though new methods address specific challenges, the recentness of the publications in the related work indicates that the challenges are still relevant. Thus, the present thesis focuses on four challenges – namely the missing scenario, small training data size, concept drift, and periodicity challenge – in forecasting time series with calendar-driven periodicities and proposes novel approaches to cope with these challenges.

3.2 Considered Challenges

This section describes the four challenges considered in this thesis. These challenges are the missing scenario, small training data size, concept drift, and periodicity challenge. Thereby, each challenge is described as follows: First, the challenge is introduced. Afterwards, an overview of how existing solutions handle the respective challenge and their shortcomings are provided.

3.2.1 Missing Scenarios

The first considered challenge is the missing scenario challenge. The missing scenario challenge describes the problem that recorded time series only cover the scenario associated with the context of the measurements. E.g., in a business use case, if sales figures are only recorded during a stable market situation, only the scenario of a stable market is covered by the recorded time series. However, there might exist further scenarios, yet unobserved scenarios, that might arise in the future. E.g., in business use cases, this challenge comprises the question of how a certain market assumption (e.g. growing or shrinking market) of the future affects the sales of a product [67]. Furthermore, this challenge also comprises unobserved scenarios that might occur due to concept drifts, longer periods with unusual exogenous variables, or periods with unusually low or high values. Such scenarios are important to assess the forecasting models' quality, reliability, and robustness [67]. Thus, this thesis provides an approach to create synthetic time series that follows specific scenarios. Whereby the scenario can be determined or controlled by the user. I.e., the user can specify if a time series with an increasing or decreasing trend should be created. As a special form of scenario generation, mid-term and long-term forecasting with exogenous variables can be considered since such forecasts are scenarios that should fit well with the provided exogenous data. Thus, this thesis considers such forecasts together with the missing scenario challenge.

Related Work

To solve the missing scenario challenge, synthetic time series can be generated. Popular methods for time series generation are deep generative models. The following describes the three most common deep generative model categories.
The first considered category comprises the different kinds of Generative Adversarial Networks (GANs). For time series generation, there exist several specialised architectures. E.g., Xu *et al.* [147] introduce COT-GAN, a generative model that uses causal optimal transport theory for implementing a temporal constraint in the loss function of the GAN. TimeGAN, presented by Yoon *et al.* [151], adapts the traditional GAN architecture by including a stepwise supervised loss function using the original data to capture stepwise conditional distributions in the data and recreates the transition dynamics from real sequences [151]. Moreover, Yoon *et al.* [151] use an embedding network to create reversible mappings between features and latent representations for reducing the dimensionality of the adversarial learning space. The Recurrent Conditional GAN (RCGAN) introduced by [31] focus on synthetic medical data. This model uses LSTM networks for the discriminator and the generator. Moreover, RCGAN can consider conditional information about the time series that should be generated. A further GAN that uses conditional information is proposed by [75]. This GAN uses a Wasserstein distance for creating time series conditioned on date-related information like the month of the year and further information about the time series that should be created.

The second category are Variational Autoencoder (VAE)s [72], [117] and conditional Variational Autoencoder (cVAE) [125]. An exemplary work on electrical load data is done by [108], which uses a convolutional network for the encoder and the decoder of the VAE. A recurrent VAE is proposed by Das *et al.* [23]. This architecture is based on LSTMs for the encoder network, a dilated convolutional layer, and additional LSTM layers for the decoder. A fully connected network-based VAE is presented by Esling *et al.* [30] to generate novel audio.

The third category this related work considers is Invertible Neural Network (INN). Kim *et al.* [69], for example, propose FloWaveNet for generating raw audio by combining a conditioned flow-based network and a non-causal WaveNet. More precisely, FlowWaveNet uses as coupling layers those from RealNVP [27] and includes a non-causal WaveNet [135]. WaveGlow proposed by [116] is another INN applied to sequential data. It combines elements from WaveNet [135] and Glow [71]. The third considered INN in this thesis is proposed by [38], which uses it for generating daily synthetic load time series. It uses coupling layers from NICE [26]. Furthermore, the proposed INN is based on the processing of images and thus needs to convert input time series into images and generate images back into time series. A further INN used for scenario generation is proposed by [20]. The authors uses RealNVP [27] for generating daily wind power scenarios to optimise the profit of wind farm operators.

Shortcomings of Generation Methods

The models presented in the previous related work focus on generating time series that are as realistic as possible and useful for downstream applications such as forecasting models. However, to the best of the author's knowledge, they do not focus on the problem of providing controllability for the non-stationarity and periodicities, which is essential in generating controlled scenarios. Both shortcomings are explained in the following[1]. For both shortcomings, this thesis considers the generation process $g_\theta(Z, \mathbf{x}^{\text{cal}})$ introduced before in Section 2.3. Regarding the missing non-stationarity, consider that the generation process uses samples $\mathbf{z}$, $i \in [1, N]$ from the latent space as realisations of the random variable Z with the known distribution P_Z and calendar information $\mathbf{x}^{\text{cal}}$. In this case, the generated time series samples $\hat{\mathbf{x}}$ are realisations of the random variable transformation $g_\theta(Z, \mathbf{x}^{\text{cal}})$, which expected value according to the Law Of The Unconscious Statistician (LOTUS) [122] is

$$\mathbb{E}[g_\theta(Z, \mathbf{x}^{\text{cal}})] = \int g_\theta(\mathbf{z}, \mathbf{x}^{\text{cal}}) P_Z \, d\mathbf{z}. \tag{3.1}$$

Thus, if the mapping $g_\theta(Z, \mathbf{x}^{\text{cal}})$ only depends on the random variable Z, fixed parameters θ, and $\mathbf{x}^{\text{cal}}$, all generated samples with the same calendar information (e.g. hour of the day) have the same expected value, i.e. $\mathbb{E}[\hat{\mathbf{x}}] = \mathbb{E}[g_\theta(\mathbf{z}, \mathbf{x}^{\text{cal}})]$. The same argument applies to the variance since the variance of a random variable X is defined as $\sigma_X^2 = \mathbb{E}[X^2] - (\mathbb{E}[X])^2$. Thus, LOTUS [122] can be used again to show that the variance is also the same for all generated time series segments (for details, see [52]).

Concluding, existing generation methods cannot vary the statistical properties of generated time series segments with the same calendar information. Therefore, these methods cannot control non-stationarity in generated time series[2]. The second

[1]Note, for better readability, this subsection omits the subscript (t_i, h) when writing the samples, i.e. this subsection writes $\mathbf{z}$ and $\mathbf{x}^{\text{cal}}$ instead of $\mathbf{z}_{t_i,h}$ and $\mathbf{x}_{t_i,h}{}^{\text{cal}}$.

[2]Existing methods, such as a recurrent neural network with teacher forcing, can generate non-stationary time series segments of arbitrary length if trained on non-stationary data sets. However, such methods adapt their weights to reproduce the non-stationarity in the training data (i.e., subsume the non-stationarity in the parameters θ). Thus, all the time series segments generated by such a method contain only the non-stationarities identical to those in the training data. Therefore,

shortcoming is the not controllable periodicities in generated time series. All generated time series samples are realisations of the same random variable transformation $g_\theta(Z, \mathbf{x}^{\text{cal}})$. Thus, the corresponding autocovariance simplifies to the variance, i.e.

$$\text{cov}(g_\theta(Z, \mathbf{x}^{\text{cal}}), g_\theta(Z, \mathbf{x}^{\text{cal}})) = \sigma^2_{g_\theta(Z, \mathbf{x}^{\text{cal}})}. \tag{3.2}$$

Together with the equality of the variance, the resulting autocovariance is also the same for all generated time series segments with the same calendar information. As a result, existing generation methods cannot create additional recurring and different autocovariance structures not defined by calendar information. Furthermore, these periodicities cannot be varied in their magnitude since the calendar information fully determines them. Thus, with the observation that periodic time series have a reoccurring autocovariance structure, it is possible to conclude that these methods cannot control periodicities in generated time series. To overcome both shortcomings, this thesis proposes a mechanism by using conditioning mechanisms in generative models not only to improve the results as Esteban *et al.* [31] and Lan *et al.* [75], but also to control the non-stationarity and periodicity in generated time series. The resulting research questions are:
RQ1: Is it possible to use the conditioning mechanism to control the non-stationarity and periodicity in generated time series?
RQ2: Is it possible to apply generative models conditioned on appropriate statistical and weather information to generate mid and long-term peak energy load forecasts?

3.2.2 Small Training Data Size

The second considered challenge is the small training data size challenge. This challenge refers to the problem that machine learning models require training data, e.g., one year [146], two years [150], or three years [89] of data to perform time series forecasts. However, for specific time series and one scenario, more than the available data is often needed to train a machine learning forecasting model. Such limitations of the training data occur, for example, in building-level electricity load forecasts, for a new building for which no measurements or only a few exist yet. Furthermore, changes in the data, such as concept drift, can make old data unusable for the training and thus also limit the amount of available data.

it is impossible to control the non-stationarity across multiple time series segments to generate time series with desired non-stationarities.

Related Work

Different approaches exist to solve the small training data size challenge. Commonly used and successful approaches are Transfer Learning (TL) [61] and Data Augmentation (DA) [7]. TL aims to transfer knowledge from source data to target data. For example, encoding features for the source and the target data might be similar, and thus, the knowledge of encoding features for the source data could be reused (or transferred) for the target data. Thereby, two tasks need to be solved. The first is about how knowledge is transferred, and the second is about identifying what knowledge (source data) should be transferred. Since this thesis does not propose a new transfer scheme; instead, it proposes a new way of creating knowledge that can be transferred, this related work focus on the second task. For the second task, different methods aim to find similar time series by analysing the correlation between source and target data [40], [83], [107], [128], by identifying information-rich time series [78], by aligning the time series with dynamic time warping and calculating the Jensen-Shannon divergence to determine the similarities [149], or by comparing the performance on the target data of encoders trained on different source data [148]. Besides identifying the most appropriate source data, other methods aim to make the source and the target data more similar, e.g., by normalising the data and removing trend and seasonal information [118] or by decomposing the data [140]. Data augmentation is the second approach often used to solve the small training data size challenge. In the following, this thesis focuses on three different data augmentation approaches. The first is the deep generative model-based approach. Thereby, for example, GANs [24], [111] or VAEs [25] can be used to generate additional time series data. The second considered DA approach is the time series combination. This approach uses different existing time series and recombines them to create new ones, e.g., by averaging [22], [34], recombining frequency components using empirical mode decomposition [2], [153], or by tuning the parameters of a mixture autoregressive model [66]. The last data augmentation approach is the application of transformation on a single time series, e.g., by adding noise [88] or using manipulations such as rotation, permutation, jittering, and scaling [32].

Shortcomings

As mentioned, the considered time series class depends on exogenous data (at least calendar information). Thus, the transformation and combination-based approaches are not well applicable since it is difficult to model the unknown dependency structures. Note if enough diverse data is available, these data augmentation techniques can be combined with seasonal bins to overcome this shortcoming. In contrast, the deep generative model approaches seem more promising since they can learn hidden dependencies. However, they are mainly for two reasons difficult to apply. First, often, they are trained on existing time series. Thus, newly created time series are similar to other time series in the source data, but there is no guarantee that they are similar to the specific target time series as required in this challenge. Second, deep generative models can also be trained on the target time series directly; however, if the amount of data of the target time series is limited, as in this challenge, deep generative models do not generalise well. This limitation of available data for the target also often hinders the application of the transformation-based approaches since they require a sufficiently diverse data set to work well.

This thesis aims to overcome the issue of deep generative models and enable their generation capabilities to augment limited data of a specific time series. Therefore, this thesis proposes a new sampling strategy in the deep generative models' latent space. The resulting research question this thesis aims to answer is:

RQ3: Is it possible to locate a specific time series in a deep generative model's latent space and use this information to guide the data augmentation process?

3.2.3 Concept Drift (CD)

The third challenge is the CD challenge. Normally, forecasting methods are trained on historical data and applied to future data afterwards. This approach implicitly assumes that the future data looks similar to the past data. However, this is not always the case since data can change over time. Such changes are called Concept Drift (CD). They can severely harm forecasting models' accuracy and negatively affect downstream applications' performance. The following four kinds of CD are normally distinguished. A sudden concept drift describes an abrupt change in a time series. Such concept drifts may occur in the street lane occupancy time series if a lane is closed due to construction. The gradual CD refers to a transition phase, where an old and a new concept are alternately active with increased activity of the new concept over time. An example might be that a new server system replaces an old one. In such cases, both systems are used for a certain amount of time, whereby the load on the newer one is increased with time. The third concept drift is the incremental CD. It occurs when slightly different concepts consecutively replace the former concepts. E.g., hiring staff could lead to an incremental drift in the electricity load time series of an office building[3]. In time series forecasting, incremental and

[3]Assuming that not all staff is hired at once.

gradual concept drifts result in similar changes. Thus, this thesis does not distinguish between them in the following and uses the term gradual CD. Last, the recurring CD describes the situation if an old concept reappears. Such CDs may arise in the electricity consumption of factories that can produce different products that require different amounts of energy.
This challenge is relevant since all these concept drifts can occur in time series and harm the forecasters' performance.

Related Work

There exist different approaches to cope with concept drift. Thus, this thesis provides an overview of the three most widely used approaches to cope with concept drift. For an extensive overview of concept drift handling, see [36], [87], [156].
The first approach comprises online and incremental learning algorithms. This approach updates the forecasting model after each n incoming data $(n \in \mathbb{N})$. While, in general, this approach is applicable to all forecasting models, there also exist specialised online forecasting models. E.g. the Online Adaptive Recurrent Neural Network (OARNN) consists of RNN in an online learning framework, which updates the RNN and the normalisation parameters after each time step [33], the Online Sequential Extreme Learning Machine (OS-ELM) extends the Extreme Learning Machine (ELM) to be online capable and robust against concept drifts [79], or for fuzzy forecasting models, the borders of the fuzzy set can be dynamically adapted [10], [82]. A disadvantage of this approach is that updates are performed even if there is no concept drift, leading to higher computational effort and often worse forecasts since the models are trained on small datasets. The second approach aims to fix that issue by detecting concept drift. E.g., Cavalcante and Oliveira [13] equips the OS-ELM and Cabello-López *et al.* [11] a linear regression with a drift detection algorithm to avoid these unnecessary updates. In general, detection-based retraining is applicable to all forecasting models. Thereby, different detection methods exist such as the Page-Hinkley (PH) detector [29], Adaptive Windowing (ADWIN) detector [9], or Feature Extraction for Explicit Concept Drift Detection (FEDD) [12].
The last approach to cope with concept drifts considered in this thesis is ensembling. Thereby, the ensemble must be diverse [14], i.e. the different concepts and scenarios should be covered by different ensemble members. Exemplary ensemble models for handling concept is a Particle Swarm Optimisation (PSO)-based approach [103], switching-based ensembles that contain models that are specialised for drifting and non-drifting periods [5], [6], [86].

Shortcomings

Regarding these three approaches, all have their shortcomings. E.g., online and incremental learning approaches are often expensive since unnecessary updates of the models are regularly performed. On the other hand, detecting concept drifts is often difficult since the adaption methods need to be tuned for the specific datasets. Finally, the ensemble's performance depends on containing an expert for the drifting scenario. Furthermore, ensembles can be expensive depending on the used models in the ensemble.

Thus, this thesis presents a method that aims to be more robust against concept drift. This method assumes that in many time series, only the level is affected by concept drifts [138]. Furthermore, this method is based on profiles, which are statistics (e.g., mean or standard deviation) that consider the calendar information of different time series values when calculating the statistics. For example, if the hour of the week is the considered calendar information, profiles for Monday at 10 AM are calculated by considering only values from previous Mondays at 10 AM (for a more formal introduction, see Chapter 7). By proposing a profile-based method for handling concept drift, this thesis aims to answer the following research question:
RQ4: Can profiles improve the ability of machine learning models to cope with concept drifts?

3.2.4 Periodicities

The fourth challenge is the periodicities challenge. Thereby, periodicities describe that pattern reoccurs regularly. Examples of such periodic behaviour are traffic time series since the traffic is normally higher during the day than at night. As already mentioned, for pure machine learning methods, periodicities are challenging. However, if correctly modelled, periodicities can support machine learning models [93]. Furthermore, the interaction of multiple periodicities in one time series is difficult to handle [35], [58]. These observations show the relevance of this challenge.

Related Work

As mentioned before, in time series forecasting, there exist time series model-based, machine learning-based forecasts, and hybrid approaches. While time series model-based methods can handle periodicity often if periodic terms are part of the model (e.g., seasonal ARIMA [63]), pure machine learning-based approaches struggle with periodicities. Nevertheless, they are powerful data mining tools. Thus, this related work provides an overview of which approaches exist to improve neural networks' ability to cope with periodicities[4].

As mentioned, neural networks can struggle with periodicities. Nevertheless, most deep learning-based time series forecasting methods do not handle periodicities explicitly, e.g. [3], [77], [94]. However, they often implicitly do it using cyclic calendar information as additional input. E.g., [152] evaluates the impact on how different preprocessings can improve the ability of neural networks to handle seasonalities. Another approach are hybrid models, e.g., [131] proposes to combine SARIMA and neural networks and let the neural networks predict the residuals of the time series forecasted by the SARIMA model. Due to the consideration of classical time series modelling methods, they struggle less with such periodicities. In recent years, hybrid deep learning-based time series forecasting algorithms have been developed. E.g., [106] proposes Neural Basis Expansion Analysis for interpretable Time Series forecasting network (N-BEATS), which learns periodicities using a neural network-based decomposition, where an alternative loss function supports the seasonal component. This work is further extended by [102] proposing Neural Basis Expansion Analysis for interpretable Time Series forecasting with exogenous variables network (N-BEATSx) and [15] proposing Neural Hierarchical Interpolation for Time Series Forecasting network (N-HiTS). Another approach is the ES-RNN proposed by Smyl [124]. The ES-RNN scales the time series locally for handling periodicities. A further hybrid deep learning-based architecture that is supposed to handle periodicities is DeepAR [123]. DeepAR integrates AR terms with RNN to better forecast time series.

The last work regarding the periodicity challenge mentioned in this thesis proposes to use a new sine-based activation function for introducing an inductive bias that is suitable for periodicities [155].

Shortcomings

Concerning handling periodicities, this thesis identifies the following shortcomings. The focus on general time series, as the related work does, limits the application of specialised algorithms that uses prior knowledge of specific time series classes to improve the forecast. In this thesis, the focus is on time series with calendar-driven

[4]Since this thesis focuses on neural network-based forecasters, this related work only focuses on how neural networks handle seasonalities and not on classical methods for seasonalities.

periodicities. Thus, prior information gained by the usage of profiles (Chapter 7) is not used by those methods.

Second, to the best of the author's knowledge, the existing hybrid methods mainly focus on improving the median or mean of a forecast. In contrast, this thesis additionally uses the variance profile besides the average profile to use the information about the probability that is available in statistics. Thus, this thesis aims to answer the following research question:

RQ5: Can the usage of profiles in neural networks improve the deterministic and probabilistic forecasting performance?

Part II

Deep Generative Models for Time Series
with Calendar-driven Periodicities

4 Generation Non-Stationary Energy Time Series

Content of this chapter based on

B. Heidrich *et al.*, "Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks," *Applied Intelligence*, vol. 53, no. 8, pp. 8826–8843, 2023.

As described in Section 3.2.1, the missing scenarios challenge exists. This chapter proposes a solution for this challenge and aims to answer the corresponding research question:

RQ1 Is it possible to use the conditioning mechanism to control the non-stationarity and periodicity in generated time series?

This chapter first formally shows that the conditioning mechanism and appropriate conditioning information enable the control of non-stationarity and periodicity. Furthermore, it presents three exemplary implementations of this solution. Afterwards, these exemplary implementations are evaluated. Finally, the last section of this chapter discusses the performance, the limitations, and potential further developments of the proposed solution.

4.1 Controlling Non-Stationarity and Periodicities in Generated Time Series

This section describes the proposed approach to control non-stationarity and periodicity in time series generation and shows its practical viability by introducing three exemplary implementations that belong to different types of deep generative methods.

4.1.1 Controllable Generation

As mentioned in Section 3.2.1 without any conditioning, the generative functions always generate time series samples with identical statistics. Thus, it is impossible to control the generative function to create time series with specific properties. To control properties, conditioning information is required, which the user can vary in production. I.e., only calendar information is insufficient since the time range for which the functions generate time series determines the calendar information. Consequently, this calendar information is not variable. Thus, this thesis uses statistical information as conditioning information since, during production, this information is variable.

Note, for better readability, this subsection omits the subscript (t_i, h) when writing the samples, i.e. this subsection writes $\mathbf{z}$, $\mathbf{x}^{\text{cal}}$, and $\mathbf{x}^{\text{stats}}$ instead of $\mathbf{z}_{t_i,h}$, ${\mathbf{x}_{t_i,h}}^{\text{cal}}$, and ${\mathbf{x}_{t_i,h}}^{\text{stats}}$.

To use the statistical information as conditioning information, the mapping from the tractable distribution to the intractable distribution incorporates the statistical information $g_\theta(\mathbf{x_{t_i}}; \mathbf{x}^{\text{cal}}, \mathbf{x}^{\text{stats}})$. This function depends not only on the learned parameter θ but also on the calendar information and controllable statistical information[1]. Consequently, it is possible to control the time series generation by varying the statistical information. More formally, as it is possible to interpret the generated time series samples as realisations of a random variable X, LOTUS [122] can be applied to get the expected value of the generated time series samples, i.e.

$$\mathbb{E}[X] = \mathbb{E}[g(Z; \mathbf{x}^{\text{cal}}, \mathbf{x}^{\text{stats}}, \theta)] = \int g(\mathbf{z}; \mathbf{x}^{\text{cal}}, \mathbf{x}^{\text{stats}}, \theta) P_Z \, d\mathbf{z}. \tag{4.1}$$

This application of LOTUS [122] shows that the expected value now depends on the statistical information. Thus, varying the statistical information changes the expected value of the time series and enables a mechanism to control the properties of the synthetic time series. Similarly, the variance and the autocovariance also depend on the calendar and statistical information and can be controlled too.

4.1.2 Implementation of Generative Models to Control the Time-Series Generation

This subsection describes the implementation of the proposed approach. Figure 4.1 provides an overview of the solution. This overview shows that the solution consists

[1]This subsection is assuming that the mapping g_θ has learned to consider the information provided by $\mathbf{x}^{\text{stats}}$ and do not weight this information with zero.

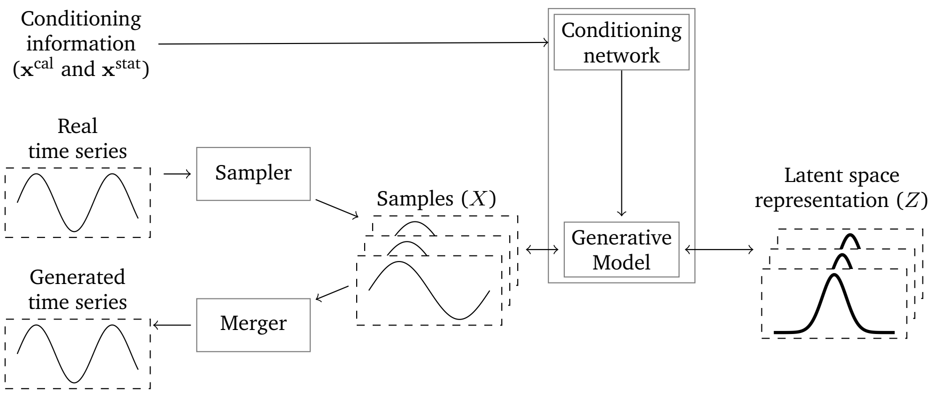

Conditioning information ($\mathbf{x}^{\text{cal}}$ and $\mathbf{x}^{\text{stat}}$)
Conditioning network
Real time series
Sampler
Samples (X)
Generative Model
Latent space representation (Z)
Generated time series
Merger

Figure 4.1.: The proposed method for generating time series with controllable non-stationarity and periodicity consists of four components: the sampler, conditioning network, generative model, and merger. Note the relationship between the latent space and the generative model, as well as the relationship between the samples and the generative model, can be bijective but has not to be. E.g., for the cTimeGAN, there is only a mapping from the latent space to the samples and not in the other direction. Furthermore, note also that conditioning network and the generative model are always jointly trained. Thus, θ comprises the parameters of both. This figure is adapted and based on [52].

of four components: the sampler, conditioning network, generative model, and merger. During training, the statistical and calendar information is extracted from the real time series and forms the conditioning information. Furthermore, during training, the sampler creates time series samples from the real time series. The conditioning information and samples are passed to the conditioning network and the generative models to train both.
During production, the time range for which the time series would be generated determines the calendar information, and the user can specify the statistical information to control the generated time series. The conditioning network encodes this conditioning information. The generative model uses this encoded conditioning information and the noise sampled in the latent space to create synthetic time series samples. The merger merges these samples to form the synthetic time series.
In the following, each of the four components is explained in more detail. Furthermore, when presenting the generative model, the three used ones are explained.

Sampler

The sampler is a preprocessing step required only in the training. The task of the sampler is to transform the time series into samples with equal lengths by applying

Table 4.1.: Implementation details of conditioning networks used by the cINN, cVAE, and cTimeGAN.

Layers	Conditioning network for cINN and cVAE	Conditioning network for cTimeGAN
1	Dense neurons: 8; activation: tanh	Dense neurons: 8; activation: tanh
2	Dense neurons: 24; activation: linear	Dense neurons: 1; activation: linear

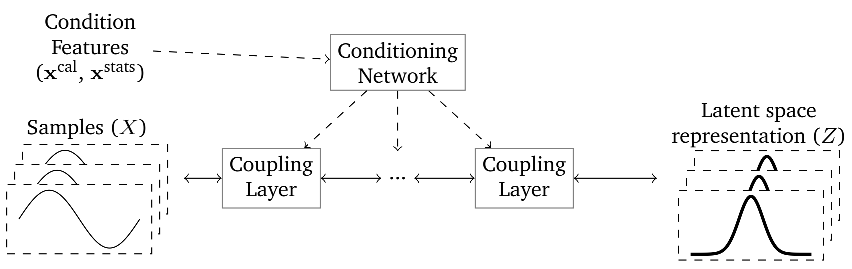
Equation (2.2). Note that this chapter uses $h = 24$ for the sampling. Transforming the time series into samples with equal lengths enables the usage of non-recurrent generative models.

Conditioning Network

The conditioning network is an encoder of the conditioning information. It is used to reduce the dimensionality of the conditioning information. The conditioning network is always trained jointly with the generative model. As input, for each of the time steps that belong to the time series sample that should be generated, it takes the calendar and statistical information. In particular, as calendar information, this thesis uses the trigonometric encoded hour of the day and month of the year, as well as the weekend as a Boolean. As statistical information, this thesis uses the rolling mean of the time series during the training and the desired rolling mean of the generated time series in the generation.
Since recurrent and non-recurrent generative models are used, this thesis uses two different conditioning networks (compare Table 4.1). These conditioning networks are distinguished in their output size since the feed-forward generative models process the conditioning information of all time steps together. In contrast, the conditioning information of the recurrent generative model is processed separately per time step.

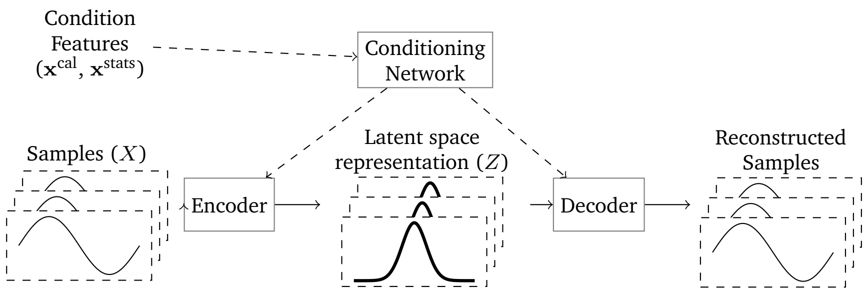
Generative Models

The proposed approach can be applied with different generative models. Thus, this thesis uses a conditional Invertible Neural Network (cINN), a conditional Variational Autoencoder (cVAE), and a conditional TimeGAN (cTimeGAN). Figure 4.2 provides an overview of the three architectures. Note, during preliminary studies, also a conditioned Conditional Optimal Transport GAN (COTGAN) is tested. However, during the evaluation, this model does not converge.



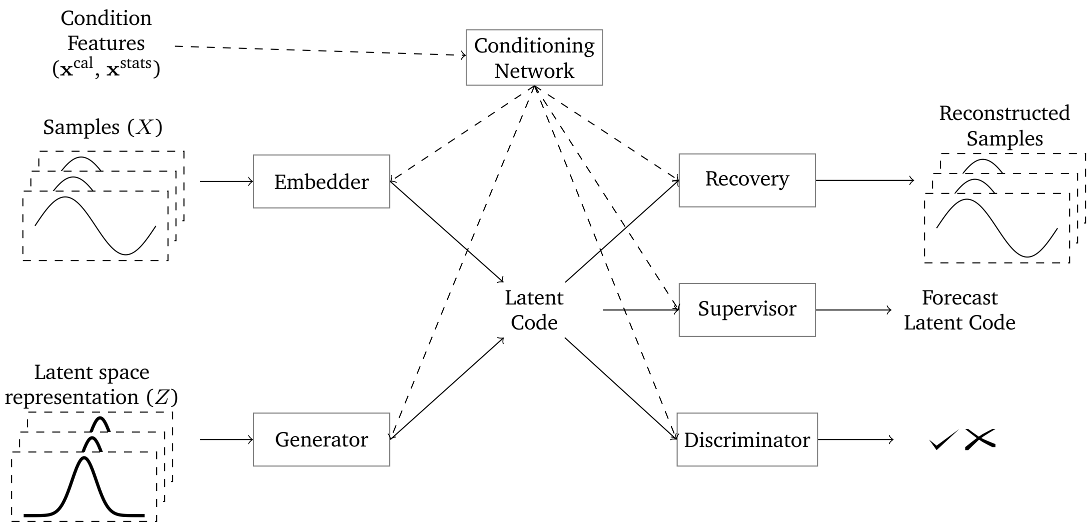
Condition Features ($\mathbf{x}^{\text{cal}}$, $\mathbf{x}^{\text{stats}}$)
Conditioning Network
Samples (X)
Coupling Layer
...
Coupling Layer
Latent space representation (Z)

(a) cINN-based exemplary implementation. This figure is adapted and based on [52].



Condition Features ($\mathbf{x}^{\text{cal}}$, $\mathbf{x}^{\text{stats}}$)
Conditioning Network
Samples (X)
Encoder
Latent space representation (Z)
Decoder
Reconstructed Samples

(b) cVAE-based exemplary implementation.



(c) cTimeGAN-based exemplary implementation.

Figure 4.2.: The three exemplary implementations of the proposed solution to control the time series generation using statistical information. In each subfigure, the dashed lines indicate the data flow of the conditioning information and the solid lines the data flow from the latent space to the real space or vice versa.

Table 4.2.: The parameters of the distributions used to provide the noise to the generative models that they require to generate data. The used parameters are either determined by the design of the algorithm or by preliminary studies.

Model	Distribution	Parameters
cINN	normal distribution	μ: 0, σ: 0.1
cVAE	normal distribution	μ: 0, σ: 0.1
cTimeGAN	uniform distribution	min: 0, max: 1

Table 4.3.: Implementation details of the cINN regarding the used subnetwork.

Layers	Subnetwork
1	Dense neurons: 32; activation: tanh
2	Dense neurons: Segment length; activation: linear

cINN The cINN provides a bijective mapping between the normally distributed latent space and the realisation space that considers calendar and statistical information (Figure 4.2a). To implement the exemplary cINN, this thesis uses FrEIA[2] and PyTorch [109]. The used cINN comprises 15 subsequent invertible coupling layers[3] and one conditioning network. Thereby, the subsequent coupling layers realise the bijective mapping between the latent and realisation space, and the conditioning network extracts features from the conditioning information and reduces their dimensionality. The coupling layers of the used cINN are the conditional affine coupling layer proposed by Ardizzone *et al.* [4], which extends RealNVP by Dinh *et al.* [27]. Each coupling layer contains two subnetworks. Table 4.3 shows their architecture. As inputs, each coupling layer takes the output of the previous coupling layer and the conditioning information $\mathbf{x}^{\text{cal}}$ and $\mathbf{x}^{\text{stats}}$ that the conditioning network (Table 4.1) encodes. To generate data, the cINN takes samples from a normal distribution and maps them to the realisation space. The distribution parameters are determined in preliminary examinations and provided in Table 4.2.

cVAE The cVAE provides a mapping from the realisation space to the normally distributed latent space (encoder) and a reverse mapping (decoder) (Figure 4.2b). Both mappings consider the encoded conditioning information. This cVAE is implemented with PyTorch [109]. Regarding the architecture, the cVAE consists of an encoder, decoder, and conditioning network. For the encoder and decoder, this thesis uses fully connected networks (Table 4.4). As input, the encoder gets the encoded conditioning information and the real-time series samples. The encoder's output is a 16-dimensional latent space vector. This vector forms, together with the encoded conditioning information, the input of the decoder, which aims to reconstruct the time series sample. The conditioning information is encoded using the same conditioning network as the cINN (Table 4.1). To generate data, the cVAE takes samples from a normal distribution. These samples are mapped by the decoder to the realisation space. The distribution parameters are determined in preliminary examinations and provided in Table 4.2.

[2]`https://github.com/VLL-HD/FrEIA`

[3]The number of subsequent coupling layers is determined in preliminary studies.

Table 4.4.: Implementation details of the cVAE regarding the used encoder and decoder.

Layers	Encoder	Decoder
1	Dense neurons: 64; activation: tanh	Dense neurons: 32; activation: tanh
2	Dense neurons: 32; activation: tanh	Dense neurons: 64; activation: tanh
μ	Dense neurons: 16; activation: linear	Dense neurons: Segment length; activation: linear
σ	Dense neurons: 32; activation: linear	

cTimeGAN The third exemplary implementation is TimeGAN [151], a state-of-the-art time series generation model. This thesis uses an own PyTorch [109] based implementation to extend the original TimeGAN. The original TimeGAN consists of five components: the embedder, recovery, generator, discriminator, and supervisor. The embedder's task is to map the samples to a so-called latent code[4]. The recovery, in contrast, reconstructs the latent code into the samples. The generator takes samples from the uniform distributed latent space to generate the latent code. The discriminator distinguishes between the generated and the embedded latent code. Finally, the last supervisor provides an additional loss by performing forecasts on the latent space and thus also forcing the generator to generate latent code with a temporal structure.
This thesis extends the TimeGAN with conditions and a conditioning network to make the time series generation controllable. In particular, the present thesis uses a conditioning network to encode the conditioning information and provides this information as additional input to each of the five TimeGAN components. In contrast to the conditioning network of cINN and cVAE, this conditioning network has only one output neuron since the encoding is performed for each time step separately. Figure 4.2c shows the structure of the proposed cTimeGAN, and Table 4.5 provides the network structure of the different components. To generate data, the cTimeGAN takes samples from a uniform distribution. These samples are mapped by the generator to the realisation space. The distribution parameters are determined by the design of cTimeGAN provided in Table 4.2.

Merger

The generative models generate overlapping time series samples. However, most applications are interested in using a time series instead of overlapping time series samples. Thus, these time series samples must be merged to create a single time series with an arbitrary length. Therefore, the merger applies Equation (2.3) and

[4]Note this is not the same as the latent space with a known and tractable distribution. Nevertheless, this thesis sticks to the notation the authors introduce.

Table 4.5.: Implementation details of the used cTimeGAN regarding the embedder, recovery, generator, discriminator, and supervisor. Note that this is a recurrent architecture using the GRU (gru) and bidirectional Gated Recurrent Unit (biGRU). This also means that the last dense layers of each component are executed for each time step separately.

	Embedder	Recovery	Generator	Discriminator	Supervisor
1	GRU units 16	GRU units 16	GRU units 16	BiGRU units 16	GRU units 16
2	GRU units 16	GRU units 16	GRU units 16	BiGRU units 16	GRU units 16
3	GRU units 16	GRU units 16	GRU units 16	BiGRU units 16	GRU units 16
4	Dense neurons 16; activation sigmoid	Dense neurons 1; activation linear	Dense neurons 1; activation sigmoid	Dense neurons 1; activation sigmoid	Dense neurons 1; activation sigmoid

calculates the median on all values that correspond to the same time index. As a result, a single time series without overlaps is generated.

4.2 Evaluation

This section empirically evaluates the exemplary implementations of the proposed approach for generating time series with controlled non-stationarity and periodicities. The following subsection introduces the experimental setup before the next one shows the results.

4.2.1 Experimental Setup

To evaluate the three exemplary implementations, this thesis applies each implementation three times[5] to each time series and reports the median, minimum, and maximum scores for each quantitative metric. In the following, this subsection presents the considered time series from the electricity and the traffic dataset, as well as the evaluation criteria.

Data Sets

To comprehensively evaluate the proposed approach and the three exemplary implementations, this thesis selects the time series corresponding to the median of the average from the Electricity and Traffic dataset (Section 2.4.2) – namely MT_206 from the Electricity and traj_401507 from the Traffic dataset. The first three years of

[5]This thesis applies it only three times since the training of some of the generative models is very expensive (see Table 4.13)

Table 4.6.: Architecture of the forecasting network for the Train-Synthetic Test-Real (TSTR) evaluation and of the classification network for the discriminative score

(a) Forecasting network for the predictive score.

Layer	Forecasting network
1	Dense neurons 10; activation ReLU
2	Dense neurons 1; activation linear

(b) Classification network for the discriminative score.

