

Automated machine learning for time series forecasting in smart grid applications

Stefan Meisenbacher



Automated machine learning for time series forecasting in smart grid applications

Zur Erlangung des akademischen Grades eines

**DOKTORS DER INGENIEURWISSENSCHAFTEN
(Dr.-Ing.)**

von der

KIT-Fakultät für Maschinenbau des
Karlsruher Instituts für Technologie (KIT)

angenommene

DISSERTATION

von

Stefan Meisenbacher

Tag der mündlichen Prüfung:

23.09.2024

Hauptreferent:

apl. Prof. Dr.-Ing. Ralf Mikut

Korreferenten:

Prof. Dr.-Ing. Veit Hagenmeyer

Prof. Dr.-Ing. Peter Bretschneider

Abstract

Climate change has forced a paradigm shift in energy systems and requires increasing the share of renewable energy generation. However, this transition to intermittent energy sources such as Wind Power (WP) and PhotoVoltaic (PV) poses electricity storage challenges, which requires a flexibilization of the electricity grid through sector coupling. Coupling energy-consuming sectors with the power-generating sector requires solving many decentralized optimization problems, which in turn are based on forecasts of local electricity demand and supply. As a result, the demand for such locally customized forecasting models is increasing rapidly, forcing the time-consuming design and application processes to be automated.

This thesis therefore presents a novel concept for automating time series forecasting in smart grid applications. First, a systematic review on automated forecasting pipelines categorizes design automation methods regarding the pipeline sections i) pre-processing, ii) feature engineering, iii) hyperparameter optimization, iv) forecasting method selection, and v) forecast ensembling. Second, the novel taxonomy AutoLVL considers automated design together with automated operation under six automation levels, which act as a guideline to develop forecasting templates and services. Third, the forecasting template AutoPQ is proposed, which automates the design process to provide high-quality and customized probabilistic forecasts for different tasks in smart grid applications. Fourth, the forecasting template AutoWP is specifically designed for the scalable deployment to decentral located onshore WP turbines subject to regular and irregular interventions in WP generation capacity. Fifth, the forecasting template AutoPV is tailored to provide PV forecasts with limited information, i. e., missing meta-information about the PV mounting configuration characterized by tilt and azimuth angles, missing historical data of the target PV plant to train the model, and exogenous impacts on the PV plant's generation capabilities occurring during operation. Finally, the application in real-world smart grid environments demonstrates the utility of the developed automated forecasting templates for downstream applications.

Kurzfassung

Der Klimawandel hat einen Paradigmenwechsel eingeleitet und verlangt eine Erhöhung der erneuerbaren Energieerzeugung. Der Übergang zu dargebotsabhängigen Energiequellen wie Windkraft und Photovoltaik macht dabei eine Flexibilisierung des Stromnetzes durch Sektorenkopplung notwendig. Die Kopplung von energieverbrauchenden Sektoren mit dem stromerzeugenden Sektor bedarf der Lösung vieler dezentraler Optimierungsprobleme, die wiederum auf Vorhersagen des lokalen Strombedarfs und -angebots angewiesen sind. Daher steigt die Nachfrage nach solchen lokal angepassten Vorhersagemodellen rasant an, sodass zeitaufwändige Entwurfs- und Anwendungsprozesse automatisiert werden müssen.

In dieser Arbeit wird daher ein neues Konzept zur Automatisierung von Zeitreihenvorhersagen in Smart Grid Anwendungen vorgestellt. Zuerst werden Automatisierungsmethoden systematisch eingeordnet in die Pipelineabschnitte i) Vorverarbeitung, ii) Merkmalsextraktion, iii) Hyperparameter-Optimierung, iv) Auswahl der Vorhersagemethode und v) Ensemblebildung. Zweitens betrachtet die neue Taxonomie AutoLVL den automatisierten Modellentwurf zusammen mit dem automatisierten Modellbetrieb in sechs Automatisierungsstufen, die als Leitfaden für die Entwicklung von Vorhersagevorlagen und -diensten dienen. Drittens wird die Vorhersagevorlage AutoPQ vorgeschlagen, die den Entwurfsprozess automatisiert, um qualitativ hochwertige und maßgeschneiderte probabilistische Vorhersagen für verschiedene Aufgaben in Smart-Grid-Anwendungen bereitzustellen. Viertens ist die Vorhersagevorlage AutoWP speziell für den skalierbaren Einsatz in dezentral gelegenen Onshore-Windkraftanlagen entwickelt, die regelmäßigen und unregelmäßigen Eingriffen in die Erzeugungskapazität unterliegen. Fünftens ist die Vorhersagevorlage AutoPV darauf zugeschnitten, Photovoltaik-Stromerzeugungsvorhersagen unter limitierten Informationen zu liefern, nämlich fehlende Meta-Informationen über die Ausrichtungs- und Neigungskonfiguration der Photovoltaikanlage, fehlende historische Daten der Zielanlage zum Trainieren des Modells und exogene Einflüsse auf die Erzeugungskapazitäten der Photovoltaikanlage, die während des Betriebs auftreten. Abschließend demonstriert die Anwendung in realen Smart-Grid-Umgebungen den Nutzen der entwickelten automatischen Vorhersagevorlagen für nachgelagerte Anwendungen.

Contents

Abstract	i
Kurzfassung	iii
1 Introduction	1
1.1 Motivation	1
1.2 Time series forecasting	4
1.2.1 Point forecasting	4
1.2.2 Probabilistic forecasting	7
1.3 Status quo in automated forecasting pipelines	8
1.3.1 Data pre-processing	8
1.3.2 Feature engineering	12
1.3.3 Hyperparameter optimization	18
1.3.4 Forecasting method selection and ensembling	26
1.4 Typical time series characteristics in smart grid applications	32
1.5 Open questions	33
1.6 Objectives	34
2 New concept to automate time series forecasting in smart grid applications	37
2.1 Methods of Automation level 1	39
2.2 Methods of Automation level 2	41
2.2.1 Hyperparameter optimization	41
2.2.2 Combined algorithm selection and hyperparameter optimization	44
2.2.3 Successive halving	46
2.2.4 Ensemble optimization	49
2.3 Methods of Automation level 3	52
2.3.1 Prior knowledge	52
2.3.2 Pre-trained models	54
2.3.3 Performance monitoring	55
2.4 Methods of Automation level 4	57
2.4.1 Cyclic model adaption	57
2.4.2 Drift detection-based model adaption	58

2.5	Service implementation	59
2.6	Related work	61
2.7	Novelty and remaining questions	65
2.8	Contributed forecasting templates	66
3	AutoPQ: Template for automated point forecast-based quantile forecasts	69
3.1	Automated design methods	70
3.1.1	Creation of the probabilistic forecast	70
3.1.2	Important task-dependent design decisions	73
3.1.3	Optimal design of the probabilistic forecast	75
3.2	Evaluation	81
3.2.1	Benchmarking	82
3.2.2	Customizability	87
3.2.3	Ablation study	92
3.3	Discussion	102
3.3.1	Benchmarking	102
3.3.2	Customizability	102
3.3.3	Ablation study	103
3.3.4	Limitations	103
3.3.5	Benefits	106
3.4	Related work	106
3.5	Novelty and remaining questions	110
4	AutoWP: Template for automated wind power forecasts	113
4.1	Automated design methods	114
4.1.1	Forecasting methods based on wind power curve modeling	114
4.1.2	Forecasting methods based on autoregression and deep learning	118
4.1.3	Post-processing	118
4.1.4	Shutdown handling	119
4.2	Evaluation	121
4.2.1	Experimental setup	121
4.2.2	Results	123
4.2.3	Insights	125
4.3	Discussion	126
4.4	Related work	128
4.5	Novelty and remaining questions	130

5	AutoPV: Template for automated photovoltaic forecasts	133
5.1	Automated design methods	134
5.1.1	Creation of the ensemble pool models	134
5.1.2	Optimal weighting of the ensemble pool models	141
5.1.3	Transformation of the ensemble output to the new PV plant	142
5.2	Automated operation methods	142
5.2.1	Cyclic adaption	142
5.2.2	Drift detection-based adaption	143
5.3	Evaluation	144
5.3.1	Benchmarking	144
5.3.2	Interpretability	149
5.3.3	Drift detection	154
5.4	Discussion	159
5.4.1	Benchmarking	160
5.4.2	Interpretability	160
5.4.3	Drift detection	161
5.4.4	Limitations	161
5.4.5	Benefits	162
5.5	Related work	163
5.6	Novelty and remaining questions	168
6	Application	171
6.1	Forecasting for smart grid applications in the Energy Lab at KIT	171
6.2	Forecasting of photovoltaic generation for demand side management in the Smart East environment	173
6.3	Forecasting of photovoltaic and wind power generation for Redispatch 2.0	177
7	Conclusion and outlook	181
	Appendix	185
A	Introduction	185
A.1	Supplementary figures	185
A.2	Further metrics	185
A.3	Energy time series transformation	186
B	AutoPQ	187
B.1	Supplementary tables and figures	187
B.2	Algorithm for the ablation study	196

C	AutoWP	197
C.1	Supplementary figures	197
C.2	Generating probabilistic WP forecasts with AutoWP using normal- izing flows	199
D	AutoPV	201
D.1	Supplementary tables and figures	201
D.2	Generating probabilistic PV forecasts with AutoPV using normaliz- ing flows	214
List of abbreviations		217
List of symbols		223
List of figures		229
List of tables		241
List of supervised student theses		247
List of own publications		249
The author's Curriculum Vitae		253
Bibliography		257

1 Introduction

The introduction of this thesis starts with the motivation of the research in Section 1.1. Then, a brief overview of time series forecasting methods in Section 1.2 is given, followed by a summary of the status quo regarding automated forecasting pipelines in Section 1.3, and an outline of typical time series characteristics in smart grid applications in Section 1.4. Finally, open questions are stated in Section 1.5 and the objectives are outlined in Section 1.6.¹

1.1 Motivation

Global warming is leading to extensive climate change and an increase of weather extremes, posing a threat to life on earth, as many animals, plant species, and humans cannot adapt to the rapid environmental changes sufficiently fast. Greenhouse gas emissions are considered as the major driving force for atmospheric temperature increase. In 2015, the European Union (EU) and 194 states have acknowledged the impending dangers of global warming and signed the *Paris Agreement* [2] to take countermeasures. These actions target to i) reduce carbon dioxide emissions, promote environmentally sustainable industries, and transform the energy supply to lower the dependence of fossil fuels, ii) increase the climate resilience and adaption abilities of agriculture and forestry, and iii) hold the global increase in the average temperature well below 2 °C above pre-industrial levels. Social movements for climate urge governments to take action and comply with the signed agreement.

In Germany, a political paradigm shift considering the energy supply is taking place at latest since 2010 with the *Energiewende* (German, energy transformation), see Figure 1.1. The German Federal Government's energy concept aims at raising the share of renewable energies in the power generation, together with reducing the primary energy demand and increasing energy efficiency [3]. Figure 1.2 depicts the historical and targeted increase of renewable energy share in the gross electricity consumption.

Recently adopted objectives are the supply of energy-intensive production with hydrogen power, increasing the domestic energy efficiency, and the electrification of the transport sector. A growing transport electrification and an increasing share of renewables in the energy generation confront the electrical power supply with major challenges. For example,

¹ Note that Section 1.1, Section 1.2, and Section 1.3 are based on [1] and contain identical phrases; a detailed list is given in the Appendix.

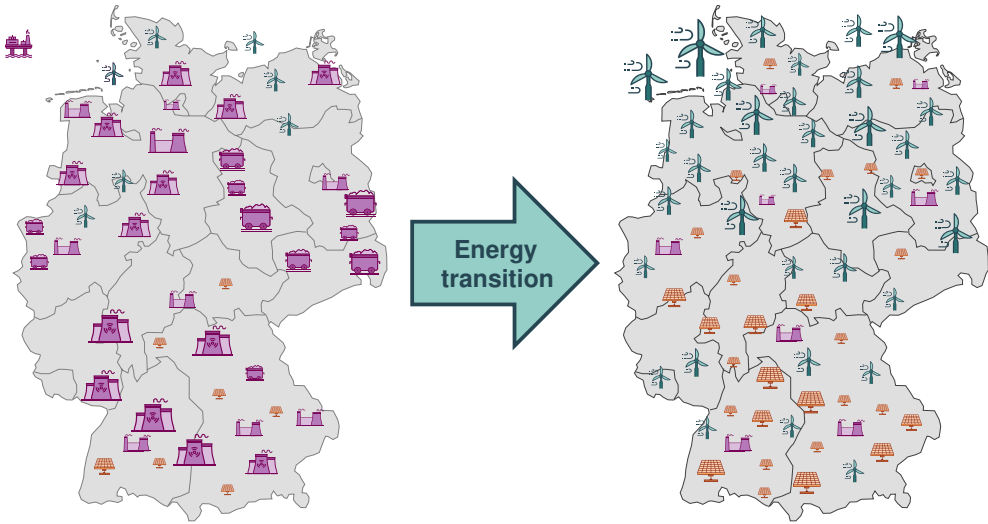


Figure 1.1: The transformation of the energy system from a primarily fossil-fueled power plant mix to renewable electricity generation; simplified, without claiming to correctly depict the electricity mix in Germany.²

uncoordinated charging of (battery) Electric Vehicles (EVs) and the intermittent generation behavior of renewables may lead to power grid congestion if high energy consumption coincides with low supply. Therefore, electrical grids are facing a major transformation, making them more flexible and decentralized, and enhance their resilience, however, thereby also increasing their complexity [4]. Hence, a smart grid is required that integrates communication, measurement, control, and automation technology. This integration enables the smart grid to monitor local power grid conditions in real-time and enables full utilization of the grid capacity by balancing demand and supply [5, 6]. The full utilization and balance of demand and supply are realizable by sector coupling, where energy-consuming sectors are interconnected with the power-generating sector to address electricity storage challenges by adding flexibility to the power grid [7]. Flexibilization is achieved through demand side management and demand response [8], along with various energy storage solutions such as stationary battery storage [9], the use of EVs as mobile storage [10], and the storage of thermal energy [11]. Yet, realizing sector coupling with the above approaches involve solving many decentralized optimization problems that, in turn, rely on forecasts of local electricity demand and supply [12, 13]. Further forecasting requirements include robustness against poor data quality, e. g., due to transmission gaps in smart meters, adaption to constantly changing systems, and fail-safe operation without manual intervention.

² The icons are adapted from [14].

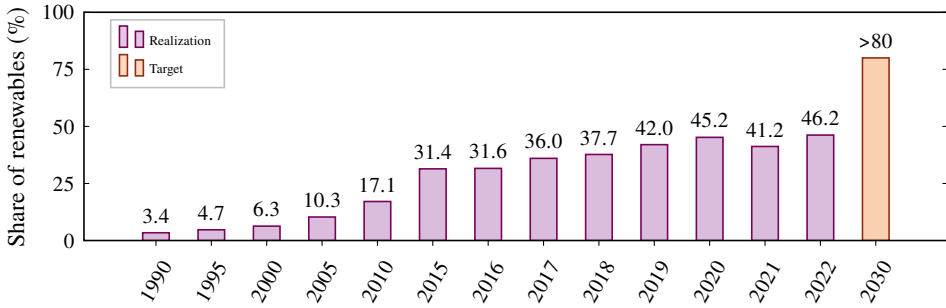


Figure 1.2: The development of renewable energy share in the German gross electricity consumption [15] and the targeted share for 2030 [16]. Explicit sectors are shown in Figure A.1 in the Appendix.

Designing such a time series forecasting model typically incorporates five sections. The first section of the design process is the data pre-processing to transform the raw data into a desirable form for the forecasting method [17, 18]. The second section is the feature engineering, which aims to extract hidden characteristics of the considered time series or to identify useful exogenous information for the forecasting method [19]. Each forecasting method contains hyperparameters that have to be set by the data scientist. Therefore, the third section, the HyperParameter Optimization (HPO), intends to improve the forecast quality over the default hyperparameter configuration [20]. Apart from the HPO, selecting the most suitable forecasting method is crucial for the forecast quality and is addressed in the fourth section [21]. The fifth section aims to increase the robustness of the forecast by forecast ensembling [22], i. e., bundling multiple forecasts of different forecasting models to avoid occasional poor forecasts [23].

The above sections of the design process are commonly organized in a pipeline structure as shown in Figure 1.3.

Manually tailoring the forecasting pipeline to a specific use case is time-consuming and challenging because selecting appropriate methods for the pipeline sections is iterative and requires expert knowledge. This expert knowledge is particularly crucial, as the forecast

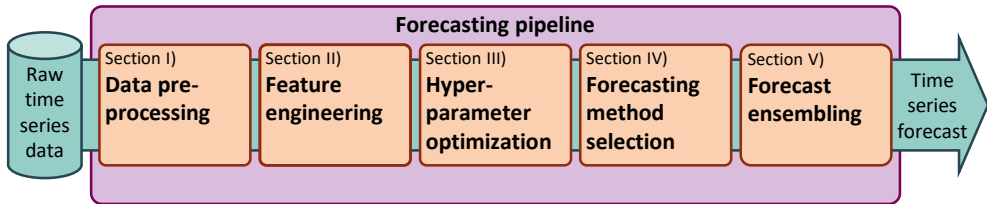


Figure 1.3: The forecasting pipeline systematizes the design process for time series forecasting using five pipeline sections. The figure is based on [1].

quality is sensitive to various design decisions [20], and the so-called forecast value often depends on requirements of the forecast's downstream application [24]. If these design decisions remain manual, it is foreseeable that the number of knowledgeable data scientists cannot handle the ever-growing demand for time series forecasts in future. Therefore, increasing the efficiency of the design process by automation is required [25].

To automate design decisions and remove the data scientist from the iterative design process, a variety of automation methods are available for each section of the forecasting pipeline. The sequential organization of these automation methods and the management of the data flow can be realized by creating a pipeline. Running the created forecasting pipeline trains a forecasting method – e. g. a Linear Regression (LR) – with historical time series data and results in a parameterized forecasting model. Thereby, the pipeline automates the design process.

1.2 Time series forecasting

A time series $\{\mathbf{y}[k], k = 1, 2, \dots, K\}$ reflects a set of $K \in \mathbb{N}_1$ observations typically measured at equidistant points in time [26]. The following two sections describe the estimation of future time series values. While a point forecasting model only estimates a single value per time point, which is the expected value, a probabilistic forecasting model also estimates the uncertainty inherent in the forecast.

1.2.1 Point forecasting

A point forecasting model $f_p(\cdot)$ estimates future expected values $\hat{\mathbf{y}}$ at origin k for one or more time points – the forecast horizon $H \in \mathbb{N}_1$ – using current and past values [27]. It is defined as

$$\begin{aligned} \hat{\mathbf{y}}[k+1, \dots, k+H] &= f_p(\mathbf{y}[k-H_1, \dots, k], \\ &\quad \mathbf{X}[k-H_1, \dots, k], \\ &\quad \hat{\mathbf{X}}[k+1, \dots, k+H], \\ &\quad \mathbf{p}), k, H, H_1 \in \mathbb{N}_1, k > H_1, \end{aligned} \tag{1.1}$$

where $H_1 \in \mathbb{N}_1$ indicates the horizon for past values $k - H_1$, \mathbf{y} represents values of the target time series, the matrix \mathbf{X} denotes values from exogenous time series (e. g. calendar information), the matrix $\hat{\mathbf{X}}$ indicates that the exogenous values originate from other forecasts (e. g. weather forecasts), and the vector \mathbf{p} contains the model's parameters [27].³

³ The vectors that include past values can be sparse, i. e., only certain time points from k to H_1 are included.

In point forecasting, the following four families of methods exist: naïve methods, Statistical Modeling (SM) methods, Machine Learning (ML) methods, and Deep Learning (DL) methods. These families and their main representatives are briefly introduced in the following.

Naïve forecasting methods. The simplest method family of forecasting are naïve methods. Common representatives are the averaging method, where the forecasts of all future values are equal to the average of historical data, and the Random Walk (RW) method, where the value of the last observation is used as forecast [28]. For RW, modifications are available for data with drift (dRW) and seasonality (sRW), which are detailed in [28].

Statistical modeling forecasting methods. More sophisticated than naïve methods are forecasting methods that use SM to forecast future time series values, which are extensively introduced in [29].

The first representatives are AutoRegression (AR) methods. The AutoRegressive Moving Average (ARMA) method [30] assumes a linear relationship between the lagged inputs and is applicable if the time series is stationary. If trends and seasonal characteristics are present, the time series is non-stationary, and the requirements for applying ARMA are not fulfilled. To address this problem, the AutoRegressive Integrated Moving Average (ARIMA) method [30] aims to remove time series trends through differencing and the seasonal AutoRegressive Integrated Moving Average (sARIMA) method aims to eliminate the seasonality by seasonal differencing [31].

The second representatives are Exponential Smoothing (ES) methods, where the forecast is determined by a weighted average of past observations, with the weights decaying exponentially with their age [28]. Simple Exponential Smoothing (SES) is a valid forecasting method for time series data without a trend or seasonal pattern. For time series with a trend, SES is adapted to Double Exponential Smoothing (DES), and Triple Exponential Smoothing (TES) is suitable for time series with seasonality. Additionally, ES variants exist which are applicable to damped or exponential trends. All variations of ES are summarized in the Error Trend Seasonality (ETS) model notation [32].

Machine learning forecasting methods. While most SM forecasting methods are based on assumptions about the distribution of the time series data, ML methods have fewer restrictions in terms of linearity and stationarity [31]. In addition to SM forecasting methods, which are specifically developed for time series forecasting, one can use regression methods based on ML to forecast multiple time points ahead using the following strategies, either solely or in combination [33]:

Recursive strategy: One trains a single regression model $f_p(\cdot)$ to forecast one time point ahead. In the operation, one recursively feeds back the output value to the input for the next time point.

Direct strategy: One trains multiple independent regression models $f_{p,h}(\cdot)$, $h = 1, \dots, H$, each to forecast the value at time $k + h$. The input is similar for each model.

Multiple output strategy: One trains a single regression model to forecast the whole horizon H at once. Consequently, the output is not a single value but a vector.

Representative ML methods are the Support Vector Regression (SVR), Decision Tree (DT)-based methods, and feedforward Artificial Neural Networks (ANNs) like the MultiLayer Perceptron (MLP). An overview of those methods applied to point forecasting is given in [34].

Deep learning forecasting methods. DL is a sub-set of ML; the methods are based on (deep) ANNs with specialized architectures, which include convolutional and recurrent layers, attention units, and residual connections.

A convolutional layer can be used to learn important patterns (i. e. features) from the input sequence. This is achieved by kernels that slide over the data, capturing specific patterns present in the data (e. g. trends, seasonality, or other temporal features) [35]. Capturing temporal dependencies can also be achieved by recurrent layers such as the Long Short-Term Memory (LSTM) [36]. Recurrent layers allow – unlike uni-directional feed-forward layers – the output from neurons to affect subsequent input to the same neuron. Such a neuron maintains a hidden state that captures information about the past observations in the time series, i. e., the output depends on the prior elements within the input sequence. Stacking many of the prescribed layers or units to a deep neural network can result in a vanishing gradient during training [37]. In this situation the gradients in a neural network diminish exponentially as they are back-propagated, making it difficult to learn and update the weights of early layers effectively. This problem can be addressed by residual blocks [38]. A residual block consists of a shortcut connection that allows the input to bypass one or more layers and be added directly to the output of these layers. Another method that allows to dynamically focus on specific parts of the input sequence to make a forecast without requiring recurrent units is the transformer architecture [39], which is based on attention units. More specifically, an attention unit computes importance scores for each element in the input sequence [40, 41], which is particularly useful when processing sequences where not all elements contribute equally to the forecast.

Representative DL architectures for point forecasting are the Profile Neural Network (PNN) [42], Neural Hierarchical Interpolation for Time Series (N-HiTS) [43], and the

Temporal Fusion Transformer (TFT) [44]. Recently, so-called foundation models are gaining attention, e. g., Time Generative Pre-trained Transformer (TimeGPT) -1 [45], which are based on the transformer architecture and are trained on numerous diverse data sets.

1.2.2 Probabilistic forecasting

A probabilistic forecasting model $f_q(\cdot)$ estimates the uncertainty inherent in the forecast of future expected values $\hat{\mathbf{y}}[k]$. This uncertainty can be quantified by estimating quantiles or Prediction Intervals (PIs), the full probability distribution, as well as scenarios of $\hat{\mathbf{y}}[k]$.

Quantile-based forecasting methods. Quantile-based methods estimate the conditional q -quantiles $\hat{\mathbf{y}}_q[k]$, $q \in (0, 1)$ for each future value $\hat{\mathbf{y}}[k]$. For example, with $q = 0.5$, the probability of the future value being smaller than the quantile forecast is 50 %, which is the median. Using pairs of quantile-based forecasts allow to estimate PIs with a given conditional probability of containing $\mathbf{y}[k]$. For example, $\hat{\mathbf{y}}_{0.05}[k]$ and $\hat{\mathbf{y}}_{0.95}[k]$ can be used to estimate the PI with the probability $95 \% - 5 \% = 90 \%$ of containing the future value $\hat{\mathbf{y}}[k]$.

Representative quantile-based methods are Quantile Regression Neural Networks (QRNNs) [46], and the Nearest Neighbor Quantile Filter (NNQF) [47].

Distribution-based forecasting methods. Distribution-based methods estimate the full Probability Density Function (PDF) or Cumulative Distribution Function (CDF) for each future value $\hat{\mathbf{y}}[k]$ in the forecast horizon H . Consequently, a distribution forecast is a H -dimensional PDF or CDF. The estimated distribution may follow a known parametric distribution (e. g. a Gaussian distribution) or be non-parametric if no assumptions are made about the underlying distribution. These distribution forecasts can in turn be used to derive quantiles and PIs.

A representative parametric distribution-based method is Deep AutoRegression (DeepAR) [48], and recent non-parametric distribution-based methods are based on normalizing flows, e. g., Bernstein-polynomial normalizing flows [49].

Scenario-based forecasting methods. Scenario-based methods combine several possible future scenarios to describe the forecast uncertainty [50]. The idea originates from weather forecasting, where the scenarios are estimated using numerical weather prediction models with varying initial conditions [51].⁴ More generally, a scenario-based forecast is

⁴ In weather forecasting, scenario-based methods are more commonly called ensemble forecasts. However, to avoid confusion with other ensemble methods in this thesis, the term is not used.

based on various trajectories in which several options for the future are considered. These scenarios can in turn be used to derive quantiles, PIs or parameters of an assumed probability distribution [52].

Representative scenario-based methods are used for weather forecasting [53], Wind Power (WP) forecasting [54], and EV load forecasting [55].

1.3 Status quo in automated forecasting pipelines

As outlined in the motivation, the increase of decentrally produced renewable energies is a major paradigm of the energy system transformation, requiring local grid monitoring and control [12]. Therefore, the need for locally adapted short-term forecasting models (e. g. for power demand, PhotoVoltaic (PV) power and WP generation) is ever growing, requiring the design of forecasting models to be automated.

The long-term objective toward full automation has motivated numerous researchers and led to promising research results. Several surveys and review studies analyze the automation of single forecasting pipeline sections such as pre-processing [17, 18], feature engineering [19], HPO and forecasting method selection [20, 21], and forecast ensembling [22]. Moreover, rather than focusing on the automated design of forecasting pipelines, existing studies on time series forecasting only consider SM or ML and DL forecasting methods themselves [29, 33, 56]. However, a comprehensive literature analysis on the entire automated time series forecasting pipeline that considers the families of SM, ML, and DL methods is lacking. Therefore, this section analyzes the literature on automated time series forecasting pipelines to investigate how to automate the design process of forecasting models, considering literature from various research directions, including SM forecasting, ML, and DL.

1.3.1 Data pre-processing

Since most forecasting methods rely on assumptions about data properties, data pre-processing is of crucial importance. Data pre-processing includes anomaly detection and handling, transforming the time series to make it stationary, and scaling the time series. In the following sub-sections, automated methods for data pre-processing are introduced, and their utilization in forecasting pipelines is exemplified with the reviewed literature, guided by Table 1.1. More detailed tables for each sub-section can be found in [1].

Anomalies

An anomaly is a value that significantly deviates from the rest of the time series [88]. Anomalies are induced by rare events or by errors in the data. Apart from anomalous

Table 1.1: Summary of automated data pre-processing methods in the literature on time series forecasting pipelines; adapted from [1].

Reference	Forecasting method family	Anomaly		Stationarity		Scaling
		detection	handling	testing	tran.	
[57]	SM	X	X	X	X	X
[58]	ML	X	X	X	X	X
[59]	ML	X	X	-	X	-
[60]	ML	X	X	-	X	X
[61]	DL	X	X	-	-	-
[62]	SM	X	X	X	X	-
[63]	ML	X	X	X	X	-
[64–69]	SM	-	-	X	X	-
[70]	SM, ML	-	-	X	X	-
[71]	SM	-	-	X	X	X
[72]	ML	-	-	X	X	X
[73]	ML	-	X	X	X	X
[74]	SM, ML	-	X	-	X	X
[75]	DL	-	-	-	X	-
[76–80]	ML	-	-	-	-	X
[81, 82]	ML, DL	-	-	-	-	X
[83, 84]	DL	-	-	-	-	X
[85, 86]	SM, ML	-	-	-	-	-
[87]	SM	-	-	-	-	X

transformation (tran.), Statistical Modeling (SM), Machine Learning (ML), Deep Learning (DL)

existing values, which we call outliers, anomalies also comprise missing values in the time series. Both outliers and missing values can degrade the performance of forecasting methods or cause the training to fail [89]. Therefore, appropriate anomaly detection and handling are necessary.

Outlier detection and handling. Automated outlier detection and handling aim to identify abnormal values and replace them with plausible values without human intervention. Liu et al. [57] define an interval based on the global mean and the variance of the time series

$$[\text{mean}(y) - \alpha_t \cdot \text{var}(y), \text{mean}(y) + \alpha_t \cdot \text{var}(y)], \quad (1.2)$$

with the threshold α_t and the anomaly detection method considers values outside the interval as outliers. Detected abnormal values are automatically substituted with the arithmetical averages of the nearest previous and posterior normal values. However, anomalous values themselves bias the estimation of the mean and the variance, and the method is only valid for stationary time series. Other authors tackle this weakness by calculating the local median

instead of the global mean. Martínez et al. [59], Yan [58], and Fan et al. [61] consider an observation as outlier if its absolute value is four times greater than the absolute medians of the three consecutive points before and after the observation. However, only extreme values above the absolute medians are detected, extreme values below the absolute medians are not identified. Widodo et al. [60] apply the Hampel method, which automatically replaces any value that deviates from the median of its neighbors by more than three Median Absolute Deviations (MAD) with that median value. Unlike previous methods, which use statistical measures, the anomaly detection and handling method of Maravall et al. [62] is based on a forecasting model. The method fits an ARIMA model, evaluates the MAD of the estimation residuals, and automatically replaces detected outliers with the forecast of the ARIMA model.

Missing value handling. Automated missing value handling aims to reconstruct absent observations without human assistance. The method of Fan et al. [61] automatically replaces missing values in the time series with the median of 12 consecutive points before and after the observation. Yet, this method is prone to larger gaps of missing data. The method of Züfle and Kounev [63] automatically imputes missing values by multiplying the known value one season before or after the missing value by the trend factor estimated between the day of the missing values and the day copied. Using this procedure in chronological order allows the imputed values for the imputation of subsequent missing values. After the missing value imputation, the authors apply a similar outlier detection method like (1.2) with $\alpha_t = 3$. Unlike [57], Züfle and Kounev [63] use the robust standard deviation between the 1st and 99th percentile of the data and replace the outliers by linearly interpolating between the two nearest non-anomalous values.⁵

Stationarity

In a stationary time series $y[k]$, the statistical properties do not depend on the time of observation k , i. e., the distribution of $y[k, \dots, k+h]$ is independent of k for all h [28]. Therefore, a time series with trends or seasonal patterns is not stationary because either the mean of the time series, its variance, or both change over time. Since some SM forecasting methods assume a stationary time series, their application to non-stationary time series requires an appropriate transformation. Stationarity tests help to identify the type of non-stationarity and support the automated selection of the appropriate transformation.

⁵ The authors disaggregated the time series into the seasonal, trend, and residual components and applied anomaly detection and handling on the stationary residual.

Autocorrelation and differencing transformations. A first approach to automatically identify non-stationarities in time series is proposed by Tran and Reed [64] based on the AutoCorrelation Function (ACF) and the Partial AutoCorrelation Function (PACF). The ACF and the PACF visualize the correlation of a time series with a delayed copy of itself. The authors automatically detect decay patterns by calculating the average rate of change in the magnitude of high frequencies in the ACF and PACF, and consider rates of less than 10 percent as slow decay. The slow decay patterns indicate trends in the time series. To remove the trends, the time series is differenced by subtracting successive observations d_1 times. After differencing, the ACF and PACF show significant peaks at regular periods s_p , if the time series is seasonal. To remove the seasonality, the time series is seasonally differenced D_1 times by subtracting observations separated by s_p . Note that the ARIMA forecasting method explicitly includes differencing as hyperparameter d_1 in the model structure and the sARIMA method additionally considers seasonal differencing with D_1 and s_p .