Layer	Classification network
1	Dense Neurons 5; activation tanh
2	Dense Neurons 1; activation sigmoid

the electricity dataset and the first year of the traffic dataset are used for training, and the remaining time series are used for evaluation.

Evaluation Criteria

The evaluation of the exemplary implementations assesses the controllability of the time series generation, the quality of the generated time series, and the computational costs.

Controllability of the time series generation To assess the controllability, this thesis generates time series with predetermined statistics that are visually inspect to evaluate if the generated time series follows the predetermined statistic.

Quality of the time series generation To assess the quality of the generated time series, this thesis examines the usefulness, fidelity, and diversity as presented in Section 2.4. For the usefulness, this thesis selects a simple neural network (Table 4.6a) as a forecaster that should predict the next value. The hyperparameters are selected so that the network is still simple but provides reasonable results. To assess the resulting forecasts and obtain the predictive score, the MAE and RMSE are used. For fidelity, this thesis uses a classification network as a discriminator. The hyperparameters of this classification network are provided by Table 4.6b. To get the discriminative score, based on the classifier's prediction, this thesis applies Equation (2.19). Finally, to assess the diversity, this thesis uses the t-SNE with the default hyperparameters of the sklearn [110] implementation.

Computational Cost To evaluate the computational cost, the training time of the evaluated generation methods in minutes are measured three times and the resulting average is reported. For comparable results, this evaluation is performed on the same hardware (setup 1 from Table 2.2).

Benchmarks

As benchmarks, this thesis selects three state-of-the-art generation methods. The first benchmark generation method is COTGAN [147]. As implementation of COTGAN, the publicly available source code[6] is used. Only the data loading functionality is adapted to apply COTGAN to the electricity and traffic datasets.
The second and third benchmark generation models are the RGAN and RCGAN [31]. As implementation, the publicly available source code of RGAN and RCGAN[7] is used. However, the setting file `test.txt` and the `data_utils.py` are adapted to load the electricity and traffic datasets. Since RCGAN can consider conditioning information, this thesis provides as input the same calendar and statistical information as for the exemplary implementations of the proposed solution.

4.2.2 Results

This thesis performs four evaluations to assess the proposed solution for generating missing scenarios. First, the next subsection examines the controllability of the exemplary implementations. Afterwards, two ablation studies to gain insights are performed. The third evaluation compares the exemplary implementations with the benchmarks. Last, the computational effort of the exemplary implementations and the benchmarks is assessed.

Controllability of the Time Series Generation

To demonstrate the controllability of the proposed solution to generate time series, this thesis generates for each exemplary implementation four time series with different predetermined statistics. More specifically, this thesis generates time series for the electricity dataset from January 2011 to December 2014 and use different functions over time as statistical information visible as orange lines in Figure 4.3a. For the traffic dataset, this thesis generates time series for January 2011 to March 2012 and use as statistical information different functions over the time that are visible as orange lines in Figure 4.3b.

Figure 4.3a and Figure 4.3b visualise the time series created with the predetermined statistics. This thesis makes three observations: First, each exemplary implementation generates time series reflecting the predetermined mean. Thereby, the mean of

[6]`https://github.com/tianlinxu312/cot-gan`
[7]`https://github.com/ratschlab/RGAN`

the synthetic time series follows accurately the predetermined mean for the traffic data. Regarding the electricity data, the mean of the synthetic time series deviates for low and high predetermined means, but follows accurately for intermediate predetermined means. Second, for the traffic dataset, the cVAE seems to have extreme values that can not be explained by the specified mean for the periodic time series. Third, in the traffic dataset, the daily minima of the occupancy have to be positive. However, the time series generated by cVAE and cTimeGAN do not always reflect this characteristic, see values below the red line that indicate zero.

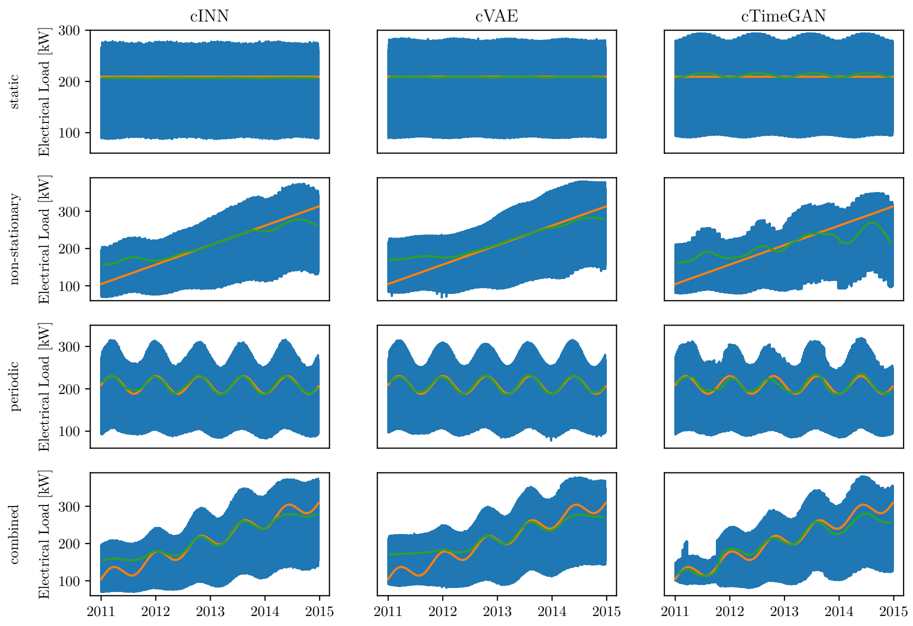
Insights

To gain insights, this thesis performs two ablation studies: The first ablation study examines the influence of the different conditioning information. The second one examines the merger's impact on the result of the exemplary implementation. For each ablation study, this thesis reports the predictive and discriminative scores as well as the diversity.

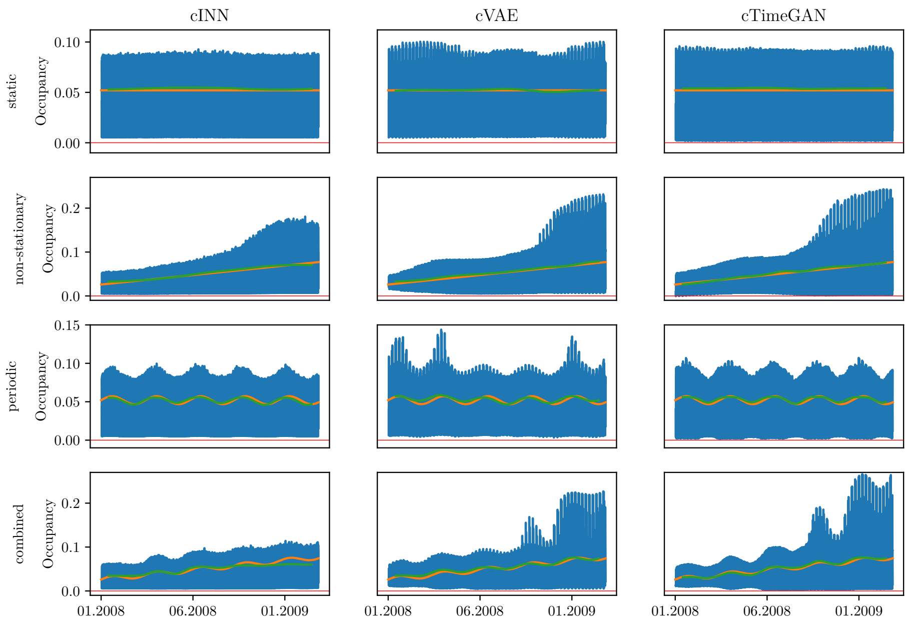
Conditioning Information's Impact In the following, this thesis compares the three exemplary implementations with different combinations of conditioning information and examine the resulting predictive scores, discriminative scores, and diversity. The combinations are calendar and statistical information (both), only calendar information (cal), only statistical information (stats), and no conditioning information (no). Table 4.7 shows the resulting predictive scores for the three exemplary implementations with varying conditioning information for the two data sets. Thereby, this thesis makes three observations: First, the variants with calendar information (cal and both) achieve the best results for all exemplary implementations, datasets, and metrics. Second, the cVAE achieves better results than the cINN and the cTimeGAN. Third, for the cINN, there exist strong fluctuations in the score for the electricity data set resulting in high maximum values in Table 4.7. Additional examinations show that a non-converging forecaster causes these high values during TSTR evaluation.

Table 4.8 shows the discriminative scores of the exemplary implementations with varying conditioning information and the two data sets. These scores show that the variants that use all information (both) achieve the lowest discriminative score, followed by the variants that use only calendar information. Using only statistical information is not beneficial.

To examine the diversity, Figure 4.4 shows the t-SNE plots for the exemplary im-



(a) Electricity



(b) Traffic. Note, the horizontal red line is the zero line.

Figure 4.3.: To demonstrate controlled non-stationarity, a time series is generated using predetermined calendar and statistical information. For both datasets, the generated time series is shown in blue, the predetermined mean is in orange, and the mean of the synthetic time series is green.

Table 4.7.: Ablation study comparing different conditioned generative models with respect to the median, minimum, and maximum predictive score. The lower, the better.

(a) Electricity

		cINN	cVAE	cTimeGAN
MAE	both	17.1 (14.1 - 189.9)	16.0 (14.2 - 19.9)	19.2 (15.5 - 22.5)
	stats	17.4 (14.7 - 21.6)	36.4 (20.7 - 50.8)	49.3 (33.1 - 111.1)
	cal	17.2 (14.2 - 189.9)	15.8 (14.0 - 18.0)	17.6 (15.6 - 21.7)
	no	20.5 (16.0 - 27.0)	73.8 (47.9 - 117.8)	107.7 (21.5 - 130.8)
RMSE	both	22.6 (18.9 - 200.1)	21.0 (19.2 - 25.0)	24.8 (20.2 - 28.8)
	stats	23.0 (20.2 - 27.5)	44.0 (26.9 - 61.3)	59.2 (42.7 - 128.8)
	cal	22.4 (18.8 - 200.1)	20.9 (18.6 - 23.8)	22.8 (20.3 - 27.7)
	no	26.9 (20.9 - 33.0)	86.8 (55.3 - 135.0)	123.3 (29.9 - 150.1)

(b) Traffic

		cINN	cVAE	cTimeGAN
MAE	both	0.013 (0.009 - 0.015)	0.011 (0.009 - 0.015)	0.014 (0.010 - 0.018)
	stats	0.034 (0.023 - 0.056)	0.028 (0.015 - 0.044)	0.050 (0.030 - 0.092)
	cal	0.012 (0.010 - 0.016)	0.017 (0.013 - 0.029)	0.016 (0.012 - 0.020)
	no	0.032 (0.023 - 0.056)	0.032 (0.018 - 0.046)	0.029 (0.019 - 0.049)
RMSE	both	0.018 (0.016 - 0.021)	0.017 (0.015 - 0.020)	0.019 (0.016 - 0.023)
	stats	0.044 (0.031 - 0.068)	0.036 (0.022 - 0.055)	0.060 (0.035 - 0.108)
	cal	0.018 (0.016 - 0.024)	0.024 (0.019 - 0.037)	0.022 (0.018 - 0.027)
	no	0.040 (0.028 - 0.066)	0.041 (0.024 - 0.058)	0.036 (0.025 - 0.057)

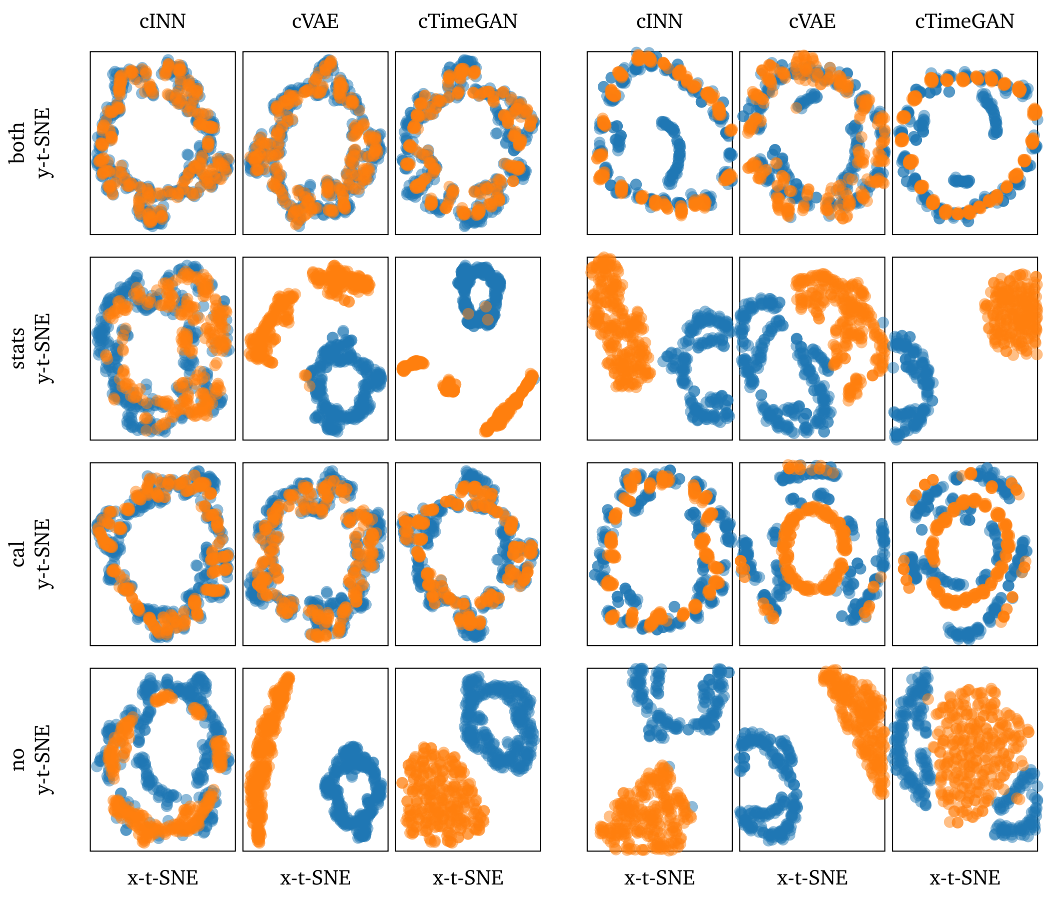
Table 4.8.: Ablation study comparing different conditioned generative models with respect to the median, minimum, and maximum discriminative score. The lower, the better.

(a) Electricity

	cINN	cVAE	cTimeGAN
both	0.18 (0.15 - 0.22)	0.16 (0.11 - 0.2)	0.23 (0.14 - 0.27)
stats	0.46 (0.45 - 0.47)	0.50 (0.50 - 0.50)	0.50 (0.49 - 0.50)
cal	0.25 (0.21 - 0.27)	0.21 (0.16 - 0.24)	0.25 (0.22 - 0.35)
no	0.47 (0.46 - 0.48)	0.50 (0.50 - 0.50)	0.50 (0.50 - 0.50)

(b) Traffic

	cINN	cVAE	cTimeGAN
both	0.32 (0.27 - 0.34)	0.17 (0.10 - 0.28)	0.34 (0.15 - 0.39)
stats	0.50 (0.49 - 0.50)	0.49 (0.48 - 0.50)	0.49 (0.48 - 0.50)
cal	0.35 (0.30 - 0.39)	0.33 (0.29 - 0.35)	0.43 (0.31 - 0.46)
no	0.50 (0.49 - 0.50)	0.50 (0.49 - 0.50)	0.44 (0.35 - 0.48)



(a) Electricity **(b)** Traffic

Figure 4.4.: Study comparing the influence of the conditioning information on the different exemplary implementations regarding the diversity of the generated data on the electricity and the traffic dataset. For both datasets, 300 randomly selected real time series samples (blue) and corresponding synthetic time series (yellow) are mapped on a two-dimensional plane using t-SNE.

plementations with varying conditioning information and the two datasets[8]. Based on these plots, this thesis makes two observations: First, if calendar information is available (cal and both), the generated time series (yellow) have a similar diversity to the real time series (blue). Moreover, the diversity on the electricity dataset seems to be better than on the traffic data. Second, if no calendar information is available, only the cINN can generate time series with a diversity similar to the real data, at least for the electricity dataset.

[8]Note that the latent spaces and the corresponding projections in the two-dimensional plane are individual. Thus, the shown representation of different methods is not comparable. Only the coverage can be compared.

Table 4.9.: Study comparing the influence of the merger on the different exemplary implementations regarding the predictive score on the electricity and the traffic dataset.

(a) Electricity

		cINN	cVAE	cTimeGAN
MAE	without merger	17.1 (14.1 - 189.9)	16.0 (14.2 - 19.9)	19.2 (15.5 - 22.5)
	merger	15.8 (14.0 - 18.7)	15.6 (14.2 - 16.8)	16.4 (14.1 - 189.7)
RMSE	without merger	22.6 (18.9 - 200.1)	21.0 (19.2 - 25.0)	24.8 (20.2 - 28.8)
	merger	21.1 (18.5 - 24.1)	20.4 (18.3 - 22.1)	22.0 (19.4 - 199.9)

(b) Traffic

		cINN	cVAE	cTimeGAN
MAE	without merger	0.013 (0.009 - 0.015)	0.011 (0.009 - 0.015)	0.014 (0.010 - 0.018)
	merger	0.012 (0.010 - 0.020)	0.010 (0.009 - 0.013)	0.013 (0.011 - 0.015)
RMSE	without merger	0.018 (0.016 - 0.021)	0.017 (0.015 - 0.020)	0.019 (0.016 - 0.023)
	merger	0.018 (0.016 - 0.025)	0.016 (0.014 - 0.019)	0.019 (0.017 - 0.022)

Table 4.10.: Study comparing the influence of the merger on the different exemplary implementations regarding the discriminative score on the electricity and the traffic dataset.

(a) Electricity

	cINN	cVAE	cTimeGAN
without merger	0.32 (0.27 - 0.34)	0.17 (0.10 - 0.28)	0.34 (0.15 - 0.39)
merger	0.31 (0.22 - 0.34)	0.15 (0.00 - 0.21)	0.32 (0.28 - 0.38)

(b) Traffic

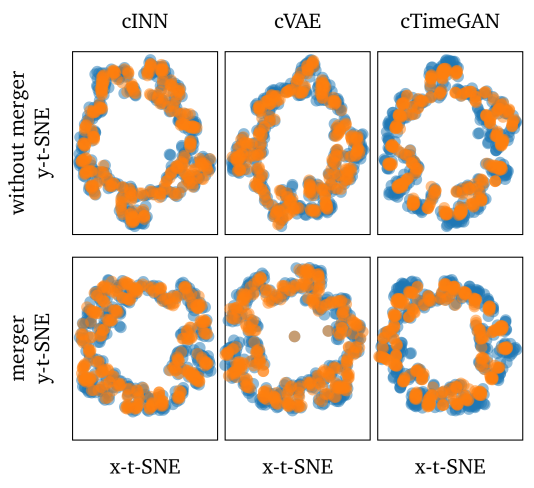
	cINN	cVAE	cTimeGAN
without merger	0.18 (0.15 - 0.22)	0.16 (0.11 - 0.20)	0.23 (0.14 - 0.27)
merger	0.12 (0.08 - 0.14)	0.09 (0.05 - 0.11)	0.20 (0.12 - 0.26)

Merger's Impact Similar to the analysis of the conditioning information's impact, this thesis also examines the predictive score, discriminative score, and diversity of the synthetic data generated by the generative models using all conditioning information (both) with and without a merger.

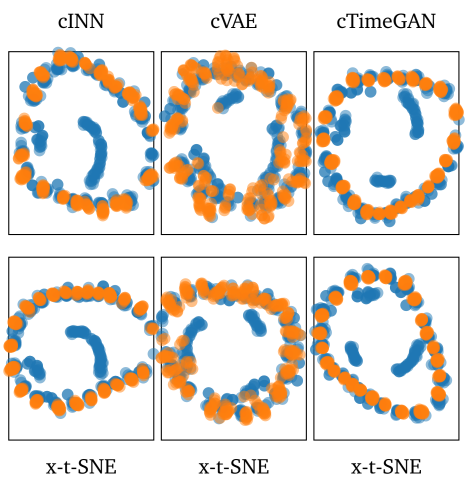
Regarding the predictive score, the merger seems to slightly improve the median. However, the fluctuations are high. Thus, there is probably no significant difference between the results with and without a merger

Regarding the discriminative score, the merger seems to improve the result of the cVAE and the cINN for the traffic dataset. For the cINN and the cVAE on the electricity dataset, as well as the cTimeGAN on both datasets, the results with and without a merger show no clear difference.

Regarding the diversity in Figure 4.5, the usage of the merger seems to have neither a positive nor negative impact.



(a) Electricity



(b) Traffic

Figure 4.5.: Study comparing the influence of the merger on the different exemplary implementations regarding the diversity of the generated data on the electricity and the traffic dataset. For both datasets, 300 randomly selected real time series samples (blue) and corresponding synthetic time series (yellow) are mapped on a two-dimensional plane using t-SNE.

Table 4.11.: The median, minimum, and maximum predictive scores of the conditioned generative models and the three benchmarks on the electricity and traffic data sets. The lower, the better.

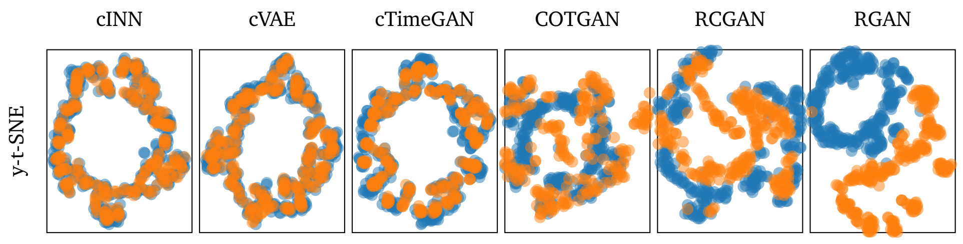
	MAE		RMSE	
	Electricity	Traffic	Electricity	Traffic
cINN	17.1 (14.1 - 189.9)	0.013 (0.009 - 0.015)	22.6 (18.9 - 200.1)	0.018 (0.016 - 0.021)
cVAE	16.0 (14.2 - 19.9)	0.011 (0.009 - 0.015)	21.0 (19.2 - 25.0)	0.017 (0.015 - 0.020)
cTimeGAN	19.2 (15.5 - 22.5)	0.014 (0.010 - 0.018)	24.8 (20.2 - 28.8)	0.019 (0.016 - 0.023)
COTGAN	17.1 (15.0 - 22.8)	0.013 (0.011 - 0.016)	23.0 (20.5 - 29.0)	0.019 (0.017 - 0.023)
RCGAN	18.5 (15.7 - 29.8)	0.017 (0.015 - 0.024)	27.1 (20.9 - 38.6)	0.025 (0.022 - 0.031)
RGAN	31.1 (20.6 - 65.5)	0.034 (0.025 - 0.049)	41.0 (25.9 - 87.3)	0.043 (0.033 - 0.061)

Benchmarking

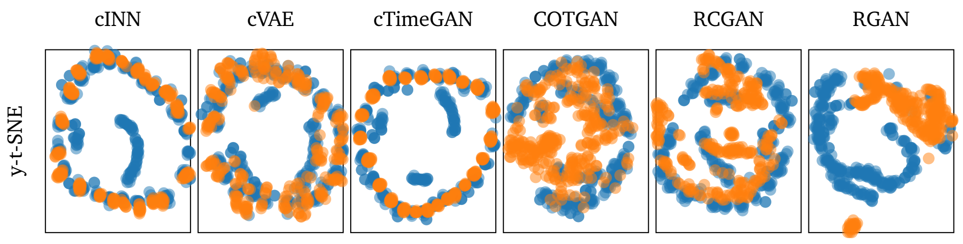
To assess the general quality of the generated time series, this thesis compares the generated time series of the exemplary implementations and benchmarks on the two data sets. More specifically, the predictive and discriminative scores as well as the diversity are compared.
Regarding the predictive score, Table 4.11 reports the results. For both datasets, the best model is the cVAE, followed by the cINN and COTGAN. Note, the RMSE for the cINN seems to be slightly better than of the COTGAN. However, their MAE perform similarly. Regarding the discriminative score, Table 4.12 reports the results. In this table, for the electricity dataset, the best model is the cVAE, followed by the cINN and cTimeGAN. For the traffic dataset, the best model is again the cVAE. However,

Table 4.12.: The median, minimum, and maximum discriminative scores of the conditioned generative models and the three benchmarks on the electricity and traffic data sets. The lower, the better.

	Electricity	Traffic
cINN	0.18 (0.15 - 0.22)	0.32 (0.27 - 0.34)
cVAE	0.16 (0.11 - 0.20)	0.17 (0.10 - 0.28)
cTimeGAN	0.23 (0.14 - 0.27)	0.34 (0.15 - 0.39)
COTGAN	0.36 (0.34 - 0.40)	0.26 (0.22 - 0.32)
RCGAN	0.46 (0.46 - 0.48)	0.41 (0.36 - 0.44)
RGAN	0.49 (0.48 - 0.50)	0.46 (0.46 - 0.47)



(a) Electricity



(b) Traffic

Figure 4.6.: Benchmarking study comparing the exemplary implementations with the benchmarks regarding the diversity of the generated data on the electricity and the traffic dataset. For both datasets, 300 randomly selected real time series samples (blue) and corresponding synthetic time series (yellow) are mapped on a two-dimensional plane using t-SNE.

the second best is the COTGAN.

The last evaluation of the benchmarking compares the diversity of the different generated time series. Figure 4.6 shows the t-SNE plots to compare the diversity. For both datasets, the three exemplary implementations using calendar and statistical conditioning information perform better than the benchmarks.

Computational Effort

Table 4.13 provides the computational cost of the exemplary implementations and the benchmarks in terms of the average training time in minutes. The results show that the cVAE needs the shortest training time, followed by the cINN. The other

Table 4.13.: The average training time in minutes of the three exemplary implementations and the three benchmarks on the electricity and traffic dataset.

	Electricity	Traffic
cINN	18.0	6.2
cVAE	2.2	0.8
cTimeGAN	463.2	157.5
COTGAN	332.5	333.8
RCGAN	205.7	96.9
RGAN	184.3	88.9

methods need more training time. Note that the COTGAN always executes the same number of iterations during training, independent of the training data size. Thus, in contrast to the training times of the other methods, no difference in training time is available for both datasets.

4.3 Discussion

This section discusses the proposed solution to create time series with controlled non-stationarity and periodicity by focusing on the performance, limitations, and potential further development.

4.3.1 Performance

Regarding the performance, this thesis discusses the impact of the different conditioning information and the merger. In addition, it also discusses the performance of the different exemplary implementations and the benchmarking results.

Regarding the conditioning information, using the calendar and statistical information provides four advantages. First, the most important advantage is that using the calendar and statistical information enables the control of the non-stationarity and periodicity of the time series as seen in Section 4.2.2. This enables generative models to generate scenarios and use these scenarios to evaluate the robustness of downstream applications on hardly observed scenarios under the assumption that the generative models generalise well and can generate time series with the new and not observed conditioning information. The experiments show that this assumption holds for the considered variations of the mean. The second advantage focuses on the calendar information. This information enables the usage of the merger, which is useful to create time series of an arbitrary length, even if the original generative model would not support it. In addition, the enabled usage of the merger also improves the generative models' predictive and discriminative scores. This improvement is likely caused by a smoothing effect of the merger that reduces noise in the generated time series. The third advantage of using conditioning information is that it improves the results of the generative models regarding the predictive score, discriminative score, and diversity. This improvement is likely caused by the increased information provided to the generative models. Thus, the generative models can generate better synthetic time series. Last, the results also show that the proposed solution is applicable to different generative model architectures and, thus, is very flexible.

Regarding the comparisons of the different exemplary implementations, the results show that different models are better depending on the evaluation criteria. E.g., the visual inspection of the generated time series with the predetermined mean shows that this mean is reflected the best by the time series generated by the cINN. However, the cVAE performs best regarding the predictive and discriminative scores. Thus, this thesis cannot recommend which model should be used. Instead, the user must carefully evaluate and compare different models for each dataset regarding the most important criteria for the considered use case.
Regarding the diversity, all models perform similarly. However, it seems that the diversity of the electricity data is better covered than of the traffic data, where some of the real data are not covered by the synthetic ones. An explanation might be that these deviating values distinguish to much from the training data. However, this hypothesis needs to be validated in future work. Regarding benchmarking, as formally proofed in the introduction, methods not conditioned on controllable conditions cannot control the generation process. Thus, only the quality of the generated time series can be discussed. During the experiments, this thesis observes that the usage of the conditioning information improves the performance of the generative models and makes their performance comparable to the performance of the benchmarks. In combination with the measured training time, using exogenous information enables the usage of simpler generative methods that require less computational effort while maintaining the generative power of the complex generative methods. Based on the result's discussion and regarding the research question

RQ1 Is it possible to use the conditioning mechanism to control the non-stationarity and periodicity in generated time series?

this thesis concludes that statistical information is useful for implementing time series generation methods that can control the non-stationarity and periodicity of synthetic time series.

4.3.2 Limitations

While the proposed solution works well to control time series generation, this thesis identifies three limitations. The first limitation is that the solution requires a merger since this thesis uses samples to enable non-recurrent methods. To overcome this limitation, future work might operate the recurrent cTimeGAN to generate time series samples with an arbitrary length or enhance COTGAN to consider conditioning information. Second, the proposed solution considers only a limited causality. More specifically, the generative method includes causality only within one sample (in this thesis the sample' length is 24). By conditioning the generative networks, this thesis uses a synthetic causality to merge the time series. However, this synthetic causality is limited since it only relies on the provided exogenous features. Thus, long-term causalities that are not covered by the provided exogenous features would not be part of the generated time series. To overcome this issue, either recurrent methods could be used to generate time series or the sampling strategy in the latent space for creating the input noise of the generative model could be adapted. Finally, the last limitation is the need for better scalability. The proposed solution needs to be trained on each time series separately. Thus, future work must handle this limitation. Possible approaches to handle this limitation could be a global model with additional information on the time series that should be created or a more sophisticated sampling in the latent space and, thus, a better exploitation of information in the latent space.

4.3.3 Further Development

Based on the previously discussed results and limitations, further open research directions exist in controllable time series generation. In the following, this thesis highlights two possible directions.
The first direction aims to integrate more causal structures in the generative process. This would provide two advantages. First, future work could use a simpler approach, i.e., the merger would become unnecessary. Second, the quality of the generated time series might increase since causality can be captured that is not reflected by the used conditioning information. A possible specific research task would be using recurrent conditioned generative models, e.g., the COTGAN. Another specific research could focus on the latent space. More specifically, future work could examine the latent space to find paths or trajectories that induce a reasonable causality in the realisation space.
The second research direction this thesis highlights is scalability. Scalability is crucial

for applying different generative models since their computation time is high, but the available computing resources are often limited. A possible specific research task would be to train a global generative model and exploit the latent space for better time series generation. The first steps towards implementing this research task are examined and discussed in Chapter 6.

5 Mid-term Forecasting Tasks in the BigDEAL Challenge

Content of this chapter based on

B. Heidrich, M. Hertel, O. Neumann, V. Hagenmeyer, and R. Mikut, “Using conditional Invertible Neural Networks to Perform Mid-Term Peak Load Forecasting,” *IET Smart Grid*, 2023.

The BigDEAL challenge is a mid-term electricity peak load forecasting challenge organised by Tao Hong in 2022. In contrast to most forecasting challenges, the organiser asked not to achieve the best overall accuracy. Instead, they organised the challenge into three tasks and asked to predict three peak characteristics corresponding to each task. The characteristics that the BigDEAL challenge focuses on are the daily peaks' shape, magnitude, and timing. For these characteristics, the competitors had to provide point forecasts for six different rounds. At the beginning of each round, the ground truth and the actual weather information of the previous round, as well as the weather forecasts for the new round, are provided (i.e., rolling-based release of the data). For the leaderboard, for each characteristic and round, a ranking – based on characteristic-specific metrics – is calculated. The final leaderboard is created by averaging the five highest-ranked rounds for each team. In each round, the competitors could use historical data, time stamps, and exogenous temperature information from nearby weather stations as input. During the six rounds, the forecast horizon varies between one and three months. Under this challenge setup, the competitors could select arbitrary models without limitations, such as the model's training or inference time.
Thus, for the model selection, this thesis considers the following points.

1. The long forecast horizon of one up to three months makes it difficult to use forecasting methods that rely on lag features.

2. The competitors provided different exogenous variables suitable to form a conditioned distribution of the time series.

3. The design of the challenge, especially the shape task, seems well suited for generative models since they are designed to provide time series that are indistinguishable from real time series.

Based upon these three points, one observes that the task is related to the task of controllable time series generation (Chapter 4). Thus, this thesis uses the conditional generative model to predict the peak characteristics and answers the research question

RQ2 Is it possible to apply generative models conditioned on appropriate statistical and weather information to generate mid and long-term peak energy load forecasts?

In the following, this thesis describes how to apply generative models for time series forecasting. Thereby, the thesis focuses on the contributions made by the author of this thesis. These contributions mainly comprise the generative model-based forecasters, the best-performing members in the ensemble used for the submissions. The other team members of the BigDEAL challenge provided the benchmarks that were also part of the ensemble. Afterwards, this thesis evaluates the generative model-based forecasters. The last section of this chapter discusses the proposed solution and provides an outlook on possible further improvements.

5.1 Time Series Forecasting with Conditional Generative Models

The proposed approach for the BigDEAL challenge is based on generative models. Figure 5.1 provides an overview of the proposed approach, which consists of three steps. These steps are the preprocessing step, the generative model, and the postprocessing step.
During training, the preprocessing extracts the conditioning information from the real time series to form the conditioning information. Furthermore, the preprocessing creates time series samples from the real time series. The conditioning information and samples are passed to the conditioning network and the generative models to train both. Note that postprocessing is not needed for the training.
During production, based on the exogenous weather time series and the time range for which the time series should be forecasted, the preprocessing extracts the conditioning information. The conditioning network encodes this conditioning information, and the generative model uses this encoded conditioning information

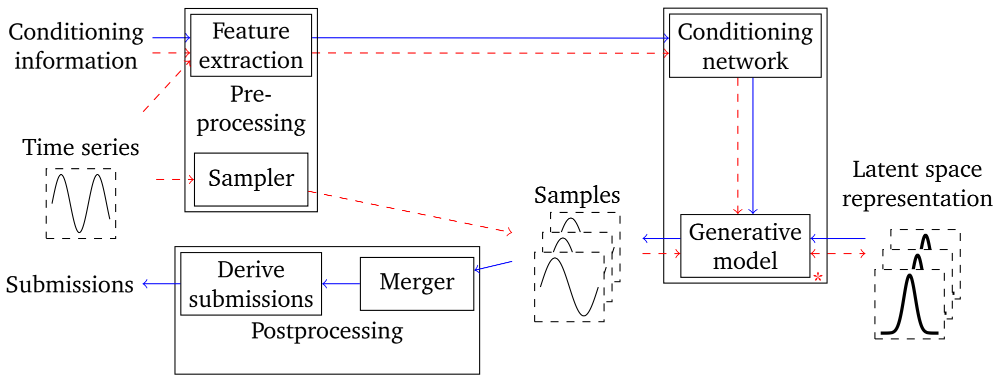


Figure 5.1.: The generative model-based forecaster for the daily peak characteristics. Note the edge annotated with an asterisk; for the cTimeGAN, the data flow is always from the latent space to the model; for the cINN and cVAE, the data flow is from the model to the latent space during training and vice versa during inference. The reason is that GAN-based approaches do not provide a reversible mapping between realisation and latent space. This figure is adapted and based on [52].

and the noise sampled in the latent space to create synthetic time series samples. In the postprocessing step, the samples are merged, and the submissions are derived. In the following, this section explains each of these steps in more detail.

5.1.1 Preprocessing

The preprocessing comprises the feature extraction and the sampler.

Feature Extraction

The extracted features are the conditions for the generative model. The considered extracted features are calendar information, statistical information, and weather-based features.

To extract the calendar information, the feature extraction takes the time index as input. Based on the time index, this thesis extracts the trigonometric encoded month of the year, day of the week, hour of the day, and the three one-hot encoded features, if the timestep corresponds to a workday, weekend, or federal US holiday (resulting in nine features). This calendar information supports the generative model in creating time series with calendar-driven periodicities. In addition, the calendar information also enables the usage of the merger.

The challenge organisers provided temperature information from six weather stations as exogenous weather features. This thesis uses this temperature information to extract three kinds of additional weather features. First, this thesis calculates the average, median, standard deviation, minimum and maximum as extra features based on six temperature values of the weather stations for each hour. Second, the average temperature time series is linearised by squaring the difference of the temperature time series and the saddle point between temperature and the target for each hour. Third, a rolling mean of the average and average linearised temperatures are calculated to capture the buildings' heat inertia that contributes to the time series.

As statistical information, this thesis uses forecasts of the rolling mean of the time series to control mid-term trends and periodicities. To forecast this rolling mean, two different sine-based functions are used. Both functions are first fitted and then forecast the rolling mean. During the training, these functions are fit on the rolling mean using the index as input. Based on the index, these functions forecast the rolling mean during the inference. The first function is a sine function with three parameters

$$s_1 = \sin(\frac{x \cdot 4 \cdot \pi}{365 \cdot 24} + p_1) \cdot a_1 + d_1. \tag{5.1}$$

x is the index input, and p_1, a_1, and d_1 are the trainable parameters of the function. More specifically, p_1 is the phase of the periodicity, a_1 is the amplitude of the rolling mean, and d_1 is the offset. This thesis refers to this function as Sine-1. This function captures the half-yearly periodicity of the time series that is probably caused by electric heating in winter and electric cooling in summer and lower electrical demand in spring and autumn. However, in the time series there exists two additional yearly periodicities with opposite trends of their magnitude. These periodicities superimpose the basic half-yearly periodicity. To capture these additional periodicities, this thesis extends Sine-1 with two additional sine terms, resulting in the Sine-3 function:

$$\begin{aligned} s_2 = s_1 \\ &+ (\sin(\frac{x \cdot \pi \cdot 2}{365 \cdot 24} + 365 \cdot 24 \cdot \pi + p_2) + 1) \cdot a_2 \cdot x \\ &+ (\sin(\frac{x \cdot \pi \cdot 2}{365 \cdot 24} + p_2) - 1) \cdot a_3 \cdot x, \end{aligned} \tag{5.2}$$

where s_1 is described in Equation (5.1), and p_2, a_2, and a_3 are additional parameters. Note both sines are shifted to each other by half a year. To ensure a shift by half a year regardless of the optimal parameters, $p2$ is used in both sines in Equation (5.2).

Table 5.1.: The parameters of the distributions used to provide the noise to the generative models that they require to generate data. The used parameters are either determined by the design of the algorithm or by preliminary studies.

Model	Distribution	Parameters
cINN	normal distribution	μ: 0, σ: 0.6
cVAE	normal distribution	μ: 0, σ: 0.6
cTimeGAN	uniform distribution	min: 0, max: 1

Thus, the final conditioning information consists of the nine calendar features per time step, six provided exogenous weather time series, eight derived weather time series (average, median, minimum, maximum, standard deviation, linearised average time series, rolling mean of the average and the linear averaged time series), and the forecast of the rolling mean. In total, this are 24 features for each time step of the sample (in this thesis, $h = 24$) that should be generated. This results in a conditioning information vector of size $24 \cdot 24 = 576$.

Sampler

In analogy to Chapter 4, this thesis uses a sampler to enable non-recurrent generative models. The sampler transforms the time series into samples with equal lengths by applying Equation (2.2) with $h = 24$.

5.1.2 Conditional generative models

To forecast the peak characteristics, the models with the same architectures introduced in Chapter 4 are used – namely, the cINN, cVAE, and cTimeGAN. This also means that the internal data flows, i.e., how the conditioning information is passed to the generative models, is the same as in Chapter 4. Since we use the same models, the distribution parameters must be determined again to generate noise used by the generative model to create data. For the cTimeGAN, the parameters are determined by the design of cTimeGAN, and for the cINN and cVAE are determined by preliminary evaluations (Table 5.1).

5.1.3 Postprocessing

The generative model provides overlapping time series samples with a fixed length. However, the challenge requires, as a submission, the daily peaks' magnitude, the daily peaks' timing, and the time series for the forecasting horizon. Thus, for providing such a submission, the postprocessing first applies the merger and derives the submissions afterwards.

Merger

The generative model provides time series samples. These time series samples overlap and have a fixed length (in this chapter, the length is 24). However, the challenge requires forecast lengths from one to three months. Thus, in analogy to Chapter 4, the merger combines the time series samples to create a time series with an arbitrary length by applying Equation (2.3).

Derive Submission formats

The output of the merger is the forecasted time series for the forecasting horizon. However, the submission also requires submission of the daily peaks' magnitude and the daily peaks' timing beside the forecasted time series. Thus, this step derives both characteristics. To derive the magnitude, the maximal value of each day is extracted. For the shape, the complete forecast is submitted. For the timing task, the argmax for each day is calculated.

5.2 Evaluation

In the following, this section describes the specific experimental setup to evaluate the generative model-based forecasters and present the corresponding results.

5.2.1 Specific Experimental Setup

The experimental setup for this chapter is mainly built upon the design of the BigDEAL challenge. Thus, this subsection first describes the design of the challenge. Afterwards, it presents the evaluation criteria. Finally, this subsection presents the variants of the generative models.

BigDEAL Challenge Design

The BigDEAL challenge uses the previously introduced BigDEAL dataset (Section 9.2.1) during the qualification and the final. More specifically, the competitors could use the past values of the time series, the past actual weather data, weather forecasts for the forecasting horizon, and information derived from the timestamps to provide the peak forecasts. Since, this chapter is presenting a forecasting method used in the BigDEAL challenge, it uses also the BigDEAL dataset. In the following, the challenge's course is described before the three tasks are presented.

The course of the BigDEAL Challenge The BigDEAL challenge consists of two parts: the qualification and the final.[1] In the qualification, the competitors have to provide one submission. All competitors that beat a benchmark in at least one task are qualified for the final. Of the 121 teams in the qualification, only 14 qualified for the final.

The final of the BigDEAL challenge consists of six submissions. All six submissions together span the period of one year (January - December 2018), whereby a single submission spans one, two, or three months. At the beginning of each round, the ground truth and the actual weather data of the previous round was provided. In addition, the organisers also provided weather forecasts for the forecasting horizon. For each round, the competitors have three or four days to prepare their submissions. E.g. the KIT-IAI team used this time mainly for evaluating the performance of the previous round and to train the models on the training dataset that includes the training data of the previous round.
The final leaderboard consists of the best five ranks for each submission. I.e., the worst result of each team is cancelled. The average ranks for each subtask determine the overall rank. Of the 14 teams that qualified for the final, 13 are eligible for the final leaderboard.

BigDEAL Challenge Tracks The submission of the final comprises three tasks, which correspond to the following three peak characteristics:

1. The daily peaks' magnitude is the maximal load of the considered day.
2. The daily peak's timing is the position of the daily peak in the hours of the day.

[1]Since the tasks differ slightly for the qualification and the final, and no ground truth data for the qualification is provided, this thesis focuses on the final only.

3. The daily peaks' shapes are the five values around the daily peaks divided by the daily peaks' magnitude.

Evaluation Criteria

This thesis uses the same metrics as the BigDEAL challenge to evaluate the forecasted peak's characteristics. Besides that, this thesis also uses two additional metrics to examine the generative model-based forecaster's bias and assess the computational effort.

Challenge Metrics To evaluate the predictions of the daily peaks' magnitude, the organisers of the BigDEAL challenge use the MAPE (Equation (2.12)) and apply it to the predicted and the actual daily peaks' magnitudes. Note that the MAPE of the forecast performances on different time series are comparable since the MAPE is a normalised metric.
To evaluate the forecasts of the daily peak's timing, a wMAE (Equation (2.9)) is used. This wMAE is applied to the daily peaks' actual and predicted timing. To weigh the error (difference in hours between the actual and the predicted timing), the organisers use

$$\mathrm{weighted}(x) = \begin{cases} x & \text{for } x \in \{0,1\} \\ 2x & \text{for } x \in \{2,3,4\} \\ 10 & \text{for } x \geq 5. \end{cases} \tag{5.3}$$

Note that the value range of the daily peaks' timing of different time series with the same resolution is the same (1 to 24 for a time series with an hourly resolution). Thus, the wMAE of the daily peaks' timing of different time series are comparable.
To evaluate the forecast of the daily peaks' shape, the organisers of the challenge use a MAE (Equation (2.8)), which is applied to the actual and the forecasted shape of the daily peak. Note that the shape's definition contains a normalisation step. Thus, the MAE of the shape across different time series are comparable.

Further Metrics Besides the metrics used in the BigDEAL challenge, this thesis also applies the MPE (Equation (2.15)) to assess the bias and measures the training time to assess the computational effort of the generative model-based forecasters.

Considered Conditional Generative Models' Implementations

This thesis uses four different variants of each generative model-based forecaster in the evaluation and refers to them as generative model variants. In particular, this thesis uses each generative model with the Sine-1 and Sine-3 statistics. Furthermore, both variants of each generative model are trained with Transfer Learning (TL) and without Transfer Learning. For Transfer Learning, this thesis pre-trains the generative model variants on all three time series that are provided by the challenge's organiser. Afterwards, the pretrained model is fine-tuned for each time-series separately.

5.2.2 Results

The results consist of four different evaluations to assess the proposed generative model-based forecasters. The first evaluation aims to get insights. Second, the proposed generative model-based forecasters are benchmarked. Third, their computational effort is assessed. Finally, the last evaluation investigates the final leaderboard of the BigDEAL challenge.

Insights

The insights evaluation comprises four analyses to gain insights: a comparison of the different generative model-based forecasters variants, an analysis of their per round performance through the six final rounds, a bias analysis, and a visualisation of the forecasts.

Variants of the Generative Models Table 5.2 provides the performance of the generative model variants on each task and time series. In the following, this thesis describes the observations separately for each task.

Regarding the shape task, this thesis makes three observations: First, the cINN performs better than the cVAE and cTimeGAN. Moreover, even each cINN variant performs better than all variants of the cVAE and cTimeGAN. Second, regarding the impact of the different variants, the impact is always smaller than 10 %. However, transfer learning seems to positively impact the cINN and cVAE. Such an impact is not observable for the cTimeGAN. Third, comparing the performance on the different time series, all models perform best on LDC3 and worst on LDC2.
Regarding the magnitude task, this thesis makes three observations again: First,

Table 5.2.: The average scores of each generative model and variant for the three time series (LDC1, LDC2, and LDC3) and the three tasks of the BigDEAL challenge. The best values are highlighted in bold.

(a) Average scores for the forecasts of the daily peak shape.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
Sine-1	0.062	0.102	0.111	0.088	0.126	**0.126**	0.055	0.094	0.103
Sine-1 TL	**0.061**	0.101	**0.102**	**0.084**	0.124	0.132	**0.051**	**0.091**	0.096
Sine-3	0.062	**0.100**	0.107	0.089	0.125	0.133	0.055	0.095	**0.094**
Sine-3 TL	**0.061**	**0.100**	0.107	**0.084**	**0.122**	0.137	**0.051**	0.092	0.102

(b) Average scores for the forecasts of the daily peak magnitude.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
Sine-1	**3.5**	4.2	5.0	4.3	5.4	5.7	**3.9**	5.8	6.3
Sine-1 TL	**3.5**	**3.8**	**4.7**	3.9	5.1	5.2	4.1	**5.3**	**5.9**
Sine-3	**3.5**	4.2	4.9	3.7	**4.8**	5.9	4.1	**5.3**	6.1
Sine-3 TL	3.7	4.5	5.2	**3.5**	**4.8**	**4.9**	**3.9**	**5.3**	6.1

(c) Average scores for the forecasts of the daily peak timing.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
Sine-1	**1.099**	**1.446**	1.955	0.943	1.354	1.659	0.879	**1.202**	**1.213**
Sine-1 TL	1.180	1.486	1.553	0.934	**1.231**	**1.460**	0.790	1.238	1.338
Sine-3	1.159	1.461	1.900	0.899	1.358	1.556	**0.762**	1.245	1.296
Sine-3 TL	1.170	1.461	**1.531**	**0.866**	1.393	1.474	0.871	1.230	1.413

similar to the shape task, all variants of the cINN are better than each variant of the cVAE and cTimeGAN. Second, for the cINN and cVAE, the best variant seems to be the Sine-3 variant with transfer learning. The best variant for cTimeGAN is Sine-1 with transfer learning. Regarding the differences between the different variants of each model, they are clearer than for the shape. Third, comparing the performance on the different time series, each model achieves the best result on LDC1.

Regarding the timing task, this thesis also makes three observations: First, similar to both previous tasks, each variant of the cINN is better than each variant of the other generative models. Second, the performance of the generative model fluctuates strongly for each generative model. However, there is no clear trend indicating that one variant is superior. Third, comparing the performance on the different time series, the best timing forecasts are achieved for LDC3.

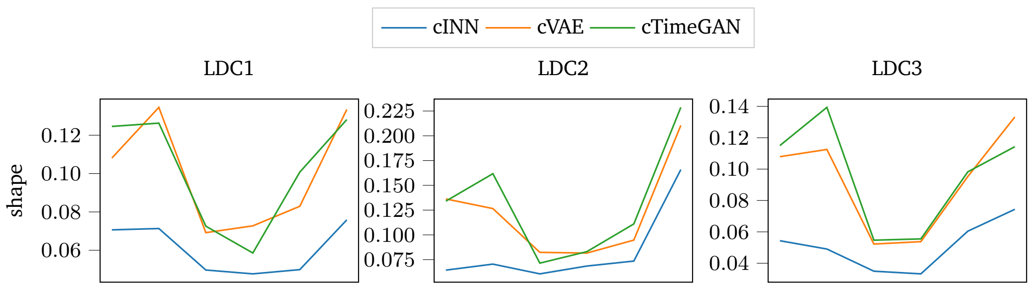
Wrapping up, the different variants have no big impact on the forecast performance. However, for the cINN and cVAE, there are some indications that the variant with the Sine-3 and Transfer Learning performs best and for the cTimeGAN that the variant with the Sine-1 and Transfer Learning performs best. Thus, this thesis selects these variants for further evaluation.

Performance through the rounds To gain insights into the seasonal behaviour of the generative model-based forecasters, this thesis compares their performance on the six rounds of the BigDEAL final in Figure 5.2.

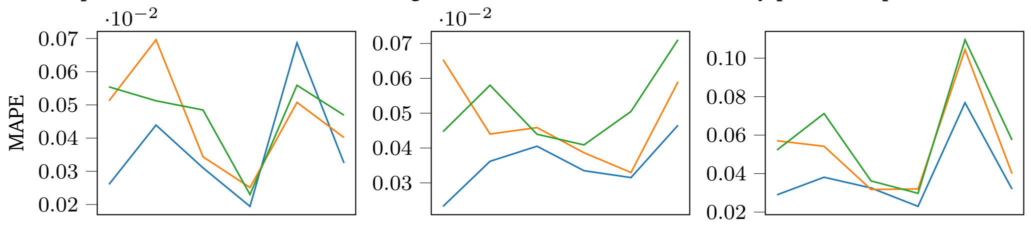
Regarding the shape, this thesis makes three observations: First, the quality of the prediction differs during the six rounds. For each substation and model, the best round achieves a score of almost half of the score of the worst round. Second, there are seasonal fluctuations, with lower scores in summer (rounds 3 and 4) and higher in winter (rounds 1 and 2). Third, the cINN performs better than the cVAE and cTimeGAN in each round.
Regarding the magnitude, this thesis makes two observations: First, similar to the shape task, forecast performance fluctuates strongly across the six rounds. These fluctuations seem to be seasonal for the cVAE and cTimeGAN. However, for the cINN, no seasonal behaviour is observable. Second, the performance on the time series LDC1 and LDC3 in the fifth round is worse than in the other rounds. For LDC2, such a behaviour is not observable.
Regarding the timing task, there are seasonal fluctuations again in all six rounds – with lower scores in summer and higher in winter. Furthermore, the scores in round 6 are worse than in the other rounds.

Examining the Bias To examine the bias, this thesis analyses the MPE for each generative model-based forecaster, each time series and task. Table 5.3 provides the MPEs. The remainder of this paragraph presents the observations for each task separately.

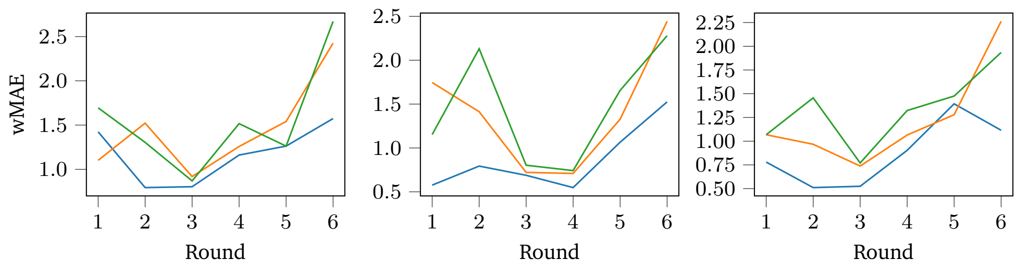
Regarding the shape task, the overall biases for each model and time series are similar and small. However, examining the different rounds in more detail, there are variations between the different rounds. For LDC1 and LDC3, the bias in round 5 is high, and for LDC2 in round 6 compared to the other rounds.
Regarding the magnitude task, this thesis makes two observations: First, the overall biases are small. Examining the different rounds, the fluctuations do not follow a clear pattern. Second, for LDC2, each model always underestimates the magnitude except for cTimeGAN in round 5 (MPE is greater than 0).
Regarding the timing task, this thesis makes three observations: First, compared to the other tasks, the overall bias is higher. Furthermore, in total, each model overestimates each time series. Second, examining the different rounds, seasonal fluctuations are observable, with an underestimation more likely in summer and an



(a) The performance of the conditional generative models on task 1 (daily peaks' shape).



(b) The performance of the conditional generative models on task 2 (daily peaks' magnitude).



(c) The performance of the conditional generative models on task 3 (daily peaks' timing).

Figure 5.2.: For each task and substation, lineplots over the six different rounds shows the performance of the conditional generative models.

Table 5.3.: The MPE of each conditional generative model and variant for the three time series and the three tasks of the BigDEAL challenge.

(a) MPE for the forecasts of the daily peak shape.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
1	-0.35	-0.22	-2.90	-0.46	2.83	0.19	1.23	1.44	1.63
2	0.97	1.49	-0.71	1.64	0.62	1.25	3.30	2.97	3.03
3	-1.26	-2.87	-1.75	2.84	0.02	1.72	2.36	1.11	2.24
4	-0.08	-0.44	0.74	2.04	-0.12	2.55	1.91	1.78	3.60
5	-7.13	-5.18	-1.74	1.54	-0.51	1.19	-7.65	-5.17	-3.12
6	0.61	0.87	1.47	3.95	3.41	5.90	2.20	0.91	3.92
Total	-1.21	-1.06	-0.82	1.92	1.04	2.14	0.56	0.51	1.88

(b) MPE for the forecasts of the daily peak magnitude.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
1	-0.61	2.41	-0.68	0.09	5.68	2.48	0.58	4.22	3.95
2	1.42	4.17	1.92	2.06	2.90	3.74	3.24	4.96	5.12
3	-0.68	-0.93	-0.16	3.09	2.38	1.38	2.79	2.53	1.51
4	0.68	1.50	1.48	3.00	3.58	3.18	1.82	2.96	1.69
5	-6.12	-3.00	-1.36	1.85	0.42	-0.31	-4.10	-1.10	-1.60
6	0.51	1.27	1.51	3.02	4.28	6.31	1.76	1.46	4.58
Total	-0.80	0.90	0.45	2.19	3.21	2.80	1.02	2.51	2.54

(c) MPE for the forecasts of the daily peak timing.

	LDC1			LDC2			LDC3		
	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN	cINN	cVAE	cTimeGAN
1	-0.99	-5.91	-1.93	2.12	3.01	2.59	-9.91	-10.18	3.16
2	-3.13	-4.74	-2.14	-6.68	-9.04	-15.51	-3.14	0.29	-5.63
3	-0.30	0.63	0.40	-0.42	-0.24	-0.44	-1.09	0.04	0.31
4	-2.66	-1.82	-2.96	0.02	-0.54	-1.03	-2.34	-0.23	-3.85
5	-4.65	-4.96	-2.35	-5.43	-5.31	-5.07	-32.27	-31.11	-31.56
6	-4.94	-35.40	-12.94	-7.35	-10.53	-3.05	-21.01	-31.69	-9.97
Total	-2.78	-8.70	-3.65	-2.96	-3.78	-3.75	-11.63	-12.14	-7.92

overestimation more likely in winter. Third, in rounds 5 and 6, the bias is comparable strong.

Visualisation of the predicted properties Figure 5.3 provides a visualisation of the forecasts of the whole time series[2], the magnitude, and the timing.

In this visualisation, two observations are made. First, the predictions are almost narrow to the ground truth (black line) regardless of the task, except for a dip in September for LDC3.
Second, for LDC2 and LDC3, the overall prediction and the magnitude seem almost

[2]Note, a visualisation of the whole time series is used instead of the shape visualisation since the shape is scaled for each day.

always underestimated. However, such an underestimation is not observable for LDC1.

Benchmarking

To benchmark the generative model-based forecasters, this thesis compares them to Support Vector Regression (SVR) and a fully connected NN[3].

Table 5.4 shows the results of the benchmarking. This thesis presents three observations: First, the cINN-based forecaster is the best model for each task and time series. For the shape task, it is even the best model for each of the six rounds. Second, except for the cINN-based forecaster, there are no clear differences in the performance of the other models. Third, examining the distance between the cINN and the other models, the distances are smaller in summer than in winter.

Computational Effort

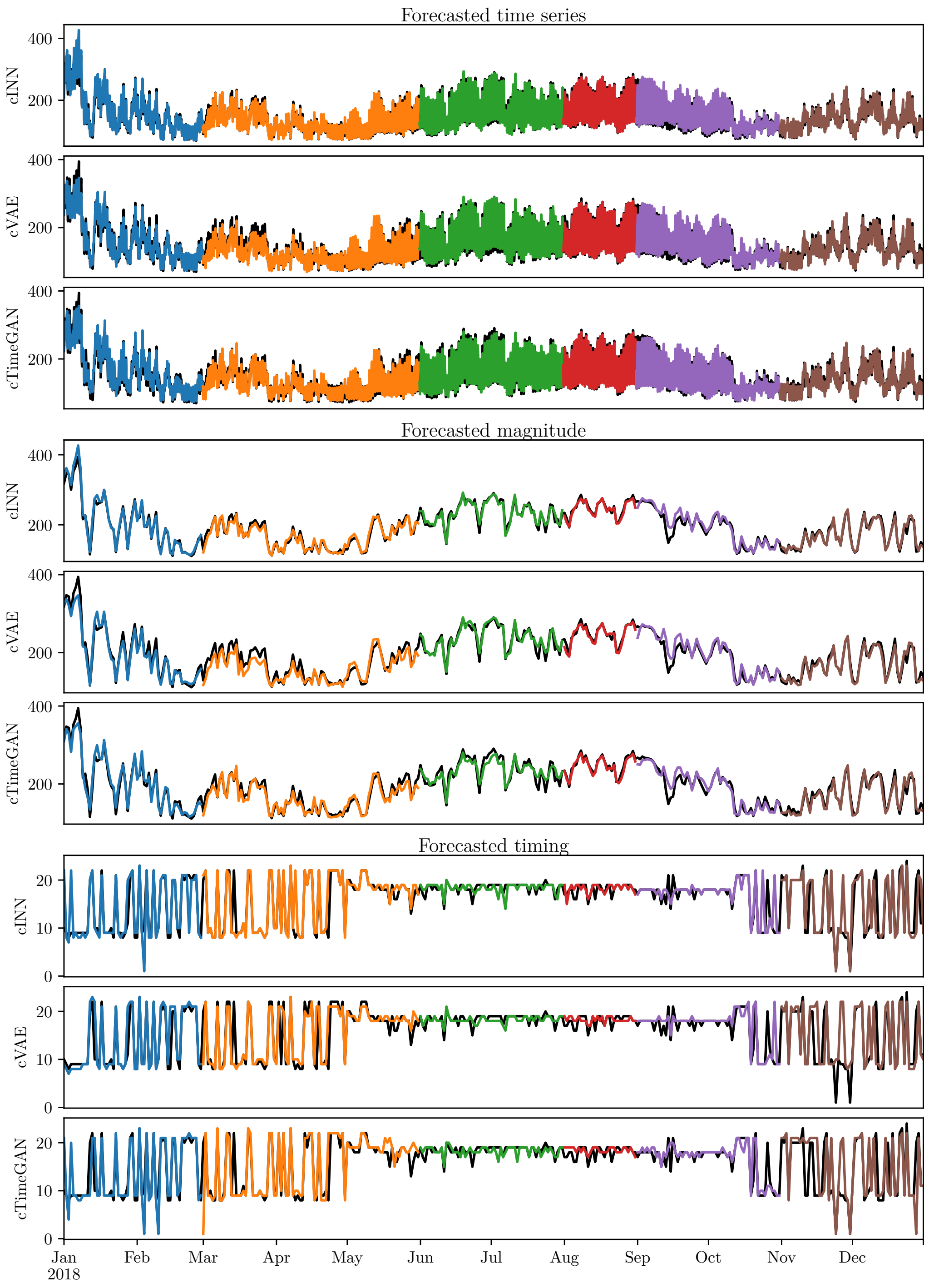
Table 5.5 contains the training time in seconds of each model and all three substations. In this table, this thesis makes three observations. First, the proposed generative models-based forecaster requires more training time than both benchmarks except for the cVAE, which requires a similar amount as the neural network. In contrast, the cTimeGAN requires the most training time and clearly more than the benchmarks. Second, analysing the training times of the different generative model variants, the transfer learning lead to an increased training time since they use a bigger dataset and the training time includes the sum of training on all data as well as the fine-tuning. Concerning the Sine-1 and Sine-3 statistics, there are no differences. Finally, the training time increases with each round since more training data is available.

Results of the Challenge

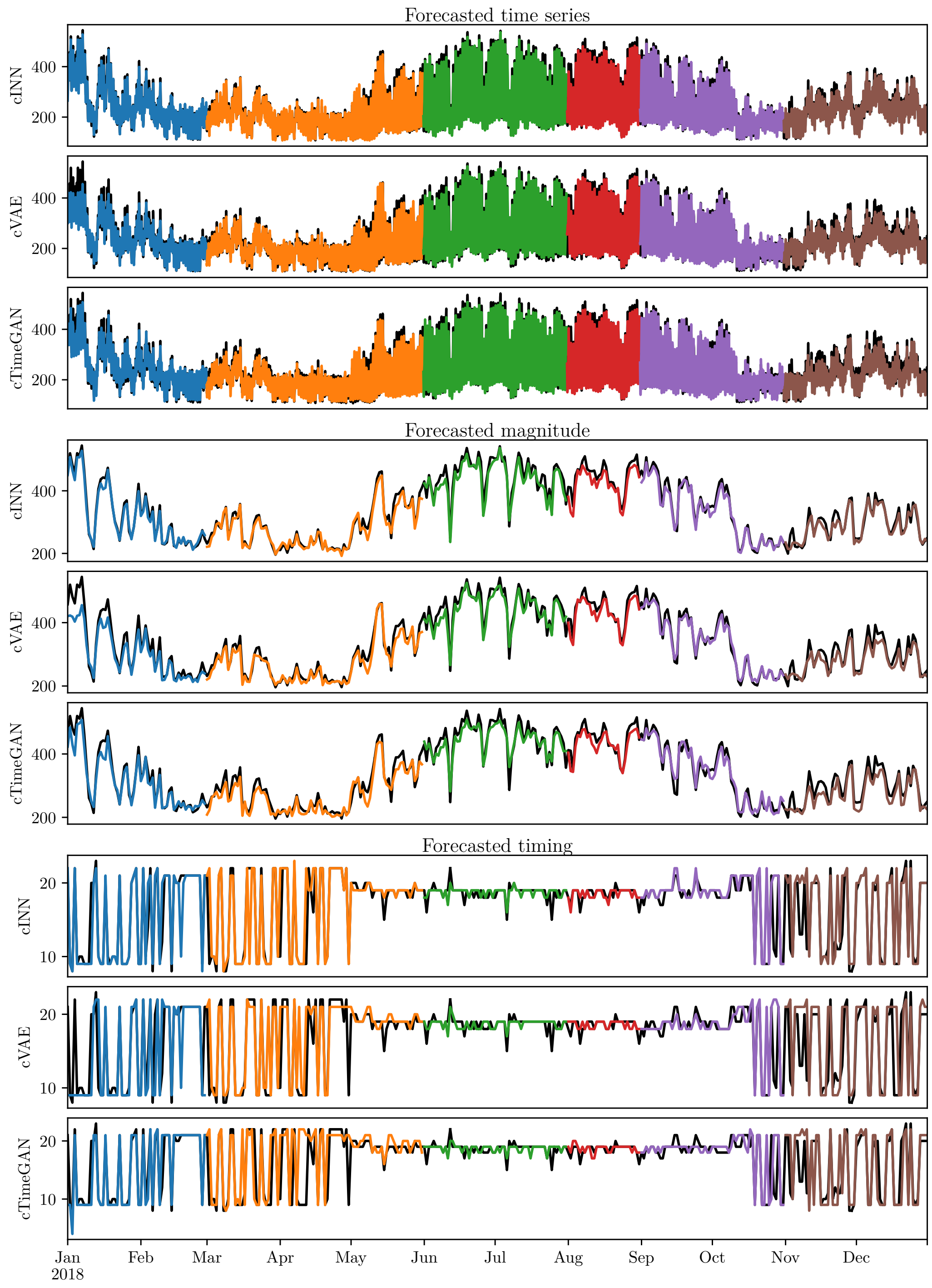
The last part of the evaluation briefly summarises the BigDEAL challenge's results. Note that during the challenge, an ensemble, with a NN, SVR, and cINN as ensemble members was used[4]. However, the cINN-based forecaster was always the most important member of this ensemble and performed even better than the ensemble

[3]In the benchmarking, simple benchmarks are used since the comparison with more sophisticated methods is performed during the BigDEAL challenge and presented in Section 5.2.2.

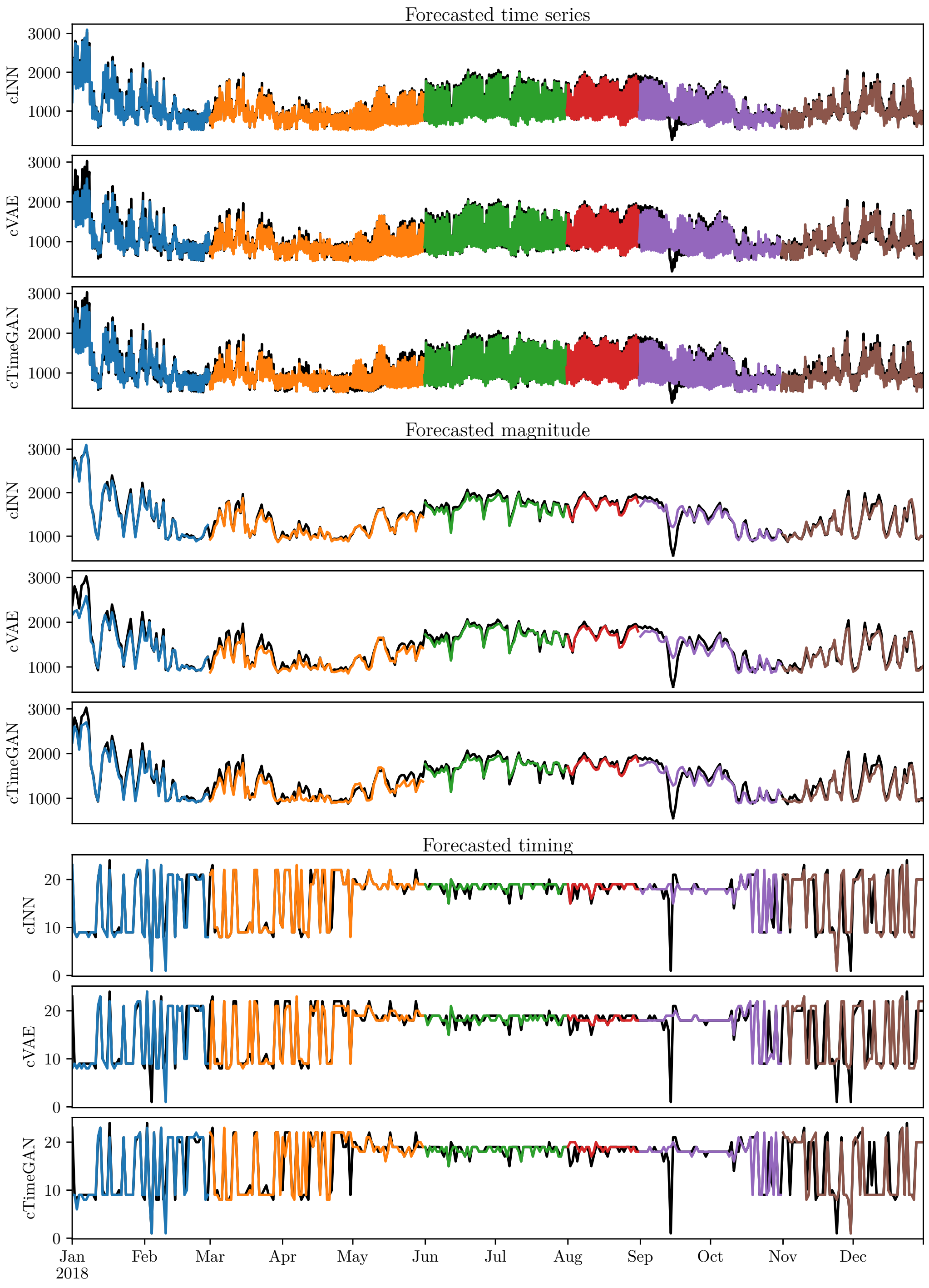
[4]The other generative models are added for the completeness of this thesis.


Forecasted time series
cINN
cVAE
cTimeGAN
Forecasted magnitude
cINN
cVAE
cTimeGAN
Forecasted timing
cINN
cVAE
cTimeGAN
400
200
20
10
0
Jan 2018
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

(a) LDC1


Forecasted time series
cINN
cVAE
cTimeGAN
400
200
Forecasted magnitude
Forecasted timing
20
10
Jan 2018
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

(b) LDC2



(c) LDC3

Figure 5.3.: The predicted values of the conditional generative models for each task and time series for the year 2018. To visualise each round separately, they are coloured differently. The ground truth is in black.

Table 5.4.: The average score and rank of the best variants of the conditional generative models and the benchmarks for the three time series and the three tasks of the BigDEAL challenge.

(a) The average scores for the forecasts of the daily peak shape.

	LDC1					LDC2					LDC3				
	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN
1	0.097	0.131	**0.073**	0.108	0.125	0.115	0.108	**0.065**	0.136	0.134	0.088	0.106	**0.049**	0.108	0.115
2	0.116	0.108	**0.070**	0.135	0.126	0.118	0.099	**0.074**	0.127	0.162	0.098	0.088	**0.052**	0.113	0.139
3	0.078	0.092	**0.051**	0.069	0.073	0.090	0.077	**0.058**	0.082	0.071	0.068	0.066	**0.038**	0.052	0.055
4	0.089	0.077	**0.043**	0.073	0.059	0.101	0.076	**0.069**	0.082	0.083	0.076	0.069	**0.033**	0.054	0.056
5	0.096	0.102	**0.053**	0.083	0.101	0.122	0.107	**0.074**	0.095	0.111	0.123	0.110	**0.061**	0.095	0.098
6	0.117	0.119	**0.074**	0.133	0.128	0.190	0.201	**0.167**	0.211	0.229	0.106	0.110	**0.072**	0.133	0.114
Total	0.099	0.105	**0.061**	0.100	0.102	0.123	0.111	**0.084**	0.122	0.132	0.093	0.091	**0.051**	0.092	0.096

(b) The average scores for the forecasts of the daily peak magnitude.

	LDC1					LDC2					LDC3				
	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN
1	4.789	3.538	**3.198**	5.124	5.543	4.322	3.524	**2.630**	6.537	4.469	6.189	4.403	**2.794**	5.707	5.229
2	6.212	4.193	**3.717**	6.962	5.123	6.273	4.329	**3.498**	4.402	5.803	6.346	4.461	**3.453**	5.420	7.121
3	5.074	4.023	**3.260**	3.436	4.847	8.171	4.456	4.491	4.586	**4.395**	**2.702**	3.722	3.873	3.176	3.620
4	2.407	**1.770**	2.152	2.513	2.302	5.604	4.303	4.977	**3.864**	4.089	2.113	**2.010**	3.124	3.204	2.976
5	6.770	5.564	5.101	**5.076**	5.592	5.602	4.744	3.339	**3.298**	5.050	8.819	9.133	**8.375**	10.441	10.956
6	5.444	5.157	**3.346**	4.017	4.691	6.102	4.903	**4.192**	5.897	7.107	4.940	3.601	**3.240**	4.000	5.746
Total	5.116	4.041	**3.462**	4.521	4.683	6.013	4.376	**3.855**	4.764	5.152	5.185	4.555	**4.143**	5.325	5.941

(c) The average scores for the forecasts of the daily peak timing.

	LDC1					LDC2					LDC3				
	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN	SVR	NN	cINN	cVAE	cTimeGAN
1	1.797	1.678	1.271	**1.102**	1.695	1.475	1.169	**0.898**	1.746	1.153	0.847	1.119	**0.559**	1.068	1.068
2	1.326	1.130	**0.696**	1.522	1.304	1.543	**0.815**	0.848	1.413	2.130	0.946	0.804	**0.533**	0.967	1.457
3	0.967	1.098	**0.836**	0.918	0.869	0.852	0.721	**0.656**	0.721	0.803	0.623	0.852	**0.607**	0.738	0.770
4	1.548	1.677	**1.226**	1.258	1.516	1.000	0.613	**0.581**	0.710	0.742	1.258	1.290	**0.548**	1.065	1.323
5	1.541	1.787	1.279	1.541	**1.262**	1.426	1.164	**1.033**	1.328	1.656	1.820	1.475	1.311	**1.279**	1.475
6	2.098	2.459	**1.770**	2.426	2.672	2.393	2.197	**1.590**	2.443	2.279	1.738	1.590	**1.180**	2.262	1.934
Total	1.546	1.638	**1.180**	1.461	1.553	1.448	1.113	**0.934**	1.393	1.460	1.205	1.189	**0.790**	1.230	1.338

Table 5.5.: The training time in seconds of the best conditional generative model variant and the benchmarks for the three time series and the three tasks of the BigDEAL challenge. For comparing the training times, all models are executed on the hardware setup 1 Table 2.2.