Frequency filters. Apart from the ACF and PACF, methods based on frequency filters are used to identify seasonality. Bauer et al. [90] use the periodogram to automatically retrieve all frequencies within the time series, iterate over the found frequencies, and match each frequency with “reasonable” frequencies (e. g. daily, hourly, and yearly) with tolerance to determine seasonal frequencies. Kourentzes and Crone [74] propose the Iterative Neural Filter (INF) to automatically identify seasonal frequencies. The filter distinguishes between stochastic and deterministic components and iteratively removes seasonalities, trends, and irregularities in the time series.

Unit root tests. Statistical unit root tests are used to identify non-stationarities before applying transformation methods. The unit root tests used in the literature include the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test [91] or the Cox-Stuart test [92] for testing if the time series is stationary around a deterministic trend, the Augmented Dickey-Fuller (ADF) test [93] for the existence of stochastic trends in the time series, and the Canova-Hansen test [94] or Osborn-Chui-Smith-Birchenhall (OCSB) test [95] for the existence of seasonal patterns over time. According to the trends and seasonalities detected in the unit root tests, the time series can be automatically differenced or seasonally differenced, respectively.

Logarithm and normality transformations. Apart from unit root testing, Maravall et al. [62] and Liu et al. [57] propose log-level tests to automatically evaluate if a log-transformation of the time series is beneficial. Amin et al. [69] use the Kolmogorov-Smirnov (KS) test [96] to determine whether a time series is normally distributed – if not, the log-transformation is applied. Other authors apply the log-transformation without testing to

reduce the variance [68, 75] or apply various transformations until critical values are satisfied in t-tests of ACF and PACF [67]. For achieving normality and stabilizing the variance, the Box-Cox transformation [97] is often applied without preceding testing, like in [59, 63, 69, 90].

Time series decomposition. In addition to transformation operations to achieve stationarity, time series can be decomposed into the components trend, season, and irregular (i. e. the residual). Afterward, each component can be handled by an individual forecasting model [98]. Prominent decomposition methods are the Seasonal and Trend decomposition using LOESS (STL) [99] used in [59, 75, 90], and the additive or multiplicative decomposition applied by the authors of [60, 70].

Scaling

The scale of the time series influences the performance of many forecasting methods based on ML and DL. If the range of values is large, learning methods based on gradient descent may converge much slower or fail due to instability. Additionally, in the case of multiple inputs with different scales, the inputs with larger variance dominate the others in the calculation of many distance metrics [100].

The general formalization for time series scaling is

$$y^* = \frac{y - a}{b}. \quad (1.3)$$

A common approach is min-max scaling, where the time series are scaled in the range $[0, 1]$ with $a = \min(y)$, $b = \max(y) - \min(y)$. While min-max scaling guarantees that all time series are scaled to the same range $[0, 1]$, the Z-score normalization scales the time series to zero-mean and unit-variance with $a = \text{mean}(y)$, $b = \text{var}(y)$. Both scaling methods are sensitive to outliers and thus require a preceding anomaly detection and handling. Alternatively, robust scaling methods like *Scikit-learn's RobustScaler*⁶ can be used, which removes the median and scales the data according to the Inter-Quartile Range (IQR).

1.3.2 Feature engineering

A time series feature reflects the observations of an explanatory variable in the process being forecasted (i. e. the target variable). Figure 1.3 shows feature engineering as a sub-section of the time series forecasting pipeline, consisting of extraction and selection of features. The following analysis of automated feature extraction and selection in forecasting pipelines is guided by Table 1.2. A more detailed table can be found in [1].

⁶ <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html>

Table 1.2: Summary of automated feature engineering methods in the literature on time series forecasting pipelines; adapted from [1].

Reference	Forecasting method family	Energy domain	Feature extraction				Feature selection			Feature aggreg.
			lag	cyc.	tran.	exo.	filter	wrapper	embedded	
[63]	ML	-	X	X	-	X				
[90]	ML	-	-	X	X	-	X			
[101]	ML	-	X	-	-	-	X			
[82]	DL	-	X	-	-	-	X			
[58]	ML	X	X	-	-	-		X		
[61]	DL	-	X	-	-	-		X		
[102]	ML	-	X	-	-	-		X		
[59]	ML	-	X	-	-	-		X		
[103]	SM	-	-	-	-	X		X		
[104]	ML	X	-	-	-	X		X		
[79, 80]	ML	-	X	-	-	-		X		
[83]	DL	-	X	-	-	-			X	
[105]	ML	X	X	-	-	X			X	
[77]	ML	X	X	-	-	X			X	
[78]	ML	X	X	-	X	X	X	X		
[60]	ML	-	X	-	-	-	X		X	
[74]	ML	-	X	X	-	-	X			
[76]	ML	-	X	-	-	X		X	X	
[106]	SM	-	-	-	-	X				X

cyclic (cyc.), transformation (tran.), exogenous (exo.), aggregation (aggreg.),
Statistical Modeling (SM), Machine Learning (ML), Deep Learning (DL)

Feature extraction

The feature space of the training data set contains for each explanatory variable a time series of the same resolution and length as the target variable.⁷ Feature extraction aims at automatically enriching the input space of a forecasting method (i. e. the feature space) with additional explanatory variables. In the following, time series features are introduced, and their usage in automated forecasting pipelines is explored with literature references.

Lag features. In autocorrelated time series, past observations can be valuable explanatory variables to forecast the target variable. Lag features provide values from prior time points for the forecasting method, i. e., at time point k , the model also processes values that date back a certain time horizon H_1 . Lag features are useful if the target variable has inertia that is significantly reflected in the resolution of the time series or if exogenous influences affect the target variable in periodic patterns [78].

Table 1.2 reveals that lag features are the most applied type of features.

⁷ If the features are originally in a different resolution or length, they must be transformed accordingly.

Cyclic features. In the training data, the time stamps t (e. g. [YYYY-MM-DD hh:mm:ss]) of the target variable are unique and specify the sequence of the observations. Calendar information and cyclic patterns that humans can detect in this format, like the hour of the day, the day of the week, weekday and weekend, or month and year, cannot be processed by ML methods without encoding.⁸ Cyclic features provide this cyclic relationship by ordinal, interval, or categorical encoding. In the following, we exemplify these encodings for the day of the week (see Table 1.3). Ordinal encoding assigns an integer numerical value to individual days (x_{ord}). For an interval encoding, one may utilize a periodical sine-cosine encoding to establish similarities between related observations, e. g., for encoding the day of the week, one adds two time series features that encode each weekday ($x_{\text{sin}}, x_{\text{cos}}$).⁹ A categorical encoding is achieved through so-called one-hot encoding. In this example, we create a time series feature for each day of the week, being one on that day and zero otherwise ($x_{\text{Mon}}, x_{\text{Tue}}, x_{\text{Wed}}, x_{\text{Thu}}, x_{\text{Fri}}, x_{\text{Sat}}$). Since the last category – the Sunday – is already implicitly represented if each other categorical feature is zero, one may omit an explicit feature.

In the literature reviewed, cyclic features are not widely applied.¹⁰ Züfle and Kounev [63] apply Random Forest (RF) and Gradient Boosting Machine (GBM) methods, using lag features and the cyclic features hour of the day, day of the week, and the exogenous feature holiday. However, the encoding of the cyclic features is not described. Sin-cos encoded cyclic features are used by Kourentzes and Crone [74] as input for a MLP forecasting method.

Transformation features. Time series transformations are widely used to make time series stationary, and can also be applied to extract explanatory variables that we term transformation features. These features include calculating time derivatives, the decomposition into trend, seasonality and residual, moving averages [108], as well as applying mathematical operations on existing features, e. g., multiplication of exogenous features [78].

Exogenous features. In addition to features that are endogenously derived from the target variable, the forecast can be improved by using exogenous features if the target variable is subject to exogenous influences. In addition, lag, cyclic, and transformation features can also be extracted from exogenous features. Whether and which exogenous influences exist depends on the data domain.

⁸ Although some DL architectures can process temporal information, they can still benefit from cyclic features that contain calendar information [42, 107].

⁹ Two time series are necessary because, otherwise, the encoding would be ambiguous for several days.

¹⁰ The sparse use of cyclic and transformation features can be explained because most of the analyzed forecasting pipelines already consider this information by automatically selecting appropriate lags.

Table 1.3: Exemplification of cyclic encoding methods for the day of the week of a time series; adapted from [1].

	Ordinal	Sin-cos interval		One-hot categorical					
	x_{ord}	x_{sin}	x_{cos}	x_{Mon}	x_{Tue}	x_{Wed}	x_{Thu}	x_{Fri}	x_{Sat}
Mon	0	1.000	0.000	1	0	0	0	0	0
Tue	1	0.623	0.782	0	1	0	0	0	0
Wed	2	-0.223	0.975	0	0	1	0	0	0
Thu	3	-0.901	0.434	0	0	0	1	0	0
Fri	4	-0.901	-0.434	0	0	0	0	1	0
Sat	5	-0.223	-0.975	0	0	0	0	0	1
Sun	6	0.623	-0.782	0	0	0	0	0	0

For example, energy data often depends on exogenous weather measures [77, 78, 104, 105], human access data underlies the influences of public holidays and weather [63, 103], and sales data often correlates with economic indicators [76, 106].

Feature selection

After automatically extracting several features, they typically undergo a selection to remove features that provide no additional information value, e. g., because of non-correlation or redundancy. In the following, methods for automated feature selection are presented, and their application in forecasting pipelines is explored based on the reviewed literature.

Filter methods. Filter methods use metrics to rank single features or feature combinations and automatically select a promising feature set based on a threshold [109]. Hence, the filters rely on the general characteristics of the training data and are independent of the forecasting method and other subsequent sections in the forecasting pipeline.

The characteristics used for filtering are manifold. Chakrabarti and Faloutsos [101] propose an automated filtering method to determine the optimal lag features based on a threshold of the time series' fractal dimension. Martínez et al. [82] automatically select features by identifying significant lags in the PACF. Kourentzes and Crone [74] propose the INF method to identify seasonal frequencies in time series, which automatically selects lag and sin-cos encoded cyclic features. The method of Bauer et al. [90] automatically filters frequencies of the time series using the periodogram with the threshold of a spectral value greater than 50 % of the most dominant frequency. Additionally, cyclic features are extracted by calculating Fourier terms of the dominant frequencies. The most dominant frequency is used to decompose the time series into trend, season, and residual components by STL.

Finally, the cyclic features and the seasonal component as a transformation feature are used to forecast the de-trended time series.

Wrapper methods. In contrast to filter methods, the wrapper methods assess candidate features based on the forecasting performance on a validation data set. The best-performing feature set is selected by a search method, which tailors the feature set to the forecasting method or the entire forecasting pipeline, respectively. The search methods and performance metrics used in the literature are diverse. Independent from the chosen search method and metric, wrapper methods commonly take more computing time than filter methods, as training and validation require considerably more computational effort than calculating statistical measures. However, empirical studies show that the computational effort pays off in a better performance of the wrapper approach compared to filter methods [109].

In the literature reviewed, automated feature selection based on the performance of the forecasting pipeline is addressed by different methods. The method of Yan [58] and Fan et al. [61] automatically determines the optimal lag features based on the forecast error by searching over candidate lags. The candidate lags are either predefined individually (grid search) or drawn randomly between specified boundaries (random search). Balkin and Ord [102] and Martínez et al. [59] apply forward selection to automatically identify the optimal lag features. First, several forecasting pipelines are trained for each individual lag feature. Second, the forward selection selects the best-performing lag feature and repeats the first procedure by adding another lag feature. It retains the combination with the highest improvement and repeats the procedure until the forecasting performance stops increasing. Besides search heuristics, optimization can be applied for feature selection. For automatically selecting exogenous features, Lowther et al. [103] adopt a Mixed Integer Quadratic Programming (MIQP) problem [110] and Son and Kim [104] apply Particle Swarm Optimization (PSO). Three evolutionary search strategies for automatically selecting the optimal lag features of MLPs are evaluated by Donate et al. [79, 80], where the Estimation Distribution Algorithm (EDA) yields the best convergence speed and the lowest forecast error.

Embedded methods. Embedded methods integrate the feature selection into the training process of the forecasting method [109].¹¹ During the training, the embedded feature selection commonly estimates the feature importance and weights the features accordingly. Embedded methods require less computational effort than wrapper methods, but more than filter methods [78].

¹¹ Since the embedded feature selection is specifically designed for the training algorithm of the respective forecasting method, the transfer to other forecasting methods is not straightforward.

The embedded methods in the literature are specific to the forecasting method. The approach of Panigrahi and Behera [83] automatically determines the optimal lag features by a Differential Evolution Algorithm (DEA). Instead of using gradient-based methods for training an MLP, the authors integrate the weight estimation into the DEA, aiming to increase the convergence speed. Valente and Maldonado [105] consider the automated selection of lag and exogenous features by a forward selection embedded into the SVR training process. The forward selection is based on a contribution metric that takes into account lags whose inclusion minimizes the metric. An automated backward elimination for SVR using embedded kernel penalization is described by Maldonado et al. [77]. Lag and exogenous features that are irrelevant for the forecasting performance are successively removed during training.

Hybrid methods. Hybrid methods aim to combine the advantages of the above methods.

Rätz et al. [78] evaluate several filter, wrapper, embedded, and hybrid feature selection methods. Based on their experiments, they propose to use a filter method to automatically remove all features with low variance in the first step. Afterward, they apply Bayesian Optimization (BO) to assess the remaining feature candidate combinations, consisting of lag, transformation, and exogenous features based on the forecasting performance. The approach of Widodo et al. [60] filters significant lags using the ACF, associates the remaining lag features with a kernel and automatically assigns appropriate weights to the kernels during training of the Multiple Kernel Learning (MKL) method. Sagaert et al. [76] first filter candidate lag features using automated forward selection, similar to the stepwise search for the automated ARIMA design of Hyndman and Khandakar [66]. Then, the identified set of lag features together with exogenous features are used as inputs of the embedded Least Absolute Shrinkage and Selection Operator (LASSO) method that automatically selects features by shrinking the parameters of irrelevant ones.

Feature aggregation

Feature aggregation aims to transform the feature space into a low-dimensional representation while retaining the primary properties of the time series. The transformation is advantageous in high-dimensional feature spaces to reduce processing time and avoid the curse of dimensionality.¹² The Principal Component Analysis (PCA), e. g., maps the data to a lower-dimensional space, maximizing the variance in the lower-dimensional representation.¹³

¹² As the dimensionality of the feature space increases, the available training data becomes sparse.

¹³ Since the PCA assumes a normal distribution, anomalies must be identified and handled beforehand and appropriate transformations need to be performed to obtain a stationary time series [111].

Dellino et al. [106] pre-whiten the time series data with a PCA, i. e., the PCA aggregates exogenous features and discards the principal components with a low variance that are assumed as noise.

1.3.3 Hyperparameter optimization

Forecasting methods incorporate a wide range of hyperparameters about the forecasting model's structure, training regularization, and algorithm setup; parameters whose values are not directly derived from the data and must be selected by the data scientist. The hyperparameter configuration λ includes all considered hyperparameters and their selected values [112]. By tailoring λ to the specific problem using HyperParameter Optimization (HPO), one may improve the model performance over the default setting of common forecasting libraries [78].

Performance metrics and validation sample

Most HPO methods assume that the performance of the model is quantifiable. For point forecasting and probabilistic forecasting, different performance metrics exist, which are introduced in the following.

Performance metrics for point forecasting. To evaluate the model performance in point forecasting, metrics from information theory and error metrics are used. Information Criteria (IC) measure the amount of information lost by a SM model, taking into account the Goodness Of Fit (GOF) and the model complexity. The less information a model loses, the higher the model performance. One IC is the Akaike Information Criterion (AIC)

$$\text{AIC} = 2N_p - 2 \ln(\hat{\mathcal{L}}) \quad (1.4)$$

with the number of estimated parameters N_p and the model's maximum value of the likelihood function $\hat{\mathcal{L}}$. It rewards GOF but penalizes high numbers of model parameters. The penalty is required since adding more parameters may increase the likelihood without being justified by the data (overfitting). Another popular IC is the Bayesian Information Criterion (BIC)

$$\text{BIC} = N_p \ln(K) - 2 \ln(\hat{\mathcal{L}}) \quad (1.5)$$

that additionally considers the number of data points K .

Popular error metrics are the Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{K} \sum_{k=1}^K (y[k] - \hat{y}[k])^2, \quad (1.6)$$

and the Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{K} \sum_{k=1}^K |y[k] - \hat{y}[k]|, \quad (1.7)$$

where $\hat{y}[k]$ is the forecast and $y[k]$ the realized value at time point k . The smaller the error of the point forecast, the higher the model performance.¹⁴

Performance metrics for probabilistic forecasting. To evaluate the model performance in probabilistic forecasting, so-called scoring rules evaluate the properties of the forecast, such as sharpness, calibration, and dispersion according to the definitions of [113]. Sharpness describes the narrowness of the forecasted probabilities around observations, i. e., a sharp forecast has probabilities concentrated near the observations. Calibration describes the statistical compatibility between forecasts and observations, i. e., the probabilities of a well-calibrated forecast match the observed occurrences of observations. Dispersion describes the variability of the forecasted probabilities around observations, i. e., a low dispersion indicates a narrow range of probabilities and a higher forecast confidence. The joint assessment is possible with proper scoring rules [113], which leverage the close relationship between calibration and dispersion and use the calibration as proxy instead of measuring the dispersion specifically.

The Continuous Ranked Probability Score (CRPS) is a proper scoring rule that measures the calibration and sharpness of a forecasted CDF \hat{F}_Y [114]

$$\text{CRPS} = \frac{1}{K} \sum_{k=1}^K \int_{\mathbb{R}} (\hat{F}_Y[k](x) - \mathbb{1}\{y[k] \leq x\})^2 dx, \quad (1.8)$$

averaged over all time points K , where the realized value $y[k]$ at time point k is translated into a degenerate distribution with the indicator function $\mathbb{1}\{y[k] \leq x\}$ that is one if $y[k]$ is smaller than the variable of integration x and zero otherwise.

The Mean Absolute Quantile Deviation (MAQD) measures the calibration of probabilistic forecasts by calculating the absolute deviations of the forecasted q -quantiles from the theoretical quantiles averaged over all considered quantiles \mathbf{q} [115]

$$\text{MAQD} = \frac{1}{|\mathbf{q}|} \sum_{q \in \mathbf{q}} |\text{QD}(q)| = \frac{1}{|\mathbf{q}|} \sum_{q \in \mathbf{q}} \left| \left(\frac{1}{K} \sum_{k=1}^K \mathbb{1}\{y[k] \leq \hat{y}_q[k]\} \right) - q \right|, \quad (1.9)$$

¹⁴ Further error metrics that are derived from the MSE and the MAE, such as the normalized Mean Absolute Error (nMAE) (A.2), the Root Mean Squared Error (RMSE) (A.1), and the normalized Root Mean Squared Error (nRMSE) (A.4) can be found in the Appendix.

with the observation $y[k]$, the forecast for the q -quantile $\hat{y}_q[k]$, and the indicator function $\mathbb{1}$. While QD being zero is ideal, a positive value indicates the quantile forecast overestimates the theoretical quantile, and negative value indicates the opposite.

The smaller the CRPS or the MAQD of the probabilistic forecast, the better the respective probabilistic properties assessed by the metric.

Validation sample. Error metrics of point forecasts and scoring rules of probabilistic forecasts can be calculated **in-sample** or **out-of-sample**. In-sample means that the forecast error is determined on data points that are part of the training data set, while out-of-sample uses unseen points from the validation data set. To increase the robustness, Cross Validation (CV) can be performed by splitting the data several times into different training and validation sets and averaging the validation error across folds. The out-of-sample error validation is generally considered to be a more trustworthy empirical evidence, as in-sample error validation is prone to overfitting.

Since ICs of point forecasts try to prevent overfitting implicitly with the penalty term, they are computed in-sample.

Optimization methods

Given validation data and metrics, the hyperparameter configuration of forecasting methods can be optimized using different optimization methods. In the following sub-sections, different automated HPO methods are introduced, and their application in forecasting pipelines is exemplified using literature references, guided by Table 1.4. A more detailed table can be found in [1].

Grid search and random search. The most elementary method for HPO is the exhaustive **grid search**, where the data scientist defines a finite set of $n = 1, \dots, N_\lambda$ hyperparameter values to be evaluated, resulting in a full factorial configuration space $\mathbf{\Lambda} \in \mathbb{R}^{N_\lambda}$. As the grid search evaluates the Cartesian product of these sets, the number of computations B_i grows exponentially with the dimensionality of \mathbb{R}^{N_λ} . Hence, increasing the discretization resolution increases the computational effort substantially [112]. The **random search** [123] is an alternative to the grid search. It irregularly samples the hyperparameter set until a certain number of computations B_i is exhausted. The random search may perform better than the grid search if some hyperparameters are much more important than others.¹⁵ Figure 1.4 shows a comparison of both methods with two hyperparameters and an equal number of computations B_i .

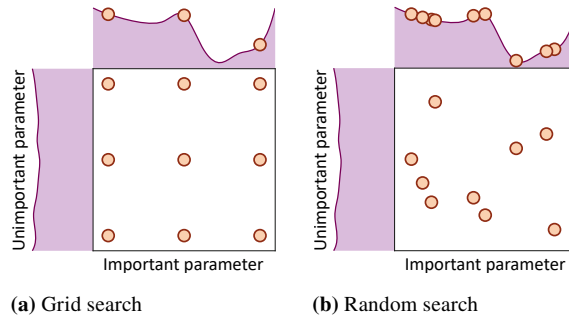
¹⁵ The greater importance of a few hyperparameters over others applies in many cases, e. g., [78, 123].

Table 1.4: Summary of HPO methods in the literature on time series forecasting pipelines; adapted from [1].

Reference	Forecasting method family	Performance metric	Validation sample		Optimization method
			in-sample	out-of-sample	
[64, 69]	SM	-	-	-	heuristic
[32, 65, 66, 71, 116, 117]	SM	IC	X	-	grid search
[118]	SM	IC, error	X	-	grid search
[103, 119]	SM	user-defined	X	X	grid search
[87]	SM	error	-	X	BO
[120]	SM	error	X	-	NLP
[121]	SM	error	X	-	heuristic, grid search
[58, 101]	ML	-	-	-	heuristic
[60, 76, 77, 104, 105]	ML	error	-	X	grid search
[86]	SM, ML	error	-	X	grid search
[61]	DL	error	-	X	random search
[83]	DL	error	X	X	metaheuristic
[79, 80]	ML	error	-	X	metaheuristic
[81, 122]	ML, DL	error	-	X	metaheuristic
[78]	ML	error	-	X	BO
[75]	DL	error	-	X	BO

Statistical Modeling (SM), Machine Learning (ML), Deep Learning (DL), Information Criteria (IC), Bayesian Optimization (BO), Non-Linear Programming (NLP)

Hyndman et al. [32] generalize the formulation of ES forecasting methods with the ETS method and suggest a grid search that automatically selects the hyperparameter configuration with the lowest in-sample AIC. Utilizing grid search for HPO is also applied to SM forecasting methods based on AR and Moving Average (MA). Sekma et al. [71] optimize the hyperparameters of an AR (p_{AR}, s_p) and a Vector AutoRegression (VAR) (p_{AR}, s_p) method, respectively, using grid search. The hyperparameter configuration resulting in the

**Figure 1.4:** Comparison of a grid search and a random search for minimizing a function with one important and one unimportant parameter [1]. The figure is based on [112, 123].

minimal BIC, calculated in-sample, is automatically selected. A similar HPO method for MA (q_{MA}) methods is proposed by Svetunkov and Petropoulos [116], where the data scientist needs to define the in-sample optimization criterion. An advanced method, combining pre-processing and HPO of ARIMA (p_{AR}, d_I, q_{MA}) methods is proposed by Alzyout et al. [65]. First, a stationarity test determines the differencing order d_I . Second, the maximal AR -lag order $p_{AR, \max}$ and the maximal MA -lag order $q_{MA, \max}$ are determined by automatically evaluating the PACF and the ACF, respectively. Finally, a grid search from 0 to $p_{AR, \max}$ and 0 to $q_{MA, \max}$ selects the optimal hyperparameter configuration based on in-sample AIC or BIC. Combining pre-processing and HPO is also proposed by Hyndman and Khandakar [66]. First, the authors propose to automatically determine the differencing and seasonal differencing order d_I and D_I and the seasonal period s_p with a stationarity test. Second, instead of an exhaustive grid search over the hyperparameters p_{AR}, q_{MA}, P_{AR} , and Q_{MA} of the sARIMA (p_{AR}, d_I, q_{MA}) (P_{AR}, D_I, Q_{MA}) $_{s_p}$, they propose a two-stage grid search, reducing the number of evaluations. In the first stage, four candidate hyperparameter configurations are evaluated, whose hyperparameters depend on the previously determined s_p . The hyperparameter configuration with the smallest in-sample AIC value proceeds to the second stage, where p_{AR}, q_{MA}, P_{AR} , and Q_{MA} are varied ± 1 . If a configuration with a lower AIC value is found, it becomes the active configuration of stage two and the variation is repeated until the AIC stops improving. Pedregal et al. [117] adopt this method except for the pre-processing step. Instead of stationarity tests, the variance of the time series is minimized to identify the differencing orders d_I and D_I .

Grid search is also applied for HPO of forecasting methods based on ML and DL. Sagaert et al. [76] determine the optimal shrinkage factor λ_s of the LASSO method for embedded feature selection in terms of the Mean Absolute Percentage Error (MAPE) of a ten-fold CV (out-of-sample). Several authors apply a grid search to determine the optimal hyperparameter configuration of an SVR. Son and Kim [104] optimize the hyperparameters C_{SV} and γ_{SV} based on the RMSE, Maldonado et al. [77] and Valente and Maldonado [105] based on the MAPE. The grid search used by Widodo et al. [60, 86] is based on a five-fold CV, using the mean symmetric Mean Absolute Percentage Error (sMAPE) value across folds as the forecast error metric. They optimize the hyperparameters C_{SV} and ε_{SV} of the SVRs used in a MKL approach with embedded feature selection. Combining automated feature selection and HPO is also suggested by Fan et al. [61]. The authors identify the optimal lag features using random search and link the number of hidden neurons N_h to the number of input neurons N_i .

Bayesian optimization. Rather than evaluating a finite search grid, Bayesian Optimization (BO) explores and exploits the configuration space $\mathbf{\Lambda} = \Lambda_1 \times \Lambda_2, \dots, \Lambda_{N_\lambda}$ of N_λ

hyperparameters. BO uses a probabilistic surrogate model to approximate the objective function Q that maps the forecasting pipeline's performance Q on Λ . More specifically, an observation $Q(\lambda)$ of the objective function reflects the pipeline's performance with a particular hyperparameter configuration $\lambda \in \Lambda$ of the forecasting methods used. In each iteration, the optimization updates the surrogate model with the new observation and uses an acquisition function to decide on the next hyperparameter configuration $\lambda \in \Lambda$ to be explored (see Figure 1.5). The acquisition function trades off the exploration against the exploitation of Λ by determining the expected benefit of different hyperparameter configurations using the probabilistic distribution of the surrogate model [112].

For surrogate modeling, various approaches exist, ranging from Gaussian Processes (GPs) and their modifications to ML approaches, e. g., RFs or Tree Parzen Estimators (TPEs). Feurer and Hutter [112] recommend using a GP-based BO for configuration spaces with real-valued hyperparameters and computationally expensive training, and an RF or TPE-based BO for configuration spaces with categorical hyperparameters and conditions, e. g., the choice of a forecasting method and its conditional (sub-)configuration space.

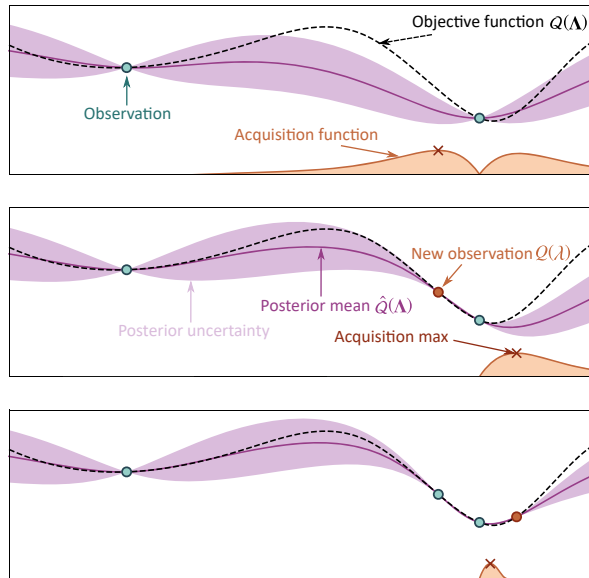


Figure 1.5: Two exemplary iterations of a BO on a 1D function. The BO minimizes the predicted objective function $\hat{Q}(\lambda)$ (purple line) by maximizing the acquisition function (light orange surface). The acquisition value is high where the value of $\hat{Q}(\lambda)$ is low, and its predictive uncertainty (light purple surface) is high. The true objective function (dashed line) might lie outside of the predicted uncertainty interval [1]. The figure is based on [112]

Dellino et al. [87] apply a BO with a GP surrogate model to optimize the hyperparameters of the sARIMA $(p_{AR}, d_I, q_{MA})(P_{AR}, D_I, Q_{MA})$ using an out-of-sample validation data set and the MAE. They compare the BO to an exhaustive grid search, where the BO achieves lower forecast errors but requires more computing time. In their experiment, however, the comparison is unequal as the configuration space of the BO is larger, and the best-performing model of the BO results in a hyperparameter configuration that the grid search does not evaluate.

Bandara et al. [75] use a BO with a GP surrogate model to optimize the recurrent neural architecture and hyperparameters of a deep ANN with LSTM layers. The performance of each hyperparameter configuration is evaluated out-of-sample based on the Mean Absolute Scaled Error (MASE). Rätz et al. [78] optimize multiple forecasting methods, combining feature selection, HPO, and forecasting method selection using a BO with a TPE surrogate model. The hyperparameter configurations' performances are estimated using the mean MAE of a five-fold out-of-sample CV.

Non-linear programming. Mathematical programming can be applied for HPO if calculating the performance metric is solvable in closed-form. Non-Linear Programming (NLP) solves an optimization problem, where at least one of the objective functions or constraints is a non-linear function of the decision variables. The objective function may be convex or non-convex, where non-convex NLP incorporates multiple feasible regions and multiple locally optimal solutions within them [124]. Depending on the formulation of the objective function and its constraints, different solving methods are appropriate.

Bermúdez et al. [120] apply the generalized reduced gradient method to solve a multiobjective NLP. They jointly minimize the in-sample RMSE, MAPE, and MAD to determine the four smoothing parameters of the TES method $\alpha_{ES}, \beta_{ES}, \gamma_{ES}, \varphi_{ES}$ and the number of periods of the seasonal cycle s_p . Lowther et al. [103] use MIQP to select suitable exogenous features for ARIMA. They combine this selection with a grid search over the sparsity parameter k_{sparse} of the MIQP formulation and the sARIMA hyperparameters $p_{AR}, d_I, q_{MA}, P_{AR}, D_I, Q_{MA}$.

Heuristics. A heuristic is an informed search technique that systematically explores a configuration space Λ subject to a constant search rule [125].

Tran and Reed [64] propose heuristics based on the ACF and PACF to determine the AR and MA lag order p_{AR} and q_{MA} . A similar approach is published by Amin et al. [69]. To determine the transformation parameter θ_* of the Theta forecasting method, Spiliotis et al. [121] apply the Brent–Dekker method – a root-finding method. They determine the optimal θ_* for eight different trend and season configurations, and select the configuration that minimizes the in-sample MAE.

Chakrabarti and Faloutsos [101] propose a heuristic to specify the number of nearest neighbors k_{NN} of the k-Nearest Neighbors (kNN) forecasting method after selecting the optimal lag features. A training sample-related heuristic for determining the spread factor of the General Regression Neural Network (GRNN) is introduced by Yan [58].

Metaheuristics. Metaheuristics are strategies for guiding a search according to feedback from the objective function, previous decisions, and prior performance [126], i.e., the searching behavior changes while exploring the configuration space Λ . Metaheuristics do not require assumptions about the objective function and can solve optimization problems where gradient-based methods fail.

Evolutionary Algorithms (EAs) comprise a wide range of population-based metaheuristics inspired by biological evolution [127]. A population of candidate hyperparameter configurations is evaluated using a fitness function to determine the performance of solutions. Weak solutions drop out, while well-performing solutions evolve. The mechanisms of selection and evolution differ between algorithms.

A Genetic Algorithm (GA) evolves a population of candidate hyperparameter configurations to explore and exploit the configuration space Λ . The hyperparameters of a candidate solution are encoded as genes in a chromosome. In each generation, the fitness of the population is evaluated, and the chromosomes of individual candidates are modified to create a new generation – the offspring. The modification includes recombination and mutation and depends on an individual candidate’s fitness. A part of the population is retained and forms with the offspring the next generation. Differential Evolution Algorithms (DEAs) differ from GAs in the mechanism of generating the offspring. While in GAs an individual acts as parent to generate an offspring, the DEA adds the weighted difference between three chromosomes to create a new individual. In this way, no separate probability distribution is required, making the algorithm self-organizing. In Estimation Distribution Algorithms (EDAs), the population is replaced by a probability distribution over the choices available at each position in the chromosome of the individuals. A new generation is obtained by sampling this distribution, avoiding premature convergence and making the representation of the population more compact.