(a) LDC1

	SVR	NN	cINN				cVAE				cTimeGAN			
			Sine-1		Sine-3		Sine-1		Sine-3		Sine-1		Sine-3	
				TL		TL		TL		TL		TL		TL
1	19	194	953	3616	891	3506	64	190	62	188	12548	35838	13031	37897
2	19	78	963	3487	937	3714	66	195	67	197	13203	37027	13044	38794
3	19	237	1052	3814	1066	3916	75	228	81	224	15183	69657	15688	42371
4	25	99	1369	4134	1305	4078	86	235	88	230	16323	43459	16658	41762
5	25	63	1376	4178	1422	4055	81	212	80	211	15910	39441	15141	40715
6	25	68	1549	3970	1298	3618	84	225	81	212	16641	41823	16769	42634

(b) LDC2

	SVR	NN	cINN				cVAE				cTimeGAN			
			Sine-1		Sine-3		Sine-1		Sine-3		Sine-1		Sine-3	
				TL		TL		TL		TL		TL		TL
1	18	280	989	3626	955	3685	64	196	64	192	12401	36510	12955	37589
2	22	127	1069	3793	1027	3629	67	195	65	194	13227	36751	13255	39905
3	21	77	1039	3356	1090	4016	80	223	82	230	15729	70343	15732	42479
4	23	55	1322	4171	1398	4070	86	232	87	229	16589	44117	16080	42206
5	23	99	1409	4135	1423	3803	82	224	84	221	16148	39382	14918	41550
6	24	74	1290	4093	1504	3974	97	217	85	213	16701	42629	16351	41980

(c) LDC3

	SVR	NN	cINN				cVAE				cTimeGAN			
			Sine-1		Sine-3		Sine-1		Sine-3		Sine-1		Sine-3	
				TL		TL		TL		TL		TL		TL
1	15	161	746	2685	752	2580	10	27	9	29	12668	36214	12723	37238
2	16	93	744	2715	774	2513	66	193	66	195	12942	36378	13130	39228
3	18	75	853	2789	834	2850	74	220	81	220	15159	70129	15581	42042
4	20	54	995	2943	936	2840	86	221	81	230	16318	43634	16491	41332
5	19	46	1006	2918	983	2976	81	219	84	212	16429	39069	15107	40482
6	21	145	1061	2936	1067	2794	92	234	91	207	16639	42191	16714	42205

Table 5.6.: First five teams on each task of the BigDEAL challenge.

Overall	Daily peaks' magnitude task	Daily peaks' timing task	Daily peaks' shape task
Amperon	Amperon	**KIT-IAI**	**KIT-IAI**
KIT-IAI	Overfitter	Amperon	Amperon
Overfitters	peaky-finders	BelindaTrotta	Overfitters
peaky-fitters	Team SGEM-KIT	Overfitters	X-Mines
X-Mines	**KIT-IAI**	X-Mines	SheenJavan

in some rounds [46].
The first five places of the final leaderboard are shown in Table 5.6[5]. In this table, this thesis makes three observations. First, the proposed solution achieves the overall second place in the challenge. Second, it won two out of three tasks –namely, the daily peaks' shape and positioning tasks. Third, in the daily peaks' magnitude task, the proposed solution achieved fifth place out of thirteen competitors.

5.3 Discussion

This section discusses the performance, limitations, and the further potential of the generative model for mid-term peak load forecasting.

5.3.1 Performance

Regarding the performance, four aspects are discussed: the insight results, the computational costs, the benchmarks, and the results of the challenge.
Regarding the insights, this thesis highlights four observations. First, the scores of the generative model-based forecasters fluctuates through the different seasons with lower scores in summer than in winter. This indicates that the summer months are easier to predict. The observation that the time series consumption pattern is more stable in summer than in winter confirms this conclusion. Second, the difficulty in forecasting the three time series differs. Potential reasons might be varying numbers of residential, commercial, or industrial buildings connected to the time series. Unfortunately, no corresponding meta-information is available to prove this hypothesis. Last, the dip in the forecast for LDC3 is probably caused by Hurricane Florence[6], which hit the east coast of the United States. Fourth, the bias is high, especially for the timing task, and the forecast accuracy is reduced for round

[5] `http://blog.drhongtao.com/2022/12/bigdeal-challenge-2022-final-leaderboard.html`
[6] `https://www.washingtonpost.com/weather/2018/09/20/land-transformed-by-water-north-carolina-before-after-hurricane-florence/` (last accessed 07.06.2023)

6. These observations might be due to the interplay of harder predictable winter months and not well-learned holiday information at Christmas time.
Regarding the computational costs, the cINN and cTimeGAN are expensive to train. However, these costs might be reduced by using GPUs for training instead of CPUs. Furthermore, improved pretraining strategies might reduce the required training time too.
Regarding the benchmarking and the results of the BigDEAL challenge, the cINNs seems to be good at forecasting the timing and shape of the daily peaks. However, this model struggles with the magnitude.
To wrap up the result-related discussion and to answer

RQ2 Is it possible to apply generative models conditioned on appropriate statistical and weather information to generate mid and long-term peak energy load forecasts?

This thesis concludes that generative models –in particular the cINN– if conditioned on appropriate statistics, calendar, and weather information, can generate accurate mid-term forecasts regarding the daily peak's shape and timing. However, the challenge's results also indicate that other methods might be more suitable for the daily peak magnitude forecasts.

5.3.2 Limitation

Despite the good performance on the timing and the shape task, this thesis identifies two limitations of the proposed generative model-based forecaster. First, to perform the forecast, the user has to determine a sampling parameter to sample noise from the latent space. This parameter needs to be carefully selected. For example, if the latent space is normally distributed, a too-high variance parameter would induce too much noise into the forecast. However, a too-small variance parameter reduces the variability of the samples. It thus increases the risk of forecasting an extreme value that is not fixed by the smoothing effect of the merger. Thus, future work might also examine the robustness of the generative model-based forecasters regarding the selection of the sampling parameters. Second, the presented generative-model based forecaster is a local forecaster with limited scalability. Thus, future work should overcome this issue by extending this approach to global model forecasting. A possible extension would be exploiting latent space information by introducing a smarter sampling strategy. Towards such an implementation, this thesis performs the first steps in Chapter 6.

5.3.3 Further Potential of Generative Models for Forecasting

This thesis highlights three aspects regarding the further potential of generative models for forecasting. First, future work should identify why the cINN has problems forecasting the magnitude correctly. Second, future work can also revise the merger, for example, by applying a more sophisticated merging strategy which uses a specific quantile instead of the median. E.g., it might be possible to use the training data or the most recent data in online learning to determine the quantile. Finally, future work may exploit more the latent space. E.g., the latent space provides probability information, which can be used to generate probabilistic forecasts. Moreover, meaningful areas or directions might exist in the latent space. E.g., if different areas in the latent space correspond to different time series, a global forecasting model could exploit this.

6 Synthetic Data to Enhance Forecasting

Content of this chapter based on

B. Heidrich *et al.*, “Boost short-term load forecasts with synthetic data from transferred latent space information,” *Energy Informatics*, vol. 5, 2022.

This chapter proposes a method to handle the small training data size challenge (mentioned in Section 3.2.2). In particular, the proposed method uses a globally trained generative model that provides a mapping from the data space to the latent space and vice-versa (in this thesis, we use a cINN and a cVAE). Thereby, this chapter aims to answer the corresponding research question:

RQ3 Is it possible to locate a specific time series in a deep generative model’s latent space and use this information to guide the data augmentation process?

This chapter aims to answer this research question by proposing the LSFE, which locates time series in the latent space to generate time series samples that fit the located time series. The next section presents the LSFE in detail. Afterwards, the LSFE is evaluated by gaining insights into the method and benchmarking it with other approaches to cope with the small training data size challenge. Finally, the last section of this chapter discusses the performance of the LSFE, the limitations of the LSFE, and potential further developments.

6.1 Latent Space-based Forecast Enhancer (LSFE)

The LSFE uses generative models, which generate time series to augment the few available data and handle the small training data size challenge. However, for training, generative models usually need the target time series, which is limited in the considered challenge. Thus, instead of training the generative model only on the target time series, it is trained on other related ones. The trained generative model can create different time series. However, it must be guided to generate samples of specific time series since it would create a median time series of the time series used

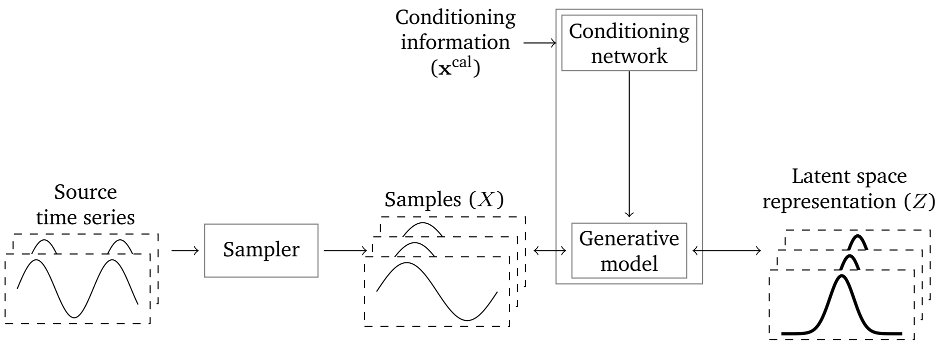
for training. For guidance and creating specific time series, the LSFE examines the latent space of a globally trained generative model. More specifically, the aim is to identify the area of the latent space in which the samples of the target time series are located. Based on this information, it is possible to create purposefully noise that the generative model uses to create the synthetic data.
Figure 6.1 provides an overview of the proposed approach. During training, the conditional generative model is trained in analogy to Chapter 4 and Chapter 5. I.e., a sampler creates time series samples from the source time series. The conditioning information[1] is encoded using a conditioning network. Finally, the encoded conditioning information and the time series samples are passed to the generative model. Again, as in Chapter 4 and Chapter 5, the conditioning network and the generative models are trained jointly.
During production, first, the available target time series samples are mapped to the latent space using f of the generative model and the encoded conditioning information that belongs to the target time series samples. The samples' latent space representation is used to generate additional samples in the latent space (seed noise sampling strategy). These additional samples are mapped back to the time series sample space using g of the generative model and the conditioning information for the time series samples that should be generated. I.e., calendar information can be used to control for which seasons synthetic time series should be generated. The output of the mapping are synthetic time series samples. The available target time series samples and the synthetic time series samples are merged (data combination strategy). This leads to an augmented dataset that can be used to train a forecaster.

The remainder of this section presents the conditional generative model, the seed noise sampling strategy, and the data combination strategy.
Note, analogue as in Section 3.2.1, for better readability, this subsection omits the subscript (t_i, h) when writing the samples, i.e. this subsection writes $\mathbf{z}$, $\mathbf{x}^{\text{cal}}$, and $\mathbf{x}$ instead of $\mathbf{z}_{t_i,h}$, ${\mathbf{x}_{t_i,h}}^{\text{cal}}$, and $\mathbf{x}_{t_i,h}$.

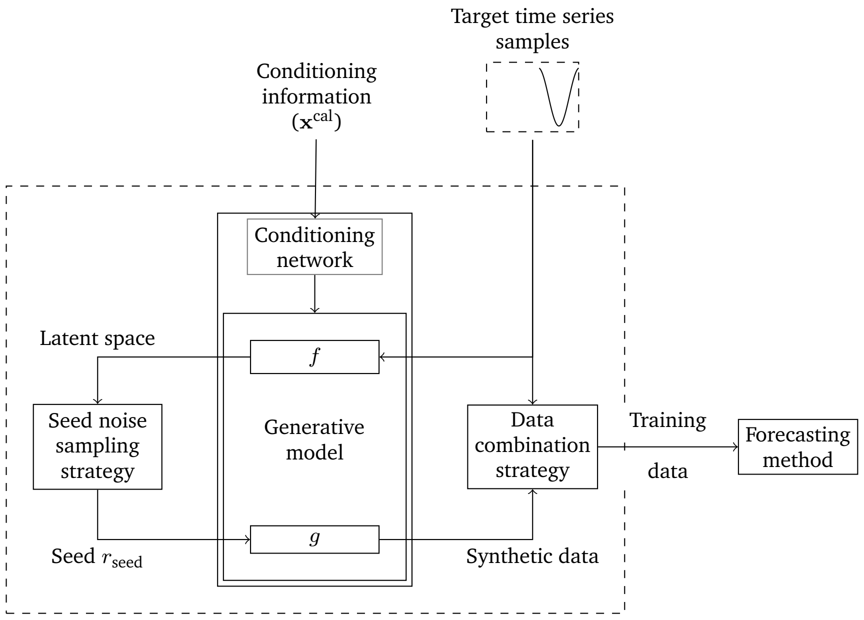
6.1.1 Conditional Generative Model

As mentioned in the introduction, the conditional generative model has two tasks. The first task is to map the available target time series samples to the latent space. The seed noise sampling strategy uses the resulting latent space representation of the samples to locate the time series in the latent space and thus create seed noise

[1]As conditional information, the LSFE considers only calendar information.


Conditioning information ($\mathbf{x}^{\text{cal}}$)
Conditioning network
Source time series
Sampler
Samples (X)
Generative model
Latent space representation (Z)

(a) Training the generative model of the LSFE.


Target time series samples
Conditioning information ($\mathbf{x}^{\text{cal}}$)
Conditioning network
Latent space
f
Seed noise sampling strategy
Generative model
Data combination strategy
Training data
Forecasting method
g
Seed r_{seed}
Synthetic data

(b) Application of the LSFE to generate additional training data for a forecaster. The LSFE (dashed) consists of three components: the conditional generative model with the mappings f and g, the seed noise sampling strategy, and the data combination strategy.

Figure 6.1.: The training of the generative model used in the LSFE and the application of the LSFE to generate time series samples to train a forecaster.

that fits the target time series. Therefore, the conditional generative model has to provide a mapping from the data space to the latent space, i.e.

$$\mathbf{z} = g_{\theta}^{-1}(\mathbf{x}, \mathbf{x}^{\text{cal}}), \tag{6.1}$$

where $\mathbf{x}$ is a time series sample, $\mathbf{x}^{\text{cal}}$ is the sample of the calendar information time series, $\mathbf{z}$ is the latent space representation of this time series sample, g^{-1} the mapping from the data space to the latent space, and θ are the trainable parameters of g.

The second task is to generate synthetic time series samples based on the seed noise

$\mathbf{z}^{\text{seed}}$[2]. Therefore, the conditional generative model maps the seed noise $\mathbf{z}^{\text{seed}}$ from the latent space to the data space, i.e.

$$\mathbf{x} = g_\theta(\mathbf{z}^r, \mathbf{x}^{\text{cal}}). \tag{6.2}$$

To implement g, one can select cINN or cVAE[3]. Note that GANs cannot be used since they only map the latent space to the data space. Thus, as specific models in the evaluations, only the cINN and cVAE architecture from Chapter 4 are used[4].

6.1.2 Seed Noise Sampling Strategy

The task of the seed noise sampling strategy is to generate seed noise, which the conditional generative model uses to generate time series samples. The seed noise sampling strategy has to ensure that the generated seed noise leads to time series samples that fit the target time series. This thesis presents the three used seed noise sampling strategies in the following three paragraphs.

Random The random seed noise sampling strategy assumes that the latent space representation of the target time series samples is normally distributed as the samples of the source time series. Therefore, this strategy samples the seed noise $\mathbf{z}^{\text{seed}} \sim \mathcal{N}(0,1)$ normal distributed.

Around The around seed noise sampling strategy assumes that few samples of the target time series are sufficient to identify the location of the samples of the target time series in the latent space. Therefore, this strategy aims to sample the seed noise around the latent space representation of the available target time series, i.e., $\mathbf{z}^{\text{seed}} = g_\theta^{-1}(\mathbf{x}, \mathbf{x}^{\text{cal}}) + \epsilon = \mathbf{z} + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \sigma)$ is normal distributed.

Shift The shift seed noise sampling strategy assumes that the latent space representation of all time series has a similar shape but are located in different areas. Thus, this strategy searches for a linear transformation between the latent space

[2]This thesis differentiates between the latent space representation samples $\mathbf{z}$ and the seed noise $\mathbf{z}^{\text{seed}}$ to make it understandable which data is currently being considered. However, both should originate from the same space and ideally have the same distribution.

[3]In case of a cVAE, g^-1 is approximated by the encoder, while g denotes the decoder.

[4]Note that the input dimension of the conditioning network differs since this chapter only considers calendar information. Furthermore, the length of generated time series samples is 48 to train forecasters to forecast the next 24 values using the past 24 values as input data.

representations of a source and target time series. This thesis uses linear regression to find such a mapping. In particular, the linear regression is trained with the latent space representations of the source time series samples as input and the latent space representations of the available target time series samples as dependent variables. To consider calendar effects, the used source time series samples must belong to the same time as the available target time series samples. For generating seed noise, this strategy applies the linear regression on the latent space representation of the source time series samples, i.e., $\mathbf{z}^{\text{seed}} = h^{\text{lr}}(g_\theta^{-1}(\mathbf{x}, \mathbf{x}^{\text{cal}})) = h^{\text{lr}}(\mathbf{z})$, where h^{lr} is the linear regression.

6.1.3 Data Combination Strategy

The data combination strategy combines the synthetic time series samples and the available target time series samples to form a training data set for the forecaster. This thesis considers three different strategies.

Synthetic The synthetic data combination strategy assumes that the available target time series samples contain only a little information compared to the synthetic time series samples. Thus, it only uses synthetic time series to train the forecasting model.

Combined The combined data combination strategy assumes that the real target time series and the synthetic time series contain equally relevant information. Thus, the combined data combination strategy uses the available target data and the generated synthetic target data.

Fine-tune The fine-tune data assumes that the available target time series samples contain more relevant information than the synthetic ones. Thus, it first trains the forecasting method on the synthetic time series samples before fine-tuning the forecasting model on the available target time series samples.

6.2 Evaluation

To properly evaluate the LSFE, the next section describes the specific experimental setup. Afterwards, the results are presented.

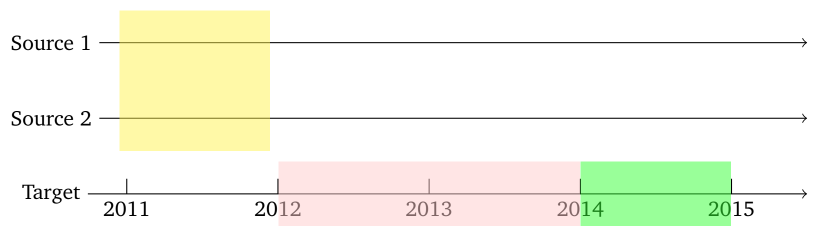


Figure 6.2.: As training data for the conditional generative method, the first year of source time series (yellow) is used, and as test data for the forecasting method the last year of the target time series (green). As available real target time series, the two years in 2012 and 2013 of the target time series (red) are considered. This figure is adapted and based on [48].

6.2.1 Specific Experimental Setup

This subsection on the specific experimental setup describes the used data, the implementation of the LSFE, the considered forecasting models and the metrics.

Data

The evaluation of the LSFE requires a specific data setup to evaluate the LSFE, described first before the used time series, preprocessing, and calendar information are presented.

Data Setup Evaluating the LSFE requires simulating the small training data size challenge. Thereby, it is essential to avoid temporal and spatial data leakages. Thus, this thesis splits the data into three subsets described in Figure 6.2. The first year of the source time series serves as a training dataset for the generative model, and the last year of the target time series is the test time series for the forecasting method. The thesis retrieves the available target time series from the remaining two years between the training data for the conditional generative model. In the evaluation, this thesis considers different amounts of available data. The considered amounts of available data are 2, 4, 8, 16, 21, 52, and 104 weeks. The available amounts of data are always the last weeks of the two years between the training dataset and the test time series. This split ensures that neither a spatial nor a temporal data leakage can occur. Unfortunately, this split is data-intense (at least four years of data). Thus, this thesis can only evaluate the LSFE on the electricity dataset.

Table 6.1.: The architecture of the three exemplary NN-based forecasters.

Layer	FCN	CNN	LSTM
1	Dense 32 ReLU	Conv1D filter=5, kernel=3, ReLU	LSTM 32
2	Dense 24 Linear	Conv1D filter=2, kernel=3, ReLU	Dense 24 Linear
3		Dense 24 Linear	

Used Time Series and Preprocessing As mentioned, this thesis can evaluate the LSFE only on time series from the electricity dataset. In particular, the three time series corresponding to the time series with the minimal, median, and maximal average – namely MT_169, MT_206, and MT_196– are used for evaluation. Each time series is used once as target data and the time series of the other two selected clients as source data, resulting in three combinations of source and target time series.
These time series are normalised. Note the target time series is normalised on the available time series. These normalised time series are transformed into samples of size 48 using Equation (2.2). For the forecasters, the samples created by the generative model are split into an input and a target part. Both parts have a length of 24. As additional features to condition the conditional generative model and support the forecasters, this thesis uses the following calendar information: trigonometric encoded month of the year and hour of the day, as well as the Boolean encoded weekend and public holidays.

LSFE

The evaluation applies the LSFE with each possible combination of seed noise sampling strategy, conditional generative model, and data combination strategy.

Used Forecasting Models

This thesis aims to show that localising the target time series in the latent space and exploiting this by using the LSFE can enhance the performance of forecasters. As exemplary forecasters, this thesis considers a Fully-Connected Neural network (FCN), Convolutional Neural Network (CNN), and Long-Short Term Memory (LSTM). Table 6.1 provides an overview of the architecture. As input, the forecasters get the past 24 hours and the calendar information. The hyperparameters for training the forecasters are provided in Table 6.2.

Table 6.2.: Hyperparameters used to train the NN-based forecasters.

Hyperparameter	Selected value
Optimiser	RMSProp
Learning rate	0.001
Epochs	1000
Early stopping	Yes
Validation size	20% of training data
Shuffle train samples	Yes
Batch size	64
Loss function	MAE (Equation (2.8))

Metrics

The forecasts generated by the three forecasters are evaluated with the MAE (Equation (2.8)) and RMSE (Equation (2.10)). Furthermore, this thesis uses skill scores (Equation (2.18)) with the persistence forecast as a simple baseline for benchmarking. The persistence forecast uses the value one week before as a forecast.

6.2.2 Results

The results are twofold. The first subsubsection presents results related to insights. Afterwards, the second subsubsection presents the benchmarking results.

Insights

The insights comprise the analysis of the influence of the seed noise sampling and data combination strategies on the result, as well as a visual inspection of the forecasted time series. Note the insights use only MT_206 as a target time series and the other as source time series (MT_169 and MT_196) since the results using the other time series as targets are similar.

Seed Noise Sampling Strategy To examine the influence of the seed noise sampling strategies and to compare them, this thesis visualises and compares the seed noise diversity against the latent space representation of the source and target time series samples. Furthermore, this thesis also examines the forecast accuracy with different amounts of available data.
The analysis of the seed noise sampling strategies' diversity is performed by using the t-SNE [136] from sklearn [110] with default parameters. The t-SNE maps the high-dimensional seed noise and latent space representation of the source and target

time series samples on a two-dimensional plane. This two-dimensional plan is visualised. Note to enable better comprehensibility, this evaluation uses only time series samples starting at midnight. Figure 6.3 provides the results. In this figure, this thesis makes four observations: First, for both generative models, the latent space representation of the target and source data does not overlap. Second, the random seed noise sampling strategy does not fit the sources and the target time series for both conditional generative models. Third, for the cINN, the around seed noise sampling strategy's diversity is the most similar to the target diversity. Fourth, for the cVAE, the latent space representations of the source and target time series samples overlap more than for the cINN. Furthermore, for the cVAE, no seed noise sampling strategy fits well to the latent space representation of the target time series samples. However, the around and shift seed noise sampling strategy generates slightly better seed noise than the random seed noise sampling strategy regarding diversity. Besides examining the latent space, this thesis also examines the forecast accuracy of the downstream forecaster using the LSFE with different seed noise sampling strategies. Therefore, three different forecasters are trained to evaluate the LSFE with different amounts of available target data for each seed noise sampling strategy and conditional generative model. As a data combination strategy, this thesis uses the combined strategy in this evaluation.
Figure 6.4 shows the corresponding accuracies in terms of the MAE and RMSE for the different amounts of available data. In this figure, this thesis makes two observations: First, for the cINN, the around seed noise sampling strategy provides the best results. In contrast, for the cVAE, all seed noise sampling strategies perform similarly. Second, the *shift* seed noise sampling strategy leads to a peak at eight weeks of available target data for the cINN. For the cVAE, there is no peak.

Data Combination Strategy For assessing the influence of the different data combination strategies, this thesis examines the forecast accuracy of the downstream forecaster using the LSFE with data combination strategies. Therefore, three different forecasters are trained to evaluate the LSFE with different amounts of available target data for each data combination strategy and conditional generative model. As a seed noise sampling strategy, this thesis selects the around seed noise sampling strategy.
Figure 6.5 shows the accuracy of the different data combination strategies for the different amounts of available target data and the accuracy of a forecaster trained only on the available data (real). Based on this figure, this thesis makes two observations. First, for the cINN, the data combination strategies have no impact on the results. The error of the LSFE using the cINN decreases during the first 21 weeks

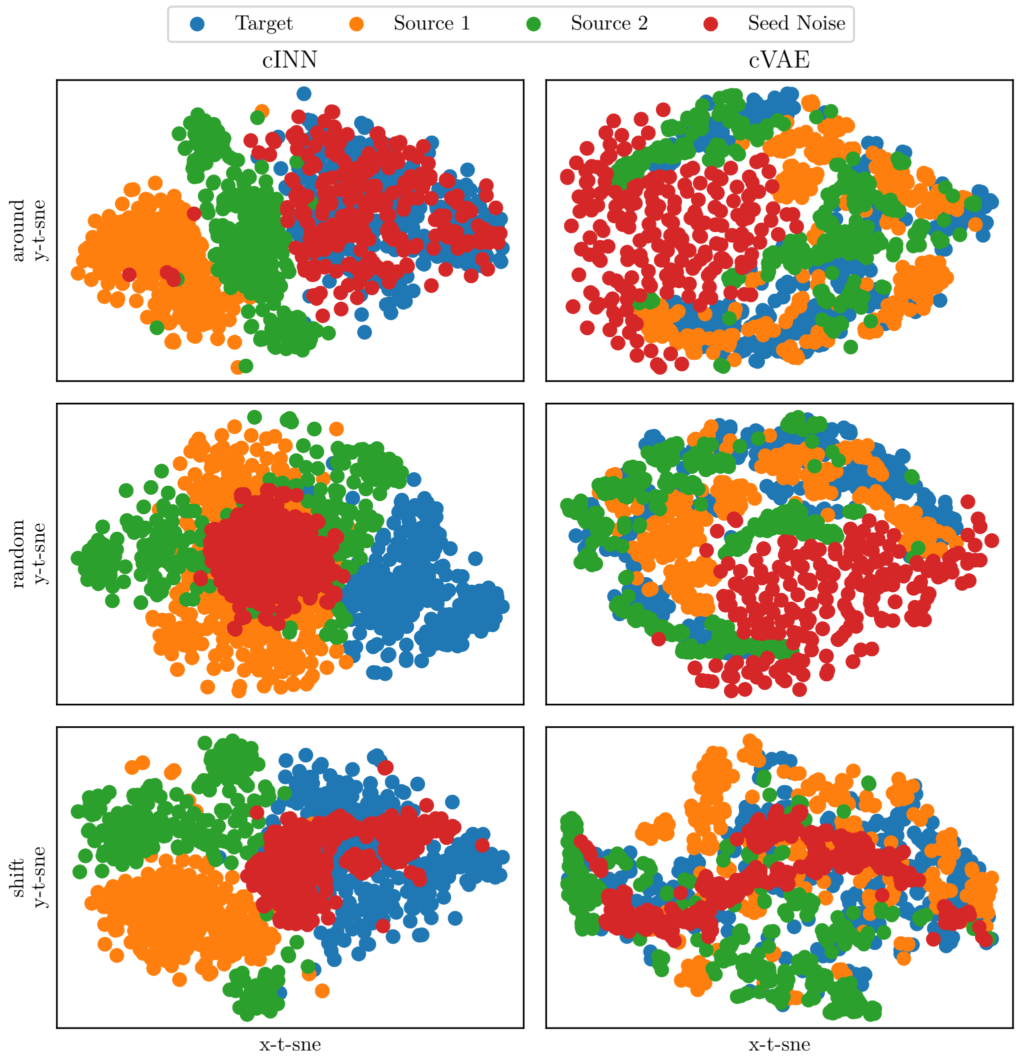


Figure 6.3.: The t-SNE visualisations of the latent space representation of 365 randomly selected samples from the target data (`MT_206`), the seed noise z^{seed}, and the source data (`MT_169` and `MT_196`) for the three seed noise sampling strategies (random, around, and shift) and the two generative models (cINN and cVAE). Note that the different subfigures can only be compared qualitatively due to randomly selected start values and different input data.

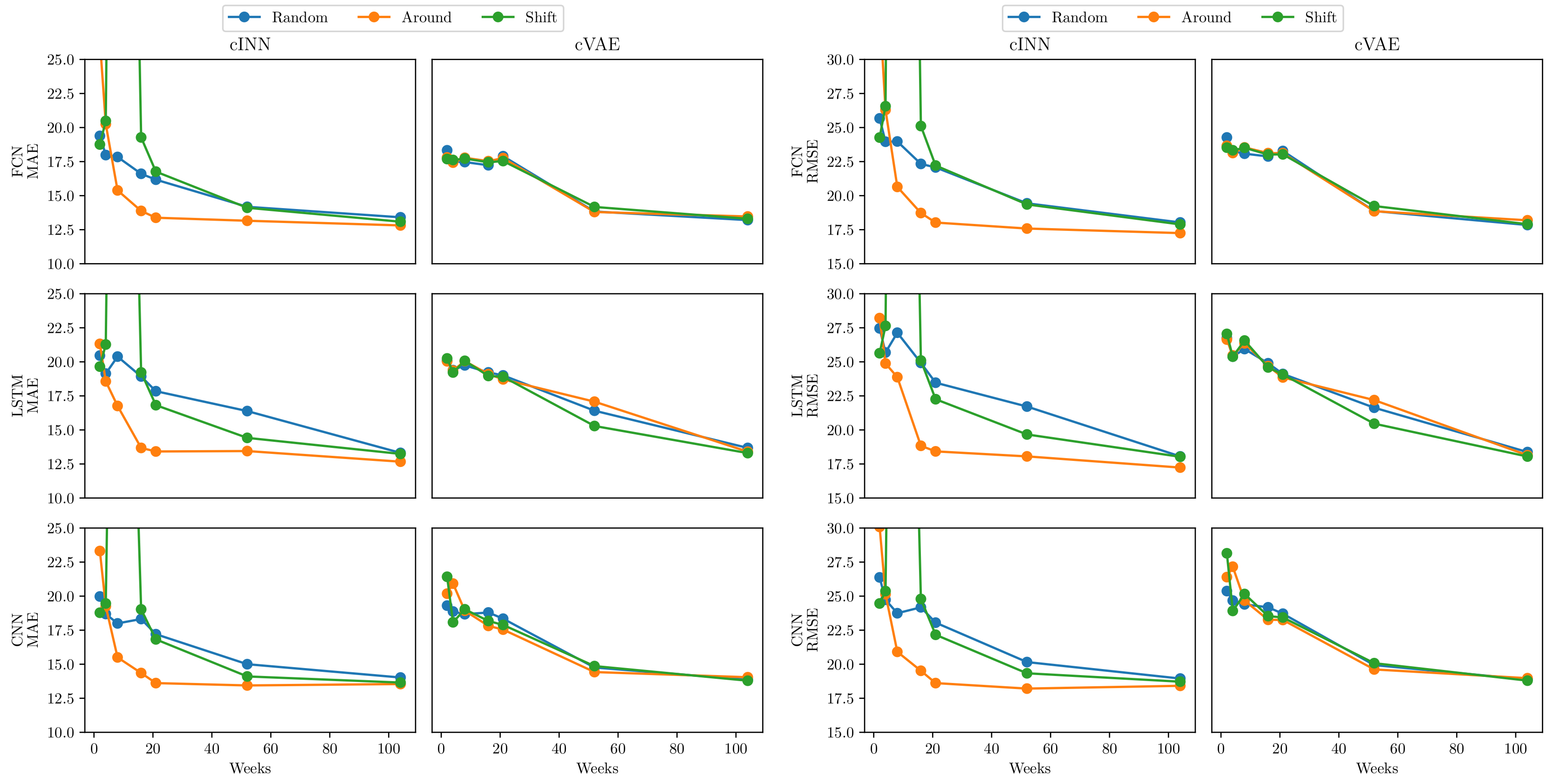


Figure 6.4.: The MAE and RMSE of the three NN-based forecasters using the LSFE with the three seed noise sampling strategies and the two generative models given different amounts of available target data (`MT_206`). The used data combination strategy is the combined data combination strategy.

and stays stable afterwards. Second, in contrast to the LSFE using the cINN, the results for the LSFE using the cVAE are more diverse. In particular, the different data combination strategies lead to differing errors without showing a clear trend, except for the observation that using only synthetic data leads to the highest error.

Visualisation This thesis also visualises the forecasts the forecasters using the LSFE created. In particular, this thesis shows the 24-hour ahead values. This thesis uses the LSFE with both conditional generative models. Still, it fixes the seed noise sampling strategy to the around seed noise sampling strategy and the data combination strategy to the combined data combination strategy. Furthermore, this thesis also visualises the forecast of a forecaster trained only on the available target time series. Figure 6.6, shows the resulting forecasts. Thereby, it is observable that the LSFE improves the forecasts of the forecaster. The improvement is stronger for periods that differ from the data available in training (e.g. July and August).

Benchmarking

The benchmarking compares the performance of the LSFE with other methods for coping with small amounts of available data. In particular, the considered methods are noise-based data augmentation and transfer learning, which are already mentioned in Chapter 3. As LSFE configuration, the configuration with a cINN (LSFE-cINN) and a with a cVAE (LSFE-cVAE) as generative models are used. Both LSFE variants use the around seed noise sampling strategy and the combined data combination strategy.
Figure 6.7 shows the results for the three forecasters and both LSFE configurations regarding the RMSE and MAE based skill score. This thesis makes two observations: First, the LSFE-cINN performs best if 16 to 52 weeks of data are available. If less data is available, the TL approach is better; if more data is available, all approaches perform similarly. Second, each approach outperforms the persistence forecasts after a few weeks (positive skill score).

6.3 Discussion

This section discusses the evaluation performance, the limitations, and the further potential of the LSFE.

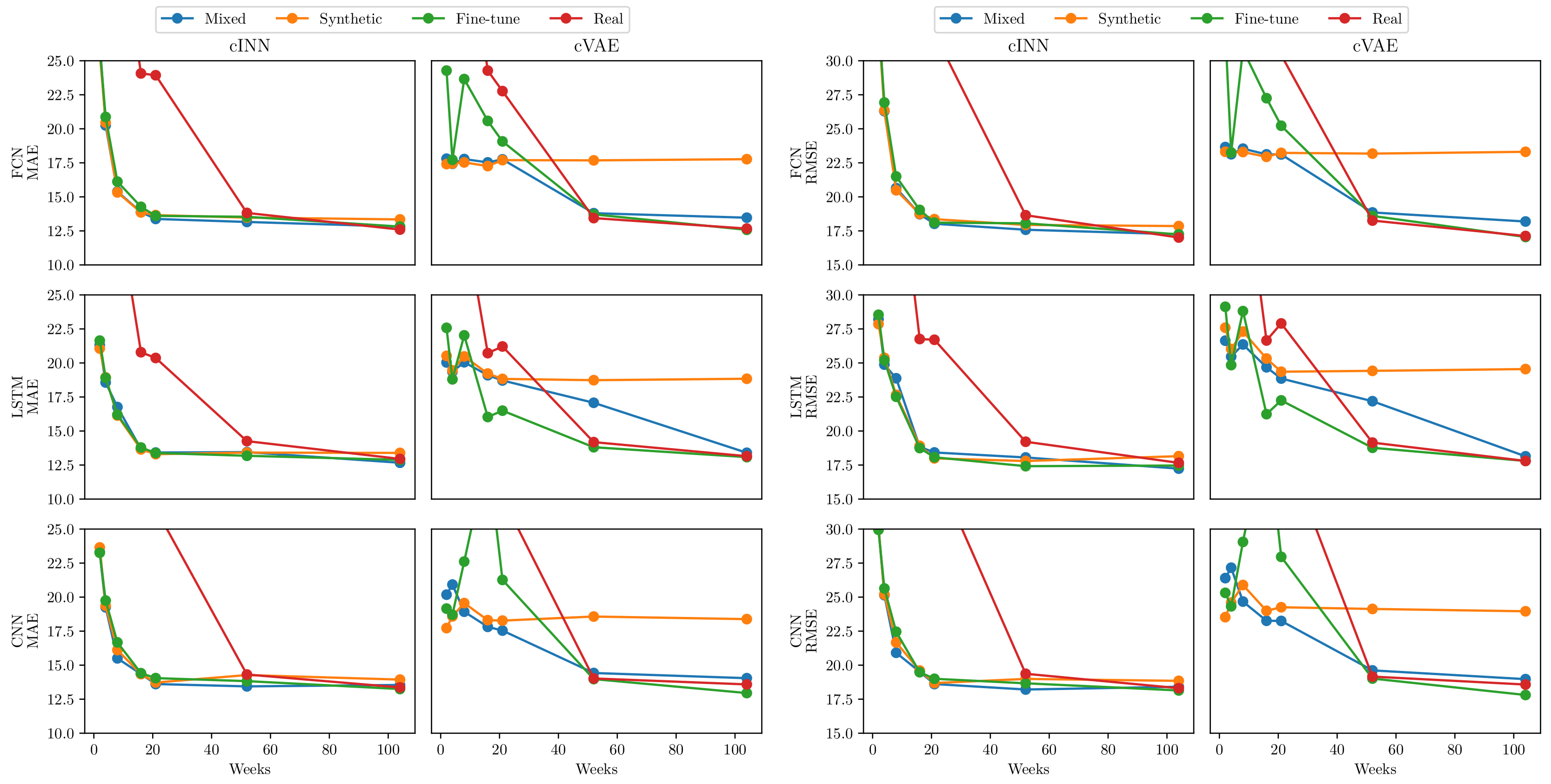


Figure 6.5.: The MAE and RMSE of the three NN-based forecasters using the LSFE with the three data combination strategies and the two generative models given different amounts of available target data (`MT_206`). The used seed noise strategy is the around seed noise strategy.