Donate et al. [80] evaluate a GA, a DEA, and an EDA for optimizing the hyperparameter configuration of an MLP, including the number of input and hidden neurons N_i and N_h , as well as the training hyperparameters learning rate α and the parameter initialization seed s_{init} . The results of the experiments show that the DEA and the EDA require more than 100 generations to improve significantly over GA. After 200 generations, the EDA achieves the lowest forecast error, followed by the DEA and the GA. In a later publication [79], the authors adapt the chromosome encoding and replace s_{init} with the hyperparameter Δ_{max} of

the used training algorithm. In both publications, the fitness of each individual is evaluated by calculating the MSE on an out-of-sample validation data set. Panigrahi and Behera [83] apply a DEA to optimize N_i and N_h of an MLP, combining in-sample and out-of-sample validations. The fitness of each individual (RMSE) is calculated in-sample, and the DEA is terminated when the RMSE on the validation data set increases¹⁶, indicating the beginning of overfitting.

In **Particle Swarm Optimization (PSO)**, a population of hyperparameter candidate configurations – the swarm – is evaluated. The candidates move through the configuration space Λ , where the movement of the swarm is guided by the best-performing candidates so far.

Sergio et al. [81] apply PSO to optimize the hyperparameter configuration of multiple forecasting methods, combining the best hyperparameter configurations afterward to an ensemble.

Memetic algorithms combine the global search capabilities of EAs with local refinement methods to enhance solution quality. This enables a large search coverage in hyperparameter space as well as the thorough investigation of identified local optima.

Fischer et al. [122] apply evolutionary stochastic gradient descent [128] to optimize the hyperparameter configuration of MLPs and temporal Convolutional Neural Networks (CNNs). The comparison with random search shows a moderate reduction of the forecast error with significantly reduced computational effort.

1.3.4 Forecasting method selection and ensembling

Not only optimizing hyperparameters of a forecasting method but also selecting the appropriate method is crucial for the forecast quality. Consequently, the forecasting method selection is often combined with an individual HPO.¹⁷ Forecast ensembling aims to bundle the forecasts of several methods, thereby reducing the impact of occasional poor forecasts – which can even occur with the best-selected and optimally configured forecasting method.

Forecasting method selection

For automatically selecting the best-performing forecasting method, there are several approaches that we divide into heuristic, empirical, and decision model-based selection. The selection is based on experience, a determined performance, or meta-features. The determined performance reflects IC or error metrics. Meta-features describe properties of the time

¹⁶ This regularization is also called *early stopping*.

¹⁷ Selecting the optimal forecasting method and finding the optimal hyperparameter configuration is also called the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem.

series to be forecasted and provide meta-information, including statistical characteristics of the target time series, such as the skewness, kurtosis, and self-similarity; and domain information, such as physical properties of the system and environmental characteristics. In the following, methods for automated forecasting method selection are presented, and their application in forecasting pipelines is examined based on the reviewed literature, guided by the summary in Table 1.5. A more detailed table can be found in [1].

Heuristic forecasting method selection. The heuristic selection of the forecasting method relies on fixed rules. The basis of these rules is experience and statistical tests that examine the time series for certain characteristics. Therefore, heuristic selection requires neither a determined performance nor meta-features.

Shcherbakov et al. [129] propose decision rules that consider the amount of available training data. For less than 672 observations, a naïve method is selected that uses the previous day’s value as the forecast. For 672 – 2688 observations, a MA method is applied, whereas an ANN is selected for more than 2688 observations. Besides the amount of training data, the availability of calendar information and exogenous time series also determines the chosen forecasting method.

Table 1.5: Summary of automated method selection and ensembling methods in the literature on time series forecasting pipelines; adapted from [1].

Reference	Forecasting method family	Forecasting method selection category	Selection basis		Forecast ensembling	
			performance	meta-features	selection	ensemble
[59]	ML	-	-	-	use all	avg.
[129]	SM, ML	heuristic	-	-	-	-
[130]	SM	heuristic	-	-	use all	wgt.
[63, 78, 102]	ML	empirical	X	-	-	-
[69–71, 131]	SM	empirical	X	-	-	-
[73]	ML	empirical	X	-	k-best	avg.
[132]	SM	heuristic, empirical	X	-	k-best	avg.
[133]	SM, ML	empirical	X	-	k-best	wgt.
[134]	ML	empirical	X	-	k-best	wgt.
[81]	ML, DL	heuristic, empirical	X	-	dyn. best	avg., wgt.
[135]	SM	decision model	X	-	-	-
[136]	SM	decision model	X	X	-	-
[86, 137, 138]	SM, ML	decision model	-	X	-	-
[72, 139, 140]	ML	decision model	-	X	-	-
[75]	DL	decision model	-	X	-	-
[141]	SM, ML	decision model	-	X	use all	wgt.
[84, 142]	DL	decision model	-	X	use all	wgt.
[85]	SM, ML	decision model	-	X	k-best	wgt.
[143]	SM	decision model	X	X	-	-

Statistical Modeling (SM), Machine Learning (ML), Deep Learning (DL), averaging (avg.), dynamic (dyn.), weighting (wgt.)

Empirical forecasting method selection. The empirical forecasting method selection determines the performance of several forecasting methods during training (in-sample) or on a validation data set (out-of-sample) and automatically selects the best-performing forecasting method.¹⁸

Balkin and Ord [102] train an AR, an MLP, and a naïve RW method, and select the forecasting method with the smallest in-sample BIC. Similarly, Sekma et al. [71] decide between AR and VAR forecasting methods based on the smallest in-sample BIC; Amin et al. [69] use the AIC to choose between ARIMA and Self Exciting Threshold AutoRegressive Moving Average (SETARMA).

An out-of-sample validation is used by Pereira et al. [131]. They calculate the MAE of six forecasting methods on a validation data set and select the method with the lowest MAE. A similar selection strategy is used by Züfle and Kounev [63]. They select the best-performing candidate forecasting method in terms of the R^2 score on validation data. Robustness can be increased by CV. Rätz et al. [78] combine feature selection, HPO, and the selection of the optimal forecasting method using BO with the mean MAE over five CV folds.

The effectiveness of in-sample and out-of-sample empirical forecasting method selection is evaluated by Fildes and Petropoulos [70]. They assess the following four empirical selection methods: i) minimal one-point-ahead in-sample MSE, ii) minimal one-point-ahead out-of-sample MAPE, iii) minimal h -point-ahead out-of-sample MAPE, and iv) minimal 1-18-points-ahead out-of-sample MAPE. The latter method proves to be better than the one-point-ahead validation and even than adjusting the validation to the corresponding forecast horizon H .

Decision model-based forecasting method selection. Instead of a heuristic or empirical forecasting method selection, one may train a decision model to automatically select the optimal forecasting method. As decision models, regression, classification, and clustering methods can be used to establish a relationship between performance information or meta-features and the optimal forecasting method.

In a decision model, the selection and aggregation of meta-features can improve decision accuracy. The principle of feature selection and aggregation methods corresponds to the descriptions in Section 1.3.2. Similar to the optimization of forecasts, the decision model can also be improved by HPO.

Taghiyeh et al. [135] propose a decision model-based selection method that relies on a classification method. The authors suggest to select the optimal forecasting method from a pool of candidates based on their in-sample and out-of-sample MSE. None of the three classification methods, including Logistic Regression (LogR), DT, and Support Vector

¹⁸ For a detailed description of the in-sample and out-of-sample validation, refer to Section 1.3.3.

Machine (SVM), outperforms the others. Kück et al. [136] propose a decision model-based selection based on the out-of-sample sMAPE and meta-features. They apply an MLP classifier as decision model and select its inputs using a grid search over 127 feature sets that include error metrics and meta-features.

The approaches above [135, 136] have the disadvantage that all candidate forecasting methods must be trained on the target data set to determine the applied error metrics. Computing meta-features, in contrast, does not require training. Hence, relying only on meta-features for decision model-based selection saves computing time. Widodo and Budi [86] propose a kNN classifier to select the forecasting method for a target time series based on the meta-features introduced in [144] such as the strength of the trend and the seasonality, as well as statistical measures like skewness and kurtosis. In the design of the classifier, the authors apply forward selection for meta-feature selection. Another approach based on the same meta-features is introduced by Scholz-Reiter et al. [137]. They use a meta-feature aggregation instead of meta-feature selection, and a Linear Discriminant Analysis (LDA) as classification method to select the optimal forecasting method. Shahoud et al. [139] introduce statistical meta-features for different aggregation levels of the time series to extract characteristics at different time scales. They select suitable meta-features by a backward elimination and aggregate them using an autoencoder. RF and ANN classifiers are compared for decision-making, both optimized with a grid search, where the ANN classifier achieves better performance in terms of selecting the best forecasting method. In addition to statistical and time series meta-features, Cui et al. [140] include domain information. The domain-based meta-features describe physical properties of buildings for which energy consumption is to be forecasted. Bauer et al. [72] evaluate three types of decision models, i. e., classification, regression, and hybrid. In the classification decision model, an RF is trained to map meta-features to the forecasting method with the lowest forecast error. In the regression decision model, an RF learns how much worse each forecasting method is compared to the best method (i. e. forecast quality degradation). The hybrid decision model combines this RF regression with an RF classifier that maps the RF regression prediction to the best method. In the evaluation, the hybrid approach achieves the best performance in terms of forecast quality degradation. Instead of training an individual model for each new time series, Bandara et al. [75] suggest clustering time series using meta-features and training only one model for each cluster, which is applied to all time series in the cluster.

Forecast ensembling

Forecast ensembling aims to improve the forecast robustness by bundling multiple forecasts of different models. We differentiate forecast ensembling from ensemble learning methods

that build an ensemble of weak models¹⁹, such as RF and GBM. Ensembling the forecasts from a pool of different forecasting models aims to avoid occasional poor forecasts, rather than outperforming the best individual forecasting model [23]. In the following, methods for ensembling in automated forecasting pipelines are introduced and exemplified using the reviewed literature, guided by the summary in Table 1.5. A more detailed table can be found in [1].

The benefit of forecast ensembling is empirically demonstrated in many cases. For example, in the analysis of the so-called M3 forecasting competition [145], averaging the model output of all submitted forecasting methods performs better than each individual method itself. Simple forecast ensembling through averaging is used by Martínez et al. [59]. They average the output of three kNN models with $k_{NN} \in \{3, 5, 7\}$ after the identification of optimal lag features with forward selection.

To improve averaging, one may weight the forecasting methods according to the expected individual performance. An ensemble can also be improved by only considering the k -best candidate methods, ranked by a forecasting method selection beforehand (i. e. heuristic, empirical, or decision model-based methods).

Selecting the candidate methods based on a heuristic forecasting method selection is proposed by Pawlikowski and Chorowska [130]. They categorize the time series data of the M4 forecasting competition [146] in terms of their frequency, and the existence of a trend and seasonality. Depending on the category, they select a distinct pool of candidate forecasting methods. The hyperparameters of the candidate methods are optimized and the weight for each candidate is determined based on the sMAPE error, validated out-of-sample with a rolling origin evaluation.

The following authors use empirical forecasting method selection to consider only the k -best candidate methods in the ensemble. Crone and Kourentzes [73] empirically determine the forecasting performance of candidate MLP architectures in a grid search (N_h , activation function) with a rolling origin evaluation (out-of-sample) and average the outputs of the ten best candidates to reduce the impact of overfitting. Kourentzes et al. [132] propose a forward selection heuristic to decide on the number of ranked candidates to be considered for averaging. They calculate the performance metric's rate of increase Q' assigned to each forecast and include all candidates until the first steep increase $Q' > \alpha_1$. To detect the increase, they use the same approach used for detecting outliers in boxplots, i. e., $\alpha_1 = Q3 + 1.5 \cdot IQR$, where $Q3$ is the 3rd quartile.

¹⁹ The forecast of a weak model, e. g., a DT, is only slightly superior to a random estimate. Ensemble learning aims to combine many weak models to achieve a good estimate.

Instead of averaging the k_{best} candidates, Shetty and Shobha [133] assign weights to the filtered candidates before averaging. Candidates with low forecast errors Q receive more weight, i. e.,

$$w_i^* = \frac{\prod_{j=1}^{k_{\text{best}}} Q_j}{Q_i \sum_{j=1}^{k_{\text{best}}} Q_j}, \quad w_i = \frac{w_i^*}{\sum_{j=1}^{k_{\text{best}}} w_j^*} \quad (1.10)$$

Wu et al. [134] propose a multi-objective optimization to determine the optimal weights for averaging candidates. They apply the flower pollination metaheuristic to minimize

$$\min \sum_{i=1}^4 w_i \cdot Q_i, \quad \text{subject to} \quad \sum_{i=1}^4 w_i = 1 \quad (1.11)$$

with the error metrics Q_i including the MAE, RMSE, and their relative formulations.

In the decision model-based method selection, one can also include weighted averaging directly in the decision model. Montero-Manso et al. [141] and Li et al. [142] train a GBM classifier with softmax-transformed outputs corresponding to the weights of the candidates for averaging. Similarly, Ma and Fildes [84] compare a CNN and a Fully Connected Neural Network (FCNN) classifier using output neurons with softmax activation to predict the weights for averaging. Instead of softmax outputs, Züfle et al. [85] use an LR as a decision model. Its output reflects the probability of how well the forecasting method fits the time series and determines the weight for averaging. For filtering the k -best candidate methods, they post-process the LR output.

Above-mentioned forecast ensembling methods with weighted average determine the weights statically, i. e., the weights do not change as the time series evolves. Sergio et al. [81] propose a dynamic weighting method for the ensembling of forecasts. For every single forecast, the method searches for the k -nearest patterns in the training data similar to the given input data. Three ensemble functions – average, median, and softmax – are evaluated on the found similar patterns, and the method with the best performance is chosen for making forecasts. A dynamic decision model-based approach is also introduced by Villegas et al. [143]. In their work, a binary SVM classifier is trained on performance metrics and meta-features to predict the best forecasting method from a pool of candidates for each forecast origin. In their experiment, the dynamic selection achieves the best performance compared to the mean and the median ensembling of all candidate forecasting methods.

1.4 Typical time series characteristics in smart grid applications

Previous sections provide a general overview of the status-quo in automated time series forecasting pipelines. In the following, we focus specifically on energy time series, which are subject to the three key characteristics.

Exogenous influences. Exogenous influences strongly impact both the energy demand and the renewable generation. Regarding renewable energy generation, environmental influences – especially the weather – determine the generation capabilities [12]. While PV power generation is mainly dependent on solar irradiance including cloud cover [147], additional minor influences exist due to air temperature and wind speed [148, 149], which affect the cell temperature [150]. WP generation is strongly dependent on the wind speed [151], with further minor influences resulting from the wind direction [152, 153] and the air density, which in turn depends on air temperature and atmospheric pressure [152, 154]. Weather influences also have an impact on energy demand time series, but these are less pronounced [12] and can often also be treated as seasonality, especially if they are related to calendar-driven exogenous effects.

Multi-seasonality. Most energy demand time series are subject to multiple seasons [12], which are often calendar-driven [42, 155]. A typical electrical demand time series is subject to three seasons [12]: First, a daily season exists, shaping the common time series progression of a day. Second, there is a weekly season with typical working day and weekend patterns.²⁰ Third, a long-term annual season exists, which coincident with exogenous influences, especially the average air temperature. Apart from calendar-driven seasonalities on energy demand, there are also influences on renewable energy generation. For PV, the sunrise and sunset, as well as the sun trajectory is calendar-driven [156, 157] (and depends on the plant's location). For WP, time-controlled shutdowns may exist, e. g., due to bat protection in the twilight [158].

Physical limitations. For many renewable energy generation and energy demand time series, physical limitations exist. In terms of PV power generation, no power is generated if there is no solar irradiance and negative power generation is impossible [159]. For WP generation, no power is generated if the wind speed is too low or too high, because the wind turbine is then shut down using pitch or azimuth control [160]. The power generation capabilities of both is limited by their peak power rating. With regard to energy demand

²⁰ Additionally, there are irregular patterns due to holidays, where behavior is often similar to the weekend.

time series and energy prices, the assumption of limitations requires caution. For example, negative prices are possible if there is electricity overproduction, and negative demand can occur for so-called prosumers [161], e. g., if they feed self-generated PV electricity into the grid.

1.5 Open questions

Based on the analysis of the status quo, the following open problems arise for automated time series forecasting in smart grid applications:

- **How is holistic design automation realized with forecasting pipelines?** The majority of the introduced papers incompletely cover the five forecasting pipeline sections, which are i) the data pre-processing, ii) the feature engineering, iii) the HPO, iv) the forecasting method selection, and v) the forecast ensembling. It follows that there is untapped potential regarding approaches that holistically consider the design automation in time series forecasting.
- **What does automation imply in time series forecasting and which scopes exist?** The individual forecasting task of a use case determines the scope of automation that is suitable. Frequently recurring tasks have a high potential for automation, while tasks with many specific challenges may still require human expertise. Another important aspect regarding the scope of automation is not only automating the design process but also automating the operation of the resulting models including performance monitoring and model adaption.
- **How can the design and application of recurring forecasting tasks be automated to enable a scalable application?** Recurring forecasting tasks for smart grid applications such as electrical load, electricity prices, mobility, WP and PV forecasts can potentially be generalized. It is therefore an open question how such tasks can be addressed with an automated forecasting pipeline that is generalized for the specific task.
- **How can automated forecasting pipelines be transferred into smart grid applications?** Manual tailoring of forecasting models is still a common practice in smart grid applications despite first successful applications of automated forecasting pipelines. Hence, there seems to be a lack of awareness of the automation potentials in real-world applications.

The scientific contribution of this chapter is summarized in Figure 1.6.

1.6 Objectives

The target of the present thesis is the development of a new, holistic concept for automating the entire process of time series forecasting in smart grid applications. The new concept pursues the following objectives:

- **Holistic design automation with forecasting pipelines:** Consider the five pipeline sections in the development of automated forecasting pipelines. That is, include i) data pre-processing, ii) feature engineering, iii) HPO, iv) forecasting method selection, and v) forecast ensembling.
- **Definition of automation levels for time series forecasting:** Describe the different scopes of automation by fusing automated design and automated operation of time series forecasting.
- **Development of automated forecasting templates for common tasks in smart grid applications:** Generalize recurring tasks in automated forecasting templates, i. e., automated forecasting pipelines that integrate domain knowledge and enable scalable application.
- **First applications of automated forecasting pipelines in smart grids:** Apply the developed automated forecasting templates in real-world smart grid environments to serve downstream applications.

Chapter 2 tackles the first two objectives. First, an automation level taxonomy for time series forecasting is defined in Chapter 2 (AutoLVL [162, 163]). Subsequently, contributions toward these five automation levels are presented, including templates for automated quantile forecasting in Chapter 3 (AutoPQ [164, 165] – Automation level 2), automated WP forecasting in Chapter 4 (AutoWP [166, 167] – Automation level 3), and automated PV forecasting in Chapter 5 (AutoPV [168, 169] – Automation level 4). Each of these chapters first motivates the scientific challenges, then describes the method, and evaluates the method regarding these challenges before discussing the results. Finally, related work is reviewed to assess the novelty of the method. Chapter 6 addresses the fourth and last objective by presenting software implementations of the novel methods and their application to real-world problems. Finally, Chapter 7 concludes and gives an outlook of the present thesis.

Analysis of status quo

Review on automated forecasting pipelines



Publication

S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut, "Review of automated time series forecasting pipelines", WIREs Data Mining and Knowledge Discovery, vol. 12, no. 6, e1475, 2022.

Figure 1.6: The contributions of this chapter with the review on automated forecasting pipelines.

2 New concept to automate time series forecasting in smart grid applications

For systematically approaching the objectives defined in Section 1.6, this chapter presents an overview of a new concept for automated time series forecasting in smart grid applications. The new concept aims to automate design and operation of forecasting models to reduce the human effort and satisfy the increasing demand for time series forecasts in smart grids. To address different scopes of automation in time series forecasting with a special focus on smart grid applications, the novel automation level taxonomy AutoLVL is defined in Figure 2.1, which fuses automated model design and operation.¹

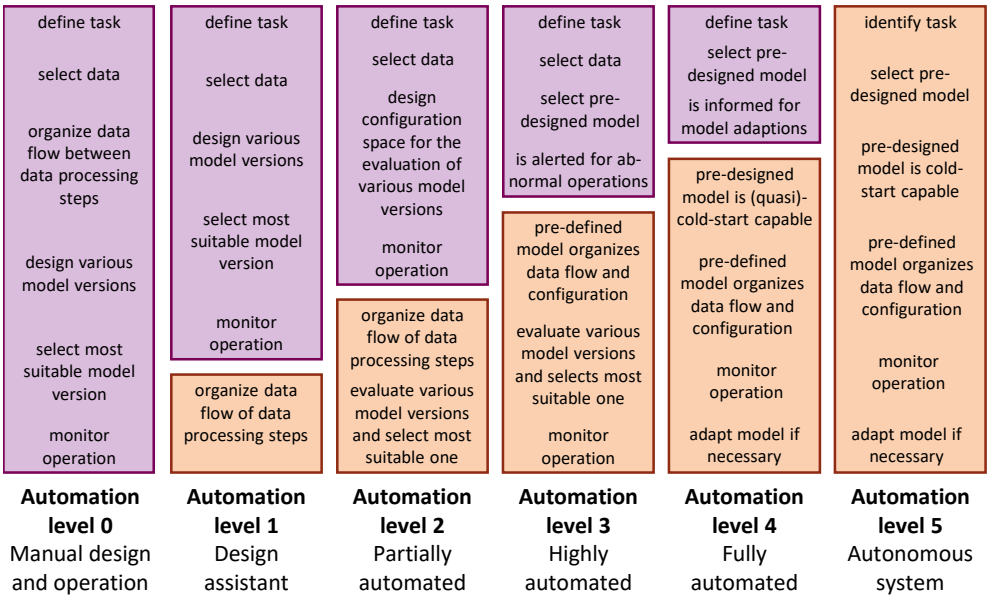


Figure 2.1: The six levels of the automation level taxonomy AutoLVL for time series forecasting [162], inspired by the SAE standard for autonomous driving of vehicles [170]. The purple bars indicate manual steps and the orange bars automated steps in the workflow.

¹ Note that this chapter is based on [1, 162, 163, 165–169] and contain identical phrases; a detailed list is given in the Appendix.

AutoLVL defines the six levels of automation in time series forecasting, where manual design and operation reflect **Automation level 0**. That is, any type of Machine Learning Operations (MLOps) for real-world applications inevitably requires a higher level of automation to fulfill its requirements.

In **Automation level 1**, the workflow is systematized by organizing the data processing steps and managing the data flow between the steps. Systematizing the workflow is necessary, as many data processing steps are generalizable, such as accounting for seasonality and adding calendar-based features, which are both often useful in the forecasting of energy data [171–173]. Still, the model design is manually performed by a data scientist to meet task-specific requirements.

Automation level 2 enables partially automated forecasting, where Automated Machine Learning (AutoML) methods support the data scientist in the model design for tailoring the model to the specific task. Tailoring is crucial as the forecast quality is sensitive to various design decisions [20], and the forecast’s downstream application [24]. Still, the data scientist needs to define the configuration space of automation methods and, after the model design, monitor the model during operation.

Automation level 3 reaches highly automated forecasting by automatically designing the model and monitoring the forecast during operation. By limiting the scope of a forecasting model to a specific task, domain knowledge can be integrated, e. g., the Prior Knowledge (PK) of physical limitations of the underlying system [174]. Thus, the data scientist only needs to select a pre-designed model – a *forecasting template* – and provide the data. Subsequently, the operation is monitored and the data scientist is alerted when suspicious model input or output data is detected. Apart from selecting training data, the data scientist must take action in the operational phase if the model performance degrades.

In **Automation level 4**, the problem of missing training data (cold-start) or limited training data (quasi-cold-start) for the model design is addressed, and in the operational phase, the model is automatically adapted if necessary. The former is essential for the scalable deployment of forecasts with growing needs, e. g., as with the expansion of locally distributed PhotoVoltaic (PV) plants [168]. The latter ensures timely adaption to changing conditions, e. g., in the case of drifting electricity demand, as experienced during the Covid-19 pandemic [175]. To ensure effective adaption to a changing condition, it becomes necessary to select the set of all available training data that represents this condition.

Finally, **Automation level 5** achieves a fully autonomous system that independently identifies the task (e. g. day-ahead PV forecasting) based on the downstream application’s request), creates the forecasting model, and detects and resolves issues during operation. This corresponds to the situation where artificial intelligence reaches human intelligence

and data scientists are no longer required. A first step toward Automation level 5 would be domain-specific autonomous systems before extending to a non-domain-specific system.

The above-introduced automation levels of AutoLVL are built on each other, i. e., to reach a higher level, all previous levels must be achieved, see Figure 2.1. Although it would be possible to fulfill only certain aspects of a specific automation level, the fulfillment of previous levels' requirements is obligatorily for an efficient realization of the next higher level.

To achieve the proposed automation levels, different automation methods are required that can be combined. The automation methods used in this thesis are introduced in the following for the Automation levels 1 to 4 with illustrative examples.² These illustrative examples are based on the mobility data set [176] using Gradient Boosting Machine (GBM) and MultiLayer Perceptron (MLP) estimators, as these are universal estimators that achieve good forecasting performance while requiring moderate computational effort [177, 178]. Both estimators are implemented with the Python package Scikit-learn [100]. Accordingly, we refer to the default hyperparameters using the default values of these implementations.

The actual usage of the automation methods introduced in the following is summarized in Section 2.8, which gives an outline of the developed forecasting templates in this thesis that are detailed in Chapter 3, 4, and 5. AutoLVL not only serves as a guideline for the development of these forecasting templates but also supports their implementation, as shown in Section 2.5 with the definition of a generic web service. Such web services are necessary to serve downstream smart grid applications, as demonstrated in Chapter 6.

2.1 Methods of Automation level 1

To assist the design process of forecasting models in smart grid applications, an interface is required that systematizes the workflow and enables generalization of repetitive steps.

Structuring data processing steps as a workflow is achieved through a forecasting pipeline. The forecasting pipeline represents the workflow as a graph, as many workflows in time series forecasting are non-sequential, where the graph nodes represent the data processing steps and the graph edges represent the data flow between steps. Each data processing step includes a method that either only transforms the data or a method that must be fitted using training data before making transformations. Running the created forecasting pipeline executes all included data transformation steps and trains the data fitting steps, e. g., a Linear Regression (LR), with historical time series data, resulting in a fitted forecasting model.

² Importantly, as AutoLVL defines the scope of automation and not the solution, the methods described for the Automation levels 1 to 4 do not claim to be the only possibility to reach a specific automation level.

Thereby, the forecasting pipeline organizes the data processing steps and manages the data flow through the steps' methods.

For creating graphical forecasting pipelines, we use the open-source Python Workflow Automation Tool for Time Series (pyWATTS) [171]. It has a uniform interface for various methods applied to the data processing steps of the pipeline and enables seamless integration of existing Machine Learning (ML) algorithms from Scikit-learn [100] or Deep Learning (DL) models implemented in Keras [179] and PyTorch [180]. Furthermore, it provides built-in functions for pre-processing such as missing value handling (see Section 1.3.1) and feature extraction including lag features and cyclic features. Lag features provide historical context, i. e., values at time point k that date back a certain time horizon H_1

$$x_{\text{lag}, H_1} [k] = x [k - H_1], \quad (2.1)$$

and cyclic features provide seasonal information, e. g., with a trigonometrical encoding of the month, the day of year, the hour or the minute of the day:³

$$x_{\sin, 12} [k] = \sin \left(\frac{2\pi \cdot \text{month} [k]}{12} \right), \quad x_{\cos, 12} [k] = \cos \left(\frac{2\pi \cdot \text{month} [k]}{12} \right), \quad (2.2)$$

$$x_{\sin, 365} [k] = \sin \left(\frac{2\pi \cdot \text{day} [k]}{365} \right), \quad x_{\cos, 365} [k] = \cos \left(\frac{2\pi \cdot \text{day} [k]}{365} \right), \quad (2.3)$$

$$x_{\sin, 24} [k] = \sin \left(\frac{2\pi \cdot \text{hour} [k]}{24} \right), \quad x_{\cos, 24} [k] = \cos \left(\frac{2\pi \cdot \text{hour} [k]}{24} \right), \quad (2.4)$$

$$x_{\sin, 1440} [k] = \sin \left(\frac{2\pi \cdot \text{minute} [k]}{1440} \right), \quad x_{\cos, 1440} [k] = \cos \left(\frac{2\pi \cdot \text{minute} [k]}{1440} \right). \quad (2.5)$$

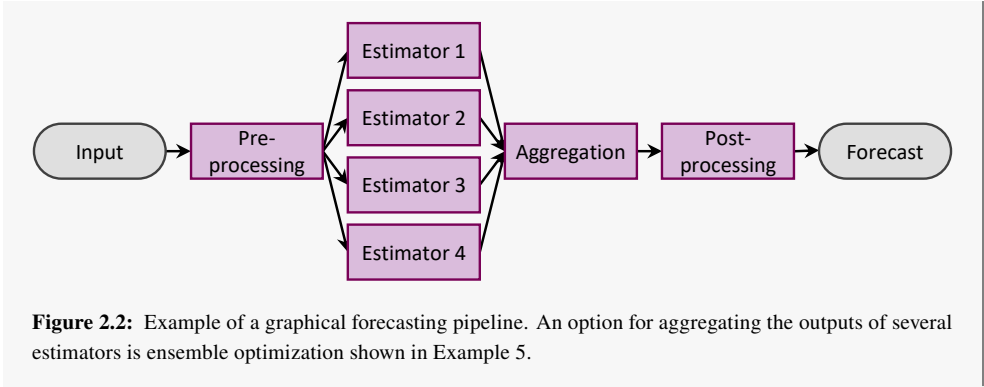
Furthermore, work and non-work days can be encoded through an ordinal encoding:

$$x_{\text{ord}} [k] = \begin{cases} 1, & \text{if work_day}(k) \text{ is True,} \\ 0 & \text{otherwise.} \end{cases} \quad (2.6)$$

Example 1: Graphical forecasting pipeline

The composition of a graphical forecasting pipeline is shown in this example. Graphical forecasting pipelines allow that different data processing steps may share the same predecessor or provide their outputs to the same successor. That is, the data flows can branch and merge, as shown in Figure 2.2.

³ The counting for each feature starts with one. In leap years, (2.3) takes 366 values into account.



2.2 Methods of Automation level 2

To support data scientists and to automatically select the optimal hyperparameter configuration of the forecasting pipeline, automation methods are required. Depending on the specific forecasting task, different automation methods are suitable. In the forecasting tasks in this thesis, HyperParameter Optimization (HPO), successive halving, Combined Algorithm Selection and Hyperparameter optimization (CASH), and ensemble optimization are used.

2.2.1 Hyperparameter optimization

As detailed in Section 1.3.3, forecasting methods include a wide range of hyperparameters that influence the forecast quality, and tailoring the hyperparameter configuration λ to the specific task using HPO may improve the forecast quality over the default configuration.

Instead of manually searching for a λ that leads to a forecast with the desired properties, optimization methods can be used. The optimization method searches for the (estimated) optimal hyperparameter configuration λ^* within a given configuration space Λ for a given objective function Q based on the forecasts \hat{y} and the realized values y of a validation data set:

$$\lambda^* = \min_{\lambda \in \Lambda} Q(\hat{y}(\lambda), y). \quad (2.7)$$

Consequently, HPO can be applied to an arbitrary Q , but the selection of an appropriate optimization method depends on the forecasting task's characteristics and available computing resources.

Since the vast majority of HPO problems are black-box optimization problems, we only consider optimization methods based on Evolutionary Algorithms (EAs) and Bayesian Optimization (BO):

If training the forecasting model with a Graphics Processing Unit (GPU) is advantageous compared to Central Processing Units (CPUs) and multiple GPUs are available, it is possible to evaluate multiple trials of Q in parallel. In this situation, we use the Python package Propulate [181], an asynchronous EA for global optimization and HPO on High-Performance Computing (HPC) systems over configuration spaces that may include real-valued, and discrete dimensions. The EA allows independent processing workers by using the island model with asynchronous propagation of populations and asynchronous migration. More specifically, and in contrast to classical synchronous Genetic Algorithms (GAs), Propulate maintains a continuous population of already evaluated hyperparameter configurations with a softened notion of the typically strictly separated, discrete generations. Based on this continuous population, new candidates of λ_p are generated based on mechanisms from biological evolution, such as selection, recombination, and mutation.

Otherwise, if the forecasting model is trained with CPUs, we evaluate trials of Q sequentially using the Python package Hyperopt [182], a BO algorithm for sequential HPO over configuration spaces that may include real-valued, discrete, and conditional dimensions.

Example 2: Hyperparameter optimization

The present example uses the mobility data set [176] to show the HPO of a forecasting method. In the forecasting pipeline, we consider a GBM estimator, see Figure 2.3.

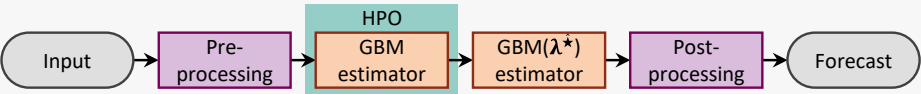


Figure 2.3: Example of HPO of the estimator within a forecasting pipeline. While this example focuses on HPO, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1).

In the HPO, we consider the configuration space Λ shown in Table 2.1.

Table 2.1: HPO configuration space Λ of the GBM forecasting method in Scikit-learn [100] naming convention.

Hyperparameter	Value range	Default value
learning_rate	[0.01, 1]	0.3
n_estimators	[10, 300]	100

The HPO objective is to find the optimal hyperparameter configuration λ^{\star} that minimizes the Mean Squared Error (MSE) (1.6):

$$\lambda^{\star} = \min_{\lambda \in \Lambda} \text{MSE}(\hat{y}(\lambda), y). \tag{2.8}$$

A trial of the objective function includes initializing the GBM with λ , training the GBM on a training data set, and evaluating the MSE on a validation data set. To solve the HPO problem we use BO with the Tree Parzen Estimator (TPE) as surrogate model, implemented with the Python package Hyperopt [182]. Figure 2.4 shows the assessed trials of the objective function.

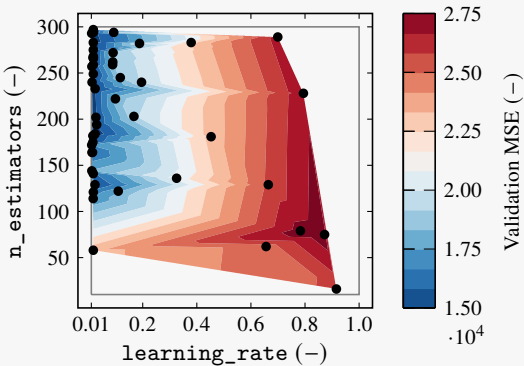


Figure 2.4: The assessed trials on the validation data set in the HPO. The limits of the configuration space (Table 2.1) are visualized as gray lines, the assessed configurations are represented by black dots, and the color contour is calculated by linear interpolation on the triangular grid spanned by these dots. In this example, HPO stops after 100 iterations, leaving unexplored areas, where BO-TPE chose to exploit areas near already-evaluated high-performing configurations rather than to explore.

The figure shows that Hyperopt balances between exploration of unknown areas and exploitation of promising areas in the two-dimensional Λ . After the HPO, we evaluate the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) on the hold-out test data set, with the results shown in Table 2.2.

Table 2.2: The forecast error of the GBM’s optimized configuration λ^{\star} compared its default configuration λ_{default} . The error metrics are calculated on the hold-out test data set and are averaged over five HPO runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).

	HPO	Default	Improvement	p-value
MAE	86.6±2.3	98.8±0.0	12.4 %	0.0003
RMSE	138.7±3.5	155.1±0.0	10.5 %	0.0005

The optimal hyperparameter configuration λ^{\star} improves both error metrics in this example significantly compared to the default configuration λ_{default} .

2.2.2 Combined algorithm selection and hyperparameter optimization

Not only optimizing hyperparameters of a forecasting method but also selecting the appropriate method is crucial for the forecast quality. Consequently, the forecasting method selection is often combined with an individual HPO, i. e., selecting the best forecasting method from a pool of candidates whose hyperparameters, in turn, are optimized.

For combining both, we define a CASH problem. In the CASH problem, we aim to minimize the objective function Q (2.7) by selecting the optimal configuration λ^* of the configuration space Λ . Compared to HPO, however, CASH selects the optimal algorithm and corresponding optimal hyperparameters simultaneously. Consequently, Λ includes conditional dimensions as the selection of a forecasting algorithm conditions the associated hyperparameters.

Based on the recommendations of Feurer and Hutter [112], we thus use BO with the TPE as surrogate model that estimates the distribution of well-performing configurations in relation to underperforming ones for exploring Λ [183]. BO-TPE is beneficial for CASH because it enables efficient exploration of the high-dimensional and conditional configuration space, balances exploration and exploitation, and can handle noisy, non-convex and non-differentiable objective functions.⁴

Example 3: Combined algorithm selection and hyperparameter optimization

The present example uses the mobility data set [176] to show the CASH of a forecasting method. In the forecasting pipeline, we consider the GBM and the MLP estimator, see Figure 2.5.

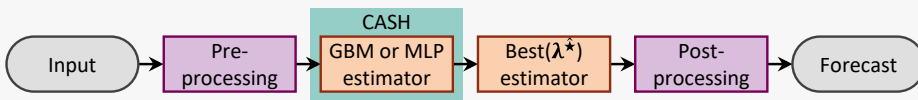


Figure 2.5: Example of CASH within a forecasting pipeline. While this example focuses on CASH, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1).

In the CASH, we consider the configuration space Λ shown in Table 2.3.

⁴ Formulating a CASH problem using the EA-based Propulate implementation, which currently does not support configuration spaces with conditional dimensions, would lead to many unused genes and slow convergence. In Example 3, the estimator selection could be implemented as categorical parameter, and the four hyperparameters as continuous and ordinal parameters, respectively. However, if the categorical parameter of an individual is the GBM, a modification of the batch size and the number of neurons has no effect.

Table 2.3: CASH configuration space Λ of the GBM and the MLP in Scikit-learn [100] naming convention. The batch size of the MLP estimator is chosen between $2^i, i \in \{3, 4, \dots, 9\}$.

Regression estimator	Hyperparameter	Value range	Default value
MLPRegressor	batch_size	{8, 16, ..., 512}	min(200, n_samples)
	n_neurons	[10, 300]	100
GradientBoosting - Regressor	learning_rate	[0.01, 1]	0.3
	n_estimators	[10, 300]	100

As in the HPO (Example 2), the CASH objective is to find the optimal configuration λ^\star that minimizes the MSE (1.6). A trial of the objective function (2.8) for CASH includes selecting the forecasting method based on λ , initializing it with the corresponding conditional hyperparameters, training the forecasting method on a training data set, and evaluating the MSE on a validation data set. To solve the CASH problem we use BO-TPE, implemented with the Python package Hyperopt [182]. Figure 2.6 shows the assessed trials of the objective function.

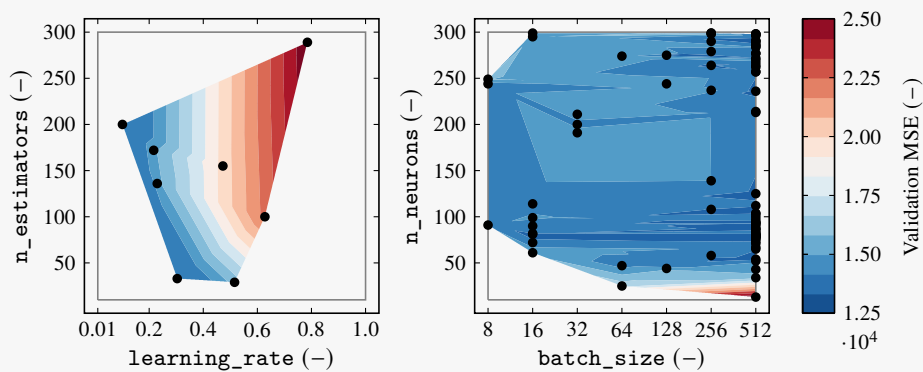


Figure 2.6: The assessed trials on the validation data set in the CASH using BO-TPE. The limits of the configuration space (Table 2.3) are visualized as gray lines, the assessed configurations are represented by black dots, and the color contour is calculated by linear interpolation on the triangular grid spanned by these dots. In this example, CASH stops after 100 iterations, leaving unexplored areas, where BO-TPE chose to exploit areas near already-evaluated high-performing configurations rather than to explore.

The figure shows that BO-TPE balances between exploration of unknown areas and exploitation of promising areas in the Λ , which consists of two choices and two dimensions per choice. As the optimization is terminated after 100 iterations, unexplored areas remain that may contain higher-performing configurations. After the CASH, we evaluate the MAE and the RMSE on the hold-out test data set, with the results shown in Table 2.4.

Table 2.4: The forecast error of the optimized configuration λ^* compared to the default configurations λ_{default} of the GBM and the MLP. The error metrics are calculated on the hold-out test data set and are averaged over five CASH runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).

	CASH		Default	Improvement	p-value
MAE	73.5 \pm 0.4	GBM	98.8 \pm 0.0	25.6 %	0.0000
		MLP	75.5 \pm 2.2	2.7 %	0.1159
RMSE	107.2 \pm 0.8	GBM	155.1 \pm 0.0	30.9 %	0.0000
		MLP	110.0 \pm 2.5	2.5 %	0.0821

The optimal configuration λ^* improves both error metrics in this example significantly compared to the GBM’s default configuration λ_{default} , while the improvement over the MLP’s default configuration is not significant. Note that CASH automatically selects the estimator that is more promising and optimizes its hyperparameters.

2.2.3 Successive halving

In both HPO and CASH, exploring and exploiting large configuration spaces demand a high amount of computing resources. Pruning strategies are therefore required to re-allocate computing resources from unpromising areas in the configuration space to promising areas.

Different pruning strategies exist that either abort long-running and poorly-evolving trails or narrow down unpromising search directions. The former requires the trial configuration to report about its progress, e. g., the performance of an MLP on a validation data set in the current training epoch. The latter does not require reporting the trials at runtime and can also process the final results of the trials only, e. g., the performance of an MLP on a validation data set after the training is completed.

In this dissertation, shrinking the size of the configuration space in CASH over the runtime with a total time budget B_t is applied. More specifically, the successive halving strategy is used to remove unpromising forecasting methods from the configuration space and concentrate the computing resources on the remaining ones, as outlined in Algorithm 1. The algorithm starts with initializing the result store for $\Lambda = \Lambda_1 \times \Lambda_2 \times \dots \times \Lambda_{N_p}$, a configuration space with choices for N_p considered forecasting methods and different conditional hyperparameters per choice. In each pruning round, the following applies:

- The computation budget per forecasting method is determined $B_{t,i} = \frac{B_t}{N_p}$, where N_p is the number of remaining forecasting methods, i. e., the dimension of Λ .

- Each remaining dimension of Λ is an independent objective function Q (2.7) that is evaluated in parallel using an HPO algorithm running independently until $B_{t,i}$ is exhausted.
- The results store is updated with the HPO results of the current round and the worse half of forecasting methods are removed from Λ . Importantly, the HPO of a remaining dimension is not restarted in each pruning round but continues running from the checkpoint of the previous round.

That is, each pruning round consists of the set of remaining forecasting methods, and the respective configuration spaces grouped together for competitive evaluation within an allocated time budget. These pruning rounds are repeated until only one forecasting method remains and the total time budget B_t is exhausted. In this way, promising forecasting methods receive more computing time for the HPO, while underperforming methods drop out early.

Algorithm 1 The successive halving pruning strategy to re-allocate computing resources from unpromising areas in the CASH configuration space to promising areas.

Input: $\Lambda, B_t, \mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}$

```

1: result_store( $\Lambda$ ) ▷ Initialize result store
2: while  $\text{len}(\Lambda) > 1$  do
3:   for  $\Lambda_i$  in  $\Lambda$  do ▷ Parallel evaluation
4:      $B_{t,i} \leftarrow B_t / \text{len}(\Lambda)$  ▷ Split computation budget
5:     hpo_algorithm(result_store) ▷ Initialize HPO algorithm with stored results
6:     hpo_algorithm.run( $\Lambda_i, \mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}, B_{t,i}$ ) ▷ Run until budget is exhausted
7:      $Q_i^{\hat{\star}}, \lambda_i^{\hat{\star}} \leftarrow \text{hpo\_algorithm.get\_best\_configuration}(\Lambda_i)$ 
8:     result_store.update( $Q_i^{\hat{\star}}, \lambda_i^{\hat{\star}}$ ) ▷ Update result store
9:   end for
10:   $\Lambda \leftarrow \text{result\_store.get\_best\_half}()$  ▷ Shrink configuration space
11: end while

```

Output: $\lambda^{\hat{\star}}$

Example 4: Successive halving

The present example uses the mobility data set [176] to show the CASH of a forecasting method using the successive halving strategy. As in the previous Example 3, we consider the GBM and the MLP estimator in the forecasting pipeline. Additionally, we consider the Random Forest (RF), the Ridge, and the Support Vector Regression (SVR) estimator, see Figure 2.7.

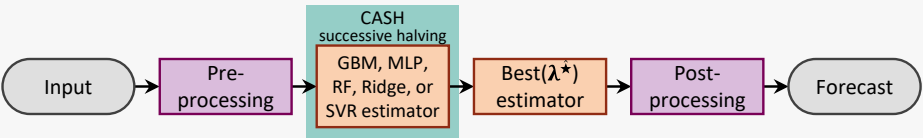


Figure 2.7: Example of CASH using the successive halving strategy within a forecasting pipeline. While this example focuses on successive halving, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1).

In the CASH using successive halving, we consider the configuration space Λ shown in Table 2.5.

Table 2.5: CASH configuration space Λ in Scikit-learn [100] naming convention. The batch size of the MLP estimator is chosen between 2^i , $i \in \{3, 4, \dots, 9\}$.

Regression estimator	Hyperparameter	Value range	Default value
MLPRegressor	batch_size	{8, 16, ..., 512}	min(200, n_samples)
	n_neurons	[10, 300]	100
GradientBoosting - Regressor	learning_rate	[0.01, 1]	0.3
	n_estimators	[10, 300]	100
RandomForest - Regressor	max_depth	[1, 10]	None
	n_estimators	[10, 300]	100
Ridge	alpha	[0.05, 1]	1.0
MSVR	C	[1e-2, 1e2]	1.0
	epsilon	[1e-3, 1.]	0.1

As in the previous Example 3, the CASH objective is to find the optimal configuration λ^* that minimizes the MSE (1.6). Now, the successive halving strategy (Algorithm 1) is used to shrink the size of the configuration space over the number of iterations. Figure 2.8 shows the evolution of the MSE on the validation data set with the iteration budget $B_i = 50$.

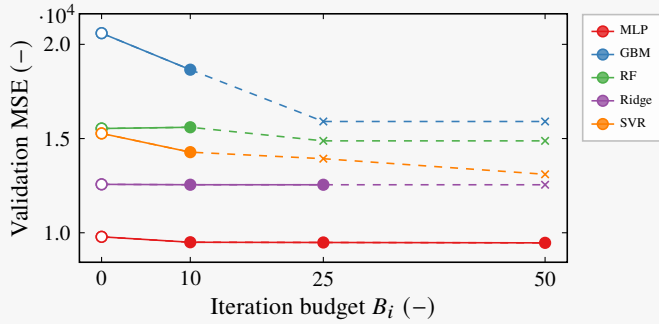


Figure 2.8: Evolution of the considered estimators' validation performances in the successive halving-based CASH with the performance at 0 iterations reflecting the default configurations, shown as a reference and therefore not considered within the iteration budget B_i . The solid lines with dot markers show the performance of active methods, while the dotted lines with cross markers show the performance of inactive methods as it would have evolved without pruning. The progress in the successive halving pruning rounds is assumed to be linear for simplicity's sake.

The figure shows that the successive halving strategy effectively removes unpromising estimators from the configuration space and concentrate the computing resources on the remaining ones. Since the optimal estimator is the MLP, the improvement of CASH is comparable to Example 3. However, three additional estimators were taken into account and the CASH was performed in parallel instead of sequentially.

2.2.4 Ensemble optimization

As detailed in Section 1.3.4, the idea of creating an ensemble is to increase the robustness of models by combining multiple outputs from a pool of different models.

Apart from weighting each model in the ensemble pool equally (averaging), one may give more weight to models from which we expect good performance and give less weight to models from which we expect poor performance, i. e.,

$$\hat{y}[k] = \sum_{n=1}^{N_m} \hat{w}_n \cdot \hat{y}_n[k] \quad (2.9)$$

is a linear combination with $\hat{y}_n[k]$ and \hat{w}_n , $n \in \mathbb{N}_1^{N_m}$ being the output and weight of the n -th model in the ensemble pool.⁵ To estimate the expected future performance of the models in the ensemble model pool, we search for $\hat{\mathbf{w}}$ that minimize the objective function Q based

⁵ Alternatives for the transition of model contributions are, e. g., fuzzy clustering [184] or validity domain-based transitions as in [185, 186].

on the forecasts \hat{y}_n of the ensemble pool models and the realized values y of a validation data set. With (2.9), this results in the optimization problem:

$$\begin{aligned} \min_{\hat{\mathbf{w}}} Q(\hat{y}_n, y) \quad n \in \mathbb{N}_1^{N_m} \quad \text{subject to} \\ \hat{w}_n \in [0, 1] \quad \forall n \in \mathbb{N}_1^{N_m}, \quad \sum_{n=1}^{N_m} \hat{w}_n = 1. \end{aligned} \quad (2.10)$$

We solve (2.10) using the Python package SciPy [187] with $\hat{w}_n = 1/N_m$, $\forall n \in \mathbb{N}_1^{N_m}$ as initial values, and normalize the weights afterward to hold the constraints. The selection of the optimization method depends on the objective function's Q characteristics. If Q is convex, continuous, smooth, and can be formulated as the sum of squared errors or residuals, the least squares method is suitable. Otherwise, if Q is non-convex, discontinuous, noisy, or lacks a smooth gradient, global optimization methods like the Differential Evolution Algorithm (DEA) are better suited, as they are less likely to get trapped in local minima.

Example 5: Ensemble optimization

The present example uses the mobility data set [176] to show the optimization of an ensemble of forecasting models. As ensemble pool models, we consider two GBM and two MLP estimators with different hyperparameter configurations as shown in Figure 2.9.

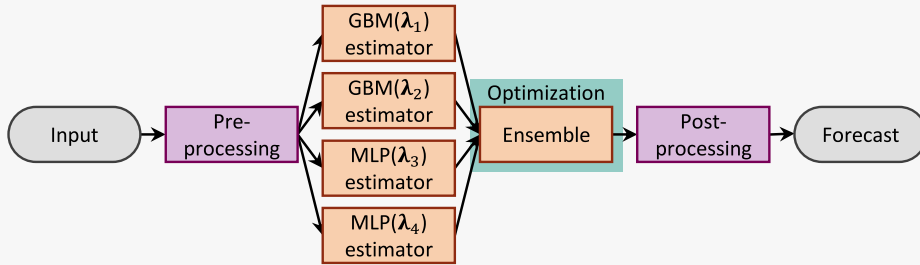


Figure 2.9: Example of ensemble optimization within a forecasting pipeline. While this example focuses on ensemble optimization, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1).

The hyperparameters of the two GBM and the two MLP estimators are shown in Table 2.6 and are manually selected around the respective default values (Table 2.3).

Table 2.6: Hyperparameter configurations λ of the two GBM and the two MLP estimators in the ensemble model pool in Scikit-learn [100] naming convention.

Hyperparameter	Values of λ_1	Values of λ_2	Values of λ_3	Values of λ_4
learning_rate	0.5	0.1	-	-
n_estimators	50	150	-	-
batch_size	-	-	128	256
n_neurons	-	-	50	150

In the optimization of the ensemble weights $\hat{\mathbf{w}}$ (2.9), the objective is to minimize the MSE (1.6):

$$\begin{aligned} \min_{\hat{\mathbf{w}}} \text{MSE}(\hat{y}_n, y) \quad n \in \mathbb{N}_1^{N_m} \quad \text{subject to} \\ \hat{w}_n \in [0, 1] \quad \forall n \in \mathbb{N}_1^{N_m}, \quad \sum_{n=1}^{N_m} \hat{w}_n = 1. \end{aligned} \quad (2.11)$$

The ensemble optimization includes training the four ensemble pool models on a training data set, and optimizing the ensemble weights $\hat{\mathbf{w}}$ to minimize the MSE on a validation data set. The estimated ensemble weights $\hat{\mathbf{w}}$ together with an exemplary ensemble forecast $\hat{\mathbf{y}}$ are shown in Figure 2.10.

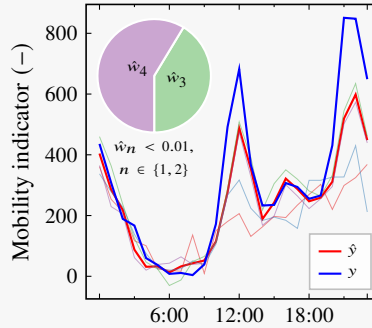


Figure 2.10: Exemplary 24 h-ahead forecast using ensemble optimization. Both MLP models receive high weights, whereas the weights of the GBM models are below 1 %. The forecasts of the ensemble pool models \hat{y}_n , $n \in [1, 2, 3, 4]$ are shown opaquely.

After the ensemble optimization, we evaluate the MAE and the RMSE on the hold-out test data set, with the results shown in Table 2.7.

The ensemble optimization improves the MAE in this example significantly compared to the GBM's and the MLP's default configuration λ_{default} , while the improvement of the RMSE is not significant for the MLP. For the hyperparameter values of the default configurations, see Table 2.3.

Table 2.7: The forecast error of the ensemble forecast compared to the default configurations λ_{default} of the GBM and the MLP. The error metrics are calculated on the hold-out test data set and are averaged over five ensemble optimization runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).

	Ensemble	Default	Improvement	p-value	
MAE	72.4±1.2	GBM	98.8±0.0	26.7 %	0.0000
		MLP	75.5±2.2	4.2 %	0.0465
RMSE	106.5±1.3	GBM	155.1±0.0	31.3 %	0.0000
		MLP	110.0±2.5	3.1 %	0.0515

2.3 Methods of Automation level 3

For recurring forecasting tasks in smart grids, specific forecasting templates can be defined. More specifically, limiting the scope of a forecasting template enables integrating domain-specific knowledge. The forecasting templates proposed in this thesis leverage Prior Knowledge (PK) for feature engineering and post-processing, use pre-trained models and apply performance monitoring to detect suspicious forecasting model behavior during operation.

2.3.1 Prior knowledge

Prior Knowledge (PK) can be used for feature engineering and post-processing to incorporate the understanding and expertise of a data scientist about the domain of the forecasting task. It can significantly impact the success of forecasting pipelines by enhancing their ability to extract meaningful patterns and increase the forecast quality.

PK-based feature engineering involves knowing which features are relevant and how to transform or create new features to improve forecast quality. PK-based post-processing includes knowing physical limitations and transforming the forecast accordingly to ensure physical consistency.⁶

When designing a forecasting template for a specific task in this thesis, a robust default set of features is defined and the output of the forecasting pipeline is post-processed to ensure physically consistent forecasts.

Example 6: Prior knowledge

The present example uses the mobility data set [176] to show the use of PK in time series forecasting. In the forecasting pipeline, we consider an MLP estimator with default configuration (for the hyperparameter values, see Table 2.3), and use PK in the feature engineering and the post-processing step, see Figure 2.11.

⁶ Apart from explicit post-processing, PK can be integrated into the training process such the model implicitly learns to hold constraints (physics-guided ML, e. g., [174, 188]).

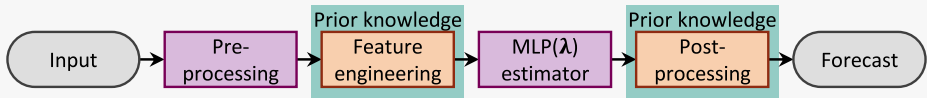


Figure 2.11: Example of applying PK within a forecasting pipeline. While this example focuses on PK, the pre-processing includes data scaling (Section 1.3.1), and the MLP estimator’s configuration may be optimized using HPO (Section 2.2.1).

In the feature engineering, we consider weather features from an exogenous forecast as well as information about holidays, knowing that bicycles are more likely to be ridden on warm, sunny days than on rainy, cold days, and mobility cycles are related to work days. If no PK for feature engineering exists, a correlation analysis can be applied. With the correlation analysis, irrelevant features can be screened out, i. e., the estimator only receives a sub-set of relevant features. In the post-processing, we set negative values in the model output to zero, knowing that a negative number of active bike rentals is not possible. An exemplary forecast considering PK \hat{y} on the hold-out test data set is shown in Figure 2.12, and Table 2.8 compares the calculated error metrics MAE and RMSE.

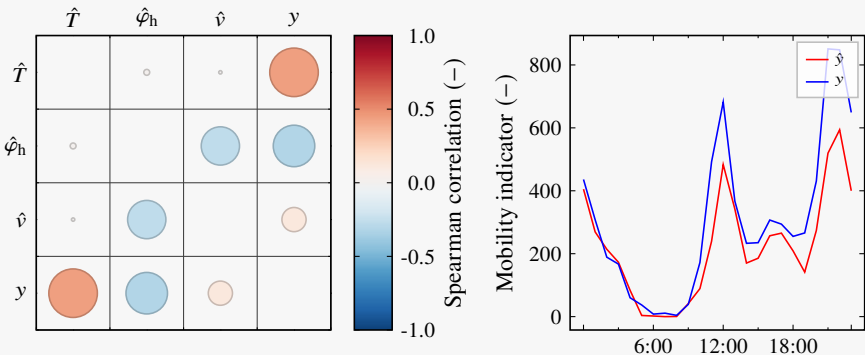


Figure 2.12: Correlation of air temperature \hat{T} , relative humidity $\hat{\varphi}_h$, and wind speed \hat{v} of an exogenous forecast $\hat{\mathbf{X}}$ to the target time series \mathbf{y} and exemplary 24 h-ahead forecast using these features together with post-processing.

Table 2.8: The forecast error of the MLP forecaster using PK for feature engineering and post-processing compared to not using PK. The error metrics are calculated on the hold-out test data set and are averaged over five training runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).

	PK	No PK	Improvement	p-value
MAE	71.8±1.8	75.5±2.2	5.0 %	0.0417
RMSE	106.2±2.4	110±2.5	3.4 %	0.0738

Using PK for feature engineering and post-processing improves the MAE in this example significantly compared to not using PK, while the improvement of the RMSE is not significant.

2.3.2 Pre-trained models

Pre-trained models are models that have been trained on a sufficiently large and related data set before being selected or fine-tuned (transfer learning) for specific forecasting tasks.⁷ By selecting or fine-tuning a pre-trained model based on a smaller, task-specific data set, better forecast quality can be achieved due to the prior generalization to the large related data set, reducing the risk of overfitting.

In this thesis, pre-trained models are used for creating the model pool of the ensemble optimization (Section 2.2.4). Since the weights are adaptable online, cold-start problems can be addressed. That is, if no data of the target time series are available, each model in the ensemble pool has an equal contribution to the forecast. During forecast operation, data of the target time series that becomes available can be used to adapt the ensemble weights. In this way, the weights are shifted toward models that are representative for the target time series, and the selection of the pre-trained models is automated.

The pre-trained models used in the forecasting templates leverage Prior Knowledge (PK) and are automatically designed using CASH. In addition to the pre-trained models, we use models based on Physical-inspired Modeling (PM) that do not require training.

Example 7: Pre-trained models in the ensemble optimization

The combination of pre-trained models and the ensemble optimization is shown in this example using the mobility data set [176]. As ensemble pool models, we consider four MLP models, each trained on a different data set that is related to the smaller task-specific data set. Thus, the ensemble pool consists of expert models, representing the related data sets. Note that the related data sets were artificially generated by applying various transformations to the target time series.

As in Example 5, the objective of the ensemble optimization (2.11) is to minimize the MSE (1.6) evaluated on the task-specific data set by varying the ensemble weights $\hat{\mathbf{w}}$ (2.9). The estimated ensemble weights $\hat{\mathbf{w}}$ together with the ensemble forecast $\hat{\mathbf{y}}$ and the forecasts of the ensemble pool models $\hat{\mathbf{y}}_n$, $n \in [1, 2, 3, 4]$ are shown in Figure 2.13.

⁷ Alternatively, related data can be generated synthetically [107, 189] to be used for pre-training models [190].

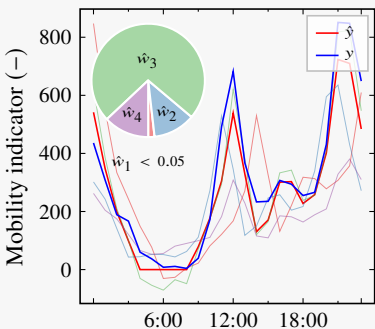


Figure 2.13: Exemplary 24 h-ahead forecast using pre-trained models in the ensemble pool.

The ensemble optimization allows to apply the learned behavior of the large and related data sets to the smaller task-specific data set, which enables solving overfitting problems that may occur with small data sets and cold-start problems.

2.3.3 Performance monitoring

Performance monitoring is a sub-set of anomaly detection⁸ and is required to detect suspicious forecasting model performance during operation, e. g., when the model exceeds the limits of familiar and learned behavior.

An established drift detection method is the ADaptive WINdowing (ADWIN) algorithm [198]. ADWIN maintains an adaptive window of observations during the online operation. The window is adaptive in the sense that the algorithm grows this window as long as there is no drift detected, and it shrinks the window by removing old observations whenever a drift is detected. During operation, the algorithm first adds a new observation to the adaptive window. Then, ADWIN iterates over all possible combinations of two sub-windows, as shown in Figure 2.14.

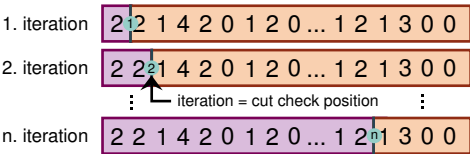


Figure 2.14: Iterations of the cut check procedure with two sub-windows of the ADWIN algorithm [198]; figure is based on [199]. The sequence shown is exemplary and represents the values of an arbitrary time series [169].

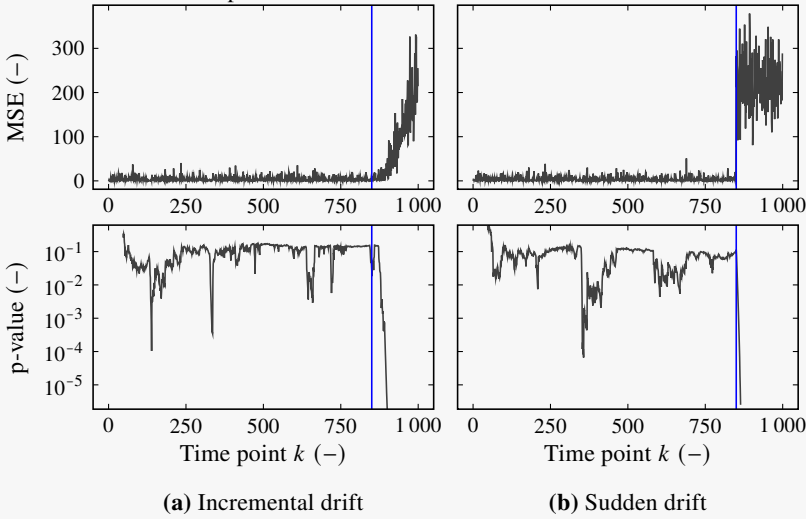
⁸ For example, anomaly detection is used for disturbance detection [191, 192] and intrusion detection [193, 194], or to identify anomalies in training data [195, 196], to be corrected using anomaly handling methods [89, 197].

For each of these combinations, ADWIN tests the null hypothesis that the mean of the observations in both sub-windows is equal with significance value $\alpha_t \in (0, 1)$. If the null hypothesis is rejected, i. e., the observations favor the alternative hypothesis stating that the mean values differ, a drift is detected. If a drift is detected, ADWIN searches for the cut position between the old and the new behavior. More specifically, the oldest observation is removed from the adaptive window, and the sub-window comparison is repeated. ADWIN repeats removing old observations until the observations in the adaptive window no longer indicate a drift. Then, the remaining observations are considered to belong to the same behavior, which is the new behavior.

Regarding performance monitoring, ADWIN can be applied to observe an arbitrary one-dimensional performance measure Q during operation such as the model's rolling MSE. ADWIN is advantageous compared to a fixed threshold, as ADWIN automatically adjusts to the error level during the operation.⁹

Example 8: Performance monitoring

This example shows the monitoring of a forecasting pipeline's accuracy using synthetically generated error time series. We consider an incremental and a sudden increase of the MSE (1.6), shown in Figure 2.15. Below the MSE plots, the minimal p-value of all ADWIN sub-windows for each time step is shown.



⁹ A fixed threshold should be configured based on the performance measure on unseen data, as it is generally higher than the training error.

Figure 2.15: Example of an incremental and a sudden MSE drift at $k = 850$ and corresponding p-values of the ADWIN drift detection.

We see that in the inconspicuous range of the MSE, drops of the p-value exist, which can potentially trigger false positives.^a Furthermore, it can be observed that the incremental increase in MSE is longer undetected than the sudden increase. Since the data was generated synthetically, the actual onset of the drift is known and we can evaluate the detection delay. Figure 2.16 shows the detection delay against the number of false positives for different significance levels α_t .