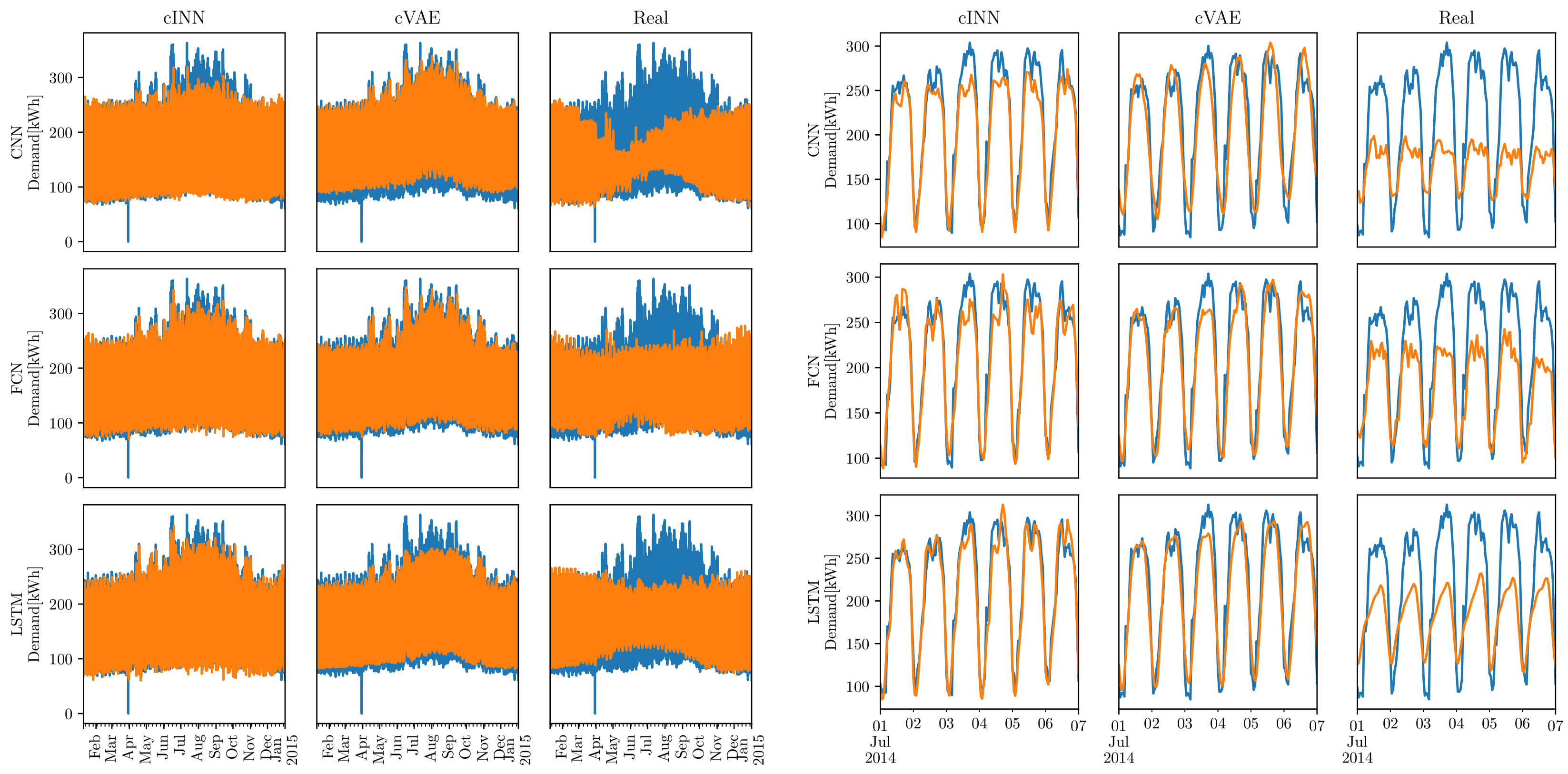


Figure 6.6.: The 24-hour forecasts of the three NN-based forecasters using the LSFE with the cINN and cVAE. Also, the forecasts of the NN-based forecasters trained on the last twelve weeks of data from 2013 are shown. The forecasts are visualised in orange, and the actual values are in blue.

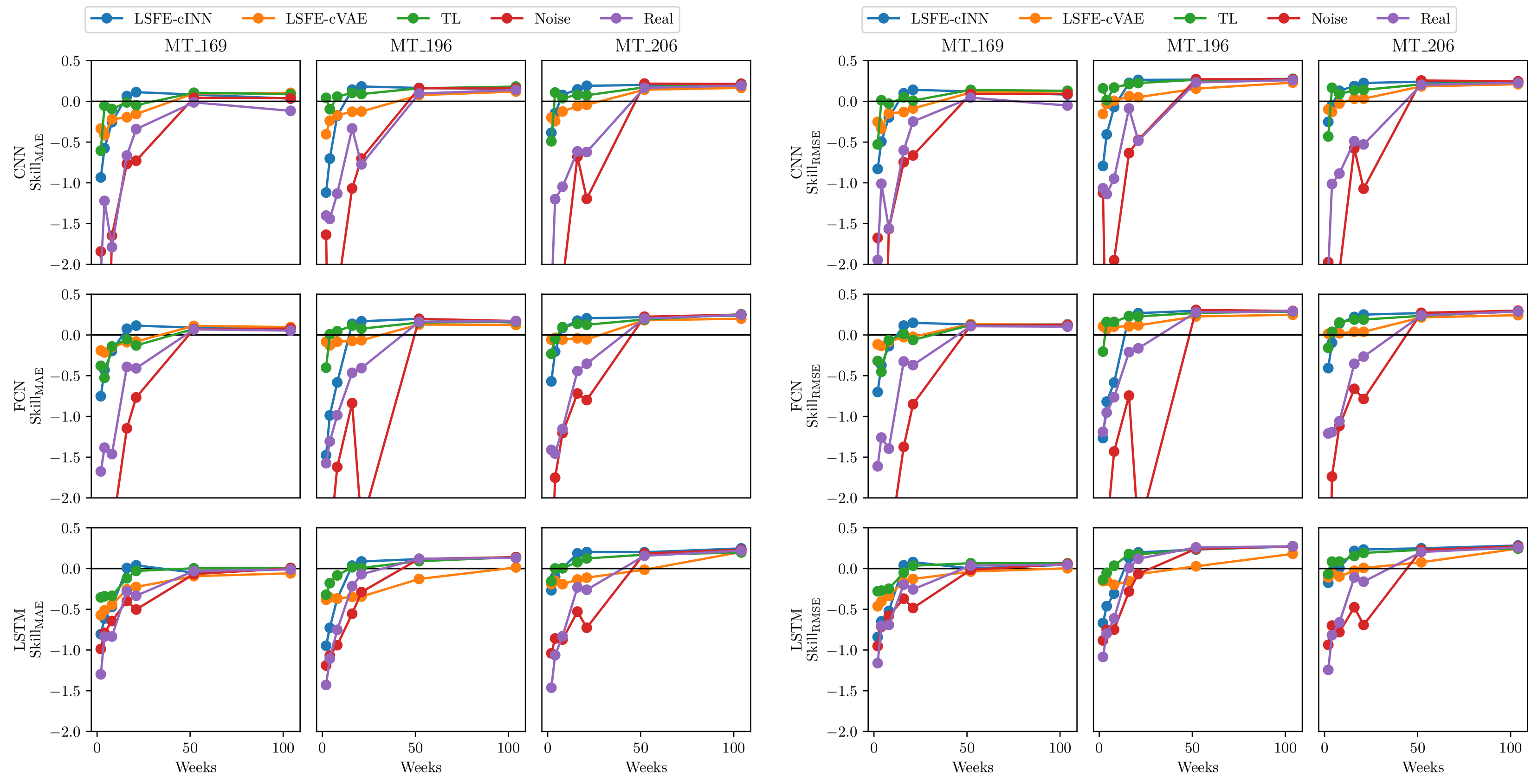


Figure 6.7.: The MAE and RMSE skill score with the persistence forecast as a baseline for the forecasts of the NN-based forecasters using the LSFE with the two conditional generative models and the benchmarks.

6.3.1 Performance

The discussion of the performance focuses on the different seed noise sampling strategies, data combination strategies, exemplary forecasts, and benchmarks.
Regarding the results from the diversity analysis of the seed noise sampling strategies, the around seed noise sampling strategy matches the diversity of the target time series samples' latent space representation better than the other seed noise sampling strategies. In line with that, this seed noise sampling strategy also leads to better convergence. In contrast, when using the cVAE, no seed noise sampling strategies do not match the diversity of the target time series samples' latent space representation. Consequently, forecasters using the LSFE-cVAE perform worse than forecasters using the LSFE-cINN. Concluding, this result shows that it can be beneficial to examine the latent space since this enables the localisation of the target time series when using the LSFE-cINN, which leads to better results.
Regarding the data combination strategy, the different data combination strategies barely impact the forecasts when using the LSFE-cINN in contrast to using the LSFE-cVAE. This observation aligns with the previous observation that the cINN leads to seed noise that better matches the diversity of the target time series samples' latent space compared to the cVAE. Consequently, the LSFE-cINN with the around seed noise sampling strategy generates time series that closely resemble the target data. Thus, the data combination strategy when using the LSFE-cINN combines more similar data than when using the LSFE-cVAE and thus has less impact for the LSFE-cINN. The lower similarity between generated and target time series when using the cVAE also explains why the synthetic data combination approach does not work for LSFE-cVAE. Further analytical investigations need to be performed to figure out if these differences can be traced to a principle difference between the INN and VAE, such as the bijectivity that the INN provides.
Regarding the visualisation, LSFE-cINN and LSFE-cVAE improve the forecasters compared to forecasters trained only on limited available data. The improvement is more substantial for periods that differ from the period of the limited available data. This observation might be explained by the interplay of two aspects: First, the usage of conditional information enables the LSFE to generate data for periods not covered by the available target time series. Second, the seed noise sampling strategy ensures that the generated synthetic data matches the target. Thus, the resulting generated time series samples contain information about these periods while matching the target time series. Thus, the forecaster can learn the target time series' behaviour in periods not covered by the available data.
Regarding the benchmarking, the LSFE is beneficial if the available target time series is limited. However, the benchmarking also shows that other approaches (e.g.

TL) lead to similar results. Thus, the LSFE is an additional tool in the time series forecaster's toolbox, extending the existing tools for handling the small training data size challenge. However, it must always be investigated if the LSFE provides benefits in the considered use case.
Finally, to answer the research question

RQ3 Is it possible to locate a specific time series in a deep generative model's latent space and use this information to guide the data augmentation process?

this thesis concludes that locating specific time series in a conditional generative model's latent space is possible. This localisation supports creating synthetic time series samples that fit the target time series as the LSFE shows.

6.3.2 Limitations

The most important limitations are related to the seed noise sampling strategies, which either introduce new, not obvious-to-choose hyperparameters or new assumptions. The around seed noise sampling strategy adds noise to the latent space representations of the available time series samples. Thus, it requires the sampling parameter σ. While this evaluation uses $\sigma = 0.1$, optimising this hyperparameter for different datasets might improve the results. Regarding additional assumptions, the shift seed noise sampling strategy assumes linear relationships between the sample's latent space representations of different time series. However, in general, this relationship is not linear.

6.3.3 Further Potential of LSFE

This thesis highlights three aspects regarding potential improvements of the LSFE. First, additional information about the time series for the conditioning could lead to better synthetic time series samples. Such information in building-level energy forecasting could be the building size. Second, future work should focus on hyperparameter optimisation of the LSFE, e.g. the hyperparameters of the generative models or seed noise sampling strategy. Furthermore, the conditional generative models should be trained on more data. Training on more data might lead to better latent space representations. Finally, the idea of a global trained conditional generative model together with the localisation of time series in the latent space could be applied to the controllable time series generation (Chapter 4) or the peak load forecasting (Chapter 5) to improve their scalability.

Part III

Profiles for Time Series Forecasting

7 Profiles as Statistics Representatives

As mentioned in Chapter 2, type days are popular in the energy domain to estimate the demand. An advantage of them is that they are easy to obtain and do not require new measurement data by using standardised daily consumption patterns. However, using type days requires meta-information about the time series to assign the correct type days to each time series (e.g. in the energy domain if the building is an office building). Furthermore, they are inflexible, not adaptive to specific time series, and cannot adapt to the time series behaviour changes. Thus, this chapter extends them to be more flexible. More precisely, the first section introduces profiles, and the second section briefly evaluates them. The last section of this chapter discusses the performance, limitations and the further potential of profiles.

7.1 Profiles

As mentioned earlier, the class of time series with calendar-driven periodicities enables specialised algorithms such as type days. Unfortunately, these type days are inflexible. Thus, this thesis uses profiles. Profiles are grouped statistics considering that different time series values have different calendar information. This thesis proposes four different calculation methods for the average profile (p_t^{μ}) and the variance profile ($p_t^{\sigma^2}$)[1]. These calculation methods differ in the amount of past data used and the weights (see Figure 7.1).

[1]Note, to obtain the standard deviation profile, it is possible to calculate $p_t^{\sigma} = \sqrt{p_t^{\sigma^2}}$

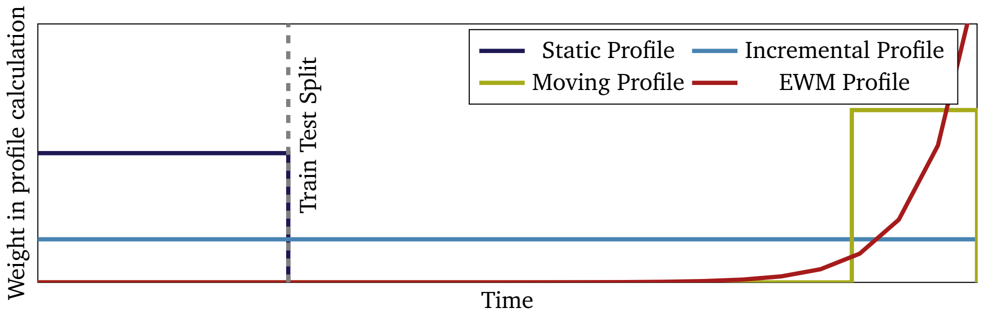


Figure 7.1.: The four considered profile calculation methods use and weight past data differently. Source: [47].

7.1.1 Static profile

The static profile is determined on the training set and remains unchanged throughout the test period. It is defined as

$$p_t^{\mu} = \sum_{i \in \mathcal{T}_{\text{train}}} \frac{1}{n_{t,\mathcal{T}_{\text{train}}}} \begin{cases} x_i, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else}, \end{cases} \tag{7.1}$$

where t is the current time, $\mathcal{T}_{\text{train}}$ is the set of all indexes that belong to the training data, x_t is the value of the time series at t, and $n_{t,\chi}$ is the number of elements in χ with $h_{\text{cal}}(t) = h_{\text{cal}}(i)$. $h_{\text{cal}}(t) = h_{\text{cal}}(i)$ describes that timestamps t and i have the same considered calendar information. I.e., if the hour of the day is the considered calendar information, it means that t and i must have the same hour of the day (e.g. 10 AM). Analogue to the static average profile, the static variance profile is calculated as

$$p_t^{\sigma^2} = \sum_{i \in \mathcal{T}_{\text{train}}} \frac{1}{n_{t,\mathcal{T}_{\text{train}}}} \begin{cases} (x_i - p_i)^2, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else}. \end{cases} \tag{7.2}$$

7.1.2 Incremental Profiles

In contrast to the static profiles, the incremental profiles use at each time step t all available data for the calculation (the corresponding index set is denoted with $\mathcal{T}_{<t}$). More formally, the incremental average profile is defined as

$$p_t^{\mu} = \sum_{i \in \mathcal{T}_{<t}} \frac{1}{n_{t,\mathcal{T}_{<t}}} \begin{cases} x_i, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else}, \end{cases} \tag{7.3}$$

where t, l_t, h_{cal} and $n_{t,\chi}$ are defined as in the static case. The incremental variance profile is defined analogously as

$$p_t^{\sigma^2} = \sum_{i \in \mathcal{T}_{<t}} \frac{1}{n_{t,\mathcal{T}_{<t}}} \begin{cases} (x_i - p_i)^2, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else.} \end{cases} \tag{7.4}$$

7.1.3 Moving Profiles

The moving profiles apply a moving window to forget outdated data and focus on the most current data. I.e., for each time index t, this profile considers only the data in the most recent window of indexes (denoted as $\mathcal{T}_{t-w,t-1}$), with w being the window length. This results in the moving average profile defined as

$$p_t^{\mu} = \sum_{i \in \mathcal{T}_{t-w,t-1}} \frac{1}{n_{t,\mathcal{T}_{t-w,t-1}}} \begin{cases} x_i, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else,} \end{cases} \tag{7.5}$$

where t, l_t, h_{cal} and $n_{t,\chi}$ are defined as for the Static Profile. The corresponding moving variance profile is defined as

$$p_t^{\sigma^2} = \sum_{i \in \mathcal{T}_{t-w,t-1}} \frac{1}{n_{t,\mathcal{T}_{t-w,t-1}}} \begin{cases} (x_i - p_i)^2, & h_{\text{cal}}(t) = h_{\text{cal}}(i) \\ 0, & \text{else.} \end{cases} \tag{7.6}$$

7.1.4 Exponential Weighted Moving (EWM) Profile

Lastly, this thesis introduces the EWM profiles, which consist of rolling subprofiles for each possible outcome of h_{cal}[2]. For the Exponential Weighted Moving Average (EWMA) profile, these results in

$$p_t^{\mu} = p_{t,h_{\text{cal}}(x_t)}^{\mu}, \tag{7.7}$$

where t and h_{cal} are defined as before and the average subprofiles ($p_{t,h_{\text{cal}}(x_t)}^{\mu}$) as

$$p_{t,\phi}^{\mu} = \begin{cases} (1-\alpha) \cdot p_{t-1,\phi}^{\mu} + \alpha \cdot x_{t-1}, & h_{\text{cal}}(t) = \phi \\ p_{t-1,\phi}^{\mu}, & \text{else,} \end{cases} \tag{7.8}$$

[2]These subprofiles are required for the definition of the EWM average and variance profile which needs to be recursive since there is no closed formula.

where t, x_t, and h_{cal} are defined as before, and $\alpha \in (0,1)$ is the smoothing factor of the EWMA profile[3]. The calculation of the Exponential Weighted Moving Variance (EWMV) profile is similar to that of the EWMA profile. It is calculated as

$$p_t^{\sigma^2} = p_{t,h_{\text{cal}}(x_t)}^{\sigma^2}, \tag{7.9}$$

where t, h_{cal} are defined as before and the variance subprofiles ($p_{t,h_{\text{cal}}(x_t)}^{\sigma^2}$) as

$$p_{t,\phi}^{\sigma^2} = \begin{cases} (1-\alpha) \cdot p_{t-1,\phi}^{\sigma^2} + \alpha \cdot (x_{t-1} - p_{t-1}^{\mu})^2, & h_{\text{cal}}(t) = \phi \\ p_{t-1,\phi}^{\sigma^2}, & \text{else.} \end{cases} \tag{7.10}$$

Note if no subprofiles are available for $t-1$, they are initialised with the first time series value that corresponds to ϕ.

7.2 Evaluation

The profiles are an extension of the so-called type days. This thesis evaluates them by considering the average profiles as simple forecasters and thus assesses their forecasting performance. This thesis does not benchmark with forecasters in this section since the profiles are a methodological basis for other methods, not an independent method. In the following, the experimental setup is introduced, and the results are presented.

7.2.1 Experimental Setup

The experimental setup presents the data and used metrics. Furthermore, benchmarking with type days is impossible since no meta-information of the time series is available that allows assigning the correct type days to each time series. As hyperparameters for the moving and EWM profile calculation methods, the window length is set to 28 and the smoothing parameter $\alpha = 0.3$. As calendar information h_{cal}, this thesis considers the hour of the day for each dataset. Furthermore, note that for each time series, separate profiles are calculated and that the profile calculation methods are not applied globally.

[3]The smoothing factor has to be set by the user.

Table 7.1.: The average nMAE and nRMSE scores and ranks for the electricity and traffic dataset for the four presented profiles calculation methods.

(a) nMAE

		Static	Incremental	Moving	EWM
Electricity	nMAE-Score	0.43	0.3	0.22	0.19
	nMAE-Rank	3.95	3	1.97	1.08
Traffic	nMAE-Score	0.41	0.34	0.31	0.31
	nMAE-Rank	3.53	2.58	1.59	2.3

(b) nRMSE

		Static	Incremental	Moving	EWM
Electricity	nRMSE-Score	0.53	0.42	0.32	0.30
	nRMSE-Rank	3.89	3.06	1.97	1.08
Traffic	nRMSE-Score	0.66	0.62	0.59	0.60
	nRMSE-Rank	3.16	2.70	1.52	2.63

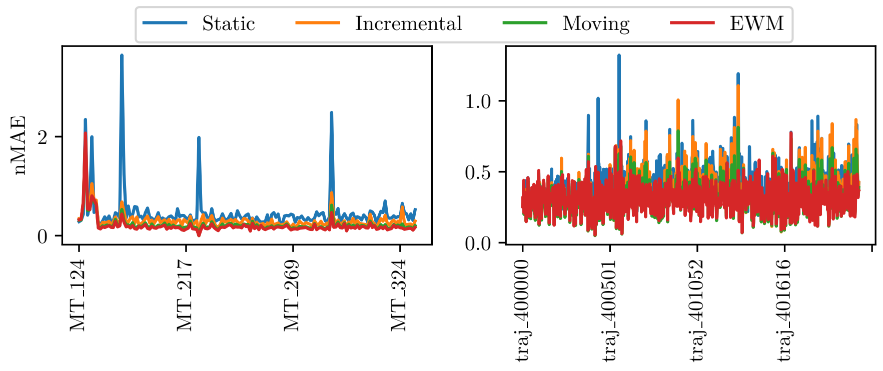
Data As data, this thesis uses the electricity and the traffic dataset. The static profile is created using the data from January 2011 to December 2013 for the electricity dataset and from January 2008 to December 2008 for the traffic dataset. As test data, this thesis uses the period from January 2014 to December 2014 for the electricity data and from January 2009 to March 2009 for the traffic dataset.

Metrics This thesis uses the average scores and ranks of the nMAE and nRMSE to evaluate the profiles.

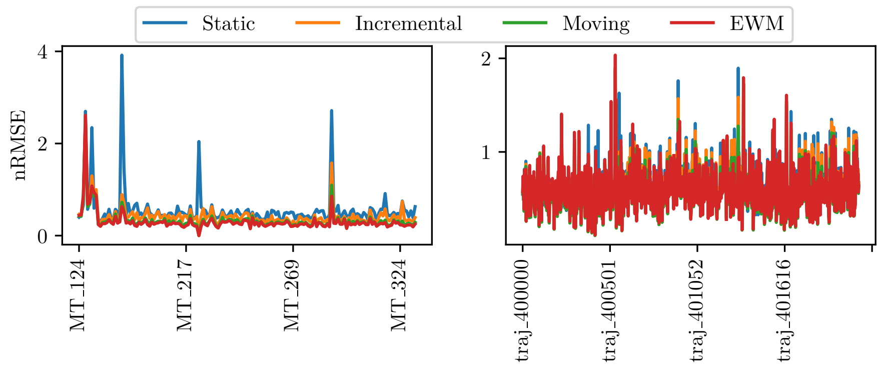
7.2.2 Results

The results are twofold. The first part examines the average score and ranks. The second takes a more detailed look at the per-time series results.

Average ranks and scores Table 7.1 provides the average scores and ranks for all four profile calculation methods and both datasets. This thesis makes two observations: First, the exponential weighted moving average and moving average profiles perform better than the static and incremental average profiles on both datasets. Moreover, the static average profile always leads to the worst results. Second, comparing the ranks of the moving and EWM average profile on the electricity data set, the EWMA profile is better. Still, on the traffic, the moving average profile is better. However, regarding their scores, they perform similarly.



(a) nMAE



(b) nRMSE

Figure 7.2.: The nMAE and nRMSE for each time series in both datasets for each of the four presented profile calculation methods.

Per Time Series Performance The accuracy of the per time series results in Figure 7.2 for the electricity dataset seems more stable across the different time series than for the traffic time series. In this figure, for the electricity dataset, we observe more error peaks for the static profile than for the other profile calculation methods.

7.3 Discussion

This section discusses the presented profile calculation methods' performance, limitations, and potential.

7.3.1 Performance

Regarding the performance, this thesis discusses three aspects. First, the adaptive profile calculation methods provide better forecasts than the static and incremental calculation methods. Second, regarding the different datasets and time series, there is not always a clear order between the EWMA and moving average profile. Thus, it is not possible to provide a recommendation on which is the best profile calculation method. Third, regarding the per-time series evaluation, there are error peaks. Investigating exemplary error peaks in more detail, probably concept drifts leads to them in the static profile.
For wrapping up the discussion on the results, this thesis concludes that profiles are an easily calculable representative of the average statistic. Furthermore, it is a flexible extension of the type days that do not require meta-information on the time series.

7.3.2 Limitations

Regarding the limitations of the profile calculation methods, this thesis highlights three aspects.
First, the profile calculation methods require that there are time series measurements. In contrast, such measurements are not required for the type days; instead, they need meta-information on the time series. Second, despite being computationally cheap, profiles must be calculated per time series. In contrast, type days are precalculated and need not be calculated separately for each time series. However, since the profile calculation is efficiently implementable, this should not arise in scaling issues. Third, the introduction of profiles may also introduce additional hyperparameters. E.g., the EWM and moving profiles require a smoothing parameter α or a window length.

7.3.3 Further Potential of Profiles

Regarding the further potential of profiles, this thesis highlights two aspects. First, the profile's adaptive behaviour seems promising if integrated as a preprocessing to cope with concept drifts. Chapter 8 examines this in more detail. Second, the average and variance profiles can be integrated as additional input for neural networks. Integration would also enable to learn the best weights for combining profiles and other components that correct the profiles. Chapter 9 investigates this potential.

8 Enhancing Handling of Concept Drifts with Profiles

Content of this chapter based on

B. Heidrich, N. Ludwig, M. Turowski, R. Mikut, and V. Hagenmeyer, “Adaptively coping with concept drifts in energy time series forecasting using profiles,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 459–470.

As mentioned in Chapter 3, concept drifts are challenging in time series forecasting. E.g., in electricity time series, exchanging appliances and new buildings can lead to concept drifts. Thus, this chapter proposes the Profile-based Framework for Concept Drift Handling (PDH) framework for coping with concept drifts in time series to be able to answer

RQ4 Can profiles improve the ability of machine learning models to cope with concept drifts?

This chapter answers this research question by introducing the PDH that uses profiles to handle concept drifts. The PDH assumes that concept drifts only influence the time series level. In energy time series forecasting, Vrablecová *et al.* [138] makes a similar assumption. The underlying idea of the proposed framework is to use the average profile to represent the time series level. By selecting an adaptive profile calculation method, the remainder contains fewer concept drifts since the adaptability of the profile handles most of the concept drifts.
The remainder of this chapter is structured as follows. First, the next section presents the proposed framework. Afterwards, this thesis evaluates the PDH. Finally, the last section of this chapter discusses the proposed PDH.

8.1 Coping with Concept Drifts using Profiles

This section introduces how profiles can support machine learning models coping with concept drift under the assumption that the concept drifts mainly affect the

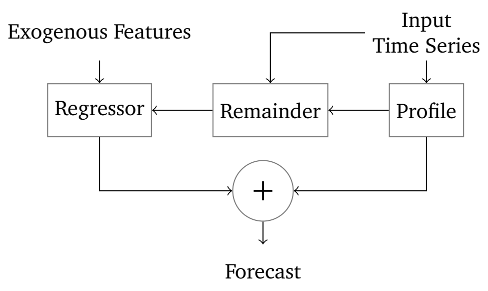


Figure 8.1.: The PDH sums a profile and the output of a regression method to obtain a prediction. This figure is adapted and based on [47].

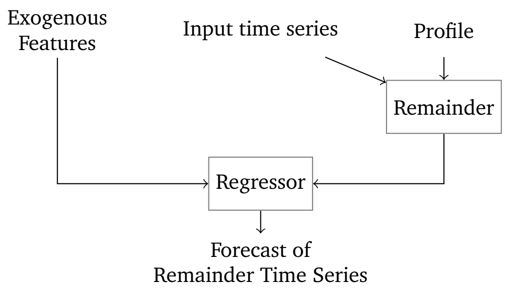


Figure 8.2.: The regressor inputs exogenous features and the historical remainder time series. The historical remainder time series is the difference between the profile and the historical time series. Different information, such as calendar and weather information, could be used as an exogenous time series. The regressor's output is the forecast of the remainder time series for the next time steps. This figure is adapted and based on [47].

time series level as Vrablecová *et al.* [138] assume for load time series. Based on this assumption, the PDH models the level and the remainder of the time series separately and adds them later (see Figure 8.1). For the time series level, the PDH uses an adaptive average profile introduced in Chapter 7 since they consider periodicities and the typical daily behaviour of the time series. For the remainder (the difference between the profile and the time series), PDH uses a regressor. Finally, the prediction of the regressor and the average profile are added to form the forecast. The PDH is not bound to a specific machine learning model as a regressor. Instead, it is applicable to various ones. Generally, the selected machine learning model takes the historical remainder time series and the exogenous features time series as input to predict the future values of the remainder time series (compare Figure 8.2).

8.2 Evaluation

This section first introduces the experimental setup and presents the results afterwards.

8.2.1 Experimental Setup

The evaluation of the PDH requires an online learning-based evaluation approach. Thus, this thesis simulates such an evaluation by running the PDH and the benchmarks mini-batch by mini-batch. Each mini-batch comprises one day of the test set (24 samples). For each time step of a mini-batch, the proposed framework and the benchmarks produce a forecast for the next 24 hours. After each mini-batch and if the PDH or benchmarks support retraining, checks are performed to detect if retraining should be triggered. After processing all mini-batches, this thesis calculates the metrics on the forecasts of the proposed frameworks and the benchmarks. The used metrics, regressors, data, hardware, retraining strategies, and benchmarks are presented in the following.

Metrics The PDH is assessed by examining the accuracy and the computational time. This thesis uses the MAE and RMSE to assess the accuracy. Moreover, to examine the accuracy over time, this thesis also calculates the nMAE and nRMSE on a rolling window with a window size of 28 days.
To assess the computational effort, the training time, retraining time, and sum of both times of the PDH and the benchmarks are measured.

Used Regressors The PDH can be used with different regressors. This thesis uses three regressors: linear regression, XGBoost regressor, and Multi-Layer Perceptron (MLP) regressor. This thesis uses the default implementation available in sklearn [110] respectively XGBoost [16] for each of these three regressors. As input, each regressor gets the past 24 hours of the remainder time series and the trigonometric encoded hour of the day, day of the week and month of the year as exogenous features for the values that should be predicted. As output, the regressors provide their forecasts of the next 24 values of the remainder time series.

Table 8.1.: The statistical conditional information to create the synthetic time series with concept drift using the approach presented in Chapter 4.

Name	Drift length	Position	Statistical Information		
	n_{drift}		Before	During	After
Grad-Inc	2920	01.05.2014 16:00	μ_{X_t}	$\mu_{X_t} + \frac{0.5*i}{n_{\text{drift}}} * \mu_{X_t} \forall i \in \{1, 2, ..., n_{\text{drift}}\}$	$1.5 * \mu_{X_t}$
Grad-Dec	2920	01.05.2014 16:00	μ_{X_t}	$\mu_{X_t} - \frac{0.5*i}{n_{\text{drift}}} * \mu_{X_t} \forall i \in \{1, 2, ..., n_{\text{drift}}\}$	$0.5 * \mu_{X_t}$
Sud-Inc	2	01.05.2014 16:00	μ_{X_t}	$\mu_{X_t} + \frac{0.5*i}{n_{\text{drift}}} * \mu_{X_t} \forall i \in \{1, 2, ..., n_{\text{drift}}\}$	$1.5 * \mu_{X_t}$
Sud-Dec	2	01.05.2014 16:00	μ_{X_t}	$\mu_{X_t} - \frac{0.5*i}{n_{\text{drift}}} * \mu_{X_t} \forall i \in \{1, 2, ..., n_{\text{drift}}\}$	$0.5 * \mu_{X_t}$

Data This thesis uses synthetic and real time series with concept drifts to evaluate the LSFE.
The approach presented in Chapter 4 is used to generate synthetic data. More specifically, the cINN is used. It is trained on the time series corresponding to the median of the time series' averages from both datasets – namely MT_206 and traj_401507. The trained cINN generates time series for the time span from January 2011 to December 2014 by using the corresponding calendar information as conditioning information. As additional conditioning information to integrate concept drifts and control the time series generation, this thesis uses statistical information. In particular, this thesis uses linear and step functions that either increase or decrease. Table 8.1 provides the exact specification of the functions.
As real time series with recorded concept drift, this thesis uses MT_188 and MT_316 from the electricity and traj_400367 and traj_402051 from the traffic dataset. Note, that we cannot provide an exact start position of the concept drifts in these time series.

Hardware and Software This thesis uses the setup 1 (Table 2.2) for all experiments to compare the computational efforts better.

Retraining Strategies As mentioned in Chapter 3, a popular approach to cope with concept drifts is retraining. Thus, during the evaluation, this thesis uses different retraining strategies for benchmarking and to assess if they provide further improvements to the PDH. As mentioned previously, the online evaluation approach operates on mini-batches. Thus, after each mini-batch, the considered retraining strategies decide if the model should be refitted. This thesis uses the following three strategies: **None** never triggers a retraining, **Periodic** triggers a retraining after every 30 days, and **Detection** triggers a retraining if ADWIN [9] detects a concept drift.

Benchmark Models For the comparison, this thesis uses twelve benchmark models. Nine of the twelve models are combinations of the three retraining strategies, and the three used regressors. The remaining three models are online models designed for drifting scenarios already mentioned in Chapter 3 – namely the Online Sequential Extreme Learning Machine (OS-ELM) [79], the Online Adaptive Recurrent Neural Network (OARNN) [33], and the Error Intersection Approach (EIA) [5]. The OS-ELM [79] is a NN with one hidden layer using random weights and output weights determined using the Least Square Method. The OARNN [33] consists of a simple RNN and normalization step that are updated after each time step. Additionally, the OARNN performs a Bayesian optimization for the RNN's hyperparameters if the forecast error is too high. EIA [5] is an ensemble approach consisting of persistence forecast and a complex model (neural network), which are exchanged if their error curves intersect[1]. For the EIA and the OARNN, this thesis uses its own implementations, while for the OS-ELM, an existing implementation is used[2].

8.2.2 Results

The results examine which and how profiles support machine learning models in handling concept drift, compare the proposed framework with benchmarks, and assess the computational costs of the PDH and the benchmarks.

Insights

The insights evaluate whether profiles support machine learning models in handling concept drifts. Therefore, three experiments are conducted on the synthetic time series for which the location and type of the containing concept drift are known. The first experiment evaluates the PDH using different profile calculation methods. The second experiment examines the retraining strategies' influence on the forecast accuracy of the PDH. The last experiment examines the error curves through time of the PDH using different profiles and regressors.

Evaluating Different Profiles This paragraph evaluates how the PDH performs with different profiles. Thus, the PDH with each average profile and regressor is evaluated.

[1]Note that the authors state that this model is only suitable for sudden concept drifts.
[2]https://github.com/leferrad/pyoselm

Table 8.2 shows the resulting MAE and RMSE. In this table, this thesis makes three observations. First, the PDH using the EWMA profile achieves the best result regarding both considered metrics with almost all regressors on almost all considered time series. The exceptions are the synthetic electrical time series with the gradual decrementing concept drift with the MLP and XGBoost regressor. The PDH with a moving average profile performs better in both cases. Second, the PDH using the static average profile followed by the incremental average profile achieves the worst results. Third, the different regressors are all similarly affected by using different profiles in the PDH.

Evaluating Retraining Strategies The second insights-related evaluation examines if a periodic or detection-based retraining strategy further improves the PDH. Therefore, this thesis uses the MAE skill score for the PDH with the periodic and detection-based retraining strategy. The used baseline is the PDH equipped without a retraining strategy.

Table 8.3 reports the skill scores. In this table, this thesis makes three observations: First, for the traffic time series and the EWMA profile, neither periodic nor detection-based retraining strategies provide an improvement. More generally, when examining both datasets, in most cases, the retraining strategies provide no noticeable benefit except for some cases with XGBoost as a regressor on the electricity dataset. Second, in contrast to the moving average and EWMA profile, the retraining strategies improve the results of the PDH using the static and incremental average profiles. Last, the usefulness of the retraining strategies depends on the regressors. For XGBoost, the retraining impacts the errors more positively than for the Linear Regression (LR).

Evaluating Profiles over Time This thesis also examines the rolling nMAE and nRMSE for all synthetic time series, all profiles, and all regressors to get more insights into how the profiles impact the adaption to concept drifts.

Based on the results in Figure 8.3 and Figure 8.4, this thesis makes three observations: First, the four concept drifts permanently affect the PDH with static or incremental average profiles. I.e., when the concept drift occurs, the error rises and stays high. In most cases, the error of the PDH with the static profile is higher than that of the PDH with the incremental average profile. Furthermore, in contrast to the PDH with the static profile, the incremental profile's error slightly decreases. Second, in contrast to the PDH with the static and incremental profile, with the moving and

Table 8.2.: The MAE and RMSE of the PDH using different profile calculation methods and regressors on synthetic time series with concept drifts.

(a) Electricity

		Grad-Inc			Grad-Dec			Sud-Inc			Sud-Dec		
		LR	MLP	XGBoost	LR	MLP	XGBoost	LR	MLP	XGBoost	LR	MLP	XGBoost
MAE	Static	30.84	48.36	36.57	22.61	13.28	29.05	37.73	59.43	45.43	27.49	20.41	28.26
	Incremental	28.61	32.41	34.03	20.86	11.75	25.35	34.33	42.52	43.58	25.13	21.83	28.60
	Moving	4.53	4.86	4.47	3.51	**3.08**	**3.24**	5.17	5.94	4.90	4.29	4.08	3.93
	EWMA	**3.38**	**3.38**	**2.95**	**2.60**	3.20	3.40	**3.85**	**4.17**	**2.99**	**3.33**	**3.37**	**2.81**
RMSE	Static	42.56	72.71	51.75	29.29	19.11	39.26	50.87	86.29	60.55	35.62	30.30	38.16
	Incremental	39.29	47.87	47.82	26.82	16.74	34.13	46.41	60.34	57.65	32.47	31.76	38.11
	Moving	6.76	7.40	7.14	6.61	5.33	**6.07**	11.98	15.55	13.45	9.46	9.08	9.75
	EWMA	**5.50**	**4.79**	**4.66**	**5.29**	**4.77**	6.12	**7.95**	**9.49**	**7.26**	**7.12**	**6.66**	**6.23**

(b) Traffic

		Grad-Inc			Grad-Dec			Sud-Inc			Sud-Dec		
		LR	MLP	XGBoost	LR	MLP	XGBoost	LR	MLP	XGBoost	LR	MLP	XGBoost
MAE	Static	0.0086	0.0099	0.0092	0.0067	0.0089	0.0138	0.0061	0.0095	0.0100	0.0066	0.0079	0.0123
	Incremental	0.0084	0.0089	0.0087	0.0068	0.0073	0.0125	0.0069	0.0094	0.0094	0.0068	0.0061	0.0113
	Moving	0.0013	0.0014	0.0014	0.0012	0.0012	0.0015	0.0013	0.0014	0.0014	0.0011	0.0012	0.0013
	EWMA	**0.0009**	**0.0009**	**0.0010**	**0.0009**	**0.0009**	**0.0009**	**0.0010**	**0.0010**	**0.0010**	**0.0008**	**0.0008**	**0.0008**
RMSE	Static	0.0159	0.0182	0.0208	0.0093	0.0133	0.0197	0.0105	0.0152	0.0188	0.0090	0.0115	0.0170
	Incremental	0.0159	0.0174	0.0195	0.0093	0.0110	0.0177	0.0115	0.0157	0.0178	0.0093	0.0087	0.0157
	Moving	0.0031	0.0031	0.0031	0.0022	0.0022	0.0030	0.0039	0.0040	0.0045	0.0031	0.0031	0.0036
	EWMA	**0.0018**	**0.0018**	**0.0019**	**0.0012**	**0.0012**	**0.0014**	**0.0023**	**0.0023**	**0.0024**	**0.0017**	**0.0017**	**0.0017**

Table 8.3.: The MAE skill score of the retraining strategies Periodic and Detection (Section 8.2.1) applied to the PDH with different profiles and regressors for both datasets. The corresponding PDH without retraining is used as a baseline of the skill score. I.e, the PDHs with the Periodic and Detection retraining strategy are compared with the PDH without retraining. Thus, a value higher than one means that the retraining strategy worsens the PDH, while a value lower than one indicates an improvement.

(a) Electricity

		Grad-Inc		Grad-Dec		Sud-Inc		Sud-Dec	
		Periodic	Detection	Periodic	Detection	Periodic	Detection	Periodic	Detection
LR	Static	0.20	0.31	0.22	0.33	0.22	0.25	0.25	0.30
	Incremental	0.21	0.32	0.23	0.35	0.24	0.27	0.27	0.32
	Moving	0.88	1.00	0.94	1.01	1.03	1.00	1.02	1.00
	EWMA	0.99	1.00	1.00	1.00	0.98	1.00	0.98	1.00
MLP	Static	0.11	0.21	0.28	0.43	0.16	0.13	0.24	0.27
	Incremental	0.15	0.26	0.30	0.49	0.21	0.17	0.23	0.26
	Moving	0.74	1.00	0.96	1.00	0.98	0.91	0.90	0.99
	EWMA	0.92	1.00	0.76	1.00	0.96	1.00	0.90	1.00
XGBoost	Static	0.13	0.38	0.11	0.22	0.13	0.11	0.19	0.16
	Incremental	0.14	0.36	0.14	0.23	0.15	0.16	0.20	0.20
	Moving	0.64	1.00	0.81	1.00	0.97	1.06	0.93	0.97
	EWMA	0.78	1.00	0.56	1.00	0.91	1.00	0.82	1.00

(b) Traffic

		Grad-Inc		Grad-Dec		Sud-Inc		Sud-Dec	
		Periodic	Detection	Periodic	Detection	Periodic	Detection	Periodic	Detection
LR	Static	0.16	0.23	0.19	0.30	0.28	0.28	0.23	0.24
	Incremental	0.17	0.24	0.19	0.29	0.25	0.25	0.22	0.24
	Moving	0.92	1.08	0.92	1.00	1.08	1.00	1.09	1.00
	EWMA	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
MLP	Static	0.17	0.26	0.18	0.31	0.22	0.20	0.20	0.24
	Incremental	0.19	0.28	0.22	0.36	0.22	0.20	0.26	0.30
	Moving	0.86	1.07	1.00	1.00	1.00	1.00	1.00	1.00
	EWMA	1.11	1.00	1.00	1.00	1.00	1.00	1.00	1.00
XGBoost	Static	0.16	0.29	0.11	0.19	0.19	0.15	0.17	0.13
	Incremental	0.17	0.28	0.12	0.18	0.20	0.20	0.19	0.17
	Moving	0.79	1.00	0.87	1.00	1.00	1.07	1.08	1.00
	EWMA	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

EWM average profile, the rolling errors only peak when the concept drift occurs and recover afterwards. I.e., the PDH with the moving or EWM average profile achieves an error after the concept drift similar to the error before the concept drift. Third, regarding the peaks caused by the concept drifts, the peaks of the sudden concept drifts are smaller than those of the gradual concept drift.

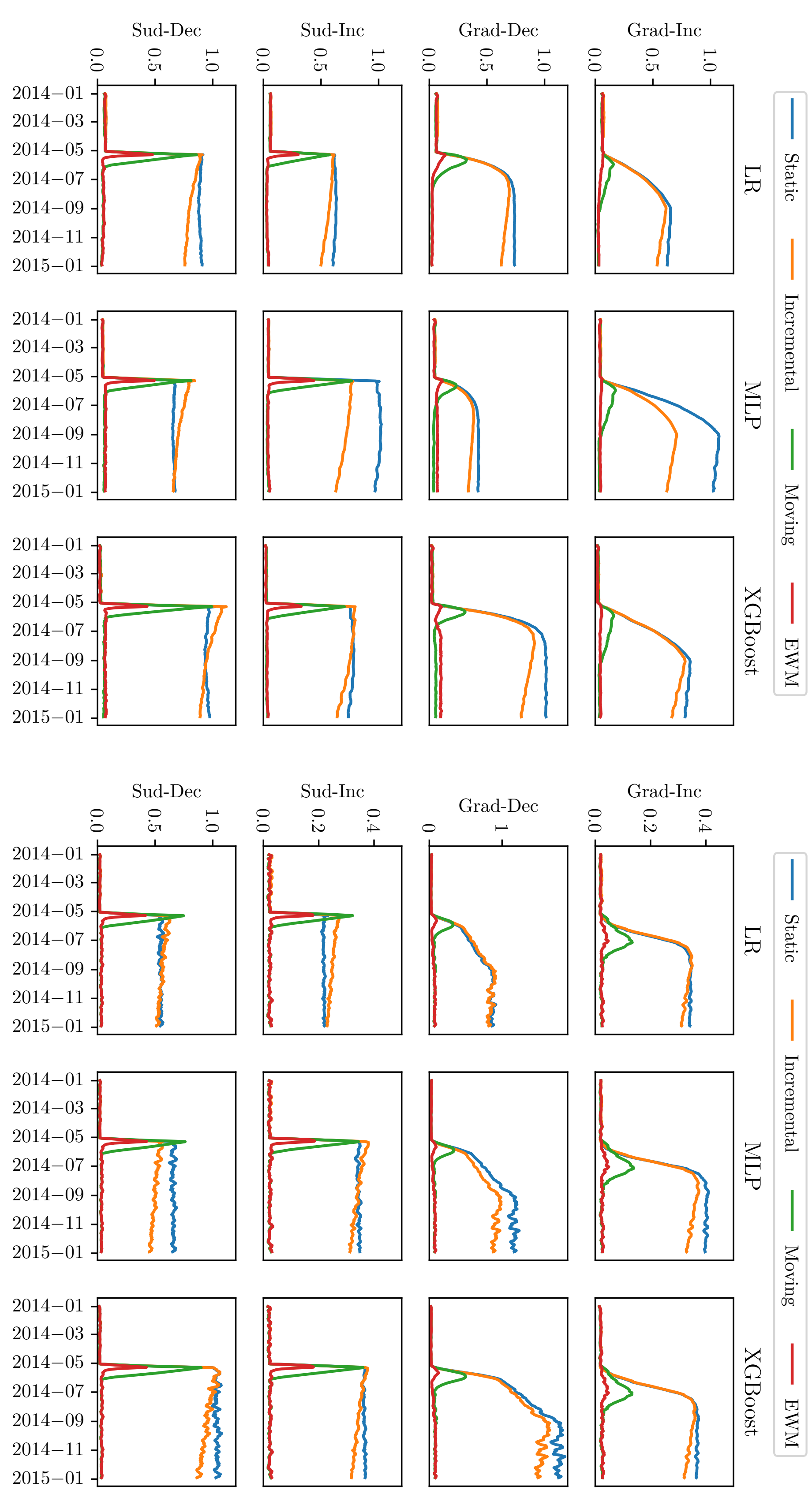
Benchmarking

The benchmarking compares the PDH using the adaptive profiles (the moving and EWM average profile) with the benchmark models. Therefore, this thesis uses two real-world time series with recorded concept drifts from each dataset: the time series MT_188 and MT_316 from the electricity and traj_402051 and traj_400367 from the traffic dataset. Table 8.4 shows the results. Based on these results, this thesis makes three observations: First, either the PDH with the moving or EWM average profiles achieves the lowest error. An exception is MAE on the time series traj_402051 from the traffic dataset. There, XGBoost, with a periodic retraining strategy, achieves the lowest error. However, the PDH with XGBoost achieves similar results. Second, while on the two time series from the electricity, all base learners perform similarly, on the two time series from the traffic dataset, XGBoost clearly outperforms the other base learners. Third, when examining the differences between the benchmarks and the PDH, the improvement of the MAE differs between the different time series (11.8% for MT_188, 12.1% for MT_316, -4.7% for traj_402051 and 15.1% for traj_400367).

Computational Costs

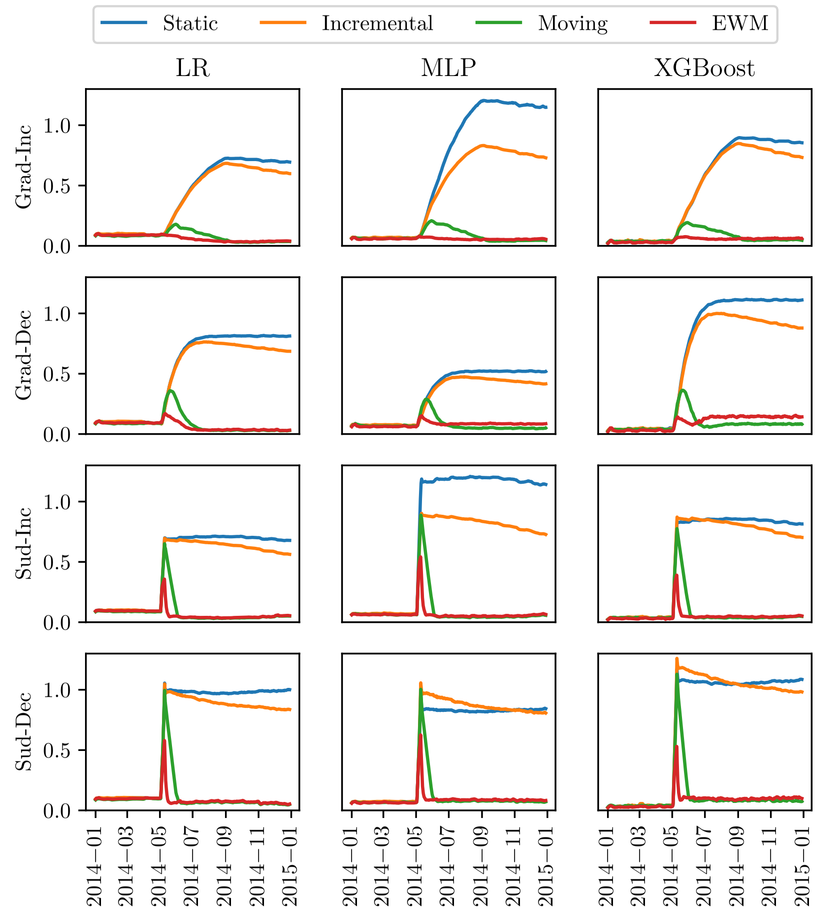
The last evaluation examines the three introduced types of computational costs – namely, the training time, the retraining time, and the sum of both (total time). These times are measured for the PDH with the moving and EWM average profile and the benchmarks.

Table 8.5 provides the computational costs. In this table, this thesis makes the following three observations: First, the models with a periodic or detection-based retraining strategy and the online learning benchmarks require more time for retraining than for the original training. An exception is the EIA that requires less retraining than training time. Second, in line with the first observation, the benchmarks with retraining have the most total time. An exception here is again the EIA and the

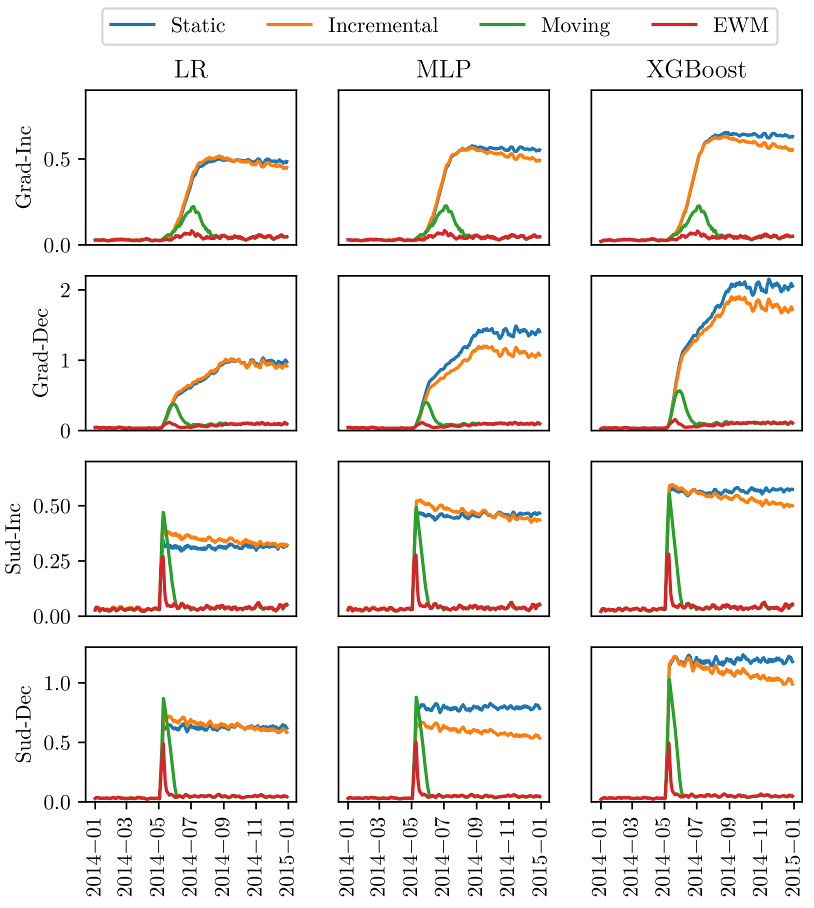


(a) Electricity **(b)** Traffic

Figure 8.3.: The rolling nMAE of the PDH with different profiles, base learners, and synthetic concept drifts for the electricity and the traffic datasets.



(a) Electricity



(b) Traffic

Figure 8.4.: The rolling nRMSE of the PDH with different profiles, base learners, and synthetic concept drifts for the electricity and the traffic datasets.

Table 8.4.: Comparison of the PDH with the benchmarks on time series with recorded concept drifts. The considered retraining strategies are presented in Section 8.2.1. As a metric, the MAE and the RMSE are considered.

(a) MAE

		MT_188	MT_316	traj_402051	traj_400367
Online	EIA	3.49	19.55	0.0164	0.0179
	OS-ELM	3.69	33.35	0.0264	0.0261
	OARNN	2.09	18.50	0.0234	0.0301
Benchmark-None	LR	3.21	20.43	0.0188	0.0222
	MLP	3.46	19.99	0.0162	0.0181
	XGBoost	5.53	37.32	0.0042	0.0038
Benchmark-Periodic	LR	1.99	16.94	0.0187	0.0223
	MLP	1.93	16.10	0.0162	0.0183
	XGBoost	1.90	16.59	**0.0041**	0.0040
Benchmark-Detection	LR	2.35	17.62	0.0188	0.0222
	MLP	2.26	20.55	0.0162	0.0181
	XGBoost	2.30	20.16	0.0042	0.0038
Method-Moving	LR	**1.70**	14.89	0.0177	0.0222
	MLP	**1.70**	**14.36**	0.0166	0.0190
	XGBoost	1.75	14.70	0.0043	0.0040
Method-EWMA	LR	1.73	14.86	0.0158	0.0227
	MLP	1.71	14.40	0.0156	0.0192
	XGBoost	1.81	14.93	0.0043	**0.0033**

(b) RMSE

		MT_188	MT_316	traj_402051	traj_400367
Online	EIA	4.67	26.90	0.0308	0.0314
	OS-ELM	5.11	43.49	0.0443	0.0420
	OARNN	3.63	26.12	0.0466	0.0455
Benchmark-None	LR	4.48	28.58	0.0387	0.0400
	MLP	4.63	27.36	0.0307	0.0314
	XGBoost	7.00	57.98	0.0075	0.0065
Benchmark-Periodic	LR	3.52	24.44	0.0387	0.0399
	MLP	3.54	22.48	0.0305	0.0315
	XGBoost	3.26	23.51	0.0073	0.0068
Benchmark-Detection	LR	3.90	25.08	0.0387	0.0400
	MLP	3.49	27.51	0.0307	0.0314
	XGBoost	3.68	28.05	0.0075	0.0065
Method-Moving	LR	3.07	21.54	0.0385	0.0384
	MLP	3.03	20.37	0.0315	0.0325
	XGBoost	3.07	20.79	**0.0069**	0.0057
Method-EWMA	LR	3.08	21.35	0.0363	0.0395
	MLP	**2.96**	**20.24**	0.0303	0.0328
	XGBoost	3.15	21.02	**0.0069**	**0.0047**

computationally very cheap LR since their refitting respectively training time is very small. Third, the training time of the time series from the electricity and the traffic datasets differ since the time series lengths differ.

8.3 Discussion

This section discusses the proposed PDH' performance, limitations, and further potential.

8.3.1 Performance

Regarding the performance, this thesis discusses three aspects: First, the results indicate that the PDH with adaptive average profiles (moving and EWM average profile) support regressors in handling concept drifts. In most cases, the PDH with these profiles needs no retraining. Consequently, the unnecessity of the retraining makes the PDH a computationally cheap framework, which is confirmed by the evaluation of the computation times. Second, despite the observation that the PDH with the moving and EWM average profile can recover from concept drift, the sliding error curves still show peaks when the concept drift occurs. These peaks are caused by the profile calculation algorithm that relies on historical data, which does not yet follow the new concept. In addition, when using EWM average profile, the peak is smaller than when using the moving average profile. The reason is that the EWM weights the more recent data higher and thus adapts faster to the profile. Last, regarding the research question

RQ4: Can profiles improve the ability of machine learning models to cope with concept drifts?

this thesis concludes that profiles can support machine learning models in handling concept drifts regardless if the MAE or RMSE is considered. However, the results also indicate that this effect depends on the time series, used regressors, and concept drift. Thus, it is a helpful framework but still needs, in some cases, retraining strategies.

Table 8.5.: The training, refitting, and total time for the PDH and the benchmarks on four time series with real concept drifts.

		MT_188			MT_316			traj_400367			traj_402051		
		Train	Refit	Total	Train	Refit	Total	Train	Refit	Total	Train	Refit	Total
Online	EIA	6.48	0.91	7.39	7.50	0.92	8.42	10.37	0.08	10.45	10.58	0.09	10.67
	OARNN	18.26	934.76	953.02	19.16	975.61	994.77	11.25	40.30	51.55	11.15	41.45	52.60
	OS-ELM	0.02	0.71	0.73	0.02	0.72	0.74	0.01	0.07	0.09	0.01	0.07	0.08
Retraining-None	XGBoost	21.44	0.00	21.44	24.58	0.00	24.58	13.88	0.00	13.88	15.51	0.00	15.51
	MLP	6.62	0.00	6.62	6.36	0.00	6.36	11.03	0.00	11.03	10.77	0.00	10.77
	LR	0.05	0.00	0.05	0.05	0.00	0.05	0.02	0.00	0.02	0.02	0.00	0.02
Retraining-Periodic	XGBoost	21.44	296.42	317.87	24.58	344.65	369.22	13.88	28.40	42.28	15.51	30.13	45.64
	MLP	6.62	129.89	136.51	6.36	107.71	114.07	11.03	20.65	31.68	10.77	21.62	32.39
	LR	0.05	0.53	0.57	0.05	0.53	0.57	0.02	0.05	0.07	0.02	0.05	0.07
Retraining-Detection	XGBoost	21.44	36.80	58.24	24.58	28.18	52.75	13.88	0.00	13.88	15.51	0.00	15.51
	MLP	6.62	4.49	11.11	6.36	6.55	12.91	11.03	0.00	11.03	10.77	0.00	10.77
	LR	0.05	0.02	0.07	0.05	0.02	0.07	0.02	0.00	0.02	0.02	0.00	0.02
Method-Moving	LR	0.05	0.00	0.05	0.04	0.00	0.04	0.02	0.00	0.02	0.02	0.00	0.02
	MLP	3.89	0.00	3.89	5.11	0.00	5.11	11.75	0.00	11.75	10.84	0.00	10.84
	XGBoost	32.35	0.00	32.35	33.29	0.00	33.29	16.40	0.00	16.40	16.43	0.00	16.43
Method-EWMA	LR	0.04	0.00	0.04	0.05	0.00	0.05	0.02	0.00	0.02	0.02	0.00	0.02
	MLP	3.47	0.00	3.47	4.53	0.00	4.53	11.70	0.00	11.70	11.19	0.00	11.19
	XGBoost	32.67	0.00	32.67	31.82	0.00	31.82	17.21	0.00	17.21	17.68	0.00	17.68

8.3.2 Limitations

This thesis highlights three aspects of the limitations of the PDH. First, as mentioned in the introduction of this chapter, the PDH relies on the assumption that the concept drifts mainly influence the time series' level as also [138] assumes for electricity load time series. However, this is a strong assumption since other concept drifts may appear that influence the time series' variance or the time series' periodicity. Thus, the PDH is a further tool in the toolbox but is not the general solution for coping with concept drifts in time series. Second, the PDH introduces new hyperparameters, namely the window size for the moving profile and the smoothing factor α for the EWM profile. Furthermore, the selection of h_{cal} – i.e., which calendar information should be extracted (e.g., day of the week and hour of the day) – is also an hyperparameter that can impact the quality of the profiles. These hyperparameters must be selected to apply the PDH.
Hereby, a wrong selection may negatively impact the performance. Suppose the window size is too small. In that case, it leads to a fast adaption capability. This might be disadvantageous for time series with a high anomaly rate since this would lead to an adaption to anomalies which is undesired and might reduce the accuracy for such time series. However, for time series with various concept drifts, a fast adaption to new concepts might be beneficial. In contrast, a larger window size leads to a slower adaption rate. While this might be beneficial for forecasting time series with a high variance, it is disadvantageous in case of concept drifts since a large window size leads to a slow adaption to the new concept. Therefore, future work may investigate automation for selecting these hyperparameters or mixing different profiles with different hyperparameters. Last, the PDH is only tested on four particular time series since the evaluation under an online learning setup is expensive. Thus, there might be concept drifts with which PDH struggles.

8.3.3 Further Potential of Profiles to Cope with Concept Drift (CD)

Regarding the further potential of the proposed profile, this thesis highlights two aspects in the following: First, as mentioned in the limitations, the performance of the PDH relies on the adaption speed to new concepts. Thus, future work might improve this proposed framework by constructing an ensemble of profiles with different adaption speeds. Such an ensemble could easily switch between profiles depending on the current data. E.g., during periods of a high anomaly rate, slower-reacting profiles might be weighted more than faster-reacting ones. However, this would increase the computational effort of the PDH. Second, this chapter only considered point forecasts. However, for many downstream applications, probabilistic forecasts are better suited. Thus, future work might extend this approach to probabilistic forecasting. A possible extension towards probabilistic forecasting could be to use the standard deviation or variance profiles, too.

9 (Probabilistic) Profile Neural Networks

Content of this chapter based on

B. Heidrich *et al.*, “ProbPNN: Enhancing Deep Probabilistic Forecasting with Statistical Information,” *arXiv preprint arXiv:2302.02597*, 2023

B. Heidrich, M. Turowski, N. Ludwig, R. Mikut, and V. Hagenmeyer, “Forecasting Energy Time Series with Profile Neural Networks,” in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, Association for Computing Machinery, 2020, pp. 220–230.

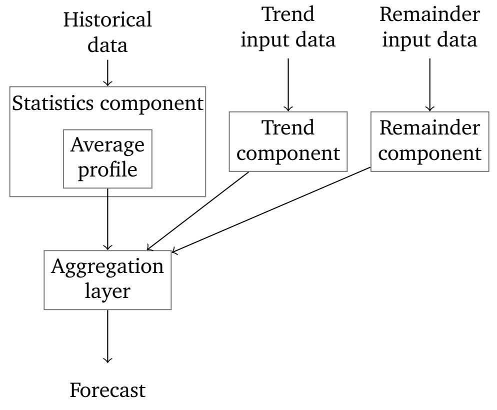
As mentioned in Chapter 3, periodicities are challenging for neural networks. In contrast, many statistical approaches can cope with periodicities but cannot extract hidden structures in the data well. Thus, this chapter provides the PNN and ProbPNN for forecasting time series with calendar-driven periodicities. These models combine the advantages of the statistical and deep learning methods by using the previously introduced profiles to enhance the forecast quality. By proposing the PNN and ProbPNN, this chapter aims to answer

RQ5 Can the usage of profiles in neural networks improve the deterministic and probabilistic forecasting performance?

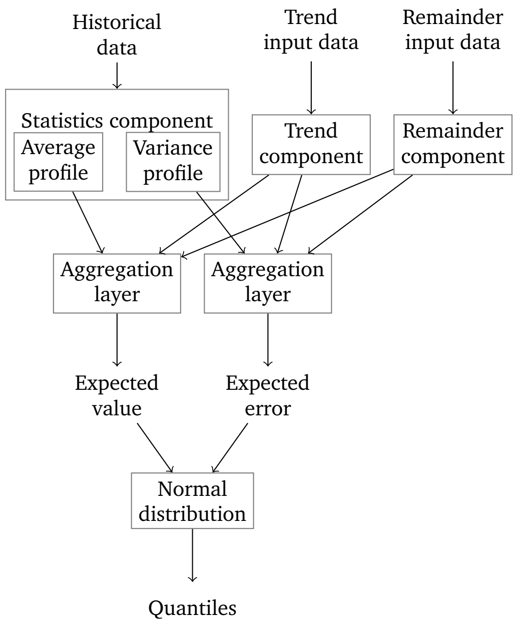
The structure of this chapter is as follows. The next section introduces the architecture of the PNN and its variants. Afterwards, the second section presents the experimental setup for evaluating the PNN and ProbPNN and the corresponding results. Finally, the last section discusses the introduced PNN and ProbPNN.

9.1 (Probabilistic) Profile Neural Networks

The general architecture of the Profile Neural Network (PNN) and its probabilistic variant (ProbPNN) is inspired by the decomposition of time series. I.e., the PNN and ProbPNN consist of three different components that model the statistics, the



(a) PNN



(b) ProbPNN. This figure is adapted and based on [50].

Figure 9.1.: The three components of PNN and ProbPNN are the statistics component, the trend component, and the remainder component. The statistics are extracted from the historical data, the trend data gets the trend input data as input, and the remainder component gets the exogenous variables and the difference between the profile and the historical data as input. The output of each component is provided to the aggregation layer as input, which combines the component's output to get the forecast.

trend, and the remainder. The PNN and ProbPNN are related to the PDH proposed in Chapter 8. The differences are the usage of the trend and that instead of a simple adding the components' output, a weighted addition is performed, whereby the weights are learned during the training. The remainder of this section first describes the PNN and afterwards the extensions to create ProbPNN, since the PNN and the ProbPNN distinguish only in the three components' output and the loss function.

9.1.1 PNN

The PNN is a hybrid neural network that combines profiles with neural networks. In the following, first, the architecture of the PNN is described, and afterwards, the training is presented.

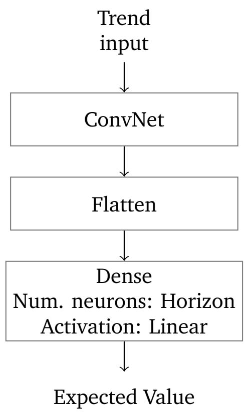
Architecture

The PNN consists of three components. An overview of the PNN architecture and the interplay of the three components is provided in Figure 9.1a. These three components are the statistics, the trend, and the remainder component. It also has an aggregation layer to combine the outputs of the three components. In the following, these four parts are described.

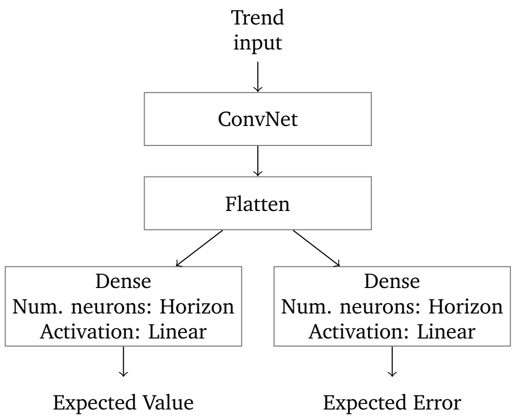
Statistics The statistics component is the first component of the PNN. This statistics component uses the average profile (Chapter 7). The average profile captures the periodicities of the time series and thus supports the neural network in coping with the periodic behaviour of the time series. The outputs of this component are the profile values for the next horizon (h) time steps. Note that for the profile calculation, only observed values are considered. I.e. for calculating p_{t+i}, only values x_j with a time stamp $j \leq t$ can be used.

Trend The second component of PNN is the trend component. For each value to be predicted, the trend component takes the last m values that match the same periodicity. For example, for hourly electrical load data with weekly periodicities, the trend component considers the values $l_{\hat{t}-s\cdot m}, l_{\hat{t}-s\cdot(m-1)}, \dots, l_{\hat{t}-s}$, where s is the length of one periodicity, and $\hat{t} \in [t+1, t+\mathrm{horizon}]$ are the timesteps of the values to be predicted. This corresponds to the input of an sAR(m,s) model in classic time series modelling. The output of the trend component is the expected value of this component. The trend component consists of a convolutional network and a dense layer. The Figure 9.2a shows the architecture of the trend component, which is determined in preliminary studies.

Remainder The remainder component first encodes the historical data and exogenous features separately. To encode the historical data, it uses the last k values of the time series as input. To encode the exogenous features, it obtains the available exogenous features as input. The encoded historical data and exogenous features are concatenated and fed into another subnetwork. This subnetwork forms the output of the remainder component, which are the expected values of the remainder for the values to be predicted. Figure 9.3a shows the architecture of the remainder component, including the historical data and the exogenous variables encoder. The used convolutional subnetwork's architecture is shown in Figure 9.4.

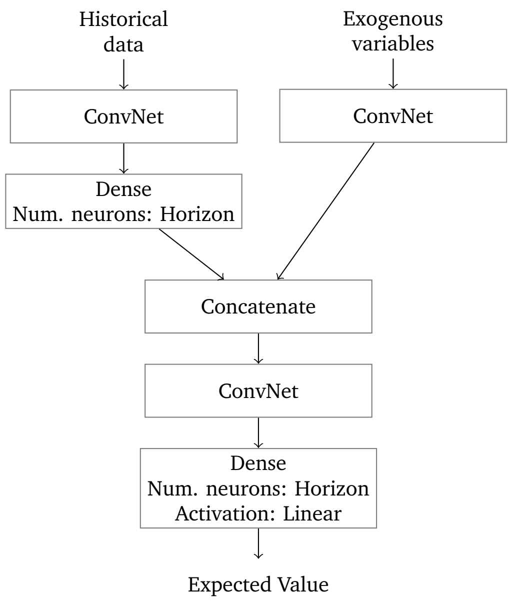


(a) PNN

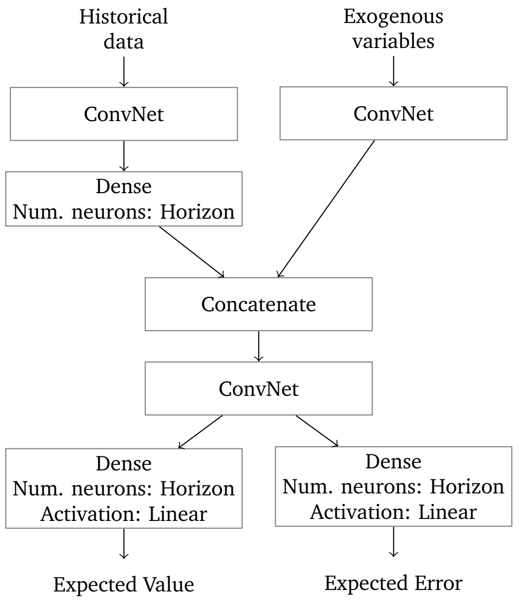


(b) ProbPNN. This figure is adapted and based on [50].

Figure 9.2.: The trend component consists of one convolutional subnetwork (ConvNet) that encodes the trend input data. The flattened output of this convolutional network is afterwards used by two concurrent dense layers to forecast the mean and the variance or standard deviation. This architecture is determined in preliminary studies. The used ConvNet is described in Figure 9.4.



(a) PNN



(b) ProbPNN. This figure is adapted and based on [50].

Figure 9.3.: The remainder component encodes the historical data and the exogenous variables using convolutional nets (ConvNet). Afterwards, the encoded historical data and the encoded exogenous variables are concatenated on a new axis. A further convolutional network (ConvNet) encodes the concatenated data. Two concurrent dense layers finally use the flattened output of this convolutional network to forecast the mean and the variance or standard deviation. This architecture is determined in preliminary studies. The used ConvNet is described in Figure 9.4.

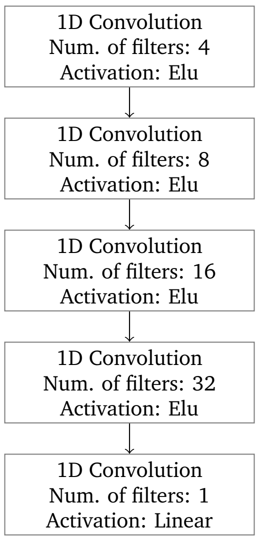


Figure 9.4.: The convolutional subnetwork (ConvNet) used in the remainder component (Figure 9.3) and the trend component (Figure 9.2) consists of five one-dimensional convolution layers with varying number of filters and either the exponential linear unit (Elu) or the linear activation function. The kernel size of each convolution is three. This figure is adapted and based on [50].

Aggregation layer The aggregation layer aggregates the three components' output introduced above (statistic, trend, and remainder component) to form the forecast $\hat{y}$. Note that the output of each component has the same shape. The aggregation layer multiplies the output of each component by a scalar weight and sums these weighted outputs together. The weights are determined in the training process of ProbPNN. Note, if the weights for each component are one the aggregation would be the same as the addition of the three components.

Training Process

The aim of the PNN is to learn as accurate as possible forecasts. Thus, the Mean Absolute Error (MAE) is used as a loss function to minimise the deviation of the forecast to the ground truth

$$L_1 = \mathrm{MAE}(\hat{y}, y), \tag{9.1}$$

where MAE is defined by Equation (2.8).

9.1.2 Probabilistic Extension (ProbPNN)

ProbPNN is based on the PNN and extends it to probabilistic forecasting (see Figure 9.1b by extending the architecture and the loss function.

Architectural Extensions

Two architectural changes extend the PNN to the ProbPNN.