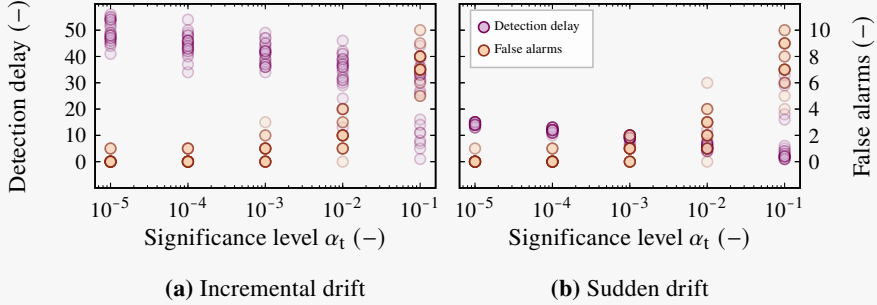


Figure 2.16: Detection delay and false positives depending on the significance level α_t .

For the evaluation, 25 synthetic MSE time series were generated assuming a normal distributed error. We see that α_t affects the detection delay and the number of false positives reciprocally. Additionally, miss detections can occur if a drift is not detected at all.

^a A common metric for evaluating the detection performance over a time series is the F1-score, which rates true positives, false positive, and false negatives [196].

2.4 Methods of Automation level 4

Fully automated forecasting – design and operation – is required to operate forecasting templates in real-world applications. It includes the automated selection of pre-trained models or training data-specific to a forecasting template and automated model adaption during operation. In this thesis, we consider the automated selection of pre-trained models, as well as cyclic model adaption and drift detection-based model adaption. While the former is accomplished in the definition of a forecasting template using domain knowledge (Automation level 3), the latter two are detailed in the following.

2.4.1 Cyclic model adaption

In the cyclic adaption, the model is adapted in a cyclic routine, regardless of the actual model performance.

In this context, there is the cycle length C_{ad} in days and the number of considered samples K_{ad} to adapt the model, which can be varied over time. C_{ad} must be adjusted to the rate of change of the process, and K_{ad} is adjusted to the amount of data required to adapt the model to the change.

In cyclic adaption with fixed C_{ad} and fixed K_{ad} , the old behavior is completely forgotten during model adaption, which we refer to as Adaptive Fixed Batch (AFB). More precisely, the learned behavior of the previous adaption is overwritten during adaption to the new samples. This can be prevented by increasing K_{ad} in each adaption i and thereby including the old behavior ($K_{ad} = i \cdot K_{ad, init}$, $i = [1, 2, \dots]$), which we refer to as Adaptive Increasing Batch (AIB).

Example 9: Cyclic model adaption

The cyclic model adaption is shown in this example. Figure 2.17 shows the cycle length C_{ad} and the number of considered samples K_{ad} in the AFB and the AIB adaption configurations for a time series with the resolution of 15 minutes (96 samples per day). While C_{ad} is the same for both, K_{ad} increases over time for AIB, whereas it remains unchanged for AFB.

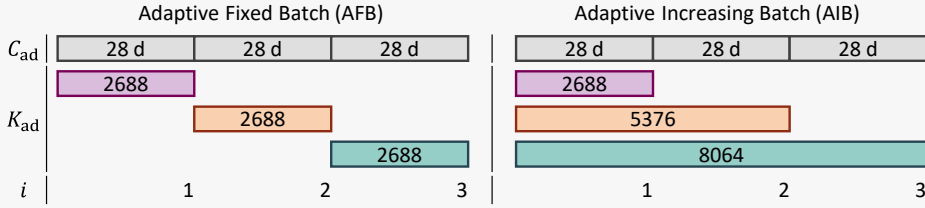


Figure 2.17: Illustration of the cycle length C_{ad} and the number of considered samples K_{ad} in the cyclic model adaption for a time series with the resolution of 15 minutes (96 samples per day). For AIB, K_{ad} increases over time for the number of adaptations $i = 1$ (purple), $i = 2$ (orange), and $i = 3$ (green), whereas it remains unchanged for AFB.

2.4.2 Drift detection-based model adaption

Different from cyclic adaption, drift detection-based adaption aims to adapt the model whenever new behavior is detected in the monitored measure.

A well-known drift detection method that detects the split between the old and the new behavior is the ADWIN algorithm [198], introduced in Section 2.3.3. This split can be used in the adaption to consider only the samples of the new behavior, thus, avoiding the confusion of old and new behavior during training.

As described in Section 2.3.3, ADWIN may be applied to monitor an arbitrary one-dimensional performance measure Q of the model.

Example 10: Drift detection-based adaption

The drift detection-based adaption is shown in this example. Figure 2.18 shows the monitoring of an error measure. After a drift is detected, only samples that belong to the new behavior are considered for the model adaption. Since the detection delay may differ, K_{ad} is not a constant but depend on the detection delay of the drift detection method. Additionally, a minimum required amount of data $K_{ad, min}$ can be defined.

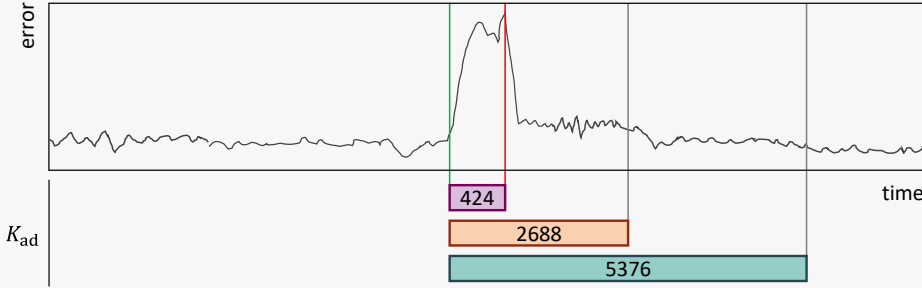


Figure 2.18: Illustration of the parameter K_{ad} in the drift detection-based adaption. Once a drift is detected (red vertical line), the model is adapted with the samples collected after the identified split (green vertical line). While these samples can be used for quick adaption after drift detection (purple), the model can be adapted once more data is collected by combining drift detection-based adaption with the cyclic adaption AIB (orange and green).

2.5 Service implementation

In smart grids, the realization of automated design and operation requires the systematic orchestration of data streams using scalable cloud services with the aim of providing forecasts for downstream applications, which we refer to as forecasting services in the following. To this end, AutoLVL is used to define the generic service architecture in Figure 2.19.¹⁰

Automation level 1 automates the data flow between data processing steps by a forecasting model that receives input data, processes it, and delivers a forecast as output data. In the implementation, the service must therefore provide the data $\{\mathbf{X}, \mathbf{y}\}$, allocate computing resources, and output the results. Due to differing computational efforts required for training a model and making forecasts (transformation), deploying separate services for each operation is recommended. The training service can query training and validation data from the time series database and provide it to the specific forecasting model in the service. Finally, the training service returns the model's performance on training and validation data, and

¹⁰ Note while the generic service architecture based on AutoLVL is the work of this thesis, the implementation of the communication infrastructure and the deployment of the service is the work of Johannes Galenzowski [163].

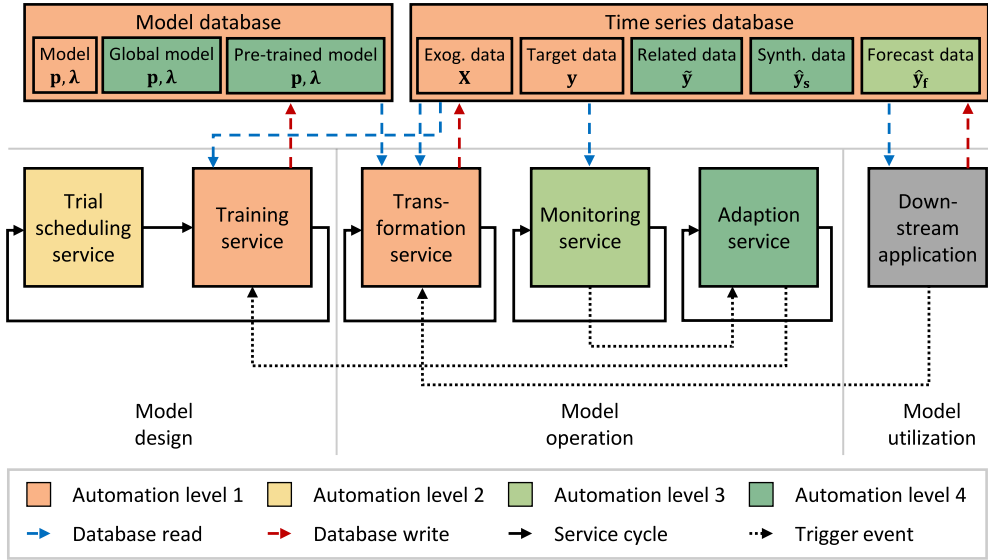


Figure 2.19: Architecture concept for the forecasting service to achieve Automation level 1, 2, 3, and 4 [163].

stores the learned model parameters p . The transformation service loads the stored model parameters p , provides the model with the input data required for making a forecast, and returns the forecast. During operation, the transformation service runs cyclically, depending on the intervals at which the downstream application requires a new forecast.

Automation level 2 enables a partially automated model design by HPO and algorithm selection during training. In the implementation, the training service must therefore be coupled with a trial scheduling service. The trial scheduling service explores and exploits a configuration space Λ given by the data scientist by training various configuration trials and reporting the performance on the validation data. After a stop condition is reached, the service returns a list of the evaluated trials, which contains the configuration λ and their validation performance. The (estimated) best-performing configuration λ^* then persists and is used to make forecasts using the transformation service.

Automation level 3 automatically designs the forecasting model and monitors its operation. In the implementation, pre-designed model templates are required which contain the model structure and its configuration space Λ or robust default configuration λ_{default} . For monitoring the model's operation, a monitoring service is required. The monitoring service monitors either the model's input, output, or both. If the forecast error is to be monitored, the output data must be stored, as the evaluation of the error is only possible with a time delay once the values have been realized. Consequently, the monitoring service must operate independently of the transformation service cycle.

Automation level 4 automatically selects a global or pre-trained model; or selects or generates training data to train a new model. Furthermore, the model is adapted during operation when the models' accuracy declines under a defined threshold. In the implementation, selecting global or pre-trained models requires a model database as described in Automation level 1, and selecting related training data, requires a time series database as described for monitoring in Automation level 3. The latter is also required to continuously collect data. Furthermore, an adaption service is required to trigger model adaption cyclically at pre-defined intervals or redirect an adaption trigger signal of the monitoring service. The data used for adaption is either selected according to a pre-defined rule or depends on the time point of the detected drift. The latter enables, e. g., to only use data that was collected after a new stable operating point was detected.

That is, each sub-service in the architecture of the forecasting service can be attributed to achieving a certain automation level. These sub-services must be coordinated via a superordinate management service, which also handles the interaction with other services. Integrating the forecasting service into a smart grid cloud service ecosystem such as [200] thus requires the alignment of the Application Programming Interface (API).

2.6 Related work

In this section, we evaluate related taxonomies in terms of automating time series forecasting. Besides forecasting, we consider taxonomies for ML, human-machine interactions and autonomous driving. The taxonomies are analyzed in terms of their coverage of automated design and automated operation. Further criteria are the characterization of the human-machine interaction and the definition of levels to describe different scopes of automation. We first focus on taxonomies only for the automated design, then on taxonomies solely for the automated operation, followed by taxonomies that consider both. Finally we review taxonomies that also consider the additional criteria of the human-machine interaction and the levels of automation. The entire analysis is guided by Table 2.9.

Taxonomies for automated design. Apart from the taxonomy for automated forecasting pipelines [1] presented in Section 1.3, which considers the automation of the different sections in the forecasting pipeline as visualized in Figure 1.3, other taxonomies for the automated design in ML exist, also known as AutoML. Chauhan et al. [201] and Mengi et al. [202] also classify the automated design into different pipeline sections, which are pre-processing, feature engineering, model selection and HPO, discussing the automation methods applied in each section. A similar classification is proposed by Karmaker et al. [203], additionally analyzing methods for evaluation and benchmarking. Apart from general

Table 2.9: Taxonomies related to automated time series forecasting [163].

Reference	Taxonomy subject	Automated		Human-machine interaction	Automation levels
		design	operation		
[1]	Forecasting	✓	✗	n. d.	n. d.
[201]	ML	✓	✗	n. d.	n. d.
[202]	ML	✓	✗	n. d.	n. d.
[203]	ML	✓	✗	++	7
[204]	ML	✓	✗	n. d.	n. d.
[205]	ML	✓	✗	++	n. d.
[206]	ML	✓	✗	n. d.	n. d.
[207]	ML	✓	✗	n. d.	n. d.
[208]	ML	✓	✗	++	n. d.
[209]	ML	✗	✓	n. d.	n. d.
[210]	ML	✗	✓	+	n. d.
[211]	ML	✗	✓	+	n. d.
[212]	ML	✗	✓	++	5
[213]	ML	✗	✓	++	5
[214]	ML	✗	✓	++	3
[215]	ML	✗	✓	++	5
[216]	ML	✓	✓	++	n. d.
[217]	ML	✓	✓	++	n. d.
[218]	Generic	✗	✗	++	5
[170]	Autonomous driving	✗	✗	++	6
AutoLVL	Forecasting	✓	✓	++	6
		not defined (n. d.)			

ML, pipeline section-based taxonomies for specialized areas such as recommender systems exist [204]. While the above taxonomies describe which pipeline sections are automated, Yao et al. [205] not only propose such a taxonomy, but also a taxonomy that describes which techniques are used for automation. These are separated into basic techniques, such as optimization and search techniques, and experienced techniques, including meta- and transfer-learning. Another way to define a taxonomy for the automated design is proposed by Bahri et al. [206]. The authors classify the automated design into supervised, unsupervised and semi-supervised learning methods. The possibilities shown above for classifying automated design, i. e., into learning methods, pipeline sections and automation techniques, are combined in one taxonomy by Barbudo et al. [207]. Another perspective on the challenges of automated design is provided by the taxonomy of Wang et al. [208]. The authors identify the challenges for data scientists in the choice of the computing environment, the data pre-processing, the experimental setup, the evaluation of the experiments, and the efficiency of the automated design. However, all these taxonomies only consider the automated design and leave out the operational phase.

Taxonomies for automated operation. A methodological classification for the automated operation of ML pipelines, also called MLOps, is provided by Klaise et al. [209]. The authors classify the methods into monitoring and diagnosis methods. While monitoring includes the observation of the deployed model’s performance, outlier detection, and drift detection, diagnosis includes methods to analyze the factors that contributed to a certain model output. A similar approach is proposed by Eck et al. [211]. The authors describe a monitoring framework that observes the model’s input and output data, as well as the performance of the model. A different taxonomy is proposed by Arnold et al. [210]. The authors suggest to classify the operation according to the life cycle into the pre-deployment test, deployment, monitoring, as well as diagnosis and improvement. Further taxonomies for MLOps are provided by Lwakatare et al. [212, 213], Google [214] and Microsoft [215]. These taxonomies describe the automated operation with different levels of automation and consider the human-machine interaction, which we discuss in the last paragraph more detailed. However above taxonomies only consider the automated operation and disregard automating the design process.

Taxonomies for automated design and automated operation. While the above taxonomies either only consider the automated design or only consider the automated operation, this paragraph analyzes taxonomies that consider both. Kreuzberger et al. [216] propose a detailed end-to-end MLOps taxonomy to describe the automated workflow. The workflow consists of four zones which are the product initiation zone, the data engineering zone,

the ML experimentation zone, and the ML production zone. In the latter three zones, the automation is further broken down, describing the processes as pipelines. These pipelines are divided into sections that are similar to those described in the previous paragraphs. A similar workflow is proposed by Testi et al. [217] that is less detailed but additionally considers model diagnosis. Although both taxonomies describe both automated design and automated operation in detail, they do not differentiate between different levels of automation. Differentiating is, however, important to describe the maturity of a process. Especially in the development of a forecasting pipeline for a specific smart grid application, the classification into automation levels helps to communicate the maturity of the pipeline.

Taxonomies that consider human-machine interaction and define levels of automation. In this paragraph, we analyze the above taxonomies in terms of the additional criteria of human-machine interaction and levels of automation. Several taxonomies describe only the human-machine interaction [205, 208, 210, 211, 216, 217]. While some of those only describe the human-machine interface briefly [210, 211], others provide detailed descriptions [205, 208, 216, 217]. Detailed descriptions are especially provided by taxonomies that define levels for the scope of automation [170, 203, 212–215, 218]. For example, Karmaker et al. [203] define seven levels for AutoML. The levels range from programming the algorithms from scratch, over ML implementations provided by programming libraries and libraries for the automated design of those to a fully automated system, which automatically formulates the task, labels the data if necessary, automates the design of the ML pipeline, and summarizes the results with a recommendation. Another example are the taxonomies for MLOps provided by Google [214] and Microsoft [215]. While Google defines three levels, Microsoft suggests five levels. Both range from manual deployment of ML pipelines, over automated ML pipeline deployment to a fully automated process, which continuously improves and delivers the ML pipeline. Apart from the above taxonomies for ML, taxonomies for other domains exist that consider the human-machine interaction and levels of automation. Simmler and Frischknecht [218], for instance, propose a general taxonomy for human-machine interaction, ranging from offering decisions, over executing and informing, to executing fully automated. Another well-known taxonomy is the SAE standard for autonomous driving [170] that ranges from SAE Level 0 (manual driving) to SAE Level 5 (full vehicle autonomy), which inspired the definition of the automation levels of the AutoLVL taxonomy in this thesis.

2.7 Novelty and remaining questions

Given the shortcomings of related taxonomies identified in the previous section, the novelty of the automation level taxonomy AutoLVL for time series forecasting are:

1. AutoLVL describes both, the automated design and the automated operation of forecasting models.
2. AutoLVL details the human-machine interaction and specifies which tasks are taken over by the machine.
3. AutoLVL allows to communicate the maturity of a forecasting pipeline in terms of the scope of automation.
4. AutoLVL serves as a guideline for the development of domain-specific templates of forecasting models, as detailed in the following chapters.

Regarding the open questions of this thesis stated in Section 1.5, the automation levels describe what automation in time series forecasting implies and which scopes of automation exist, with the scientific contribution of this chapter summarized in Figure 2.20. However, new questions may be raised:

- Which roles for data scientists are necessary to realize automated forecasts with a certain automation level?
- Are intermediate levels necessary to classify automated forecasting methods more finely?
- What further automation methods exist in addition to the ones applied in this thesis to achieve the automation levels?

AutoLVL

Automation level taxonomy for time series forecasting



Publications

S. Meisenbacher, J. Pinter, T. Martin, V. Hagenmeyer, and R. Mikut, “Concepts for automated machine learning in smart grid applications”, in Proceedings of the 31. Workshop Computational Intelligence, Berlin, Germany: KIT Scientific Publishing, 2021, pp. 11–35.

S. Meisenbacher, J. Galenzowski, K. Förderer, W. Suess, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “Automation level taxonomy for time series forecasting services: Guideline for real-world smart grid applications”, in Energy Informatics, B. N. Jørgensen, Z. G. Ma, F. D. Wijaya, R. Irnawan, and S. Sarjiya, Eds., Cham, Switzerland: Springer Nature Switzerland, 2025, pp. 277–297.

Figure 2.20: The contributions of this chapter with the proposed automation level taxonomy AutoLVL.

2.8 Contributed forecasting templates

The proposed automation level taxonomy for time series forecasting AutoLVL can be used as a guideline to pre-designed and automate forecasting models (*forecasting template*), see Table 2.10. Based on this guideline, three forecasting templates are developed in this thesis: AutoPQ [164, 165] (Automation level 2), AutoWP [166, 167] (Automation level 3), and AutoPV [168, 169] (Automation level 4). Figure 2.21 gives an overview of these three templates, showing which automation methods are utilized to achieve the corresponding automation level. The figure also refers to the provided open-source implementations and realized real-world applications.

Table 2.10: The six levels of the AutoLVL taxonomy [162] as a guideline for developing automated forecasting templates [163]. A check mark signifies that the step is automated, and a cross denotes a manual step.

Automation level	Task definition	Training data selection	Data flow orchestration	Model design	Model monitoring	Model adaptation
0	✗	✗	✗	✗	✗	✗
1	✗	✗	✓	✗	✗	✗
2	✗	✗	✓	✓	✗	✗
3	✗	✗	✓	✓	✓	✗
4	✗	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓

¹¹ The icon of the probability distributions is inspired by [219] and the icons of the Wind Power (WP) turbine and the PV plant are adapted from [14].

AutoPQ [164, 165] (Chapter 3)

Template for automated point forecast-based quantile forecasts

**Used generic automation methods of Chapter 2**

- 1 Forecasting pipeline (Section 2.1)
- 2 HPO (Section 2.2.1)
- 2 CASH (Section 2.2.2)
- 2 Successive halving (Section 2.2.3)

Open-source implementation
<https://github.com/SMEISEN/AutoPQ>
AutoWP [166, 167] (Chapter 4)

Template for automated wind power forecasts

**Used generic automation methods of Chapter 2**

- 1 Forecasting pipeline (Section 2.1)
- 2 CASH (Section 2.2.2)
- 2 Ensemble optimization (Section 2.2.4)
- 3 Pre-trained models (Section 2.3.2)

Open-source implementation
<https://github.com/SMEISEN/AutoWP>
Smart grid application

- 2 Redispatch 2.0 (Section 6.3)

AutoPV [168, 169] (Chapter 5)

Template for automated photovoltaic forecasts

**Used generic automation methods of Chapter 2**

- 1 Forecasting pipeline (Section 2.1)
- 2 HPO (Section 2.2.1)
- 2 CASH (Section 2.2.2)
- 3 Prior knowledge (Section 2.3.1)
- 3 Pre-trained models (Section 2.3.2)
- 3 Performance monitoring (Section 2.3.3)
- 4 Cyclic model adaption (Section 2.4.1)
- 4 Drift detection-based model adaption (Section 2.4.2)

Open-source implementation
<https://github.com/SMEISEN/AutoPV>
Smart grid application

- 2 Redispatch 2.0 (Section 6.3)
- 4 Smart East (Section 6.2)
- 4 Energy Lab (Section 6.1)

Figure 2.21: The three forecasting templates contributed in this thesis, achieving Automation level 2, 3, and 4.¹¹

3 AutoPQ: Template for automated point forecast-based quantile forecasts

Probabilistic forecasts are vital for downstream smart grid applications as they quantify uncertainty alongside the point forecast and thus contribute to more informed decisions (e. g. in the stochastic power flow optimization [220]). To satisfy the increasing demand for such models, automating the design process of probabilistic forecasting models is necessary. However, three major challenges exist:

First, forecasts need to quantify the inherent uncertainty in an unbiased and accurate manner. However, many state-of-the-art methods (e. g. [43, 44, 221]) are still point forecasts [222]. While methods to create a probabilistic forecast based on a point forecasting model exist, they rely on an assumed Probability Density Function (PDF) (e. g. [28, 223]), are designed to estimate Prediction Intervals (PIs) rather than providing the full PDF (e. g. [224, 225]), and are based on the point forecast's residuals (e. g. [226]) rather than mapping the uncertainty of the process directly. Although direct probabilistic methods exist that overcome these limitations to some extent, the vast majority of well-designed and established point forecasting methods then remain unused [115].

Second, due to the no-free-lunch theorem, no forecasting method exists that excels in all forecasting tasks; especially, since the forecast quality is sensitive to various model design decisions [20]. Therefore, an automated selection of the best-performing method is required as in [115, 164]. However, performance-critical smart grid applications may require further refinement by HyperParameter Optimization (HPO). Although Combined Algorithm Selection and Hyperparameter optimization (CASH) methods for time series forecasting exist, they are limited to point forecasting methods (e. g. [78, 227, 228]), or underlie the limitations stated in the first challenge (e. g. [229]).

Third, smart grid applications typically require forecasts with customized characteristics [24], such as the probabilistic properties coverage rate and sharpness [230, 231]. Unlike a stochastic optimization problem that generally requires a sharp representation of uncertainty subject to calibration [232], a robust optimization problem may only be interested in extreme events that occur with low probability [233], requiring well-calibrated forecasts with a high coverage rate. However, recent probabilistic forecasting methods create probabilistic

forecasts whose properties cannot be easily customized (e. g. [48, 234]) or lack automation (e. g. [115, 164]).

AutoPQ addresses these three challenges and performs automated design by CASH to customize the probabilistic properties of the quantile forecast (Automation level 2 according to the AutoLVL taxonomy [162, 163] in Chapter 2).¹²

3.1 Automated design methods

The underlying idea of AutoPQ is to generate a probabilistic forecast based on an arbitrary point forecast using a conditional Invertible Neural Network (cINN) [107, 115] and to make corresponding design decisions automatically, aiming to increase the probabilistic performance. In this context, three questions arise: i) how such a probabilistic forecast is created, ii) which design decisions exist, and iii) how these decisions are made automatically. These three questions are answered in the following, and Figure 3.1 illustrates the approach with automated aspects highlighted in green. To account for different computing systems and performance requirements, we consider two variants: AutoPQ-default suitable for general-purpose computing systems achieving competitive forecasting performance based on [115, 164], and AutoPQ-advanced requiring High-Performance Computing (HPC) systems to further increase performance for smart grid applications with high decision costs.

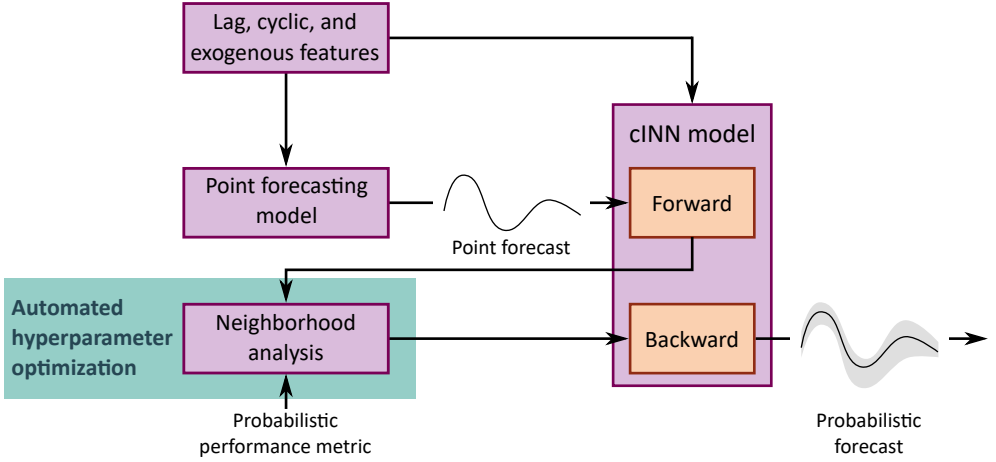
3.1.1 Creation of the probabilistic forecast

The approach to create probabilistic forecasts of Phipps et al. [115] consists of four steps, which are briefly explained in the following.

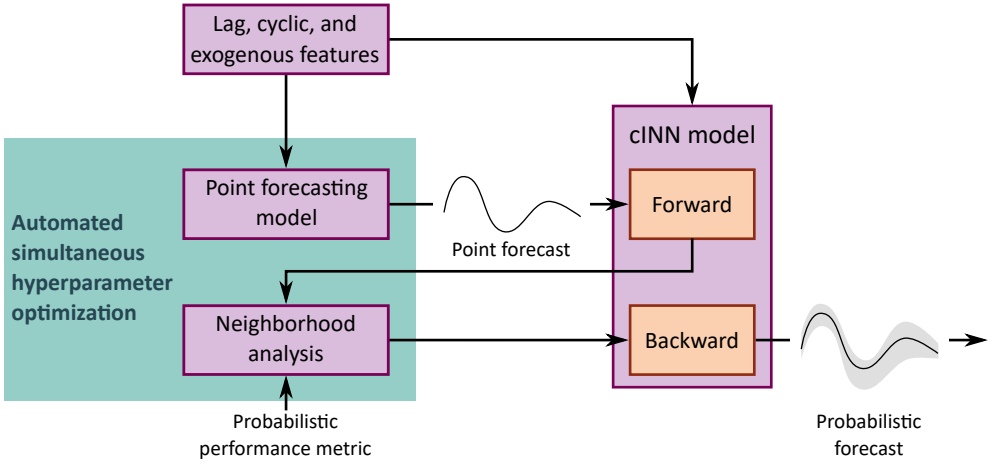
Creation of a point forecast in an unknown distribution. A point forecasting model $f_p(\cdot)$, as detailed in Section 1.2.1, estimates future values $\hat{\mathbf{y}}$ of the target time series \mathbf{y} based on current and past values. More specifically, the point forecasting model considers historical values of $\mathbf{y} [k-H_1, \dots, k]$ and values of exogenous time series $\mathbf{X} [k-H_1, \dots, k]$ and exogenous forecasts $\hat{\mathbf{X}} [k+1, \dots, k+H]$ to make a forecast $\hat{\mathbf{y}} [k+1, \dots, k+H]$ for a specified forecast horizon $H \in \mathbb{N}_1$ at origin k . Thus, for each time point k , the forecast consists of the next H values, and the target time series \mathbf{y} contains realizations of $\hat{\mathbf{y}}$. Consequently, \mathbf{y} can be interpreted as a realization of the random variable $\mathbf{Y} \sim f_Y(\mathbf{y})$ in the realization space \mathbb{Y} with an unknown H -dimensional PDF $f_Y(\mathbf{y})$.

¹ Note that this chapter is based on [164, 165] and contain identical phrases; a detailed list is given in the Appendix.

² Note that AutoPQ also allows forecasting the PDF or generate scenarios but this chapter only focuses on quantile forecasts.



(a) AutoPQ-default uses automated HPO to determine the sampling hyperparameter that is optimal for the given probabilistic performance metric. The figure is based on [164].



(b) AutoPQ-advanced uses automated and joint HPO to determine the hyperparameter configuration of the point forecasting model and the sampling hyperparameter that are optimal for the given probabilistic performance metric. The figure is based on [165].

Figure 3.1: Overview of AutoPQ: Lag features, cyclic features, and exogenous features are selected and used as inputs by a point forecasting model to create a point forecast in an unknown distribution. This point forecast and the features are combined in a cINN [107, 115], which creates a representation of the forecast in a known and tractable distribution. The neighborhood of this representation is analyzed to determine how to include uncertainty information. Finally, with the backward pass through the cINN, the uncertainty is mapped back to the unknown distribution to create the probabilistic forecast. Novel automation methods are highlighted in green [165].

Representation of the point forecast in a known and tractable distribution. To represent the point forecast $\hat{\mathbf{y}}[k+1, \dots, k+H]$ in a space with a known and tractable H -dimensional PDF $f_Z(\mathbf{z})$, we use a cINN. A cINN is an Artificial Neural Network (ANN) consisting of multiple conditional-affine coupling blocks, which ensure that the learned non-linear function $g : \mathbb{Y} \rightarrow \mathbb{Z}$ is bijective [235]. Since the coupling blocks are conditional-affine, the cINN can include additional information into the mapping [235], e. g., exogenous time series and exogenous forecasts, as described above for the point forecasting model. Using this additional information, the cINN is trained such that the non-linear bijective mapping g leads to the realization

$$\begin{aligned} \mathbf{z}[k+1, \dots, k+H] &= g(\mathbf{y}[k-H_1, \dots, k], \mathbf{y}[k+1, \dots, k+H], \\ &\quad \mathbf{X}[k-H_1, \dots, k], \hat{\mathbf{X}}[k+1, \dots, k+H], \\ &\quad \mathbf{p}), k, H, H_1 \in \mathbb{N}_1, k > H_1, \end{aligned} \quad (3.1)$$

where \mathbf{p} are the trainable parameters of the cINN. More specifically, the cINN is trained with the change of variables formula in the maximum likelihood loss function

$$\begin{aligned} \mathcal{L} = \mathbb{E} \left[\frac{1}{2} \left\| g(\mathbf{y}[k-H_1, \dots, k], \mathbf{y}[k+1, \dots, k+H], \right. \right. \\ \left. \left. \mathbf{X}[k-H_1, \dots, k], \hat{\mathbf{X}}[k+1, \dots, k+H], \right. \right. \\ \left. \left. \mathbf{p} \right\|_2^2 - \log |\det(J)| \right] + \lambda_{\mathcal{L}} \|\mathbf{p}\|_2^2, \end{aligned} \quad (3.2)$$

where $J = \partial g / \partial \mathbf{y}[k+1, \dots, k+H]$ is the Jacobian matrix [107, 235], and $\lambda_{\mathcal{L}} \|\mathbf{p}\|_2^2$ is a regularization term. This training ensures that the resulting latent space distribution $f_Z(\mathbf{z})$ follows a multi-dimensional Gaussian distribution. Consequently, the cINN can map an arbitrary distribution of the original data $\mathbf{y}[k+1, \dots, k+H]$ to a realization $\mathbf{z}[k+1, \dots, k+H]$ that is approximately Gaussian-distributed. This mapping can be used to quantify the uncertainty of a point forecast $\hat{\mathbf{y}}[k+1, \dots, k+H]$ by mapping it into the latent space $\hat{\mathbf{z}}[k+1, \dots, k+H]$, and analyze its neighborhood as detailed in the following paragraph. Importantly, the cINN is trained independently of the point forecasting model.

Neighborhood analysis of the point forecast's latent space representation. To generate probabilistic forecasts, we first pass the point forecast $\hat{\mathbf{y}}[k+1, \dots, k+H]$ together with the historical values of $\mathbf{y}[k-H_1, \dots, k]$ and exogenous time series $\mathbf{X}[k-H_1, \dots, k]$ and exogenous forecasts $\hat{\mathbf{X}}[k+1, \dots, k+H]$ through the trained cINN to obtain the latent space representation:

$$\begin{aligned} \hat{\mathbf{z}}[k+1, \dots, k+H] &= g(\mathbf{y}[k-H_1, \dots, k], \hat{\mathbf{y}}[k+1, \dots, k+H], \\ &\quad \mathbf{X}[k-H_1, \dots, k], \hat{\mathbf{X}}[k+1, \dots, k+H], \\ &\quad \mathbf{p}), k, H, H_1 \in \mathbb{N}_1, k > H_1. \end{aligned} \quad (3.3)$$

To quantify the point forecast's uncertainty, we then analyze the neighborhood of this representation in the Gaussian-distributed latent space. That is, the uncertainty in the point forecast's neighborhood is explored using

$$\begin{aligned}\hat{\mathbf{z}}_n^* [k+1, \dots, k+H] &= \hat{\mathbf{z}} [k+1, \dots, k+H] + \mathbf{r}_{\sigma, n} [1, \dots, H], \\ n &= 1, \dots, N_\sigma, \quad \mathbf{r}_{\sigma, n} \sim \mathcal{N}(0, \sigma),\end{aligned}\tag{3.4}$$

where $\mathbf{r}_{\sigma, n} [1, \dots, H]$ is a random noise vector sampled from the H -dimensional standard normal distribution $\mathcal{N}(0, \sigma)$ with zero-mean and equal variance σ for each dimension, and $n = 1, \dots, N_\sigma$ is the sample number. Taking N_σ different samples of $\hat{\mathbf{z}}_n^* [k+1, \dots, k+H]$ results in a set of realizations that depends on the sampling variance σ , which is called sampling hyperparameter λ_q in the following. This set of realizations is similar but not identical to the original point forecast $\hat{\mathbf{y}} [k+1, \dots, k+H]$, and represents the uncertainty in the neighborhood of the point forecast's latent space representation.

Generating a probabilistic forecast based on the point forecast. The generation of a probabilistic forecast based on the point forecast consists of two steps: First, the cINN is used to generate samples in the realization space by inversely mapping the samples from the latent space, thereby including the quantified uncertainty around the original representation of the point forecast, which is valid due to the equivalence of uncertainty in both spaces. Second, the quantiles \mathbf{q} of these samples in the realization space are calculated, resulting in the probabilistic forecast $\hat{\mathbf{y}} [k+1, \dots, k+H]$.

3.1.2 Important task-dependent design decisions

When creating a probabilistic forecast using the above-described approach, different design decisions must be made, which are specific to the respective forecasting task. These decisions comprise feature engineering, the selection of the point forecasting method and its hyperparameters λ_p , and the selection of the sampling hyperparameter λ_q in the Gaussian-distributed latent space of the cINN for generating a probabilistic forecast based on the point forecasting.

Lag, cyclic and exogenous features. Considering features may enrich the input space of a forecasting method with additional explanatory variables. Therefore, extracting and selecting these features is an important design decision that depends on the particular forecasting task. For example, features that are relevant for a PhotoVoltaic (PV) forecast may be irrelevant for a Wind Power (WP) forecast. As described in Section 1.3.2, different feature types exist, and we consider lag features, cyclic features, and exogenous features.

Lag features provide values from prior time points (2.1), i. e., at time point k , the model also processes values that date back a certain time horizon H_1 : While forecasting methods based on Statistical Modeling (SM) and Deep Learning (DL) implicitly consider values from prior time points, forecasting methods based on Machine Learning (ML) require the explicit definition of lag features.

In cyclic features, the cyclic relationship can be established through a trigonometrical encoding, e. g., the month or the hour of the day (2.2) and (2.4). Furthermore, work and non-work days can be encoded through an ordinal encoding (2.6).

Exogenous features are valuable if the target variable is subject to exogenous influences, e. g., weather variables such as the temperature T , the humidity φ_h , the wind speed v , and the solar irradiance G including total cloud cover TCC.

Point forecasting methods and hyperparameters. The selection of the point forecasting method and the corresponding hyperparameters influence the point forecasting performance, as detailed in Section 1.3.3, and tailoring the hyperparameter configuration λ_p to the specific task using HPO may improve the point forecasting performance over the default configuration. Therefore, we expect that HPO will also improve the probabilistic performance of the cINN approach [115, 164] for creating a probabilistic forecast based on a point forecast. As described in Section 1.2.1, different forecasting method families exist, and we consider three methods for each family [28, 222]:

- SM** seasonal AutoRegressive Integrated Moving Average with eXternal input (sARIMAX) [66]; Error Trend Seasonality (ETS) [32]; Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend, Seasonal components (TBATS) [236];
- ML** MultiLayer Perceptron (MLP) [100]; Support Vector Regression (SVR) [237]; eXtreme Gradient Boosting (XGB) [238];
- DL** Deep AutoRegression (DeepAR) [48]; Neural Hierarchical Interpolation for Time Series (N-HITS) [43]; Temporal Fusion Transformer (TFT) [44].

Depending on the selected forecasting method, different hyperparameters have to be taken into account. Since the optimal selection of the forecasting method and the optimal selection of its hyperparameters may depend strongly on the training data set, it often pays off to tailor the selection to the forecasting task. The considered hyperparameters for the three forecasting method families are given in Table B.1, B.2, and B.3 in the Appendix.

Hyperparameter for sampling in the cINN latent space. The hyperparameter $\lambda_q = \sigma$ for sampling in the latent space of the cINN is responsible for introducing quantified

uncertainty into the forecast. The selection of λ_q has a significant impact on the properties of the resulting probabilistic forecast. For example, a large λ_q includes more uncertainty, which may result in a forecast with better coverage rates. In contrast, a small λ_q only allows for a small amount of uncertainty to be included and may result in (too) sharp probabilistic forecasts. Consequently, depending on the desired task-specific probabilistic properties of the forecast, a different λ_q is optimal. The considered value range of λ_q is given in Table B.4 in the Appendix.

3.1.3 Optimal design of the probabilistic forecast

Since manually tailoring a probabilistic forecasting model to the task is iterative and requires expert knowledge, we automate this process considering the above design decisions. More specifically, we select relevant features and perform HPO to optimally configure the forecasting model. For the latter, two alternatives exist, which we refer to as AutoPQ-default and AutoPQ-advanced. While AutoPQ-default is suitable for general-purpose computing systems to achieve competitive forecasting performance, AutoPQ-advanced is designed for HPC systems to further increase the performance.

Feature selection

Since energy time series contain predominantly daily, weekly, and yearly cycles [42, 172, 173], we always consider lag features and cyclic features. These features can be created independently of the existing data, as only time stamps and calendar information are required. Additional exogenous features that are available for the forecast horizon are selected depending on the existing additional data based on [164].

AutoPQ-default for general-purpose computing systems

Since the computational effort of optimizing the hyperparameters of a point forecasting method is significantly higher than generating a probabilistic forecast based on a point forecast together with optimizing the sampling hyperparameter, we only consider the latter in general-purpose computing systems. This becomes evident in Figure 3.2 (note the logarithmic y-axis), which shows the average computational effort across the six data sets used in the evaluation of i) training a point forecasting model, compared to ii) generating a probabilistic forecast based on the point forecast. That is, the computational effort of assessing a candidate sampling hyperparameter λ_q is significantly lower. Note that [115] realizes this by using a manual HPO of λ_q . This method has been enhanced in [164] to include automated HPO of λ_q , which is referred to as AutoPQ-default and detailed in the following.

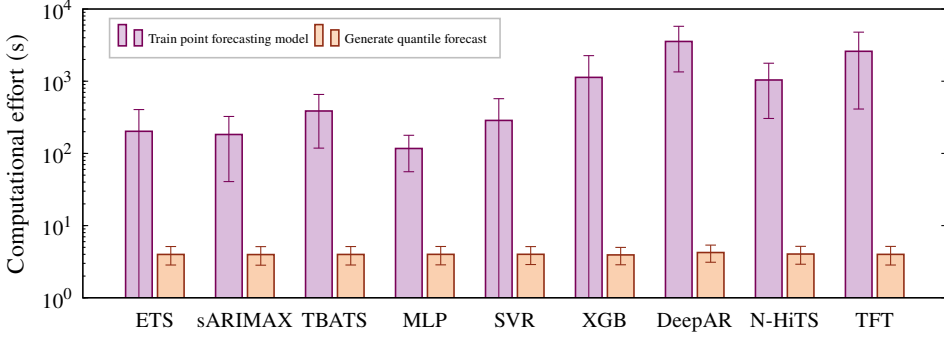


Figure 3.2: Comparison of the average computational effort in terms of runtime across the six data sets used in the evaluation: effort for training a point forecasting model with configuration λ_p and effort for generating a quantile forecast based on the point forecast using the cINN [107, 115] with λ_q . Due to the significantly smaller computational effort of generating the quantile forecast, it is worthwhile to evaluate several λ_q for one λ_p to properly balance the efforts. Note that for ETS, SVR, and XGB the standard deviation is higher than the mean value. The figure is based on [165].

Hyperparameter optimization. Instead of manually searching for the (estimated) optimal hyperparameter configuration λ_q^* within a given configuration space Λ_q , we define the HPO problem

$$\lambda_q^* = \min_{\lambda_q \in \Lambda_q} Q(\hat{y}(\lambda_q), y), \quad (3.5)$$

where the objective function Q is an arbitrary probabilistic performance metric, and λ_q is the sampling hyperparameter in the Gaussian-distributed latent space of the cINN.

Optimization algorithm. To select an appropriate optimization algorithm, we must consider the properties of the objective function Q (3.5). As Q is noisy and non-differentiable and we do not have a closed-form expression of the configuration space Λ_q , a black-box optimization method is required, which acts as HPO trial scheduler to suggest new candidate configurations to be assessed. We implement the HPO trial scheduler using Bayesian Optimization (BO) with the Tree Parzen Estimator (TPE) as surrogate model, as reasoned in Section 2.2.1.

Early stopping. When the objective function’s value Q in the optimization has reached a satisfactory value, or when the optimization has converged, it can be stopped. Consequently, we may stop the BO when it is expected that scheduling additional trial configurations for assessment will no longer improve Q . To achieve this, we define an early stopping condition

that terminates the BO if assessing of Q plateaus across trials, i. e., the value of the top five trials have a standard deviation of less than 0.0005 with a patience of five trials.³

Forecasting method selection. Not only the optimization of the hyperparameter λ_q but also the selection of the point forecasting method is decisive for the forecast performance. Consequently, we aim to select the best point forecasting method, where the corresponding λ_q in turn is already optimized. We therefore perform a straightforward empirical selection. First, we train point forecasting models of all considered methods with their default configuration $\lambda_{p,\text{default}}$. We then generate a probabilistic forecast based on each point forecast, and automatically search the corresponding optimal sampling hyperparameter λ_q^* with the above-described HPO. Finally, we select the probabilistic forecast that achieves the best score on the validation data set.

AutoPQ-advanced for HPC systems

For smart grid applications with high decision costs, it can be worthwhile to further increase the probabilistic forecasting performance. To address this requirement, we propose AutoPQ-advanced, which is detailed below.

Hyperparameter optimization. Instead of manually searching for the (estimated) optimal hyperparameter configuration $\lambda^* = \lambda_p^* \cup \lambda_q^*$ within a given configuration space $\Lambda = \Lambda_p \cup \Lambda_q$, we define the HPO problem

$$\lambda^* = \min_{\lambda \in \Lambda} Q(\hat{y}(\lambda_p, \lambda_q), \mathbf{y}), \quad (3.6)$$

where the objective function Q is an arbitrary probabilistic performance metric, λ_p is the hyperparameter configuration of the point forecasting method, and λ_q is the sampling hyperparameter in the Gaussian-distributed latent space of the cINN. To solve (3.6) efficiently, we must consider the different computational efforts of training the point forecasting model and generating a probabilistic forecast based on the point forecast. In Figure 3.2 (note the logarithmic y-axis), the average computational effort across the six data sets used in the evaluation is compared: to train different point forecasting methods and to generate a probabilistic forecast based on the point forecast using the cINN. Since the latter can be performed arbitrarily often with different sampling hyperparameters for a trained point forecasting model, we solve (3.6) with Algorithm 2, which consists of two nested loops for joint optimization. The following paragraphs first detail the joint optimization and the application of Prior Knowledge (PK) during optimization before describing the implementation details of the chosen optimization algorithms.

³ Note that we assess on normalized data sets and the values are thus in similar ranges across data sets.

Joint optimization. The outer loop of Algorithm 2 is responsible for finding the optimal hyperparameter configuration of the point forecasting method λ_p^* and runs with a timer b_t until the time budget B_t is exhausted. After receiving a candidate λ_p from the outer HPO trial scheduler, the point forecasting model $f_p(\cdot)$ is trained. Then, the inner loop of Algorithm 2 starts that is responsible for finding the optimal sampling hyperparameter λ_q^* . More precisely, a sub-optimization problem is solved, which corresponds to the objective

Algorithm 2 The HPO algorithm of AutoPQ optimizes the hyperparameters of the point forecasting method λ_p and the sampling hyperparameter λ_q of the cINN [107, 115] jointly and considers the significantly lower computational effort of the latter. The algorithm is based on [165].

Input: $\Lambda_p, \Lambda_q, B_i, B_t, \mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}, f_{\text{cINN}}(\cdot)$

```

1: trial_scheduler_outer( $\Lambda_p$ )  $\triangleright$  Initialize Propulate [181] (EA)
2:  $b_t \leftarrow 0$  s
3: while  $b_t < B_t$  do
4:    $t \leftarrow \text{get\_current\_time}()$ 
5:    $\lambda_p \leftarrow \text{trial\_scheduler\_outer.get\_configuration}()$ 
6:    $f_p(\cdot) \leftarrow \text{train\_p\_model}(\lambda_p, \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$   $\triangleright$  High computational effort
7:    $m, s \leftarrow \text{trial\_scheduler\_outer.get\_stats}()$   $\triangleright$  Distribution of  $\lambda_q^*$  in active population
8:   trial_scheduler_inner( $\Lambda_q, m, s$ )  $\triangleright$  Initialize Hyperopt [182] (BO) with PK
9:   for  $b_i \leftarrow 0$  to  $B_i$  do
10:     $\lambda_q \leftarrow \text{trial\_scheduler\_inner.get\_configuration}()$ 
11:     $f_q(\cdot) \leftarrow \text{generate\_q\_model}(f_p(\cdot), f_{\text{cINN}}(\cdot), \lambda_q)$   $\triangleright$  Low computational effort
12:     $Q \leftarrow \text{assess\_prob\_performance}(f_q(\mathbf{X}_{\text{val}}, \mathbf{y}_{\text{val}}))$   $\triangleright$  Low computational effort
13:    trial_scheduler_inner.update( $Q, \lambda_q$ )
14:     $b_i \leftarrow b_i + 1$ 
15:    if (early_stopping( $Q$ )) then
16:      break
17:    end if
18:  end for
19:   $\lambda_q^* \leftarrow \text{trial\_scheduler\_inner.get\_best\_configuration}()$ 
20:  trial_scheduler_outer.update( $Q, \lambda_p, \lambda_q^*$ )  $\triangleright$  Store  $\lambda_q^*$  associated with  $\lambda_p$ 
21:   $b_t \leftarrow b_t + (\text{get\_current\_time}() - t)$ 
22: end while
23:  $\lambda_p^*, \lambda_q^* \leftarrow \text{trial\_scheduler\_outer.get\_best\_configuration}()$ 

```

Output: λ_p^*, λ_q^*

function (3.5). The inner loop runs with a counter b_i until the iteration budget B_i is exhausted or an early stopping condition is fulfilled. After receiving a candidate λ_q from the inner HPO trial scheduler, the already trained cINN $f_{\text{cINN}}(\cdot)$ is used to generate a probabilistic forecast $f_q(\cdot)$ based on $f_p(\cdot)$. Then, its probabilistic performance Q is assessed, and the HPO trial scheduler of the inner loop is updated with Q and λ_q . The result of completing the inner loop is the optimal sampling hyperparameter λ_q^* associated with λ_p . Before starting a new iteration of the outer loop, the outer HPO trial scheduler is updated with Q , λ_p and λ_q^* . Finally, the result of the outer loop is λ_p^* and the associated λ_q^* .

Prior knowledge. In the initialization of the inner HPO trial scheduler, PK can be used as we assume that the optimal sampling hyperparameter λ_q^* associated with the hyperparameter configuration of the point forecasting method λ_p is related to previously assessed configurations. Hence, we initialize the inner HPO trial scheduler with statistical parameters computed from past well-performing hyperparameter configurations. Accessing λ_q^* of previously assessed λ_p is possible since they are stored when updating the outer HPO trial scheduler.

Optimization algorithms. As the objective function (3.6) is potentially noisy and non-differentiable and we do not have a closed-form expression of the configuration space Λ_p , black-box optimization methods are required, which act as HPO trial schedulers to suggest new candidate configurations to be assessed in the inner and outer loop of Algorithm 2.

In the **outer loop**, training the point forecasting model is computationally expensive and benefits from training using a Graphics Processing Unit (GPU). If multiple GPUs are available, it is possible to assess multiple hyperparameter configurations λ_p (i. e. multiple trials of Q) in parallel. In this situation, however, an asynchronous optimization method is required since the training duration across trials might differ, which would result in GPU idle time in synchronous optimization. To enable asynchronous execution of the outer loop, we use the Python package Propulate [181] as outer HPO trial scheduler, as reasoned in Section 2.2.1. Propulate is an Evolutionary Algorithm (EA) that maintains a continuous population to achieve an asynchronous assessment of configuration candidates. The particular algorithm setup is based on [181] and can be found in Table B.5 in the Appendix. Maintaining a continuous population not only allows scheduling new candidates of Λ_p for assessment but also enables providing PK for the inner loop.

In the **inner loop**, generating a probabilistic forecast based on a point forecast is computationally inexpensive. Furthermore, the use of PK leads to a rapid convergence, whereby the inner loop is quickly terminated by the early stopping condition. Therefore, parallelizing the inner loop would result in a large computational overhead that would not result in any reduction of the computation time. For sequentially executing the inner loop, we

use the Python package Hyperopt [182] as the inner HPO trial scheduler. Hyperopt is a BO algorithm with the TPE as surrogate model that estimates the distribution of well-performing hyperparameter configurations in relation to underperforming ones for exploring the configuration space [183].

For initializing Hyperopt, we use PK obtained from the population of Propulate by calculating statistical properties of the optimal sampling hyperparameters λ_q^* associated and stored with the active and already assessed candidates of Λ_p . More specifically, we calculate the mean m and the standard deviation s of the optimal sampling hyperparameters' natural logarithms as shown in Figure 3.3a, and initialize Hyperopt with the log-normal distribution $\log - \mathcal{N}(m, s)$ as prior distribution.⁴ The estimation of the prior distribution is exemplified in Figure 3.3a with the MLP on the Load-BW data set.

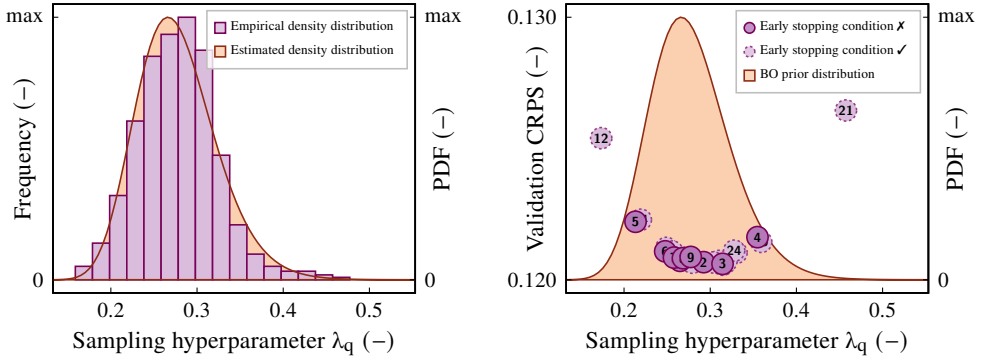
This prior distribution is initially used to randomly create a set of $\{\lambda_{q,0}, \lambda_{q,1}, \dots\}$ before fitting the TPE surrogate model, i. e., regions with high probability under the prior are more likely to be explored initially. Once the TPE surrogate model is fitted, it is used to schedule new candidates $\lambda_{q,i}$ using an acquisition function that guides the exploration and exploitation of promising regions in the configuration space. After each assessed $\lambda_{q,i}$ (i. e. a trial of Q), its performance Q is used to update the TPE surrogate model. Consequently, the prior distribution becomes less influential as more candidates are assessed, which ensures that values outside the assumed prior distribution are also considered (if the early stopping condition is not fulfilled beforehand).

The HPO with BO-TPE and PK for the noisy and non-differentiable objective function (3.5) is exemplified in Figure 3.3b with the MLP on the Load-BW data set. The figure also shows the early stopping, which is described in the following.

Early stopping. As detailed for AutoPQ-default, an optimization can be stopped when the objective function's value Q has reached a satisfactory value, or when the optimization has converged. Consequently, we may stop the inner loop when it is expected that additional trials of Q will no longer improve the objective function's value Q . To achieve this, we again define an early stopping condition that terminates the inner loop if assessing Q plateaus across trials, i. e., the value of the top five trials have a standard deviation of less than 0.0005 with a patience of five trials.

Forecasting method selection. AutoPQ-advanced aims to select the best point forecasting method, where the hyperparameters λ_p and the corresponding sampling hyperparameter λ_q , in turn, are already optimized. To combine the method selection with the HPO in a

⁴ This is advantageous because both the mean and the variance of the past optimal values are considered. Further, assuming a log-normal distribution excludes negative values, and we later show that it is a valid assumption.



(a) Estimation of the BO prior distribution based on the optimal sampling hyperparameters associated and stored with the active and already assessed candidates of the EA population, visualized as empirical density distribution. For the estimated density distribution, we assume a log-normal distribution and estimate its expected value m and standard deviation s .

(b) Initializing the BO with the estimated prior distribution already leads to a good estimate for the first trials before the TPE surrogate model fitted on these trials gains influence. To illustrate the evolution after the early stopping condition is fulfilled (9th iteration), the BO was continued until the 25th iteration (dashed dots), where the dot number is the iteration.

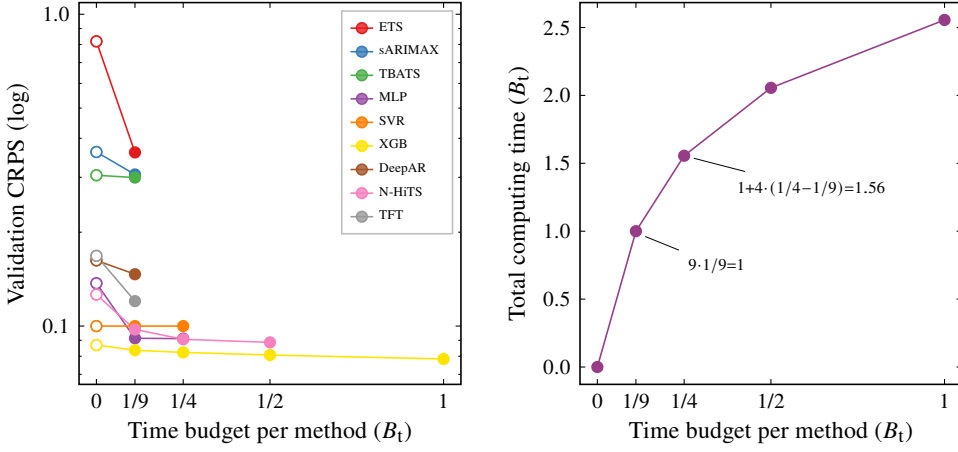
Figure 3.3: Estimation of the BO prior distribution and its utilization in the HPO of the sampling hyperparameter λ_q . The figure is based on [165].

computing resource-efficiently process, we use the successive halving pruning method (see Section 2.2.3). Successive halving prunes underperforming configurations and re-allocates resources to more promising configurations. More specifically, we apply Algorithm 1 for AutoPQ-advanced as exemplified in Figure 3.4. Starting with the nine candidates of forecasting methods, the algorithm iteratively eliminates poorly performing candidates and re-allocates computing resources to the remaining ones until a single best candidate is found and the total time budget B_t is exhausted. Importantly, the HPO of the remaining candidates is not restarted in each pruning round but continues running from the checkpoint of the previous round. In this way, an optimization run on HPC systems with a limited time budget per job can be split over several jobs.

3.2 Evaluation

In this section, we evaluate AutoPQ.⁵ First, we assess the probabilistic forecasting performance of AutoPQ with the default and the advanced configuration to explore the impact of HPO on improving probabilistic forecasts. Second, we evaluate the customizability of the probabilistic properties of forecasts with AutoPQ-advanced. Third, we perform an

⁵ A Python implementation of AutoPQ can be found on GitHub: <https://github.com/SMEISEN/AutoPQ>



(a) Evolution of the validation performance.

(b) Evolution of the computational effort.

Figure 3.4: AutoPQ-advanced with the successive halving pruning strategy exemplified for the Load-BW data set. Starting with the nine forecasting methods, underperforming methods successively drop out, while promising methods receive increased time budget for the HPO. The markers at $B_t = 0$ represent the performance of the respective default configuration, shown as a reference and therefore not considered within the time budget B_t .

ablation study to evaluate the effectiveness of optimizing the point forecasting method’s hyperparameters and the cINN’s sampling hyperparameter jointly (AutoPQ-advanced).

3.2.1 Benchmarking

In the following, we first describe the experimental setup for assessing the probabilistic forecasting performance, before showing the results and providing insights.

Experimental setup

In the experimental setup, we describe the used data, the evaluation strategy, and the considered benchmarks.

Data. We evaluate AutoPQ on six different data sets: Load-BW, Load-GCP, Mobility, Price, PV, and WP. In the following, we briefly describe each of these data sets before we describe the applied pre-processing and feature selection. Detailed information is given in Table B.6 in the Appendix.

The **Load-BW** data set [239] contains the gross electrical consumption of the state Baden-Württemberg, Germany, taken from the Open Power System Data (OPSD) portal. Thus, the

electrical consumption reflects the regional load, which is the aggregate of a large number of Grid Connection Points (GCPs).

The **Load-GCP** data set [240] includes the electrical consumption of 370 GCPs in a Portuguese distribution grid, from which we select the MT 158 time series to make local forecasts. That is, the electrical consumption reflects the local load of a single GCP.

The **Mobility** data set [176] consists of hourly records of rented bikes in Washington D.C., USA, which is an indicator for individual mobility in the smart grid context. Additionally, corresponding weather and seasonal information is available, such as air temperature and wind speed, as well as workdays and weekends or holidays.

The **Price** data set consists of the zonal electricity price data for an unknown location, taken from the price forecasting track of the Global Energy Forecasting Competition (GEFCom) 2014 [241]. The challenge comprised 14 tasks, each of which required the submission of a day-ahead forecast based on exogenous time series provided, namely the corresponding zonal price forecast and the total electrical consumption forecast. For the evaluation, we combine all 14 tasks into one data set.

The **PV** data set includes the power generation of a PhotoVoltaic (PV) plant in Australia. The data is also taken from the GEFCom 2014 [241] but from the PV power forecasting track, which consisted of 16 day-ahead forecasting tasks. Additionally, corresponding Numerical Weather Predictions (NWP) from the European Centre for Medium-Range Weather Forecasts (ECMWF) are available, such as Global Horizontal Irradiance (GHI) and TCC. All 16 tasks are also combined in one data set for the evaluation.

The **WP** data set contains the power generation of a wind farm in Australia. The data is taken from the Wind Power (WP) forecasting track of the GEFCom 2014 [241], which consisted of 16 day-ahead forecasting tasks. As in the PV data set, NWP from ECMWF are available like the wind speed and wind direction at a height of 10 m and 100 m. Likewise, we combine all 16 tasks into one data set for the evaluation.

All considered data sets originally have an hourly resolution except the Load-GCP data set, which is resampled to an hourly resolution.

Evaluation strategy. We evaluate the day-ahead forecasting performance of AutoPQ with the default and the advanced configuration using the six data sets described above. To make the results comparable across the data sets, we normalize the data sets before creating separate training, validation, and test data sets.⁶ An overview of the exact sample indexes of these splits and the selected exogenous features is given in Table B.6 in the Appendix.

⁶ The forecasting methods MLP, DeepAR, N-HiTS, and TFT additionally hold-out 20 % of the training data set for early stopping the training process when the loss on the hold-out data increases.

For AutoPQ’s point forecasting methods, we consider nine methods, specifically, three of each for the method families SM (sARIMAX, ETS, TBATS), ML (MLP, SVR, XGB), and DL (DeepAR, N-HiTS, TFT). AutoPQ-default trains all nine methods with the corresponding default configuration $\lambda_{p,\text{default}}$, generates probabilistic forecasts based on the point forecasts, and performs HPO for the sampling hyperparameter λ_q of each model to finally select the best-performing one; the results originate from [164]. AutoPQ-advanced optimizes both configurations – λ_p and λ_q – jointly and performs CASH using the successive halving pruning strategy with the time budget $B_t = 8$ h.

In the evaluation of probabilistic forecasts, the properties sharpness and calibration are crucial. As Gneiting et al. [114] state that probabilistic forecasts should maximize sharpness subject to calibration, we use the Continuous Ranked Probability Score (CRPS) (1.8) as validation and test metric, which trades-off sharpness and calibration.

We run the evaluation on each data set five times and report the arithmetic mean and the standard deviation across runs to make stochastic effects during training and optimization transparent. To ensure comparable results across runs, the same hardware is used for all runs.⁷ Additionally, we record the computing time and electricity consumption to quantify the computational effort and make it accessible for comparison regarding sustainability, as reasoned in [242].

Benchmarks. We compare the probabilistic forecasting performance of AutoPQ against multiple benchmarks. These benchmarks can be categorized into two classes: direct probabilistic forecasts and probabilistic forecasts based on existing point forecasts.

The first class of direct probabilistic forecasts includes DeepAR, Quantile Regression Neural Networks (QRNNs), and the Nearest Neighbor Quantile Filter (NNQF). DeepAR [48] is an AutoRegression (AR)-based Recurrent Neural Network (RNN) that makes probabilistic forecasts by sampling from a learned parametric PDF. We implement DeepAR using the Python package PyTorch Forecasting [243]. A QRNN [46] is trained with the Pinball Loss (PL) function to forecast a selected quantile q . To make a probabilistic forecast, we train multiple QRNNs with the different quantiles in the PL to make forecasts for a set of quantiles \mathbf{q} . We implement the QRNNs with the Python package Keras [179] and the built-in PL function. The NNQF [47] is also a quantile-based forecasting method that filters the training data to determine a set of quantiles \mathbf{q} in the data. Then, an ML-based method can be trained to make a multiple-quantile forecast with one output per quantile of the set. We implement the NNQF using an MLP using the Scikit-learn [100] Python package.

⁷ Each run schedules four trials in parallel over two Intel Xeon Platinum 8368 Central Processing Units (CPUs) with 76 cores, four NVIDIA A100-40 GPUs with 40 GB memory (GPUs are only used for DL-based forecasting methods), and 256 GB Random-Access Memory (RAM) provided by the HoreKa HPC.

The second class of point forecast-based probabilistic forecasts includes Gaussian PIs, Empirical PIs, and Conformal PIs. While all consider the residuals $r[k] = |\hat{y}[k] - y[k]|$, $\forall k \in \mathbb{N}_1^{K_{\text{val}}}$ between the point forecasts $\hat{y}[k]$ and the realized values $y[k]$ on a validation data set, they differ in how the residuals are used to determine PIs. Gaussian PIs [28] assume the residuals to be Gaussian-distributed and estimate its standard deviation σ to determine PIs centered on the point forecast by multiplying σ by a factor based on the desired confidence level. Contrarily, Empirical PIs [224] do not assume a parametric PDF but use the empirical PDF of the residuals (i. e. ranking $r[k]$ from the smallest to the largest value) to determine PIs based on the desired confidence level. Conformal PIs for multiple step-ahead forecasts [225] calibrate the PIs to the forecast horizon without assuming a parametric PDF. More precisely, a critical non-conformity score is determined for each $r[k]$, and the temporal dependence between these critical scores across the forecast horizon is ensured by applying Bonferroni correction to maintain the desired confidence level. The corrected critical scores are then used to determine PIs around the given point forecast.

The benchmark results originate from [164], where for the point forecast-based probabilistic benchmarks, we benchmark against the best-performing one for each, i. e., the point forecasting method that achieves the lowest CRPS when deriving PIs around them using the Gaussian, the Empirical, and the Conformal approach, respectively.

Results

In the following, we visualize and summarize the results of the benchmarking. An interpretation and discussion of these results is given in the next Section 3.3.

In the benchmarking results shown in Figure 3.5, we make three observations: First, the performance spread across data sets is higher for the direct probabilistic forecasting methods (DeepAR, QRNNs, NNQF) than for probabilistic forecasting methods based on point forecasts (Gaussian, Empirical, and Conformal PIs, as well as AutoPQ with the default and advanced configuration). Second, we see that the average performance across data sets for DeepAR, QRNNs, NNQF, and Gaussian PIs is worse than for Empirical PIs, Conformal PIs, AutoPQ-default, and AutoPQ-advanced. Third, although the performance difference between these latter four methods is small, a performance advantage of AutoPQ-advanced is visible.