First, each component needs to provide two outputs. The first output refers to the expected value, and the second output refers to the expected error of the first value. Thus, the trend and the remainder component have two concurrent last layers (see Figure 9.2b and Figure 9.3b). Correspondingly, the statistics component also comprises a variance or standard deviation profile in addition to the average profile (see Figure 9.1b). This variance or standard deviation profile refers to the expected error.
The second extension is the usage of two aggregation layers instead of one. Thereby, the first aggregation layer combines all components' outputs that refer to the expected value, while the second combines all that refer to the expected error of the forecast. The resulting expected value and errors are interpreted as normal distribution parameters, resulting in probabilistic forecasts.

Training ProbPNN

In contrast to the PNN, ProbPNN cannot be learned via standard loss functions such as the MAE. The reason is that a point forecast is provided and a forecast of the expected error. Thus, ProbPNN uses an adapted loss function consisting of two parts corresponding to the expected value and error forecast.

As for the PNN, the L_1 of the PNN is used to learn the expected value of the time series.
For learning the expected error forecast, this thesis distinguishes two cases. In the first case, the expected error is measured by an absolute error, whilst in the second case, this expected error is measured by a squared error. The first case should be applied when the estimated error corresponds to the standard deviation of the forecast, resulting in the ProbPNN-σ model. The second case must be applied when

the learned error corresponds to the variance. The corresponding model is called ProbPNN-σ^2. The following loss function reflects both cases

$$L_2 = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \mid \text{MAE}(\hat{y}_i, y_i) - \hat{e} \mid & \text{for ProbPNN-}\sigma \\ \frac{1}{n}\sum_{i=1}^{n} \mid \text{MSE}(\hat{y}_i, y_i) - \hat{e} \mid & \text{for ProbPNN-}\sigma^2, \end{cases} \tag{9.2}$$

where n is the number of samples in the batch, $\hat{y}$ is the forecast value of the time series, y is the actual value of the time series, and $\hat{e}$ is the estimated error of the forecast value $\hat{y}$. Note that the selection of the L_2 loss influences which statistical component is required: If the ProbPNN-σ should be learned, the standard deviation profile has to be used in the statistics component. If the ProbPNN-σ^2 should be learned, the variance profile must be used in the statistics component. Finally, the combination of L_1 and L_2 forms the final loss L. To equally weight both losses, they are multiplied with adaptive weights. These weights are calculated as

$$w_1 = \frac{L_2}{L_1 + L_2}, w_2 = \frac{L_1}{L_1 + L_2}. \tag{9.3}$$

In the final loss, the weight w_1 is multiplied by the loss L_1 and w_2 is multiplied by the loss L_2, resulting in the final loss function

$$L = w_1 \cdot L_1 + w_2 \cdot L_2. \tag{9.4}$$

9.2 Evaluation

This section presents the experimental setup used to evaluate the PNN and ProbPNN and the corresponding results.

9.2.1 Experimental Setup

The experimental setup comprises the implementation of the PNN and ProbPNN, benchmarks, datasets, metrics, and hardware.

PNN Models

The PNN and ProbPNN models are implemented using Keras [19] and Tensorflow [1] and trained to forecast the next 24 hours. While the ProbPNN directly provides probabilistic forecasts, this thesis equips the PNN with prediction intervals to also be

Table 9.1.: Hyperparameters used for the training of the PNN and ProbPNN.

Hyperparameter	Selected value
Optimiser	Adam [70]
Learning rate	0.001
Epochs	1000
Early stopping	Yes
Validation size	20% of training data
Shuffle train samples	Yes
Batch size	32

able to provide probabilistic forecasts. All models receive as inputs the last 36 hours of the target time series, the trend information calculated with $s = 168$ and $m = 5$[1], and exogenous variables. For the electricity dataset, the exogenous variables are the trigonometric encoded hour of the day, the month of the year, and a flag for weekends or public holidays. For the traffic dataset, the exogenous variables are the trigonometric hour of the day and a flag for the weekend. The used hyperparameters are described in Table 9.1. Note that the best working hyperparameters regarding the profiles are determined while evaluating the insights in Section 9.2.2.

Benchmarks

To assess the quality of PNN and ProbPNN's forecasts, they are compared with five benchmarks. These benchmarks comprise three state-of-the-art neural network-based forecasters, a nearest-neighbour-based forecaster, and a simple profile-based forecaster. All models except for the simple profile-based one get the same input as the PNN and ProbPNN. The three neural network architectures for time series forecasting are already presented in Chapter 3. These are the Temporal Fusion Transformer (TFT) [80], Neural Hierarchical Interpolation for Time Series Forecasting network (N-HiTS) [15], and DeepAR network (DeepAR) [123]. The TFT and N-HiTS are equipped with a quantile loss function to provide probabilistic forecasts. For the point forecast, this thesis considers the median of the probabilistic forecasts. The DeepAR directly predicts a distribution as a probabilistic forecast. Again, the median of the probabilistic forecast is considered a point forecast. For all three models, this thesis uses the implementation provided by PyTorch Forecasting[2] with the default hyperparameters.

The nearest-neighbour-based forecaster is the Nearest Neighbour Quantile Filter (NNQF) [104] also presented in the Chapter 3. As a simple regressor in the

[1]Note, the dataset has an hourly resolution. Thus, the parameters mean that for each value to be predicted, the corresponding values from the last five weeks are used.

[2]https://pytorch-forecasting.readthedocs.io/en/stable/

NNQF, this thesis used the MLP from sklearn [110] that directly predicts the quantiles previously extracted by a nearest neighbour quantile filter. The corresponding point forecasts are created by taking the median from the probabilistic forecast. The simple profile-based forecaster is the Profile Standard-Deviation Forecast (PSF). This baseline calculates the average profile (Equation (7.5)) with a window size of 28 days. This profile forms the point forecast. In the probabilistic forecasting case, this method also calculates the standard deviation profile (Equation (7.6)) with a window size of 28 days. Together with the average profile, the standard deviation profile is interpreted as a parameter of a Gaussian distribution, leading to a probabilistic forecast.

Datasets

The evaluation of the PNN and ProbPNN uses the electricity and the traffic dataset. The models are trained and evaluated on each time series separately. The corresponding results are aggregated by calculating the average normalised scores and ranks.

Metric

This thesis uses the nMAE and nRMSE for the point forecasts and the nCRPS and nPL for the probabilistic forecasts. Furthermore, this thesis also considers the coverage rate of the quantiles to determine the quality of the probabilistic forecasts using the DICR as well as a visualisation of the quantile ranges. As a metric for the computational effort, the training time of the forecasters is measured.

9.2.2 Results

The results are separated into four parts. The first part gains insights into the PNN and ProbPNN. The second part performs an ablation study to influence the different components' impact on the forecast quality. The third part compares the proposed methods with the benchmarks. Finally, the last part assesses the computational effort.

Insights

This evaluation performs two insights-related analyses. First, the influence of different profiles on the PNN's and ProbPNN's results are compared. Second, the influence of the hyperparameters of the profile calculation method is examined. [3]

Analysing Different Profile Calculation Methods For assessing the profile calculation method's impact on the forecast quality, this thesis equips the PNN and ProbPNN with each method calculation method for the average and variance profiles. When evaluating the calculation average profile method, the ProbPNN's average variance profile calculation method is fixed to the moving variance profile. When assessing the variance profiles calculation methods, the ProbPNNs' average profile calculation method is fixed to the moving average profile. Note, the PNN do not use the variance profiles.

Table 9.2 shows the corresponding point and probabilistic forecast results. This thesis makes three observations regarding the average profile calculation method and two observations regarding the variance profile calculation method. The first observation regarding the average profile calculation method is that the moving average profile always leads to the best results, followed by the EWMA profile, which is the second-best calculation method in most cases. Second, the static and incremental average profile performs worse than the moving average or EWMA profile. Third, the order of the profile computation methods is not influenced by the considered metrics. The increasing order regarding the error is moving, EWM, incremental and static profile calculation method. Regarding the variance profile calculation method, the differences between the calculation methods are smaller than for the average profile calculation methods. However, the moving calculation method is also the best for the variance calculation. Second. the order of the profile calculation methods seems again not to be influenced by the metrics and is the same as for the average profiles.

Analysing Profile's Calculation Methods' Hyperparameters Based on the previous evaluation, this thesis examines the hyperparameters of the best working average and variance profile calculation method (moving average and moving variance profile). Both methods rely on the window size as a hyperparameter. As window

[3]To reduce the computational effort, these two evaluations are performed only on the electricity dataset.

Table 9.2.: Impact of four profile calculation methods the forecast quality of the PNN, ProbPNN-σ, and ProbPNN-σ^2 on the electricity dataset.

(a) Impact of the average profile calculation method on the PNN and ProbPNN. For the ProbPNN, the variance profile is fixed using the moving variance profile.

	PNN				ProbPNN-σ				ProbPNN-σ^2			
	Static	Incremental	Moving	EWMA	Static	Incremental	Moving	EWMA	Static	Incremental	Moving	EWMA
nMAE	0.180	0.190	**0.172**	0.176	0.191	0.186	**0.173**	0.177	0.192	0.191	**0.174**	0.181
nRMSE	0.264	0.280	**0.256**	0.266	0.271	0.272	**0.254**	0.265	0.270	0.277	**0.253**	0.267
nCRPS	0.174	0.182	**0.170**	0.177	0.146	0.140	**0.131**	0.135	0.149	0.146	**0.132**	0.139
nPL	0.088	0.092	**0.086**	0.090	0.073	0.071	**0.066**	0.068	0.075	0.074	**0.067**	0.070

(b) Impact of the variance profile calculation method on the ProbPNN. The average profile is fixed using the moving average profile.

	ProbPNN-σ				ProbPNN-σ^2			
	Static	Incremental	Moving	EWMV	Static	Incremental	Moving	EWMV
nMAE	0.176	0.176	**0.173**	0.176	0.179	0.178	**0.174**	0.179
nRMSE	0.263	0.262	**0.254**	0.263	0.264	0.264	**0.253**	0.264
nCRPS	0.134	0.136	**0.131**	0.133	0.137	0.139	**0.132**	0.136
nPL	0.067	0.068	**0.066**	0.067	0.069	0.070	**0.067**	0.069

lengths, 7, 28, and 90 days are tested. Note to save runtime, the same window length for both profiles is selected.

Table 9.3 shows the results. The best window length for the considered dataset is 28 days for all metrics, followed by a 90-day window size. The worst tested hyperparameter is seven days. However, the differences between the best and worst window sizes are small (mostly around 5%).

Ablation Study

The ablation study assesses the influence of the different components on the forecast. I.e., the forecasting performance of the PNN, ProbPNN-σ and, ProbPNN-σ^2 are compared with variants of these networks, where specific components or subnets of the networks shown in Figure 9.1 are truncated and the corresponding inputs are omitted. These variants are the three networks without profiles (wo profile), without the trend component and input (wo trend), without the historical data encoder and corresponding input (wo hist), without exogenous features encoder and corresponding input (wo exog), and for the ProbPNN also variants without the variance (wo variance). In addition, this thesis examines if PNN, ProbPNN-σ

Table 9.3.: Impact of the windows size of the moving and the smoothing parameter of the EWM profile calculation method on the forecast quality of the PNN, ProbPNN-σ, and ProbPNN-σ^2 on the electricity dataset.

	PNN			ProbPNN-σ			ProbPNN		
	90	28	7	90	28	7	90	28	7
MAE	0.175	**0.172**	0.181	0.179	**0.173**	0.183	0.181	**0.174**	0.186
RMSE	0.264	**0.256**	0.272	0.265	**0.254**	0.272	0.266	**0.253**	0.274
nCRPS	0.174	**0.170**	0.180	0.136	**0.131**	0.138	0.140	**0.132**	0.142
nPinball	0.088	**0.086**	0.091	0.068	**0.066**	0.070	0.070	**0.067**	0.072

or, ProbPNN-σ^2 is significantly better than each variant using the one-sided paired Wilcoxon test [144] and reporting the corresponding p-value.
Table 9.4 shows the result of this ablation study, revealing the following three observations: First, this thesis identifies the most important component by examining which truncation led to the highest error. For the electricity dataset, the average profile is always the most important component and the historical data for the traffic dataset. Second, in general, all components influence the results positively regarding the scores for both datasets with exceptions for the ProbPNN-σ^2 in some cases on the traffic dataset. Regarding the ranks on the electricity dataset, all components contribute positively except for the trend component of both ProbPNNs. However, on the traffic dataset, the trend component improves the results for all models. Instead, the variance deteriorates the rank of the ProbPNN-σ^2. Consistently, the Wilcoxon test's p-values indicate that only the trend component for both ProbPNNs in the electricity dataset and the variance component for the ProbPNN-σ^2 in the traffic dataset did not lead to significant improvement. Last, the ablation study shows no qualitative differences in the results across the four metrics.

Benchmarking

The benchmarking compares the point and probabilistic forecasts of the proposed methods to the forecasts from the benchmarks. Furthermore, the last part of the benchmarking also examines the computational effort of the proposed method and the benchmarks.

Point Forecasting This thesis examines the average score and rank of point forecasts using the nMAE and nRMSE. Table 9.5 shows the results. In the following, this thesis makes two observations: First, for the electricity dataset, the best models are the PNN and both ProbPNNs, followed by the TFT as the best benchmark model regarding the average score. For the traffic data, the PNN is the best model regarding the score, followed by both ProbPNNs and the TFT. PNN's nMAE is 9.9% better

Table 9.4.: The ablation study of the PNN with prediction intervals, ProbPNN-σ, and ProbPNN-σ^2. For the original variants and the variants where one component is truncated, this table reports the corresponding average scores and average ranks of the nMAE, nRMSE, nCRPS, and nPL. Furthermore, this table reports the p-value of the one-sided Wilcoxon test that tests if the original variants improve the truncated ones.

			PNN PI						ProbPNN-σ						ProbPNN-σ^2				
				wo profile	wo variance	wo trend	wo hist	wo exog		wo profile	wo variance	wo trend	wo hist	wo exog		wo profile	wo trend	wo hist	wo exog
Electricity	nMAE	score	0.172	0.374	0.175	0.182	0.175	0.173	0.386	0.174	0.173	0.213	0.174	0.174	0.384	0.175	0.175	0.215	0.175
		rank	1.654	5.000	2.262	3.908	2.177	2.277	5.992	2.698	2.310	4.992	2.721	2.462	5.992	2.698	2.217	4.992	2.628
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.006	0.667	0.000	0.002		0.000	0.004	0.966	0.000	0.035
	nRMSE	score	0.256	0.484	0.264	0.277	0.266	0.254	0.485	0.255	0.253	0.306	0.257	0.253	0.484	0.254	0.252	0.307	0.257
		rank	1.292	5.000	2.154	3.954	2.600	2.346	5.984	2.558	1.899	5.016	3.186	2.385	5.984	2.589	1.822	5.000	3.209
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.091	0.999	0.000	0.000		0.000	0.057	1.000	0.000	0.000
	nCRPS	score	0.17	0.293	0.175	0.184	0.176	0.131	0.293	0.132	0.131	0.160	0.132	0.132	0.296	0.133	0.132	0.162	0.133
		rank	1.208	5.000	2.162	3.977	2.654	2.208	5.984	2.698	2.186	5.000	2.915	2.377	5.992	2.760	2.147	4.992	2.721
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.001	0.840	0.000	0.000		0.000	0.004	0.999	0.000	0.008
	nPL	score	0.086	0.148	0.089	0.093	0.089	0.066	0.147	0.066	0.066	0.080	0.067	0.067	0.149	0.067	0.067	0.082	0.067
		rank	1.208	5.000	2.162	3.977	2.654	2.208	5.984	2.721	2.171	5.000	2.907	2.354	5.992	2.791	2.140	4.992	2.721
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.001	0.846	0.000	0.000		0.000	0.003	0.999	0.000	0.008
Traffic	nMAE	score	0.128	0.140	0.143	0.180	0.152	0.154	0.169	0.157	0.170	0.188	0.171	0.169	0.178	0.166	0.187	0.202	0.184
		rank	1.786	2.578	2.545	4.683	3.407	2.198	3.280	2.399	3.702	5.330	4.090	2.471	3.412	2.222	3.651	5.144	4.099
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.009	0.000	0.000	0.000		0.000	1.000	0.000	0.000	0.000
	nRMSE	score	0.313	0.324	0.335	0.407	0.354	0.333	0.348	0.341	0.368	0.386	0.357	0.329	0.329	0.324	0.379	0.393	0.349
		rank	1.937	2.469	2.523	4.661	3.409	2.361	2.909	2.700	3.971	5.108	3.951	2.556	3.007	2.299	4.170	5.181	3.787
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000		0.000	1.000	0.000	0.000	0.000
	nCRPS	score	0.243	0.256	0.274	0.325	0.272	0.117	0.126	0.121	0.130	0.143	0.130	0.132	0.137	0.132	0.148	0.159	0.143
		rank	1.674	2.389	3.028	4.823	3.085	2.19	3.139	2.591	3.717	5.279	4.084	2.416	3.351	2.483	3.710	5.107	3.933
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000		0.000	0.100	0.000	0.000	0.000
	nPL	score	0.123	0.130	0.139	0.165	0.138	0.059	0.064	0.061	0.066	0.072	0.065	0.066	0.069	0.066	0.074	0.080	0.072
		rank	1.671	2.388	3.033	4.822	3.085	2.187	3.140	2.586	3.723	5.278	4.086	2.42	3.352	2.474	3.712	5.109	3.934
		wilcoxon		0.000	0.000	0.000	0.000		0.000	0.000	0.000	0.000	0.000		0.000	0.118	0.000	0.000	0.000

Table 9.5.: The average scores and ranks regarding nMAE and the average nRMSE of the PNN, ProbPNN-σ, ProbPNN-σ^2, PSF, TFT, N-HiTS, NNQF with an MLP as a regressor, and DeepAR for both data sets.

(a) nMAE

		PNN	ProbPNN-σ	ProbPNN-σ^2	PSF	TFT	N-HiTS	NNQF	DeepAR
Electricity	Score	0.172	0.173	0.174	0.219	0.189	0.253	0.235	0.211
	Rank	1.62	1.91	2.54	6.07	3.99	7.58	6.88	5.42
Traffic	Score	0.128	0.154	0.169	0.288	0.164	0.226	0.241	0.257
	Rank	1.37	2.57	3.53	7.58	2.93	5.51	6.07	6.45

(b) nRMSE

		PNN	ProbPNN-σ	ProbPNN-σ^2	PSF	TFT	N-HiTS	NNQF	DeepAR
Electricity	Score	0.256	0.254	0.253	0.318	0.276	0.348	0.323	0.300
	Rank	2.48	1.77	1.83	6.47	4.03	7.52	6.57	5.33
Traffic	Score	0.314	0.333	0.329	0.536	0.352	0.404	0.485	0.471
	Rank	2.20	2.66	2.75	7.65	3.15	4.87	6.54	6.18

than of the TFT on the electricity and 28.1 % on the traffic dataset. Regarding the nRMSE, PNN's nMAE is 7.8% better than the TFT on the electricity and 12., % on the traffic dataset. Regarding the ranks, the results are similar. PNN achieves the best rank for the MAE on the electricity dataset (1.62) and for the MAE (1.37) and nRMSE (2.2) on the traffic dataset. An exception is the nRMSE on the electricity dataset where ProbPNN-σ achieves the best average rank with 1.77.

Probabilistic Forecasting This thesis calculates the average ranks and scores regarding the nCRPS and nPL of all considered time series from the Electricity and Traffic data sets to compare the probabilistic forecasts. In addition to nCRPS and nPL, this thesis also examines the quantile coverage of the forecasts using the DICR and a visualisation. Finally, an exemplary probabilistic forecast is visualised.

Regarding the nCRPS and nPL, Table 9.6 reports the scores and ranks for all evaluated methods and data sets. In the results, this thesis makes three observations. First, the ProbPNN-σ outperforms the benchmarks concerning both scores. ProbPNN-σ's nCRPS is 6.1% better than of the TFT on the electricity and 6.8 % on the traffic dataset. Regarding the nPL, ProbPNN-σ is 6.1% better on the electricity and 6.8% better on the traffic dataset than the TFT. Second, regarding the average rank, the ProbPNN-σ outperforms the benchmarks again. Regarding the nCRPS, ProbPNN-σ achieves 1.44 as the average rank on the electricity and 1.74 on the traffic data. Regarding the nPL, ProbPNN-σ's average rank on the electricity data is 1.39 and 1.73 on the traffic data. Third, when comparing ProbPNN-σ and ProbPNN-σ^2, it is observable that using the standard deviation profile seems beneficial compared to

Table 9.6.: The average scores and ranks regarding nCRPS and nPL of the PNN with prediction intervals, ProbPNN-σ, ProbPNN-σ^2, PSF, TFT, N-HiTS, NNQF with an MLP as a regressor, and DeepAR for both data sets.

(a) Average nCRPS scores and ranks of all considered time series from the Electricity respectively Traffic data set.

		PNN-PI	PPNN-σ	PPNN-σ^2	PSF	TFT	N-HiTS	NNQF	DeepAR
Electricity	Score	0.170	0.131	0.132	0.160	0.139	0.184	0.172	0.155
	Rank	6.33	1.44	1.90	5.28	2.76	7.37	6.44	4.48
Traffic	Score	0.243	0.117	0.132	0.245	0.125	0.167	0.181	0.191
	Rank	6.85	1.74	2.73	7.47	2.07	4.38	5.28	5.48

(b) Average nPLs scores and ranks of all considered time series from the Electricity respectively Traffic data set.

		PNN-PI	PPNN-σ	PPNN-σ^2	PSF	TFT	N-HiTS	NNQF	DeepAR
Electricity	Score	0.086	0.066	0.067	0.081	0.070	0.093	0.087	0.078
	Rank	6.34	1.39	1.86	5.27	2.84	7.41	6.42	4.48
Traffic	Score	0.123	0.059	0.066	0.124	0.063	0.085	0.092	0.096
	Rank	6.85	1.73	2.71	7.48	2.07	4.43	5.29	5.45

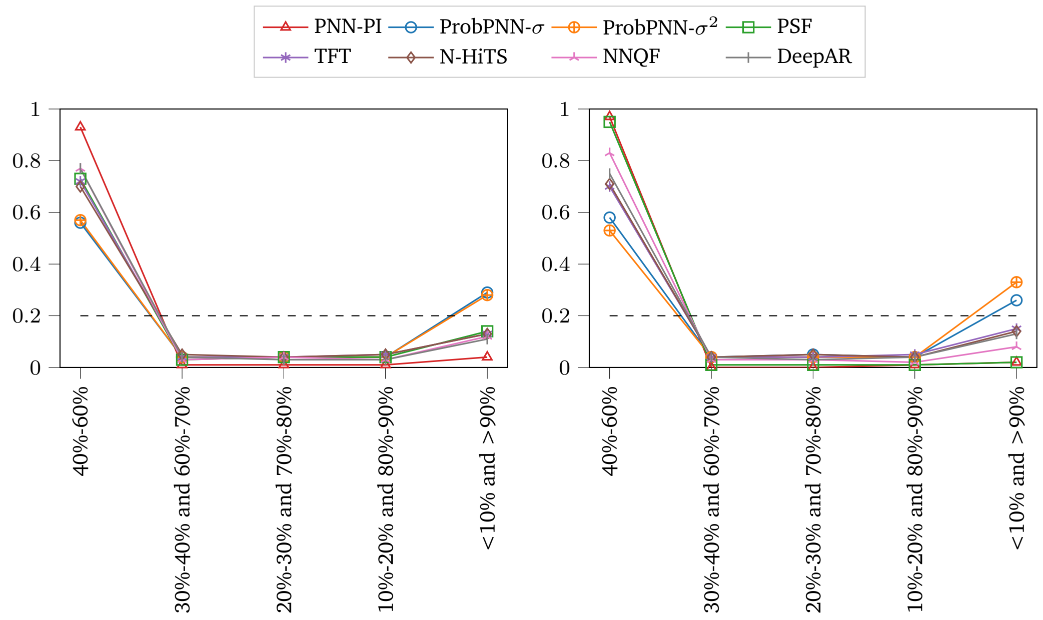
Table 9.7.: The average DICR of the PNN, ProbPNN-σ and ProbPNN-σ^2 as well as the simple PSF, the TFT, the N-HiTS, the NNQF with an MLP as a regressor, and DeepAR for both data sets.

	PNN-PI	PPNN-σ	PPNN-σ^2	PSF	TFT	N-HiTS	NNQF	DeepAR
Electricity	3.98	1.77	1.81	2.63	2.61	2.52	2.91	2.93
Traffic	4.26	1.83	1.71	4.19	2.47	2.55	3.35	2.80

the variance profile regarding all metric's scores and ranks on both data sets.

Regarding the Distance to Ideal Coverage Rate (DICR), Table 9.7 provides the results for both datasets and all benchmarks. In this table, the ProbPNN has the lowest DICR and the TFT the second lowest, whereas the PNN with Prediction Intervals has the highest DICR. However, in general, the DICR is high for all methods and also for the ProbPNN and TFT.

Examining the quantile ranges in more detail, the results in Figure 9.5 align with these results. Figure 9.5 provides the coverage of different quantile ranges that each corresponds to 20%. In particular, the quantile ranges are 40%-60%, 30%-40% and 60%-70%, 20%-30% and 70%-80%, 10%-20% and 80%-90%, and smaller 10% and greater 90%. Thus, each quantile range of an ideal forecasted distribution would cover 20% of the actual values. Regarding the coverage of the quantile ranges, this thesis makes two observations: First, all PNN, ProbPNNs, and benchmarks have a too wide quantile range, which also leads to high DICR scores. More specifically,



(a) Electricity data set. **(b)** Traffic data set.

Figure 9.5.: The coverage of different quantile ranges of the PNN with prediction intervals, ProbPNN-σ and ProbPNN-σ^2 as well as the simple PSF, the TFT, the N-HiTS, the NNQF with an MLP as a regressor, and DeepAR for both data sets. The black dashed lines indicate the optimal coverage.

the predicted quantile range around the median (between 40 and 60 percentile) covers more than 50% of the actual values. In contrast, the ranges between 10th and 40th percentile and between the 60th and 90th percentile are too small (they cover less than 20% of the actual values). The tails of the distributions cover more actual values again. Second, the ProbPNN and the benchmarks behave similarly. However, the ProbPNNs estimate the median slightly better but overestimate the distribution's tails.

These observations regarding the quantiles are confirmed by the exemplary forecasts in Figure 9.6, where the band between 40% and 60% seems to cover more than 20% of the values and are thus too wide, especially for the PNN with prediction intervals.

Computational Effort

This thesis measures the training time required by ProbPNN and the benchmarks to assess the computational effort.
Due to the various experiments and the allocation of the server, the computation

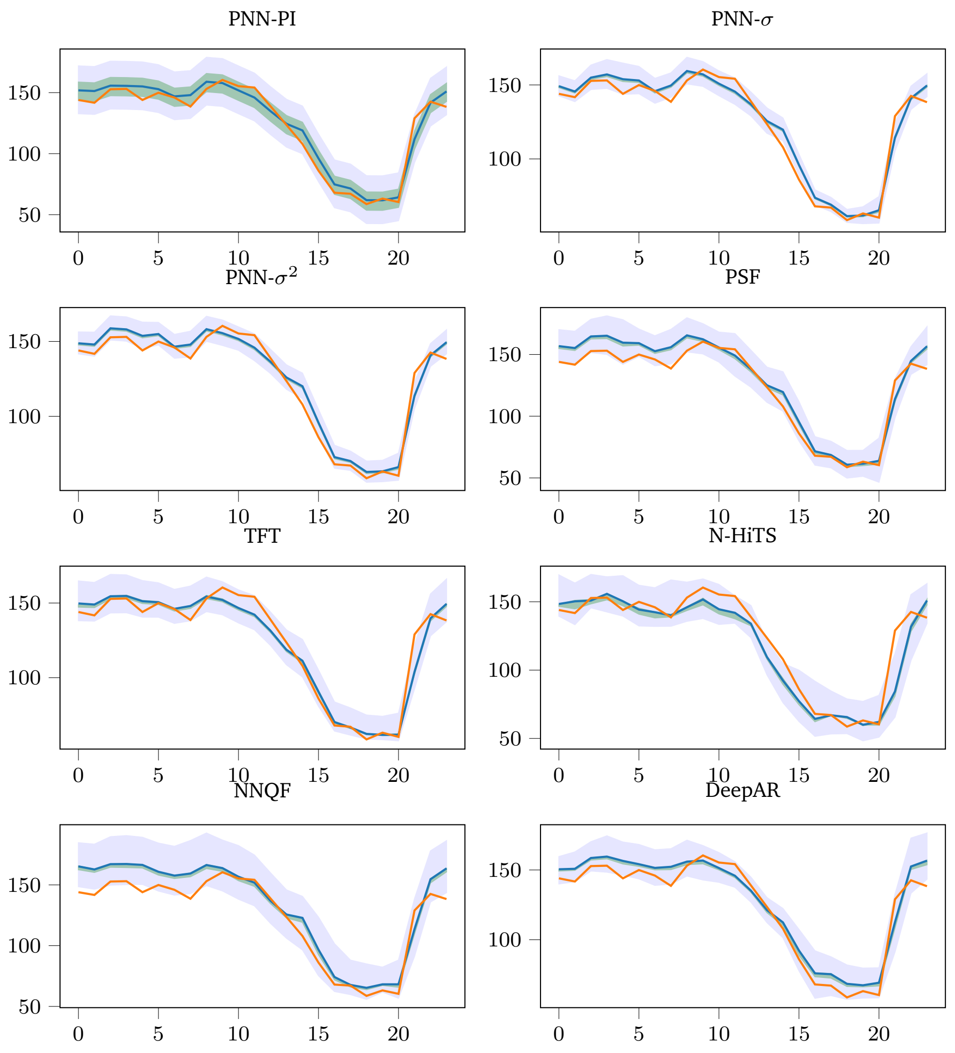


Figure 9.6.: Ground truth (orange), forecast (dark blue), the 90% (light blue) and 20% (green) confidence intervals of the PNN with prediction intervals, ProbPNN-σ and ProbPNN-σ^2 as well as the simple PSF, the TFT, the N-HiTS, the NNQF with an MLP as a regressor, and DeepAR for an exemplary day of the electricity data set.

Table 9.8.: Average training time in seconds of the PNN, ProbPNN-σ and ProbPNN-σ^2 as well as the TFT, the N-HiTS, the NNQF with an MLP as a regressor, and DeepAR for both data sets. The simple PSF is omitted as it does not need to be trained.

	PNN-PI	PPNN-σ	PPNN-σ^2	TFT	NHiTS	NNQF	DeepAR
Electricity	150	197	164	2995	549	79	2221
Traffic	677	717	451	4762	914	50	2510

time for the electricity dataset is measured on setup 1 and for the traffic dataset on setup 2 (Table 2.2).

Table 9.8 presents the forecasters' training time in seconds on both data sets. In the following, this thesis makes two observations. First, PNN and ProbPNN need less training time than the state-of-the-art deep learning benchmarks. However, they require more training time than the NNQF, which training time strongly depends on the used base learner. Second, when comparing the ProbPNN-σ and ProbPNN-σ^2, ProbPNN-σ has a higher training time than ProbPNN-σ^2.

9.3 Discussion

This section discusses the performance of the PNN and ProbPNN, their limitations, and further potential.

9.3.1 Performance

Regarding the performance, this thesis highlights and discusses four aspects.

First, the results indicate that the PNN and ProbPNN provide better forecasts than the other models regarding the considered metrics. This leads to the conclusion that using profiles and decomposition-inspired architectures provides benefits. However, the profiles also introduce new hyperparameters that might influence the results. Furthermore, the best hyperparameters might differ from dataset to dataset and time series to time series. The impact of the hyperparameters on different datasets needs to be addressed by further experiments since this thesis examines the hyperparameter selection only on one dataset. Second, the ablation study's results show that the most important component is the historical information for the traffic data set and the profile component for the electricity data set. Furthermore, the ablation study also shows that each component except for the trend component on the electricity dataset and the variance on the traffic dataset significantly improves the results. This thesis concludes from these results that each component is potentially beneficial. However, considering different combinations as a step of a hyperparameter search might lead to further improvements. Third, when analysing the quantiles, the DICR and the quantile coverage visualisation show that the quantiles are not well calibrated. Thus, adapted loss functions or additional post-processing strategies might improve the forecasted quantiles. Such an improvement could be addressed by future work since post-processing strategies are widely applied, e.g. in wind power forecasting [114]. Finally, the last aspect covers the training time. The shorter training time of the PNN and ProbPNN compared to the benchmarks might be caused by the smaller number of parameters of the PNN and ProbPNN. However, when training the models globally, the required training time becomes less important. Thus, the PNN and ProbPNN should be extended to global forecasting models. Furthermore, the usage of GPUs could improve the training times.

Finally, to answer the research question

RQ5: Can the usage of profiles in neural networks improve the deterministic and probabilistic forecasting performance?

This thesis concludes that profiles improve the deterministic and probabilistic forecasting performance for the considered class of time series, as shown in the ablation study.

9.3.2 Limitations

Regarding the limitations of the proposed method, this thesis highlights four aspects.

First, the PNN and ProbPNN exploit regular periodicities in time series to improve point and probabilistic forecasts. Thus, this method is only suitable for time series containing calendar-driven periodicities. Furthermore, the metadata of the periodicities, such as the frequency, must be known in advance for being able to calculate the profiles. Wrong assumptions about the periodicity and the calendar information (e.g., weekends and public holidays) used to build the profiles might harm the accuracy severely. E.g., if a 24-hour periodicity is assumed but the time series has a 23-hour periodicity.
Second, the proposed method assumes that the components (periodicities, trend, remainder) are independent. Future work might relax this assumption by introducing copulas.
Third, the proposed method is a local model. I.e., it is trained on each time series

separately, resulting in bad scalability. Thus, future work has to extend this model to a global forecaster.
Last, the ProbPNN assumes that the time series is normally distributed. This assumption does not seem to influence the results negatively on the tested datasets since the ProbPNN outperforms the benchmarks. However, this must be considered when applying the ProbPNN to a forecasting task.

9.3.3 Further Potential of (Prob)PNN

Finally, the discussion's last part focuses on the potential of the (Prob)PNN. This thesis highlights two aspects. First, future work could try to relax the assumption that the periodicity, trend, and remainder components are independent. A possibility to relax this assumption might be the introduction of copulas.
The second aspect focuses on the scalability. As mentioned before, the PNN and ProbPNN are local models. Local models need to be fitted on each time series separately. A possibility to extend the PNN and ProbPNN to a global model might be the usage of ensembles of average and variance profiles together with an attention module determining which profiles should be used depending on the current data.

Part IV

Discussion and Conclusion

Discussion 10

The present thesis focuses on challenges in forecasting time series with calendar-driven periodicities. In particular, it proposes methods that face four selected challenges - the missing scenario, small training data size, concept drift, and periodicity challenge. The proposed methods can be separated into two approaches. The generative models used for facing the missing scenario and small training data size challenge and the profile-based methods for the concept drift and periodicity challenge. Both approaches are not purely data-driven since they use additional knowledge, contexts, or analyses. To solve the missing scenario and perform mid-term peak forecasts, the generative models use statistical information to control the time series. To handle the small training data size challenge, the proposed Latent Space-based Forecast Enhancer (LSFE) analyses the latent space of a globally trained generative model to localise the target time series and use this additional information. To handle concept drifts affecting time-series levels and support neural networks with periodicities, the Profile-based Framework for Concept Drift Handling (PDH), the Profile Neural Network (PNN), and Probabilistic Profile Neural Network (ProbPNN) combine profiles as additional knowledge with machine learning models. Besides considering additional knowledge by all proposed methods, all chapters that correspond to the proposed methods discuss scalability as a key challenge. Scalability is essential when applying complex models to domains with millions of time series, such as the electrical grid, traffic planning, or retail. Thus, this chapter discusses the consideration of additional knowledge and scalability by focusing on global forecasting. Furthermore, the last section of this discussion focuses on the challenges that might arise when integrating additional knowledge with global forecasting.

10.1 Consideration of Context or Additional Knowledge

As mentioned, all proposed methods consider context or additional knowledge when being applied. Thereby, the consideration ranges from taking this information as additional input as in Chapter 4, gaining it via additional analysis as in Chapter 6 to propose hybrid methods as in Chapter 5, Chapter 8, and Chapter 9. However, context could also foster the explainability of machine learning methods or increase the value of forecasts for a specific downstream application. Thus, this section discusses these three forms as examples of the importance of considering additional knowledge in the following.

10.1.1 Hybrid Models

Hybrid models integrate ideas of classical time series model-based and deep learning-based forecasting. This thesis also proposes hybrid models in Chapter 5, Chapter 8, and Chapter 9. Thereby, this thesis observes two benefits and one drawback that are discussed in the following. The first advantage is that using hybrid models leads to improved forecasts, as observed in the results from the BigDEAL challenge, but also when analysing if profiles provide benefits to the PDH, PNN, or ProbPNN. This observation of improvements in the forecasts is in line with results from the M4 competition [92], which was won by a hybrid method (ES-RNN [124]). The second benefit this thesis discusses is that hybrid neural networks may allow smaller neural networks. In a hybrid neural network, a part of the time series forecasting or modelling is done by classical time series models, which also introduces some prior knowledge (context). These parts that are already modelled need not be learned by the neural network. Thus, it can focus on other parts or it can be smaller and require less training time (compare the training time for the PNN and ProbPNN). Finally, as a drawback, this thesis states that hybrid models may require more knowledge than just the time series. E.g., using the PNN requires knowledge about the periodicity of the time series or using the generative model-based forecaster used in the BigDEAL challenge (Chapter 5) requires the definition of a function that can model the rolling mean of the time series. Thus, not having such knowledge or information might restrict the applicability of hybrid models. Therefore, their application might be challenging under certain circumstances.