To analyze the observation in Figure 3.5 that the performance difference between Empirical PIs, Conformal PIs, AutoPQ-default, and AutoPQ-advanced is small, we evaluate the improvement of AutoPQ-advanced over these methods. Table 3.1 shows the percentage improvement of AutoPQ-advanced to the six considered benchmarks and AutoPQ-default for each data set. The improvement values are marked with an asterisk if the improvement is significant, i. e., the p-value of the one-tailed t-test is smaller than 0.05. In this table,

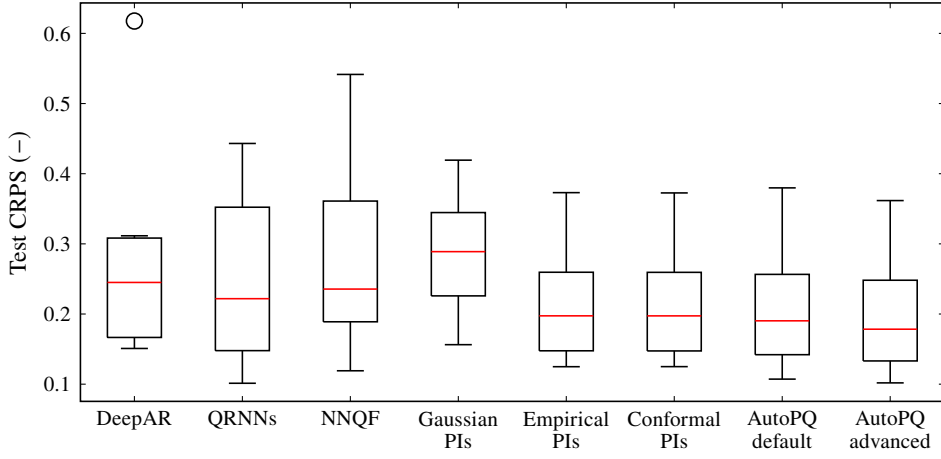


Figure 3.5: The CRPS evaluated on the hold-out test data sets with methods categorized into three classes: i) direct probabilistic benchmark methods, ii) point forecast-based probabilistic benchmark methods, and iii) AutoPQ with the default and the advanced configuration. The numerical values aggregated in this figure are detailed in Table B.7 in the Appendix.

Table 3.1: The percentage improvement of AutoPQ-advanced over the comparison methods in terms of the CRPS evaluated on the hold-out test data sets. AutoPQ-advanced uses HPO to optimize both the configuration of the point forecasting method λ_p and the cINN's sampling hyperparameter λ_q , while AutoPQ-default only optimizes λ_q . The values are marked with an asterisk if the improvement is significant (p-value < 0.05). The table is based on [165]. The actual p-values of the one-tailed t-test are detailed in Table B.8 in the Appendix.

	DeepAR	QRNNs	NNQF	Gaussian PIs	Empirical PIs	Conformal PIs	AutoPQ default
Load-BW	28.1 % *	4.8 % *	33.7 % *	11.5 % ₀ ⁽¹⁾	3.5 % ₀ ⁽¹⁾	3.5 % ₀ ⁽¹⁾	6.1 % ₀ ⁽¹⁾
Load-GCP	30.1 % *	24.0 % *	17.1 % *	27.1 % ₀ ⁽¹⁾	6.8 % ₀ ⁽¹⁾	6.8 % ₀ ⁽¹⁾	6.8 % ₀ ⁽¹⁾
Mobility	13.7 % *	41.8 % *	52.4 % *	28.3 % ₀ ⁽²⁾	3.7 % ₀ ⁽²⁾	3.7 % ₀ ⁽²⁾	1.9 % ₀ ⁽²⁾
Price	17.1 % *	16.6 % *	28.4 % *	53.0 % ₀ ⁽²⁾	18.6 % ₀ ⁽²⁾	18.6 % ₀ ⁽²⁾	6.4 % ₀ ⁽²⁾
PV	33.1 % *	0.0 %	15.1 % *	51.4 % ₀ ⁽¹⁾	19.2 % ₀ ⁽¹⁾	19.2 % ₀ ⁽¹⁾	4.7 % ₀ ⁽¹⁾
WP	41.4 % *	3.2 % *	8.1 % *	13.6 % ₀ ⁽¹⁾	2.9 % ₀ ⁽¹⁾	2.9 % ₀ ⁽¹⁾	4.0 % ₀ ⁽¹⁾
Mean	27.3 %	15.1 %	25.8 %	30.8 %	9.1 %	9.1 %	5.0 %

* improvement is significant (p-value < 0.05); best-performing base point forecasting method: (1) XGB, (2) TFT

three findings can be made: First, we see that the improvement is significant in 38 out of 42 tests (seven methods, six data sets). Second, AutoPQ-advanced improves over each direct probabilistic benchmark (on average by at least 15.1 %) and each point forecast-based probabilistic benchmark (on average by at least 9.1 %). Third, the average improvement of AutoPQ-advanced over AutoPQ-default amounts to 5.0 %.

Insights

A visual comparison of PIs resulting from AutoPQ-advanced with the nine considered point forecasting methods is exemplified in Figure 3.6 for the Mobility data set. For each method, the PIs are generated with the best-performing configuration determined during the CASH with successive halving, where the TFT prevails for the Mobility data set (as will be shown later in Figure 3.14). In Figure 3.6, we observe that the nine point forecasting methods result in PIs that differ in the width of the PIs (sharpness). Specifically, a sharp PI is narrow, meaning it indicates low uncertainty. Visually, clear differences can be recognized between methods with strongly differing CRPS (e. g. ETS vs. TFT). In contrast, differences between methods with comparable CRPS are less visible (e. g. N-HITS vs. TFT).

Another observation concerns the PIs at times when the realized values are zero or near zero. Since the mobility indicator is limited to positive real numbers including zero, negative PIs are unreasonable. The PIs generated with AutoPQ consider this restriction compared to the PIs of benchmarks visualized in Figure 3.7.⁸ However, the impact of post-processing AutoPQ’s quantiles to comply with the non-zero condition on the CRPS is not significant, as shown in Table B.9 in the Appendix.

3.2.2 Customizability

In the following, we first describe the experimental setup for evaluating the customizability of AutoPQ-advanced. Afterward, we show the results and provide insights.

Experimental setup

The experimental setup outlines the used data and describes the evaluation strategy.

Data. We evaluate the customizability of forecasting properties as in the benchmarking on six different data sets: Load-BW, Load-GCP, Mobility, Price, PV, and WP. Brief information about these data sets are given in Section 3.2.1.

⁸ A visual comparison over a period of one week is given in Figure B.1 in the Appendix.

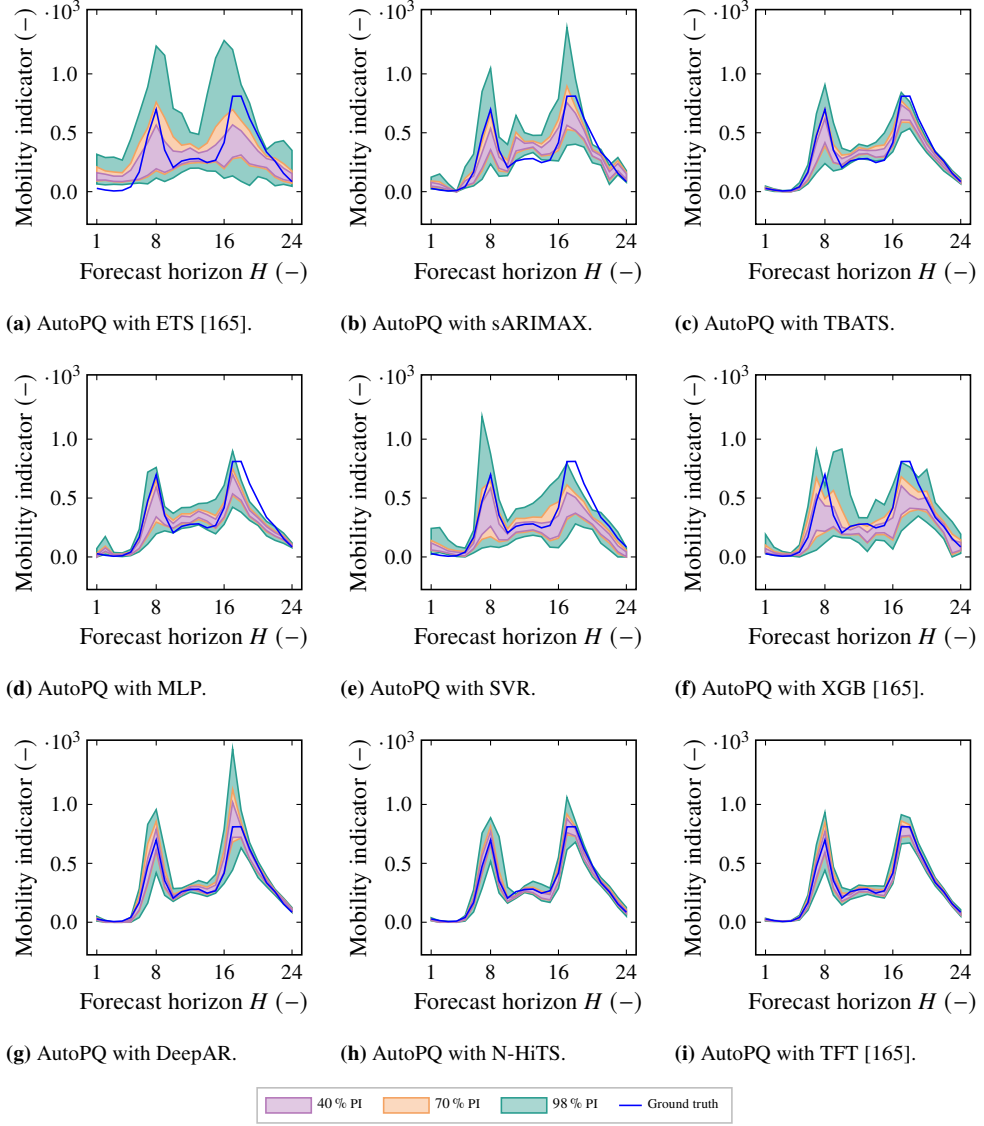


Figure 3.6: Exemplary 40 %, 70 %, and 98 % PIs of a day-ahead forecast with origin at 00:00 for the Mobility data set. The probabilistic forecasts are generated by using AutoPQ-advanced with the nine considered point forecasting methods, specifically, three of each for the method families SM (sARIMAX, ETS, TBATS), ML (MLP, SVR, XGB), and DL (DeepAR, N-HiTS, TFT). The configuration of the point forecasting method λ_p and the sampling hyperparameter λ_q are optimized jointly to minimize the CRPS. The figure is based on [165].

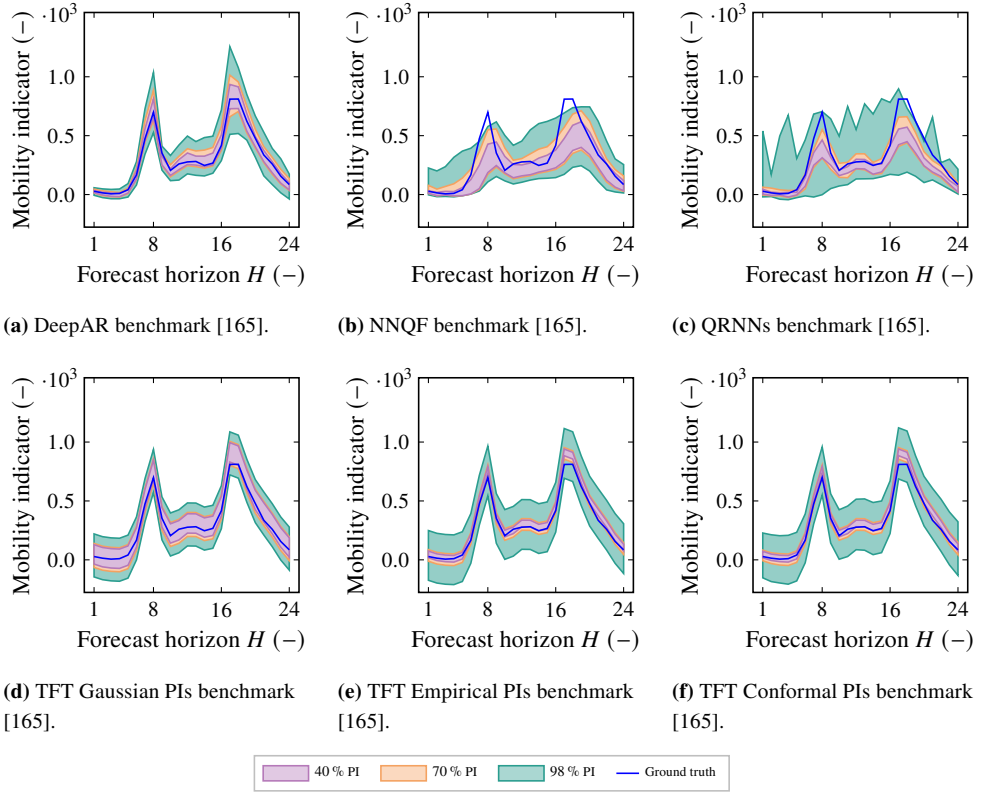


Figure 3.7: Exemplary 40 %, 70 %, and 98 % PIs of a day-ahead forecast with origin at 00:00 for the Mobility data set. The probabilistic benchmarks can be classified into direct probabilistic methods (DeepAR, NNQF, QRNNs) and point forecast-based probabilistic methods (Gaussian PIs, Empirical PIs, Conformal PIs based on a TFT forecaster). The figure is based on [165].

Evaluation strategy. We evaluate the customizability of day-ahead forecasts with AutoPQ-advanced using the six data sets described above. As in the benchmarking, the data sets are normalized and split into separate training, validation, and test data sets, which are given in Table B.6 in the Appendix.

To reduce the computational effort, we limit the study to the MLP and N-HiTS as point forecasting methods without successive halving. The selection is based on three properties: the computational effort, the performance, and the sensitivity to HPO. In terms of the first property, the MLP requires the lowest computational effort and the N-HiTS requires a moderate to high computational effort, as shown previously in Figure 3.2. Regarding the second and third properties, both achieve good performance in the benchmarking and respond well to HPO, as will be shown later in Table 3.4.

To evaluate the customizability of probabilistic properties, we consider the CRPS and the Mean Absolute Quantile Deviation (MAQD). The respective metric is used in the HPO to find the configuration that optimizes that metric on the validation data set. While the CRPS (1.8) is a metric that trades-off sharpness and calibration, the MAQD (1.9) only considers the calibration of the probabilistic forecast. For comparing these probabilistic performance metrics to a point forecasting metric, we consider the Mean Squared Error (MSE) (1.6). Further, we compare to the probabilistic PL (??).

To make stochastic effects during training and optimization transparent, we run the evaluation on each data set five times using the same hardware, as described in Section 3.2.1.

Results

In the following, we visualize and summarize the results of the customizability evaluation. An interpretation and discussion of these results is given in the next Section 3.3.

To determine the effect of the probabilistic performance metric used to determine the optimal configuration on the validation data set, we compare the performance on the test data set regarding other metrics. Table 3.2 shows the results for the MLP point forecasting method and Table 3.3 the results for N-HiTS. For both the MLP and N-HiTS, four observations can be made: First, forecasts optimized with the CRPS also result in low CRPS values on the test data set, while forecasts optimized with the MAQD result in higher CRPS values. Second, this is conversely the case for forecasts that are optimized for MAQD. Third, for five out of six data sets, the CRPS -optimized forecasts also result in a low PL, while optimizing for the MAQD increases the PL. An exception is the Load-BW data set, where optimizing for the MAQD leads to a slightly better PL than optimizing for CRPS. Fourth, the optimization for CRPS leads to lower MSE values compared to optimizing for MAQD for all data sets except for the Load-BW data set. Here, the MAQD optimization achieves a lower MSE value for N-HiTS, while for the MLP the CRPS optimization achieves a lower MSE value, as for the other data sets. For all four observations, the strength of the influence differs depending on the data set.

Insights

To visualize the impact of different metrics used for HPO, we compare the validation performances of all evaluated configurations. Specifically, we plot the point forecast's MSE over the probabilistic forecast's CRPS in Figure 3.8, and the MAQD in Figure 3.9.⁹

Comparing the relationship between CRPS and MSE in Figure 3.8, two findings can be made: First, a low CRPS generally also leads to a low MSE. Second, the lowest CRPS,

⁹ Since we already identified a strong correlation between CRPS and PL in Table 3.2 and 3.3, we omit this plot.

Table 3.2: The impact of using the CRPS and the MAQD as validation metrics in the HPO of AutoPQ-advanced with the MLP point forecasting method on different performance metrics evaluated on the hold-out test data sets and averaged across five runs. The best average value for each data set is highlighted in bold.

Val. metric	Test metric	Load-BW	Load-GCP	Mobility	Price	PV	WP
CRPS	CRPS	0.152 ± 0.005	0.246 ± 0.002	0.391 ± 0.006	0.158 ± 0.003	0.119 ± 0.001	0.406 ± 0.013
MAQD		0.157 ± 0.004	0.346 ± 0.045	0.568 ± 0.054	0.173 ± 0.009	0.159 ± 0.018	0.527 ± 0.019
CRPS	MAQD	0.105 ± 0.008	0.093 ± 0.004	0.151 ± 0.007	0.092 ± 0.011	0.158 ± 0.009	0.108 ± 0.014
MAQD		0.047 ± 0.018	0.034 ± 0.009	0.039 ± 0.015	0.053 ± 0.017	0.012 ± 0.005	0.032 ± 0.011
CRPS	PL	0.071 ± 0.002	0.111 ± 0.001	0.178 ± 0.003	0.071 ± 0.002	0.055 ± 0.001	0.184 ± 0.007
MAQD		0.069 ± 0.002	0.136 ± 0.016	0.233 ± 0.019	0.073 ± 0.004	0.063 ± 0.006	0.217 ± 0.007
CRPS	MSE	0.091 ± 0.004	0.228 ± 0.003	0.652 ± 0.032	0.146 ± 0.007	0.089 ± 0.001	0.497 ± 0.025
MAQD		0.096 ± 0.006	0.279 ± 0.042	1.183 ± 0.123	0.167 ± 0.012	0.109 ± 0.006	0.753 ± 0.077

Table 3.3: The impact of using the CRPS and the MAQD as validation metrics in the HPO of AutoPQ-advanced with the N-HiTS point forecasting method on different performance metrics evaluated on the hold-out test data sets and averaged across five runs. The best average value for each data set is highlighted in bold.

Val. metric	Test metric	Load-BW	Load-GCP	Mobility	Price	PV	WP
CRPS	CRPS	0.151 ± 0.004	0.255 ± 0.006	0.269 ± 0.004	0.131 ± 0.002	0.140 ± 0.003	0.513 ± 0.012
MAQD		0.155 ± 0.005	0.375 ± 0.019	0.375 ± 0.016	0.156 ± 0.007	0.201 ± 0.030	0.682 ± 0.058
CRPS	MAQD	0.107 ± 0.011	0.069 ± 0.005	0.111 ± 0.008	0.097 ± 0.013	0.163 ± 0.004	0.134 ± 0.015
MAQD		0.033 ± 0.009	0.009 ± 0.002	0.017 ± 0.001	0.043 ± 0.014	0.016 ± 0.004	0.022 ± 0.004
CRPS	PL	0.070 ± 0.002	0.113 ± 0.003	0.122 ± 0.002	0.060 ± 0.001	0.064 ± 0.001	0.234 ± 0.005
MAQD		0.067 ± 0.002	0.154 ± 0.007	0.153 ± 0.006	0.066 ± 0.003	0.078 ± 0.009	0.276 ± 0.019
CRPS	MSE	0.085 ± 0.003	0.270 ± 0.023	0.288 ± 0.003	0.114 ± 0.002	0.114 ± 0.001	0.832 ± 0.100
MAQD		0.082 ± 0.003	0.428 ± 0.024	0.424 ± 0.020	0.173 ± 0.025	0.157 ± 0.014	1.401 ± 0.087

however, does not necessarily lead to the lowest MSE. In this context, a considerable dependency on the data set can be recognized. While the relationship between CRPS and MSE in the Load-BW data set is approximately linear, the relationship in the PV data set is weak. Furthermore, a dependency on the point forecasting method is visible. For example in the Price data set, N-HiTS achieves considerably better CRPS values on the same MSE level than the MLP.

Comparing the relationship between MAQD and MSE in Figure 3.9, and considering the findings above, the relationship between MAQD and MSE is weaker than the relationship between CRPS and MSE. However, we also observe different relationships depending on the data set and the point forecasting method. In the Load-BW data set in Figure 3.9a, similar

MAQD values can be achieved for the MLP and N-HiTS, whereby N-HiTS achieves slightly lower MSE values. Contrarily, in the Load-GCP data set in Figure 3.9b, the MLP achieves lower MSE values than N-HiTS, whereby N-HiTS achieves notably lower MAQD values (the opposite is the case for the CRPS in Figure 3.8b).

To visualize the above finding, we exemplify the resulting PIs when using the CRPS and MAQD as validation metrics in the HPO for the Load-GCP data set in Figure 3.10. Comparing Figure 3.10c and 3.10d, N-HiTS shows smoother PIs than the MLP, which are wider compared to the PIs when optimizing the CRPS in Figure 3.10b. Comparing Figure 3.10b and 3.10a, in turn, illustrates the impact of optimizing the CRPS, resulting in sharper PIs with few realizations outside the 98 % interval.

3.2.3 Ablation study

In the following, we first describe the experimental setup for the ablation study to evaluate the effectiveness of automation methods applied in AutoPQ-advanced, before showing the results and providing insights.

Experimental setup

In the experimental setup, we describe the used data, the evaluation strategy, and the considered ablations.

Data. As in the benchmarking, the ablations of AutoPQ-advanced are evaluated on six different data sets: Load-BW, Load-GCP, Mobility, Price, PV, and WP. Brief information about these data sets are given in Section 3.2.1.

Evaluation strategy. The ablation study is performed using a similar evaluation strategy applied in the previous evaluations. That is, we use the six data sets described in Section 3.2.1, normalize the data sets, and create separate training, validation, and test data sets, which are detailed in Table B.6 in the Appendix.

As in the evaluation of the customizability of AutoPQ’s day-ahead forecasts, we limit the study to the MLP and N-HiTS as point forecasting methods to reduce the computational effort. Reasons for this selection are given in Section 3.2.2.

As in the benchmarking, the CRPS (1.8) is used as a validation and test metric, and the evaluation on each data set is run five times. Since the evaluation focuses on the runtime, the same hardware is used for all runs, as described in Section 3.2.1.

Ablation 1. We assess the effectiveness of Algorithm 2’s inner loop, responsible for finding the optimal sampling hyperparameter λ_q^* . For comparison, we use a conventional

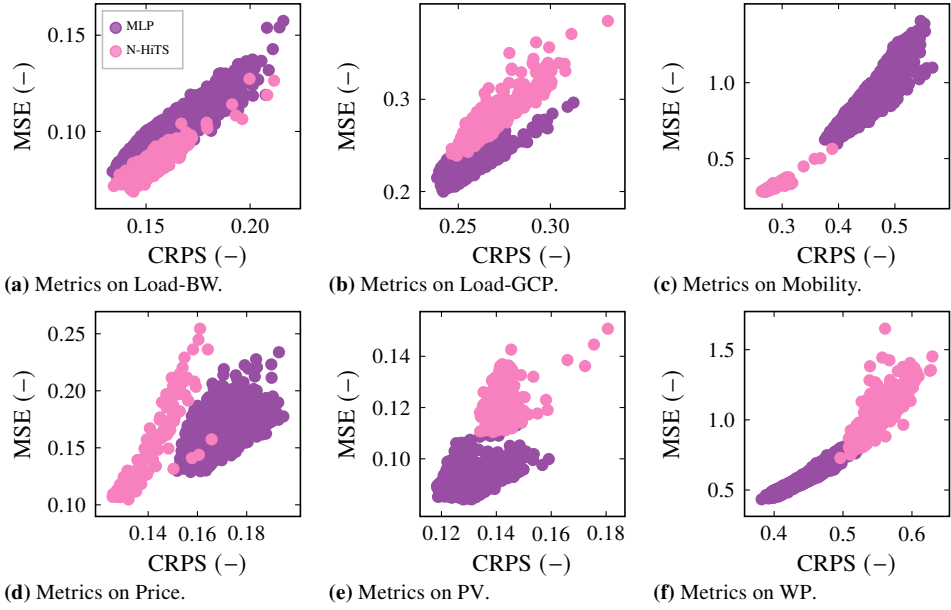


Figure 3.8: Comparison of the MSE (point forecast) with the CRPS (probabilistic forecast) on the six data sets in the HPO of AutoPQ-advanced. Each dot represents the performance of a trial configuration in the HPO and legend of (a) applies to all sub-plots.

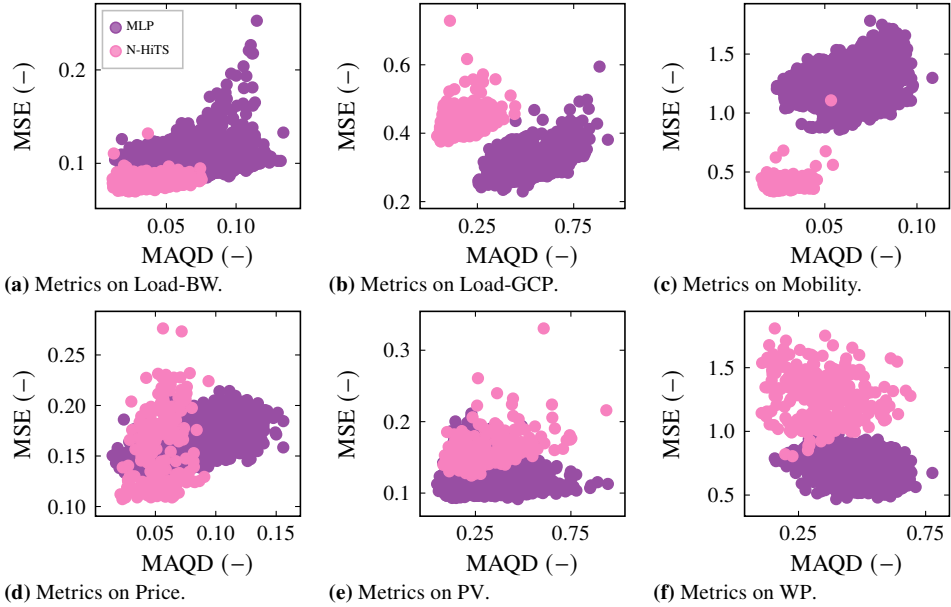


Figure 3.9: Comparison of the MSE (point forecast) with the MAQD (probabilistic forecast) on the six data sets in the HPO of AutoPQ-advanced. Each dot represents the performance of a trial configuration in the HPO and legend of (a) applies to all sub-plots.

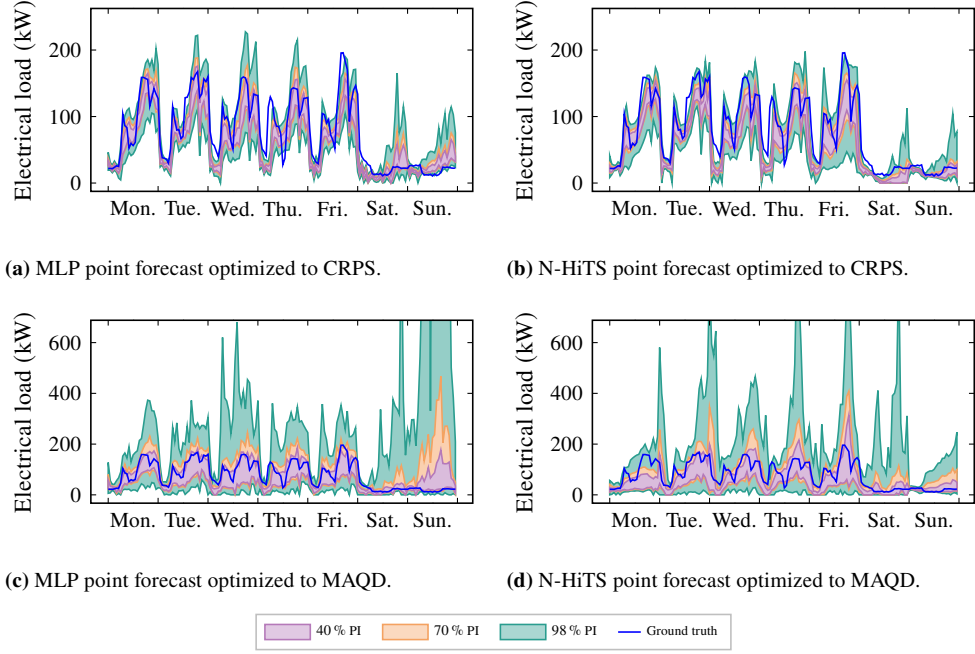


Figure 3.10: Exemplary 40 %, 70 %, and 98 % PIs of seven day-ahead forecasts with origin at 00:00 each for the Load-GCP data set. The probabilistic forecasts are generated by using AutoPQ-advanced with the MLP and N-HITS as point forecasting methods, respectively. The configuration of the point forecasting method λ_p and the sampling hyperparameter λ_q are optimized jointly to minimize the CRPS and the MAQD, respectively.

HPO, which does not consider the different computational efforts of training the point forecasting model and generating a probabilistic forecast based on the point forecast. More precisely, the inner loop of Algorithm 2 is absent, resulting in Algorithm 3 given in the Appendix. In this way, the trial scheduler suggests both $\lambda = \lambda_p \cup \lambda_q$ the configuration of the point forecasting method λ_p and the sampling hyperparameter λ_q ; consequently, only one sampling hyperparameter is assessed for each trained point forecasting model. To make results comparable, we run both Algorithm 2 and 3 without successive halving using the full time budget $B_t = 8$ h, considering the same configuration spaces.

Ablation 2. We compare different trial schedulers for the inner loop of Algorithm 2. Specifically, we compare PK-based BO to random search and BO, both without using PK. For this purpose, we compare the number of iterations until the early stopping condition described in Section 3.1.3 is fulfilled. To prevent having to repeat the entire CASH for all six data sets five times, considering all nine forecasting methods, which would consume an enormous amount of computing resources, we only repeat Algorithm 2’s inner loop. That

is, we repeat the inner loop for all six data sets five times with the best configuration of each forecasting method using the two comparison trial schedulers.

Ablation 3. We evaluate the decision quality of the successive halving, i. e., the impact of pruning underperforming forecasting methods in the CASH. For this purpose, we compare the evolution of the forecasting methods' validation performances when using successive halving to the performance evolution as it would have developed without pruning. Specifically, each pruned forecasting method of the benchmarking (see Section 3.2.1) is additionally continued until the full time budget $B_t = 8$ h is exhausted. In this way, we analyze whether the successive halving keeps forecasting methods that end up with good performance and prunes methods whose performance would not improve even with more computational budget. In addition to the decision quality of the successive halving, we also evaluate how well the nine forecasting methods respond to the HPO, i. e., the improvement over the default configuration.

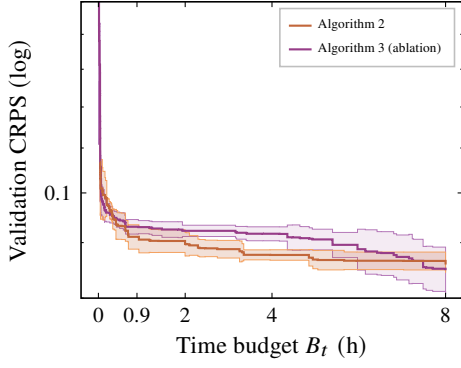
Results

In the following, we visualize and summarize the results of the ablation study. An interpretation and discussion of these results is given in the next Section 3.3.

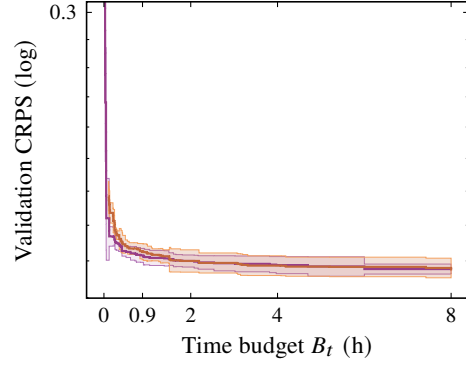
Ablation 1. The results of the effectiveness evaluation concerning Algorithm 2 and 3 (ablation) are visualized in Figure 3.11 for the MLP and in Figure 3.12 for N-HITS. These figures show the evolution of the CRPS evaluated on the validation data set over the HPO budget. To visualize the validation performance in the context of successive halving, the time budget marks corresponding to the pruning rounds are shown. For both figures, the following two points are noticeable: First, the mean value of Algorithm 2 lies below the mean value of Algorithm 3 for almost the whole HPO time budget. This observation applies to all data sets and both methods. Second, the mean value of Algorithm 2 lies especially at the beginning of the HPO often significantly below the mean value of Algorithm 3. This gap is only reduced at half to the end of the HPO time budget, and for most data sets, it is visible that both values converge toward the same value.