10.1.2 Explainability

Context and additional information can also foster the explainability of machine learning models, which is important, for example, to improve users' trust in the forecast. Although this thesis does not cover explainability directly, the proposed methods provide access points for it. E.g., the decomposition of the PNN or ProbPNN can provide insights into the forecast; such a decomposition is provided in Heidrich *et al.* [51]. Additionally, when extending the PNN and ProbPNN to a global model by using ensembles of average and variance profiles together with an attention module determining which profiles should be used depending on the current data, also the attention scores could be used to foster explainability as done for transformers [55]. Furthermore, examining the latent space (Chapter 6) might also foster explainability. This thesis only localises time series in the latent space. However, further meaningful features and directions might also be identifiable as undertaken in computer vision [137]. The decomposition and the analysis of the latent space could lead to better explainable forecasters in future work.

10.1.3 Forecast Value

Context and additional knowledge is also essential for the forecast value. Despite this thesis measuring only the forecast quality using metrics such as MAE or CRPS, the forecast value is more important in many domains. The reason is that forecasts are used in downstream applications, whose performance determines the forecast value. It is important to consider that the forecast value does not directly correspond to the forecast quality measured by metrics such as the MAE [141]. Thus, this thesis states that the proposed methods and the benchmarks should be examined regarding different forecast values in future work. Furthermore, applying the AutoML approach proposed in [115] to the generative model-based forecaster presented in Chapter 5 might be interesting. In particular, the AutoML approach of tuning sampling parameters could optimise the forecast quality by identifying the best for the downstream application.

10.2 Scalability and Global Forecasting

As mentioned, each chapter in Part II and Part III discusses the scalability of the proposed methods and states the importance of scalability in time series forecasting. However, no proposed method scales well if multiple time series are considered, except for the LSFE presented in Chapter 6. Nevertheless, each of the mentioned chapters sketches possibilities for improving the scalability of the proposed methods and training them as global forecasters. E.g. the generative models in Chapter 4 and Chapter 5 could use the idea proposed in Chapter 6 of combining a globally trained generative model with an analysis of the latent space. Furthermore, the PNN and ProbPNN could be trained globally while considering different profiles, whose importance might be weighted using an attention mechanism. Future work might address these ideas.

Besides improving the scalability, global forecasters might also solve the mentioned challenges in time series forecasting. E.g., global forecasters could solve the small training data size challenge if trained on a sufficiently diverse dataset as the results in Hertel *et al.* [54] indicate. In this case, it is possible to assume that the forecaster can forecast unknown time series that are not too different from the time series in the training dataset. Moreover, a diverse and large enough training dataset could also solve or weaken the concept drift challenge. In this case, it is possible to assume that the model has seen different concepts during training and, thus, should be more stable during inference. Finally, a larger training dataset enables larger neural networks that can learn complex patterns. Thus, guiding the network with additional knowledge, such as profiles, to better predict periodicities becomes less important.

Another scalability aspect is the technical realisation of forecasters and its embedding in software infrastructure. This also includes the following aspects of MLOps: continuous integration and deployment of such methods, workflow orchestration, reproducibility, versioning, collaboration, continuous ML training and evaluation, ML metadata tracking and logging, continuous monitoring, and feedback loops [74]. This aspect of technical realisation and MLOps is highly relevant for various domains. E.g. for the smart grid, the EnergyLab at the KIT implements a fully automated energy system [143]. In such an environment, automated forecasts and MLOps play a crucial role.

Concluding, global forecasters might be a game changer in time series forecasting. Their good performance and scalability enable forecasting-as-a-service as the TimeGPT presented by Nixtla[1].

10.3 Combine Context Information and Global Forecasting

Finally, the last section of this chapter investigates the possibility of considering context and global forecasting to exploit both's advantages. However, the combination raises new challenges that might be addressed in future work.

Considering context might be beneficial in many problems, e.g. by using hybrid models. For a global forecaster, integration of the context is difficult. For example, the PNN and ProbPNN require information on the periodicity used by the profile

[1]`https://www.nixtla.io/#tgpt` (last access 21.08.2023)

calculation methods. Thus, the requirement of periodicity information might limit its potential to apply the model to time series for which this information is unavailable. Note that this is not always a limitation since hybrid methods such as N-HiTS [15] or DeepAR [123] can be trained globally. Second, different time series are influenced by different exogenous features. E.g. for energy demand time series forecasting, the relevant exogenous features differ from those for retail time series forecasting. Forecasting might even be impossible for some time series without certain exogenous features, e.g., marketing-driven product time series. However, globally trained models cannot consider all information. Thus, this might limit their applicability, too. Third, the forecast values determined by downstream applications might also restrict global forecasters' applicability. Consider a scenario of multiple downstream applications; each of these downstream applications might focus on other aspects of a forecast (e.g., accuracy of peak level or underestimation), leading to different forecast values. However, tuning a forecast to be optimal for a specific downstream application is impossible when using a global forecaster.

In order to wrap up, considering context information in global forecasting could be difficult. Thus, this might be a topic of future research. A possibility could be to train global forecasters for specific downstream applications or groups of time series. Furthermore, in neural networks, fine-tuning only the last layers of global forecast models might also be a possibility of combining the consideration of context information and global forecasters.

11 Conclusion

This thesis focuses on challenges in forecasting time series with calendar-driven periodicities. Therefore, it provides an overview of existing challenges. Four challenges are examined in more detail: the missing scenario, small training data size, concept drift, and periodicity challenge. Based on these challenges, the thesis states five research questions:

RQ1: Is it possible to use the conditioning mechanism to control the non-stationarity and periodicity in generated time series?

RQ2: Is it possible to apply generative models conditioned on appropriate statistical and weather information to generate mid and long-term peak energy load forecasts?

RQ3: Is it possible to locate a specific time series in a deep generative model's latent space and use this information to guide the data augmentation process?

RQ4: Can profiles improve the ability of machine learning models to cope with concept drifts?

RQ5: Can the usage of profiles in neural networks improve the deterministic and probabilistic forecasting performance?

The answers to the research questions correspond with the proposed methods to face the four considered challenges. The method to face the missing scenario challenge corresponds to the answers for RQ1 and RQ2. In particular, this thesis applies conditional generative models that do not only generate time series (Chapter 4). Instead, they have a conditioning mechanism that controls their generation process with determined conditioning information to them. This control allows the generation of time series for specific scenarios and the creation of mid-term forecasts conditioned on exogenous features and forecasts of easier-to-forecast time series as the rolling mean time series (Chapter 5).
The method to face the small training data size challenge is related to RQ3. This thesis proposes the Latent Space-based Forecast Enhancer (LSFE) that applies a globally trained generative model (Chapter 6). Using such a model directly to create time series would result in creating a time series corresponding to the median of all

time series. Thus, this thesis proposes to examine the latent space to identify the localisation of the target time series in this latent space. This information is then used to generate seed noise that enables the generative model to create time series samples that fit the target model.
The Profile-based Framework for Concept Drift Handling (PDH) proposed to face the concept drift challenge also helps to answer RQ4 (Chapter 8). The PDH uses profiles to support machine learning models in handling concept drift. In particular, the PDH separates the time series into the level and a remainder. The level is modelled using an adaptive average profile, leading to an automatic adaption to concept drifts that affect mainly the level. Arbitrary regressors model the remainder.
Finally, this thesis proposes the PNN and ProbPNN to support neural networks handling periodicities and answer RQ5 (Chapter 9). Both networks are hybrid neural networks that integrate profiles into deep learning. Due to the usage of profiles that model the periodicities, networks have not to learn them. Thus, the network can focus on other parts, resulting in a comparable small, fast-to-train neural network that still provides good forecasts.

The answers and the proposed methods raise further challenges and questions that might be dealt with by future work. Thereby, this thesis highlights two possible and promising research directions.

First, it might be promising to focus on global forecasting models as discussed in Chapter 10. Thus, future work might investigate if the idea proposed in Chapter 6 is applicable to the generative model-based forecasters used for the BigDEAL challenge. Furthermore, it might also be promising to try to provide multiple profiles to the PNN and ProbPNN while using an attention mechanism to select the best profiles for the considered time series.

Second, considering time series distributions explicitly and deriving forecasts from them, as in the BigDEAL challenge, might be of interest in future work. Such an approach could be applied to other forecasting tasks as well. However, such a model would not be restricted to forecasting. It can also be applied to make point forecasts probabilistic [113] or to detect anomalies [134]. Modelling distributions enables not only global forecasting models but also global time series models from which predictions for different tasks are derivable.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, *et al.*, “TensorFlow: A system for large-scale machine learning,” in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.

[2] O. O. Abayomi-Alli, T. Sidekerskien, R. Damaeviius, J. Sika, and D. Poap, “Empirical Mode Decomposition Based Data Augmentation for Time Series Prediction Using NARX Network,” in *Artificial Intelligence and Soft Computing (ICAISC 2020)*, Springer, Cham, 2020, pp. 702–711.

[3] K. Amarasinghe, D. L. Marino, and M. Manic, “Deep Neural Networks for Energy Load Forecasting,” in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, IEEE, 2017, pp. 1483–1488.

[4] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, “Guided Image Generation with Conditional Invertible Neural Networks,” *arXiv preprint arXiv:1907.02392*, 2019.

[5] L. Baier, M. Hofmann, N. Kühl, M. Mohr, and G. Satzger, “Handling Concept Drifts in Regression Problems–the Error Intersection Approach,” in *Proceedings of 15th International Conference on Wirtschaftsinformatik*, 2020, pp. 210–224.

[6] L. Baier, V. Kellner, N. Kühl, and G. Satzger, “Switching Scheme: A Novel Approach for Handling Incremental Concept Drift in Real-World Data Sets,” in *Proceedings of the 54th Hawaii International Conference on System Sciences*, 2021, pp. 990–999.

[7] K. Bandara, H. Hewamalage, Y.-H. Liu, Y. Kang, and C. Bergmeir, “Improving the accuracy of global forecasting models using time series data augmentation,” *Pattern Recognition*, no. 120, 2021.

[8] P. Bauer, A. Thorpe, and G. Brunet, “The quiet revolution of numerical weather prediction,” *Nature*, no. 525, pp. 47–55, 2015.

[9] A. Bifet and R. Gavalda, “Learning from time-changing data with adaptive windowing,” in *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, Society for Industrial and Applied Mathematics, 2007, pp. 443–448.

[10] H. V. Bitencourt and F. G. Guimaraes, “High-dimensional Multivariate Time Series Forecasting in IoT Applications using Embedding Non-stationary Fuzzy Time Series,” in *2021 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*, 2021.

[11] T. Cabello-López, M. Cañizares-Juan, M. Carranza-García, J. Garcia-Gutiérrez, and J. C. Riquelme, “Concept Drift Detection to Improve Time Series Forecasting of Wind Energy Generation,” in *Hybrid Artificial Intelligent Systems (HAIS 2022)*, 2022, pp. 133–140.

[12] R. Cavalcante, L. Minku, and A. Oliveira, “FEDD: Feature Extraction for Explicit Concept Drift Detection in time series,” in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 740–747.

[13] R. C. Cavalcante and A. L. I. Oliveira, “An approach to handle concept drift in financial time series based on Extreme Learning Machines and explicit Drift Detection,” in *2015 International Joint Conference on Neural Networks*, 2015.

[14] V. Cerqueira, L. Torgo, M. Oliveira, and B. Pfahringer, “Dynamic and Heterogeneous Ensembles for Time Series Forecasting,” in *2017 International Conference on Data Science and Advanced Analytics*, IEEE, 2017, pp. 242–251.

[15] C. Challu, K. G. Olivares, B. N. Oreshkin, *et al.*, “NHITS: Neural Hierarchical Interpolation for Time Series Forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023, pp. 6989–6997.

[16] T. Chen and C. Guestrin, “XGBoost,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[17] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, *et al.*, “Time series forecasting for nonlinear and non-stationary processes: a review and comparative study,” *IIE Transactions*, vol. 47, no. 10, pp. 1053–1071, 2015.

[18] K. Cho, B. van Merrienboer, Ç. Gülçehre, *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014, pp. 1724–1734.

[19] F. Chollet *et al.*, *Keras*, 2015.

[20] E. Cramer, L. Paeleke, A. Mitsos, and M. Dahmen, “Normalizing flow-based day-ahead wind power scenario generation for profitable and reliable delivery commitments by wind farm operators,” *Computers & Chemical Engineering*, no. 166, 2022.

[21] L. Dannecker, *Energy Time Series Forecasting: Efficient and Accurate Forecasting of Evolving Time Series from the Energy Domain*. Wiesbaden: Springer, 2015.

[22] L. N. Darlow, A. Joosen, M. Asenov, *et al.*, “TSMix: time series data augmentation by mixing sources,” in *Proceedings of the 3rd Workshop on Machine Learning and Systems*, 2023, pp. 109–114.

[23] A. Das, S. Roy, S. Chattopadhyay, and S. Nandi, “An End-to-End Approach for Benchmarking Time-Series Models Using Autoencoders,” in *Proceedings of the Global AI Congress 2019*, 2020, pp. 319–327.

[24] A. Debnath, G. Waghmare, H. Wadhwa, S. Asthana, and A. Arora, “Exploring Generative Data Augmentation in Multivariate Time Series Forecasting: Opportunities and Challenges,” in *MileTS ’21: 7th KDD Workshop on Mining and Learning from Time Series*, 2021.

[25] S. Demir, K. Mincev, K. Kok, and N. G. Paterakis, “Data augmentation for time series regression: Applying transformations, autoencoders and adversarial networks to electricity price forecasting,” *Applied Energy*, no. 304, 2021.

[26] L. Dinh, D. Krueger, and Y. Bengio, “NICE: Non-linear Independent Components Estimation,” in *3rd International Conference on Learning Representations, ICLR 2015 - Workshop Track Proceedings*, 2015.

[27] L. Dinh, J. Sohl-Dickstein, and S. Bengio, “Density estimation using Real NVP,” in *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[28] D. Dua and C. Graff, *UCI Machine Learning Repository*, 2019.

[29] E. S. Page, “Continuous Inspection Schemes,” *Biometrika*, no. 41, pp. 100–115, 1954.

[30] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, “Generative timbre spaces: Regularizing variational auto-encoders with perceptual metrics,” in *21st International Conference on Digital Audio Effects (DAFx-18)*, 2018.

[31] C. Esteban, S. L. Hyland, and G. Rätsch, “Real-valued (Medical) Time Series Generation with Recurrent Conditional GANs,” *arXiv preprint arXiv:1706.02633*, 2017.

[32] C. Fan, M. Chen, R. Tang, and J. Wang, “A novel deep generative modeling-based data augmentation strategy for improving short-term building energy predictions,” *Building Simulation*, no. 15, pp. 197–211, 2022.

[33] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, “Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network,” *Applied Energy*, vol. 282, 2021.

[34] G. Forestier, F. Petitjean, H. A. Dau, G. I. Webb, and E. Keogh, “Generating Synthetic Time Series to Augment Sparse Datasets,” in *2017 IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 865–870.

[35] C. Fry and M. Brundage, “The M4 forecasting competition – A practitioner’s view,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 156–160, 2020.

[36] J. Gama, I. liobait, A. Bifet, M. Pechenizkiy, and A. Bouchachia, “A survey on concept drift adaptation,” *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, 2014.

[37] A. Gasparin, S. Lukovic, and C. Alippi, “Deep learning for time series forecasting: The electric load case,” *CAAI Transactions on Intelligence Technology*, vol. 7, no. 1, pp. 1–25, 2022.

[38] L. Ge, W. Liao, S. Wang, B. Bak-Jensen, and J. R. Pillai, “Modeling Daily Load Profiles of Distribution Network for Scenario Generation Using Flow-Based Generative Network,” *IEEE Access*, no. 8, pp. 77 587–77 597, 2020.

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

[40] S. Gomez-Rosero, M. A. Capretz, and S. Mir, “Transfer Learning by Similarity Centred Architecture Evolution for Multiple Residential Load Forecasting,” *Smart Cities*, vol. 4, no. 1, pp. 217–240, 2021.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, “Generative Adversarial Nets,” in *Advances in Neural Information Processing Systems*, 2014, pp. 4089–4099.

[42] P. Grego, *Blazing a Ghostly Trail: ISON and Great Comets of the Past and Future*. Cham, Heidelberg, New York, Dordrecht, London: Springer, 2014.

[43] S. Haben, M. Voss, and W. Holderbaum, *Core Concepts and Methods in Load Forecasting: With Applications in Distribution Networks*, 1st ed. 2023. Cham: Springer International Publishing, 2023.

[44] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, 2020.

[45] B. Heidrich, A. Bartschat, M. Turowski, *et al.*, “pyWATTS: Python Workflow Automation Tool for Time Series,” *arXiv preprint arxiv:2106.10157*, 2021.

[46] B. Heidrich, M. Hertel, O. Neumann, V. Hagenmeyer, and R. Mikut, “Using conditional Invertible Neural Networks to Perform Mid-Term Peak Load Forecasting,” *IET Smart Grid*, 2023.

[47] B. Heidrich, N. Ludwig, M. Turowski, R. Mikut, and V. Hagenmeyer, “Adaptively coping with concept drifts in energy time series forecasting using profiles,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 459–470.

[48] B. Heidrich, L. Mannsperger, M. Turowski, *et al.*, “Boost short-term load forecasts with synthetic data from transferred latent space information,” *Energy Informatics*, vol. 5, 2022.

[49] B. Heidrich, T. Mühlpfordt, V. Hagenmeyer, and R. Mikut, “Delay-robust Estimation of the Reproduction Number and Comparative Evaluation on Generated Synthetic Data,” *medRxiv*, 2020.

[50] B. Heidrich, K. Phipps, O. Neumann, *et al.*, “ProbPNN: Enhancing Deep Probabilistic Forecasting with Statistical Information,” *arXiv preprint arXiv:2302.02597*, 2023.

[51] B. Heidrich, M. Turowski, N. Ludwig, R. Mikut, and V. Hagenmeyer, “Forecasting Energy Time Series with Profile Neural Networks,” in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, Association for Computing Machinery, 2020, pp. 220–230.

[52] B. Heidrich, M. Turowski, K. Phipps, *et al.*, “Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks,” *Applied Intelligence*, vol. 53, no. 8, pp. 8826–8843, 2023.

[53] J. Henze, J. Schreiber, and B. Sick, “Representation Learning in Power Time Series Forecasting,” in *Deep Learning*, W. Pedrycz, Ed., vol. 865, Cham: Springer International Publishing AG, 2020, pp. 67–101.

[54] M. Hertel, M. Beichter, B. Heidrich, *et al.*, “Transformer Training Strategies for Forecasting Multiple Load Time Series,” in *12th DACH+ Conference on Energy Informatics*, 2023.

[55] M. Hertel, S. Ott, B. Schäfer, *et al.*, “Evaluation of Transformer Architectures for Electrical Load Time-Series Forecasting,” in *Proceedings 32. Workshop Computational Intelligence*, 2022, pp. 93–110.

[56] H. Hewamalage, C. Bergmeir, and K. Bandara, “Recurrent neural networks for time series forecasting: Current status and future directions,” *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, 2021.

[57] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[58] T. Hong, “Forecasting with high frequency data: M4 competition and beyond,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 191–194, 2020.

[59] T. Hong, “Short term electric load forecasting,” Dissertation, North Carolina State University, Charlotte, 2010.

[60] T. Hong, P. Wang, and H. L. Willis, “A Naïve multiple linear regression benchmark for short term load forecasting,” in *IEEE Power and Energy Society general meeting, 2011*, 2011.

[61] A. Hooshmand and R. Sharma, “Energy Predictive Models with Limited Data using Transfer Learning,” in *The Tenth ACM International Conference on Future Energy Systems (e-Energy 2019)*, 2019, pp. 12–16.

[62] S. Hoyer and J. J. Hamman, “xarray: N-D labeled Arrays and Datasets in Python,” *Journal of Open Research Software*, vol. 5, no. 1, 2017.

[63] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice, 2nd edition*. Melbourne, Australia: OTexts, 2018.

[64] S. Jadon, A. Patankar, and J. K. Milczek, “Challenges and Approaches to Time-Series Forecasting for Traffic Prediction at Data Centers,” in *The 2021 International Conference on Smart Applications, Communications and Networking (SmartNets 2021)*, 2021.

[65] G. Jayanthi and P. Jothilakshmi, “Traffic time series forecasting on highways - a contemporary survey of models, methods and techniques,” *International Journal of Logistics Systems and Management*, vol. 39, no. 1, pp. 77–110, 2021.

[66] Y. Kang, R. J. Hyndman, and F. Li, “GRATIS: GeneRAting TIme Series with diverse and controllable characteristics,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 13, no. 4, pp. 354–376, 2020.

[67] L. Kegel, M. Hahmann, and W. Lehner, “Generating What-If Scenarios for Time Series Data,” in *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*, ser. ACM Digital Library, 2017.

[68] D. Keles, J. Scelle, F. Paraschiv, and W. Fichtner, “Extended forecast methods for day-ahead electricity spot prices applying artificial neural networks,” *Applied Energy*, no. 162, pp. 218–230, 2016.

[69] J. Kim, J. Moon, E. Hwang, and P. Kang, “Recurrent inception convolution neural network for multi short-term load forecasting,” *Energy and Buildings*, no. 194, pp. 328–341, 2019.

[70] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” in *3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[71] D. P. Kingma and P. Dhariwal, “Glow: Generative Flow with Invertible 1x1 Convolutions,” in *Advances in Neural Information Processing Systems*, 2018, pp. 10 215–10 224.

[72] D. P. Kingma and M. Welling, “Auto-Encoding Variational Bayes,” *arXiv preprint arXiv:1312.6114*, 2014.

[73] I. Koprinska, D. Wu, and Z. Wang, “Convolutional Neural Networks for Energy Time Series Forecasting,” in *2018 International Joint Conference on Neural Networks (IJCNN)*, 2018.

[74] D. Kreuzberger, N. Kühl, and S. Hirschl, “Machine Learning Operations (MLOps): Overview, Definition, and Architecture,” *IEEE Access*, no. 11, pp. 31 866–31 879, 2023.

[75] J. Lan, Q. Guo, and H. Sun, “Demand Side Data Generating Based on Conditional Generative Adversarial Networks,” *Energy Procedia*, no. 152, pp. 1188–1193, 2018.

[76] L. Landberg, “Short-term prediction of the power production from wind farms,” *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 80, no. 1-2, pp. 207–220, 1999.

[77] M. Lei, L. Tang, M. Li, Z. Ye, and L. Pan, “Forecasting Short-Term Residential Electricity Consumption Using a Deep Fusion Model,” in *Proceedings of 2018 Chinese Intelligent Systems Conference*, 2019, pp. 359–371.

[78] A. Li, F. Xiao, C. Fan, and M. Hu, “Development of an ANN-based building energy model for information-poor buildings using transfer learning,” *Building Simulation*, vol. 14, no. 1, pp. 89–101, 2021.

[79] N.-y. Liang, G.-b. Huang, P. Saratchandran, and N. Sundararajan, “A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks,” *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[80] B. Lim, S. Ö. Ark, N. Loeff, and T. Pfister, “Temporal fusion transformers for interpretable multi-horizon time series forecasting,” *International Journal of Forecasting*, vol. 37, no. 4, pp. 1748–1764, 2021.

[81] B. Lim and S. Zohren, “Time-series forecasting with deep learning: a survey,” *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, no. 379, 2021.

[82] P. C. de Lima e Silva, C. A. Severiano, M. A. Alves, *et al.*, "Forecasting in non-stationary environments with fuzzy time series," *Applied Soft Computing*, no. 97, 2020.

[83] W. Lin and Di Wu, "Residential Electric Load Forecasting via Attentive Transfer of Graph Neural Networks," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*, 2021, pp. 2716–2722.

[84] J. Liu, N. Wu, Y. Qiao, and Z. Li, "A scientometric review of research on traffic forecasting in transportation," *IET Intelligent Transport Systems*, vol. 15, no. 1, pp. 1–16, 2021.

[85] Z. Liu, Z. Zhu, J. Gao, and C. Xu, "Forecast Methods for Time Series Data: A Survey," *IEEE Access*, vol. 9, pp. 91 896–91 912, 2021.

[86] Z. Liu, R. Godahewa, K. Bandara, and C. Bergmeir, "Handling Concept Drift in Global Time Series Forecasting," *arXiv preprint arXiv:2304.01512*, 2023.

[87] J. Lu, A. Liu, F. Dong, *et al.*, "Learning under Concept Drift: A Review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[88] A. Maalej and C. Rebai, "Sensor Data Augmentation Strategy for Load Forecasting in Smart Grid Context," in *18th International Multi-Conference on Systems, Signals & Devices (SSD)*, 2021, pp. 979–983.

[89] M. C. Mabel and E. Fernandez, "Analysis of wind power generation and prediction using ANN: A case study," *Renewable Energy*, vol. 33, no. 5, pp. 986–992, 2008.

[90] A. Mahmoud and A. Mohammed, "A Survey on Deep Learning for Time-Series Forecasting," in *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*, Cham: Springer International Publishing, 2021, pp. 365–392.

[91] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "Statistical and Machine Learning forecasting methods: Concerns and ways forward," *PLoS ONE*, vol. 13, no. 3, 2018.

[92] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," *International Journal of Forecasting*, vol. 36, no. 1, pp. 54–74, 2020.

[93] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: Results, findings, conclusion and way forward," *International Journal of Forecasting*, pp. 802–808, 2018.

[94] D. L. Marino, K. Amarasinghe, and M. Manic, "Building Energy Load Forecasting using Deep Neural Networks," in *42nd Annual Conference of the IEEE Industrial Electronics Society (IECON 2016)*, IEEE, 2016, pp. 7046–7051.

[95] J. E. Matheson and R. L. Winkler, "Scoring rules for continuous probability distributions," *Management Science*, vol. 22, no. 10, pp. 1087–1096, 1976.

[96] W. McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, 2010, pp. 56–61.

[97] S. Meisenbacher, B. Heidrich, T. Martin, R. Mikut, and V. Hagenmeyer, "AutoPV: Automated photovoltaic forecasts with limited information using an ensemble of pre-trained models," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 386–414.

[98] S. Meisenbacher, T. Martin, B. Heidrich, R. Mikut, and V. Hagenmeyer, "Automating day-ahead forecasting of photovoltaic power generation: Model design, monitoring, and adaption," in *ETG Congress 2023*, 2023.

[99] S. Meisenbacher, M. Turowski, K. Phipps, *et al.*, "Review of automated time series forecasting pipelines," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 12, no. 6, 2022.

[100] J. Montiel, M. Halford, S. M. Mastelini, *et al.*, "River: machine learning for streaming data in Python," *arXiv:2012.04740*, 2020.

[101] O. Neumann, N. Ludwig, M. Turowski, *et al.*, "Smart Data Representations: Impact on the Accuracy of Deep Neural Networks," in *Proceedings 31. Workshop Computational Intelligence*, 2021.

[102] K. G. Olivares, C. Challu, G. Marcjasz, R. Weron, and A. Dubrawski, "Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx," *International Journal of Forecasting*, vol. 39, no. 2, 2022.

[103] G. Oliveira, R. Cavalcante, G. Cabral, L. Minku, and A. Oliveira, "Time Series Forecasting in the Presence of Concept Drift: A PSO-based Approach," in *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2017, pp. 239–246.

[104] J. Á. G. Ordiano, L. Gröll, R. Mikut, and V. Hagenmeyer, "Probabilistic energy forecasting using the nearest neighbors quantile filter and quantile regression," *International Journal of Forecasting*, vol. 36, no. 2, pp. 310–323, 2020.

[105] B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio, "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting," in *8th International Conference on Learning Representations (ICLR)*, 2020.

[106] B. N. Oreshkin, G. Dudek, P. Peka, and E. Turkina, "N-BEATS neural network for mid-term electricity load forecasting," *Applied Energy*, no. 293, 2021.

[107] I. Ozer, S. B. Efe, and H. Ozbay, "A combined deep learning application for short term load forecasting," *Alexandria Engineering Journal*, vol. 60, no. 4, pp. 3807–3818, 2021.

[108] Z. Pan, J. Wang, W. Liao, *et al.*, "Data-Driven EV Load Profiles Generation Using a Variational Auto-Encoder," *Energies*, vol. 12, no. 5, 2019.

[109] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035.

[110] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[111] J. Pérez, P. Arroba, and J. M. Moya, “Data augmentation through multivariate scenario forecasting in Data Centers using Generative Adversarial Networks,” *Applied Intelligence*, vol. 53, no. 2, pp. 1469–1486, 2023.

[112] F. Petropoulos, D. Apiletti, V. Assimakopoulos, *et al.*, “Forecasting: theory and practice,” *International Journal of Forecasting*, vol. 38, no. 3, pp. 705–871, 2022.

[113] K. Phipps, B. Heidrich, M. Turowski, *et al.*, “Creating Probabilistic Forecasts from Arbitrary Deterministic Forecasts using Conditional Invertible Neural Networks,” *arXiv preprint arxiv:2302.01800v1*, 2023.

[114] K. Phipps, S. Lerch, M. Andersson, *et al.*, “Evaluating ensemble post–processing for wind power forecasts,” *Wind Energy*, vol. 25, no. 8, pp. 1379–1405, 2022.

[115] K. Phipps, S. Meisenbacher, B. Heidrich, *et al.*, “Loss-Customised Probabilistic Energy Time Series Forecasts Using Automated Hyperparameter Optimisation,” in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, 2023, pp. 271–286.

[116] R. Prenger, R. Valle, and B. Catanzaro, “WaveGlow: A Flow-based Generative Network for Speech Synthesis,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 3617–3621.

[117] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic Backpropagation and Approximate Inference in Deep Generative Models,” in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1278–1286.

[118] M. Ribeiro, K. Grolinger, H. F. ElYamany, W. A. Higashino, and M. A. Capretz, “Transfer learning with seasonal and trend adjustment for cross-building energy forecasting,” *Energy and Buildings*, no. 165, pp. 352–363, 2018.

[119] J. J. Roberts, A. A. Mendiburu Zevallos, and A. M. Cassula, “Assessment of photovoltaic performance models for system simulation,” *Renewable and Sustainable Energy Reviews*, no. 72, pp. 1104–1123, 2017.

[120] F. Rodrigues and A. Trindade, “Load forecasting through functional clustering and ensemble learning,” *Knowledge and Information Systems*, vol. 57, no. 1, pp. 229–244, 2018.

[121] N. Romeo. “Ancient Device for Determining Taxes Discovered in Egypt.” (May 18, 2016), [Online]. Available: `https://www.nationalgeographic.com/history/article/160517-nilometer-discovered-ancient-egypt-nile-river-archaeology` (visited on May 15, 2023).

[122] S. M. Ross, *Introduction to Probability Models*, 10th ed. Amsterdam: Academic Press, 2010.

[123] D. Salinas, V. Flunkert, J. Gasthaus, and T. Januschowski, “DeepAR: Probabilistic forecasting with autoregressive recurrent networks,” *International Journal of Forecasting*, vol. 36, no. 3, pp. 1181–1191, 2020.

[124] S. Smyl, “A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting,” *International Journal of Forecasting*, vol. 36, no. 1, pp. 75–85, 2020.

[125] K. Sohn, X. Yan, and H. Lee, “Learning Structured Output Representation using Deep Conditional Generative Models,” in *Advances in Neural Information Processing Systems*, 2015, pp. 3483–3491.

[126] A. Tealab, “Time series forecasting using artificial neural networks methodologies: A systematic review,” *Future Computing and Informatics Journal*, vol. 3, no. 2, pp. 334–340, 2018.

[127] The pandas development team, *pandas-dev/pandas: Pandas*, 2020.

[128] C. Tian, C. Li, G. Zhang, and Y. Lv, “Data driven parallel prediction of building energy consumption using generative adversarial nets,” *Energy and Buildings*, no. 186, pp. 230–243, 2019.

[129] J. F. Torres, D. Hadjout, A. Sebaa, F. Martínez-Álvarez, and A. Troncoso, “Deep Learning for Time Series Forecasting: A Survey,” *Big data*, vol. 9, no. 1, pp. 3–21, 2021.

[130] B. N. Trinh, J. Thielen-del Pozo, and G. Thirel, “The reduction continuous rank probability score for evaluating discharge forecasts from hydrological ensemble prediction systems,” *Atmospheric Science Letters*, vol. 14, no. 2, pp. 61–65, 2013.

[131] F.-M. Tseng, H.-C. Yu, and G.-H. Tzeng, “Combining neural network model with seasonal time series ARIMA model,” *Technological Forecasting and Social Change*, vol. 69, no. 1, pp. 71–87, 2002.

[132] M. Turowski, “Data-Driven Methods for Managing Anomalies in Energy Time Series,” Dissertation, KIT, Karlsruhe, 2022.

[133] M. Turowski, B. Heidrich, K. Phipps, *et al.*, “Enhancing anomaly detection methods for energy time series using latent space data representations,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 208–227.

[134] M. Turowski, M. Weber, O. Neumann, *et al.*, “Modeling and generating synthetic anomalies for energy and power time series,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 471–484.

[135] A. van den Oord, S. Dieleman, H. Zen, *et al.*, “WaveNet: A Generative Model for Raw Audio,” in *Proceedings of 9th ISCA Workshop on Speech Synthesis Workshop (SSW 9)*, 2016.

[136] L. van der Maaten and G. Hinton, “Visualizing Data using t-SNE,” *Journal of Machine Learning Research*, no. 9, pp. 2579–2605, 2008.

[137] A. Voynov and A. Babenko, “Unsupervised Discovery of Interpretable Directions in the GAN Latent Space,” in *Proceedings of the 37th International Conference on Machine Learning (ICML’20)*, 2020.

[138] P. Vrablecová, V. Rozinajová, and A. Bou Ezzeddine, “Incremental Adaptive Time Series Prediction for Power Demand Forecasting,” in *Data Mining and Big Data*, 2017, pp. 83–92.

[139] M. Weber, M. Turowski, H. K. Cakmak, *et al.*, “Data-Driven Copy-Paste Imputation for Energy Time Series,” *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5409–5419, 2021.

[140] K. Wen, G. Zhao, B. He, J. Ma, and H. Zhang, “A decomposition-based forecasting method with transfer learning for railway short-term passenger flow in holidays,” *Expert Systems with Applications*, no. 189, 2022.

[141] D. Werling, M. Beichter, B. Heidrich, *et al.*, “The Impact of Forecast Characteristics on the Forecast Value for the Dispatchable Feeder,” in *Proceedings of the Fourteenth ACM International Conference on Future Energy Systems (2023)*, 2023, pp. 59–71.

[142] D. Werling, B. Heidrich, H. K. Çakmak, and V. Hagenmeyer, “Towards line-restricted dispatchable feeders using probabilistic forecasts for PV-dominated low-voltage distribution grids,” in *Proceedings of the Thirteenth ACM International Conference on Future Energy Systems*, 2022, pp. 395–400.

[143] F. Wiegel, J. Wachter, M. Kyesswa, *et al.*, “Smart Energy System Control Laboratory – a fully-automated and user-oriented research infrastructure for controlling and operating smart energy systems,” *at - Automatisierungstechnik*, vol. 70, no. 12, pp. 1116–1133, 2022.

[144] F. Wilcoxon, “Individual Comparisons by Ranking Methods,” *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[145] H. Wu, J. Xu, J. Wang, and M. Long, “Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting,” in *Advances in Neural Information Processing Systems*, 2021.

[146] L. Wu and M. Shahidehpour, “A Hybrid Model for Day-Ahead Price Forecasting,” *IEEE Transactions on Power Systems*, vol. 25, no. 3, pp. 1519–1530, 2010.

[147] T. Xu, L. K. Wenliang, M. Munn, and B. Acciaio, “COT-GAN: Generating Sequential Data via Causal Optimal Transport,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 8798–8809.

[148] R. Ye and Q. Dai, “A relationship-aligned transfer learning algorithm for time series forecasting,” *Information Sciences*, no. 593, pp. 17–34, 2022.

[149] R. Ye and Q. Dai, “Implementing transfer learning across different datasets for time series forecasting,” *Pattern Recognition*, no. 109, 2021.

[150] A. Yona, T. Senjyu, A. Y. Saber, *et al.*, “Application of Neural Network to 24-hour-Ahead Generating Power Forecasting for PV System,” in *2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century*, 2008.

[151] J. Yoon, D. Jarrett, and M. van der Schaar, “Time-series Generative Adversarial Networks,” in *Advances in Neural Information Processing Systems*, 2019, pp. 5508–5518.

[152] G. P. Zhang and M. Qi, “Neural network forecasting for seasonal and trend time series,” *European Journal of Operational Research*, vol. 160, no. 2, pp. 501–514, 2005.

[153] X. Zhang, R. Roy Chowdhury, J. Shang, R. Gupta, and D. Hong, “Towards Diverse and Coherent Augmentation for Time-Series Forecasting,” in *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2023, pp. 1–5.

[154] H. Zhou, S. Zhang, J. Peng, *et al.*, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 11 106–11 115.

[155] L. Ziyin, T. Hartwig, and M. Ueda, “Neural networks fail to learn periodic functions and how to fix it,” in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1583–1594.

[156] I. Zliobaite, “Learning under concept drift: an overview,” *arXiv preprint arXiv:1010.4784*, 2010.

Colophon

This thesis was typeset with $\LaTeX\,2_\varepsilon$. It uses the *Clean Thesis* style developed by Ricardo Langner. The design of the *Clean Thesis* style is inspired by user guide documents from Apple Inc.

Download the *Clean Thesis* style at `http://cleanthesis.der-ric.de/`.