Ablation 2. The results of the comparison of different trial schedulers for the inner loop of Algorithm 2 are visualized in Figure 3.13. The sub-figures show the number of iterations required until the early stopping condition is fulfilled for each of the six data sets.¹⁰ For all data sets, three observations can be made: First, using the BO with the TPE surrogate model

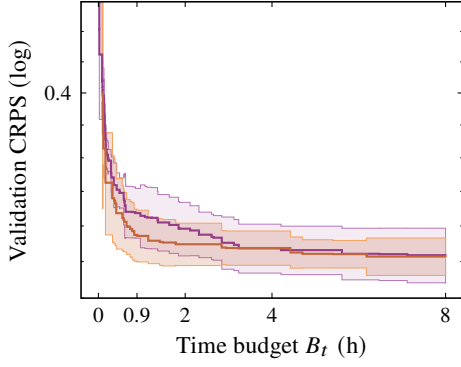
¹⁰ The computational effort increases approximately linearly with the number of iterations, as the effort for initialization is negligible.



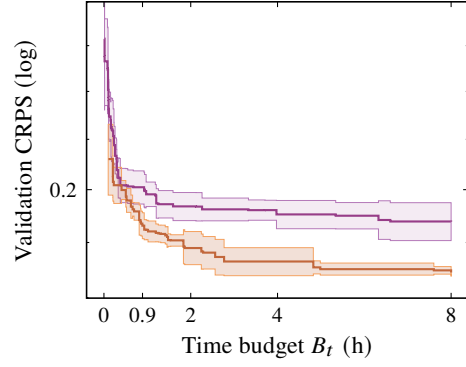
(a) Convergence of MLP on Load-BW.



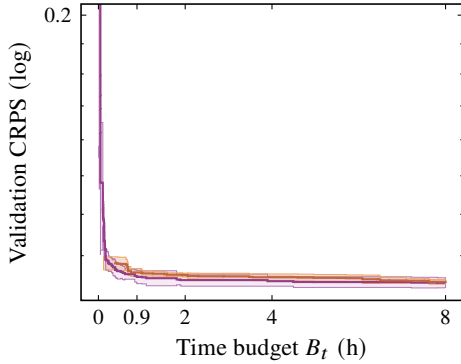
(b) Convergence of MLP on Load-GCP.



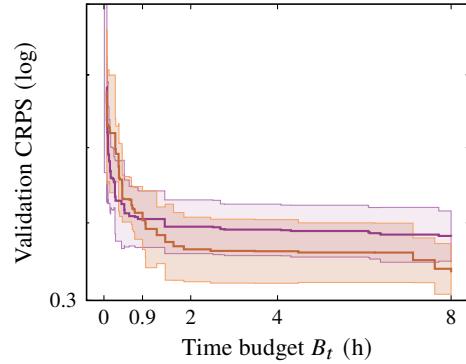
(c) Convergence of MLP on Mobility.



(d) Convergence of MLP on Price [165].

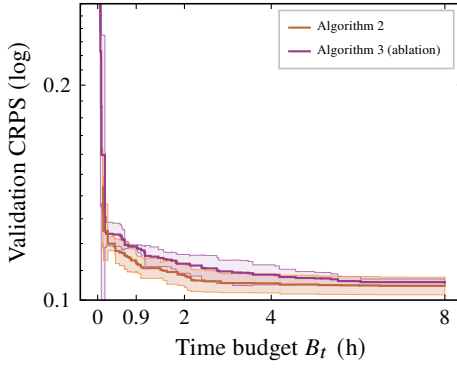


(e) Convergence of MLP on PV.

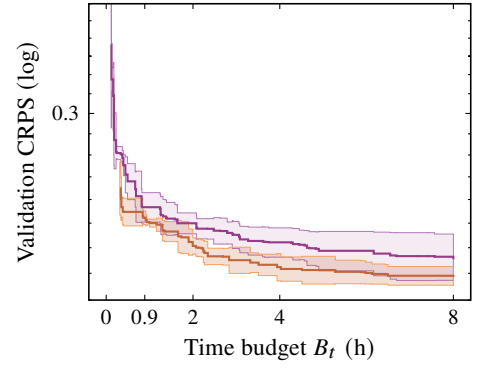


(f) Convergence of MLP on WP.

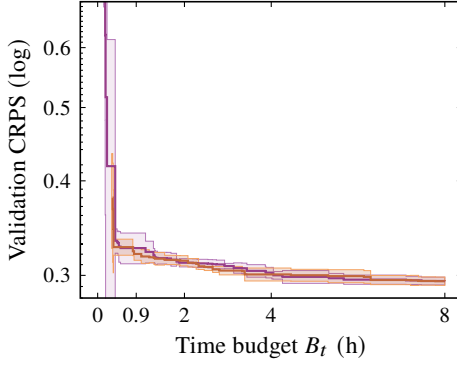
Figure 3.11: Comparison of the convergence of Algorithm 2 and 3 (ablation) for the HPO of MLP on the six data sets. The thick solid line represents the mean value, and the opaque area is the standard deviation over five runs [165]. Legend of (a) applies to all sub-plots.



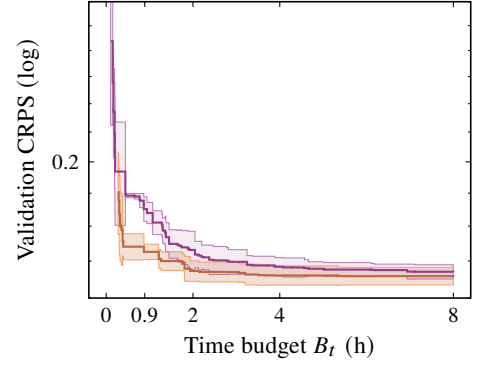
(a) Convergence of N-HiTS on Load-BW.



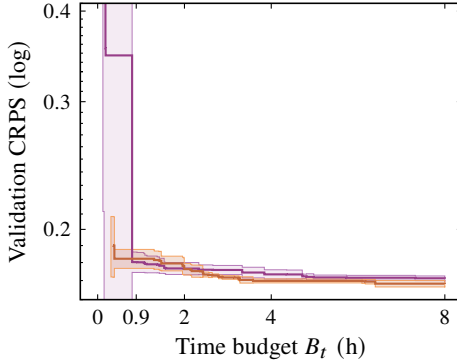
(b) Convergence of N-HiTS on Load-GCP.



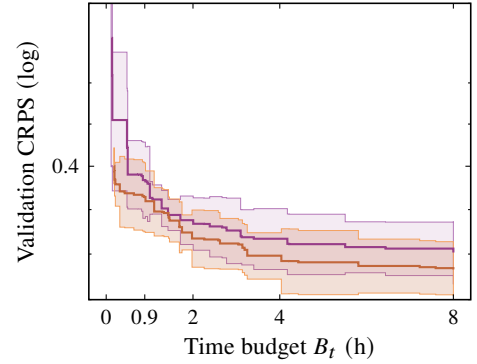
(c) Convergence of N-HiTS on Mobility.



(d) Convergence of N-HiTS on Price [165].



(e) Convergence of N-HiTS on PV.



(f) Convergence of N-HiTS on WP.

Figure 3.12: Comparison of the convergence of Algorithm 2 and 3 (ablation) for the HPO of N-HiTS on the six data sets. The thick solid line represents the mean value, and the opaque area is the standard deviation over five runs [165]. Legend of (a) applies to all sub-plots.

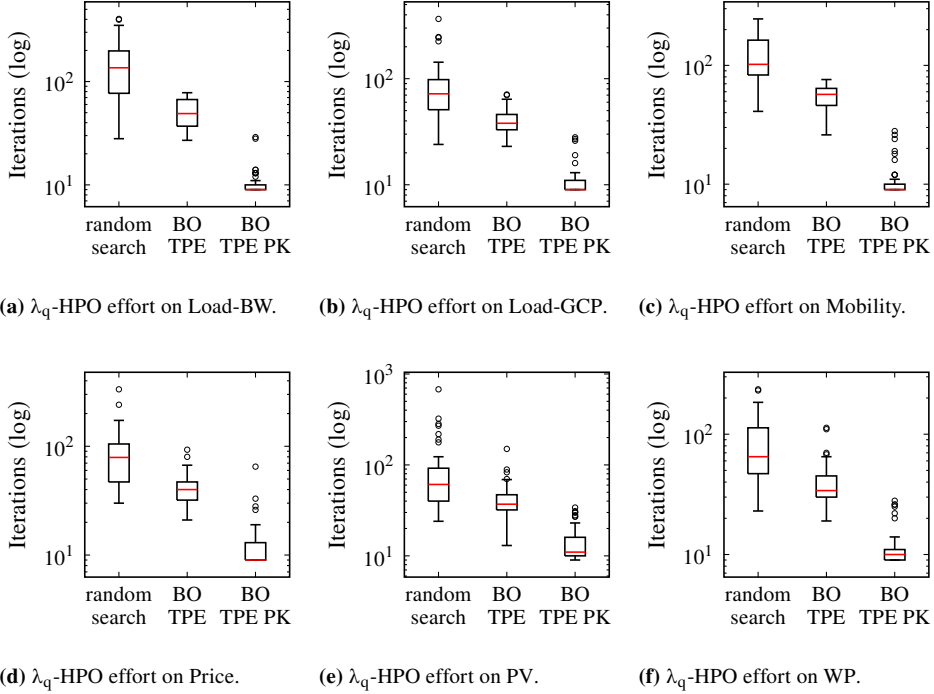


Figure 3.13: The number of iterations required for the inner loop of Algorithm 2 for optimizing the sampling hyperparameter λ_q until the early stopping criteria is fulfilled compared for three trial schedulers: random search, BO-TPE, and BO-TPE-PK. Each box plot considers all nine forecasting methods and five runs. The figure is based on [165].

already reduces the number of iterations compared to random search. Second, integrating PK into BO-TPE, in turn, leads to a notable reduction of iterations. Even the few upper outliers lie in the range of the lower whisker of BO-TPE without PK. Third, the three trial schedulers achieve comparable validation CRPS, as detailed in Table B.10 in the Appendix. That is, BO-TPE-PK achieves a similar CRPS as the two comparative methods but requires only a fraction of iterations.

Ablation 3. The results of the successive halving ablation study are shown in Figure 3.14,¹¹ where three findings stand out: First, the best-performing forecasting method prevails in all data sets, i. e., the best-performing one is not falsely deactivated during pruning. Second, this best-performing method is already the best-performing one in AutoPQ-default, while changes in the following ranks occur. Third, although SM methods often improve significantly in the first pruning round of successive halving (AutoPQ-advanced) compared to their default

¹¹ Figure 3.14 shows the results of the first of five runs, which are representative for the other runs.

configuration (AutoPQ-default), they still end up last even if they were given more time budget. In contrast, the upper and middle ranks are contested between ML and DL methods.

Insights

The effect of using PK for the BO with the TPE surrogate model is exemplified in Figure 3.15 for each of the six data sets. The plots show the CRPS on the validation data set and the prior distribution over the sampling hyperparameter λ_q for BO-TPE without and with PK. While the former BO-TPE assumes a continuous uniform PDF $\mathcal{U}(0, 3)$, the latter BO-TPE-PK assumes a log-normal PDF $\log - \mathcal{N}(m, s)$, whose parameters are derived from the EA population. For both, the PDF in Figure 3.15 is normalized to its maximum value for visualization. The plots exemplarily show the HPO for one XGB configuration, however, they are representative in terms of the other point forecasting methods regarding the following findings: The optimal sampling hyperparameter λ_q^* highly depends on the data set. Not only does the location differ, but also the sensitivity. For example, the validation CRPS valley that is apparent in Figure 3.15e is flatter than the valley in Figure 3.15a. Concurrently, the prior distribution of BO-TPE-PK shows a higher, respectively, lower spread. Apart from those findings, the figures confirm the findings of Figure 3.13 that BO-TPE-PK requires significantly fewer iterations until the early stopping condition is fulfilled. It is visible that the condition is often already fulfilled before exploitation and exploration are started being traded-off based on the TPE surrogate model that is fitted on the observations.

Taking a detailed look at the improvement in validation CRPS performance from Figure 3.14 without pruning, we can determine the percentage improvement in Table 3.4. The improvement values are marked with an asterisk if the improvement is significant, i. e., the p-value of the one-tailed t-test is smaller than 0.05. In this table, three findings can be made: First, the percentage improvement in SM methods is very inconsistent; on the one hand, large and significant improvements exist, but on the other hand, there are also little to no improvements. Second, in the ML methods, all three methods achieve significant improvements, with the MLP responding best to the HPO with an average improvement of 12.3 %. Third, among the DL methods, only DeepAR and N-HiTS achieve significant improvements with an average improvement of 14.7 % and 15.8 % respectively, while the TFT's response to the HPO is weak and not significant for five out of six data sets.

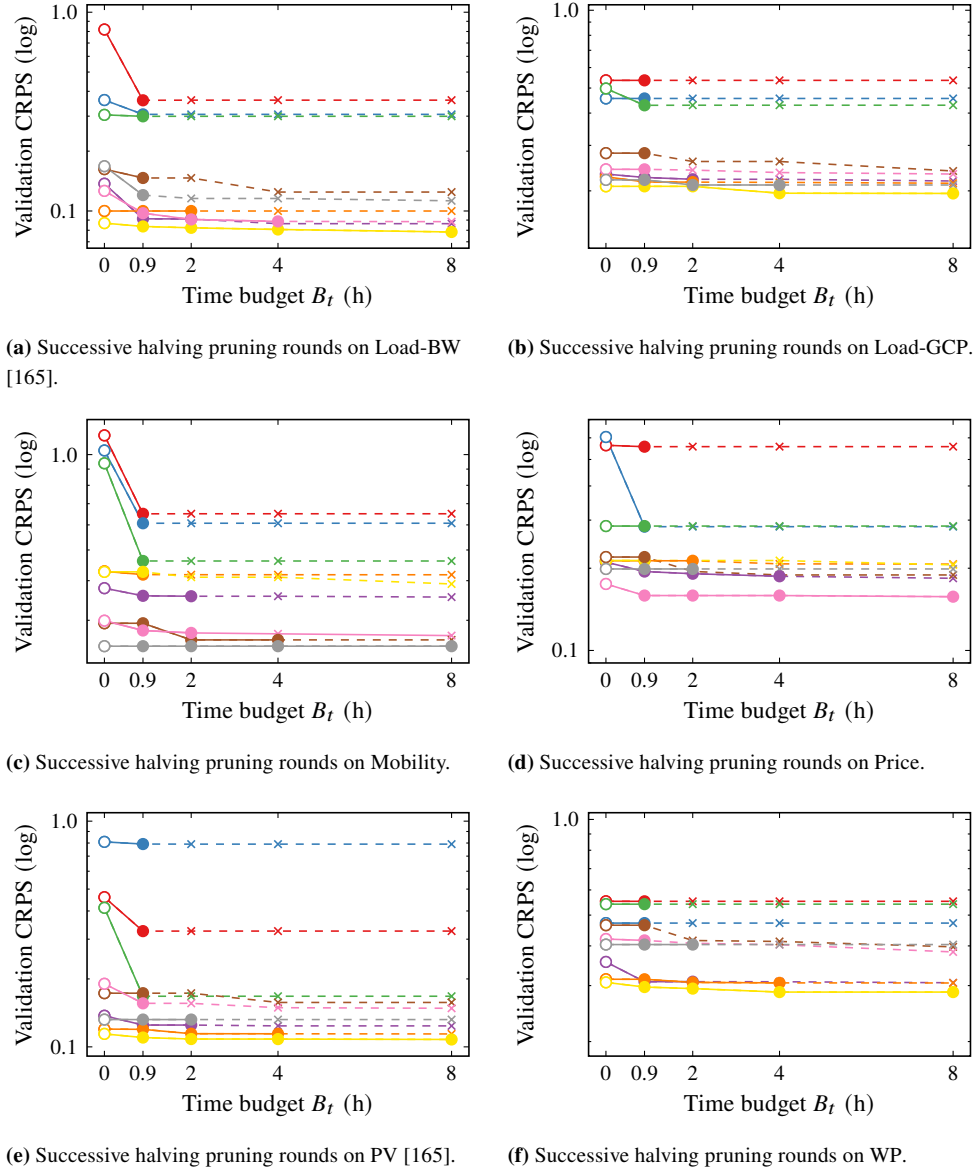


Figure 3.14: Evolution of the considered forecasting methods' validation performances in the successive halving-based CASH of AutoPQ-advanced with the performance at 0h reflecting AutoPQ-default, shown as a reference and therefore not considered within the time budget B_t . The solid lines with dot markers show the performance of active methods, while the dotted lines with cross markers show the performance of inactive methods as it would have evolved without pruning. The progress in the successive halving pruning rounds is assumed to be linear for simplicity's sake [165].

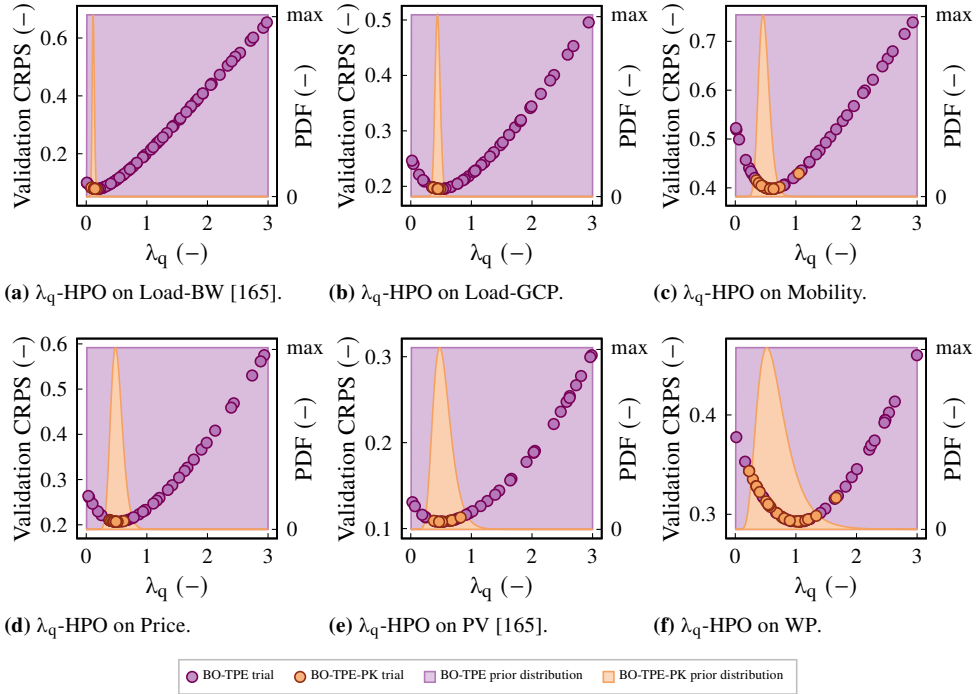


Figure 3.15: The HPO of the sampling hyperparameter λ_q with the inner loop of Algorithm 2 using BO-TPE and BO-TPE-PK. The validation performance (CRPS) over the trial configuration (λ_q) is shown as a dot, and the TPE prior distribution is shown as a filled area normalized by the maximum of the respective PDF. The plots exemplarily show the HPO for one XGB configuration but are representative in terms of the other point forecasting methods [165].

Table 3.4: The percentage improvement of each optimized configuration over the default configuration in the successive halving ablation study without pruning in terms of the CRPS evaluated on the validation data. The values are marked with an asterisk if the improvement is significant (p-value < 0.05). The actual p-values of the one-tailed t-test are detailed in Table B.11 in the Appendix. The table is based on [165].

	ETS	sARIMAX	TBATS	MLP	SVR	XGB	DeepAR	N-HiTS	TFT
Load-BW	57.7 % *	12.9 % *	0.5 %	29.2 % ₍₃₎ *	0.3 %	7.5 % ₍₁₎ *	25.6 % *	30.5 % ₍₂₎ *	23.5 % *
Load-GCP	<0.1 %	<0.1 %	8.0 %	6.6 % *	4.1 % ₍₃₎ *	4.8 % ₍₁₎ *	16.7 % *	4.8 % *	6.7 % ₍₂₎
Mobility	42.3 % *	40.5 % *	45.0 % *	4.9 % *	4.4 % *	6.8 % *	8.9 % ₍₂₎ *	14.2 % ₍₃₎ *	1.0 % ₍₁₎
Price	2.2 %	52.5 % *	<0.1 %	8.9 % ₍₂₎ *	2.3 % *	2.5 % *	7.9 % *	13.7 % ₍₁₎ *	5.3 % ₍₃₎
PV	37.5 % *	14.3 %	56.7 % *	10.2 % ₍₃₎ *	4.3 % ₍₂₎ *	6.2 % ₍₁₎ *	13.8 % *	22.0 % *	6.9 %
WP	<0.1 %	<0.1 %	<0.1 %	13.9 % ₍₃₎ *	3.6 % ₍₂₎ *	5.6 % ₍₁₎ *	15.0 % *	9.3 % *	8.5 %

* improvement is significant (p-value < 0.05); (1), (2), (3): rank number in successive halving

3.3 Discussion

This section discusses the benchmarking results, AutoPQ’s customizability, the ablation studies results, as well as the limitations and benefits.

3.3.1 Benchmarking

Regarding the results of the benchmarking on six data sets to assess the probabilistic day-ahead forecasting performance, we discuss two aspects: First, we observe a higher performance spread of direct forecasting methods (DeepAR, QRNNs, NNQF) than probabilistic forecasting methods based on point forecasts (Gaussian, empirical, and conformal PIs, as well as AutoPQ-default and AutoPQ-advanced). The reduced performance spread for the latter can be explained by the fact that multiple point forecasting methods are considered on which the probabilistic forecast is based, and the best-performing one is selected. The performance-based selection allows only the best method to prevail, which concurrently hedges against poor performance if specific methods underperform on a data set. Second, the results reveal that AutoPQ-advanced outperforms AutoPQ-default and all benchmarks. More specifically, the average improvement of AutoPQ-advanced over AutoPQ-default exceeds the improvement already achieved by AutoPQ-default compared to the direct probabilistic benchmarks and the point forecast-based probabilistic benchmarks. Since this improvement is significant for five out of six data sets, it is recommended to use AutoPQ-advanced for performance-critical smart grid applications with high decision costs.

3.3.2 Customizability

In terms of the customizability evaluation, we discuss two findings: First, it can be seen that the chosen probabilistic validation metric for HPO has a substantial impact on the forecast’s probabilistic properties, and a clear correlation between the point forecasting metric MSE and the probabilistic metrics CRPS and MAQD, respectively, does not always exist. Therefore, it can be concluded that the optimization of the desired probabilistic properties requires the selection of the best-suited point forecasting method together with the joint optimization of its hyperparameters λ_p , and the sampling hyperparameter λ_q . Second, in the exemplary comparison between the probabilistic metrics CRPS and MAQD, we see that using the CRPS results in both sharp forecasts subject to calibration, while the MAQD solely measures the calibration resulting in much wider PIs. Wider PIs can be crucial for smart grid applications that require hedging against extreme scenarios. A probabilistic forecast that is too sharp could be misleading and underestimate the risk, and a less-sharp forecast that provides more calibrated quantiles would be preferable. Importantly, calibration alone may be an insufficient measure, as it does not consider the forecast error.

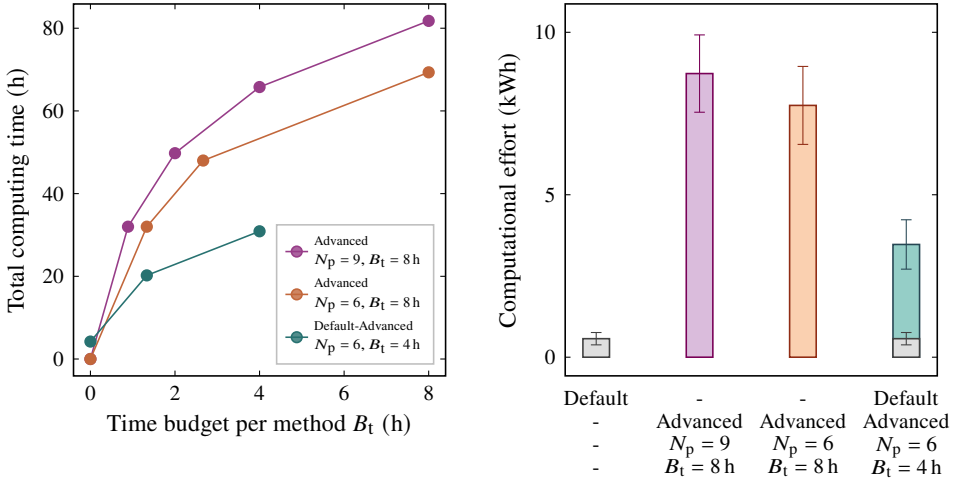
3.3.3 Ablation study

In the ablation study, we discuss the results of the three ablations: First, the performance advantage of Algorithm 2 over Algorithm 3 (ablation), especially at the beginning of the HPO, is crucial regarding the successive halving. More specifically, using Algorithm 3 could prune forecasting methods that would actually perform well, particularly because Algorithm 3 also shows for a high performance spread for the first and second pruning round of successive halving. This spread can be explained by the fact that the EA requires several generations until a suitable sampling hyperparameter is found, and the random initialization can result in both suitable and unsuitable parameters. This effect is reduced in Algorithm 2, as the most suitable sampling hyperparameter is determined for each trained model. Because the above finding applies for both N-HiTS with moderate to high computational effort and for the MLP with very low computational effort, we conclude that the inner loop of Algorithm 2 is also effective for the other point forecasting methods. Second, using PK in the inner loop of Algorithm 2 to determine the most suitable sampling hyperparameter for each trained model significantly reduces the required number of iterations. We assume that keeping the number of iterations low is necessary to gain the above-discussed performance advantage of Algorithm 2 over Algorithm 3. The reason is that the inner loop of Algorithm 2 is only effective if the computational effort of finding the most suitable sampling hyperparameter for a trained model is low compared to training the model. Third, the successive halving is effective in pruning underperforming and prevailing promising forecasting methods. As the three SM methods do not prevail in any data set and their performance is far behind the ML and DL methods, we recommend omitting SM methods to reduce the computational effort. For the ML and DL methods, the ablation study shows that the respective default configuration already provides a good indicator for the performance evolution potential of the method in the HPO. Therefore, we may conclude that if one method outperforms another by orders of magnitude, both in the default configuration, this will not alter with HPO.

3.3.4 Limitations

Regarding the limitations of AutoPQ, we discuss the computational effort required for performance improvements, the question of which probabilistic properties are desirable, and possibly untapped potential for further performance improvements.

Computational effort. The benchmarking shows that AutoPQ-advanced can significantly improve the performance over AutoPQ-default. However, the improvement comes at



(a) Evolution of the total computing time.

(b) Resulting total electricity consumption.

Figure 3.16: Comparison of the computational effort for a run of AutoPQ-default and AutoPQ-advanced for different numbers of point forecasting methods N_p in the CASH and different time budgets B_t in the successive halving pruning strategy. Since AutoPQ-advanced assesses four trial configurations in parallel while AutoPQ-default runs sequentially, we multiply the total computing time of AutoPQ-advanced by four. The total electricity consumption was recorded by the HoreKa HPC. The figure is based on [165].

the expense of a higher computational effort, measured in computing time and electricity consumption¹² and shown in Figure 3.16.

Over the six data sets, AutoPQ-default requires an average of 4.22 h and 0.57 kWh for a run, while AutoPQ-advanced with $N_p = 9$ considered point forecasting methods and the time budget $B_t = 8$ h requires an average of 81.78 h and 8.73 kWh for a run.¹³ If we omit the three SM forecasting methods due to the weak performance observed (i. e. $N_p = 6$), the computational effort reduces to an average of 69.33 h and 7.75 kWh. Furthermore, we may already consider AutoPQ-default as the first pruning round and reduce the overall time budget to $B_t = 4$ h as the performance improvement of the last pruning round is marginal. As a result, the computational effort would further reduce to an average of 30.89 h and 3.47 kWh.

To estimate the costs of using a commercial cloud computing service, we calculate the costs for AutoPQ-default and AutoPQ-advanced when using equivalent hardware of the Amazon

¹² As argued in [242], reporting the raw electricity consumption is the preferable metric, as it does not rely on geopolitical or data-center-specific factors at the time of collection. Other metrics of interest, such as the carbon footprint, can be calculated at any time based on local conditions.

¹³ Note that AutoPQ-advanced evaluates four trial configurations in parallel for each forecasting method. Thus, the total computing time is accumulated over the pruning rounds, which is the product of the active forecasting methods per round, and the time budget per round, multiplied by four.

EC2 G4 instances [244]. For AutoPQ-advanced, the four GPU instance `g4ad.16xlarge` (3.468 \$/h on demand) is suitable since four trials can be evaluated in parallel, which reduces the computing time shown in Figure 3.16 that is billed by a factor of four. For AutoPQ-default, the single GPU instance `g4ad.4xlarge` (0.867 \$/h on demand) is sufficient. Performing the computation using these instances would cost 3.66 \$ for AutoPQ-default and 26.78 \$ for AutoPQ-advanced.¹⁴ Since these costs are comparatively low compared to the personnel costs that would be required for an iterative manual design process, we conclude that both AutoPQ-default and AutoPQ-advanced are cost-effective for most smart grid applications. For smart grid applications where decision processes depend on forecasting quality, we assume that AutoPQ-advanced will amortize fast and will even be cheaper than AutoPQ-default in the long term despite higher initial costs.

Desirable probabilistic properties. The customizability evaluation shows that different probabilistic metrics for the validation of configurations in the HPO result in different probabilistic properties. However, it is underexplored which properties are actually required by smart grid applications, i. e., which probabilistic metric is to be selected. This limitation could be overcome by replacing the probabilistic metric by measuring the so-called forecast value of a configuration directly in the smart grid application. For example, the resulting costs of an electricity cost optimization problem that uses the forecast as input could measure the forecast value as shown in [24].¹⁵ In this way, the HPO assesses how well the resulting probabilistic properties suit the application instead of using the selected probabilistic metric as a proxy.

Possibly untapped performance improvement potential. Untapped potential for possible performance improvements includes the design of the configuration spaces, the feature selection, and the integration of PK. Regarding the design of the configuration space, a poorly sized configuration space may result in wasted computational resources if the space is too large and includes insensitive hyperparameters, while being too small may miss potential performance improvements by neglecting sensitive hyperparameters. Since a sensitivity analysis must be performed on a wide range of different data sets and we cannot undertake this enormous computational effort, we base our design on best practices. Specifically, the selection of hyperparameters to be optimized and their value ranges are based on the documentation of the respective forecasting methods, which provide recommendations

¹⁴ For simplicity, we assume that GPU instances are used for all forecasting methods, while they are actually only efficient for the three DL methods. A cost-optimized use would use CPU-only instances for the SM and ML forecasting methods to avoid GPU idle time, as in the present evaluation on the HoreKa HPC.

¹⁵ In this study, multiple MLP-based point forecasting models are trained with different loss functions, and the one with the best forecast value is selected.

for HPO. In terms of the feature selection, it is based on [115] (AutoPQ-default) to ensure comparability in the benchmarking. In real-world applications, however, automated selection is preferable. This limitation could be overcome by using an automated feature selection method, e. g., the Minimum Redundancy Maximum Relevance (MRMR) filter method [245].

3.3.5 Benefits

AutoPQ’s automated design of probabilistic forecasts with high quality and customized properties is crucial for smart grid applications to be deployed on a large scale given the limited number of experienced data scientists. With the above-shown affordable costs of training AutoPQ-advanced, we anticipate an amortization for many smart grid applications, especially if decision processes rely on the forecast quality. Another beneficial aspect concerns the universality of our approach, as the cINN can be used to generate quantiles or PIs for an arbitrary point forecasting method. On the one hand, this allows considering point forecasting methods of future research in the CASH in AutoPQ-advanced. On the other hand, consistently underperforming point forecasting methods can be removed from the CASH as recommended above. Further, the joint optimization of the point forecasting methods hyperparameters and the sampling hyperparameter is not limited to validation metrics that are mathematically derivable. This is crucial because it allows considering arbitrary metrics for the validation of configurations in the HPO. Apart from customized metrics, the so-called forecast value in the smart grid application can also be measured directly, as suggested above.

3.4 Related work

In this section, we evaluate related work regarding the three challenges stated in the introduction of this chapter, which are the quantification of uncertainty in time series forecasts, the automation of the design process, and the customizability of the forecast’s probabilistic properties. The entire analysis is guided by Table 3.5.

Probabilistic time series forecasting

Related work on probabilistic time series forecasting can be classified into direct probabilistic forecasting methods and probabilistic forecasts based on point forecasts. The following analysis of literature on these two types summarizes the related work of [115] and is extended by recently published literature.

Direct probabilistic forecasting methods. Direct probabilistic forecasting methods are designed to directly learn the forecast uncertainty. Depending on how the uncertainty

Table 3.5: Related work on time series forecasting in terms of uncertainty quantification, automated design, and customizable properties. The table is based on [165].

Reference	Category	Uncertainty quantification			Automated design	Customizable properties
		Quantiles/Pis	PDF	Scenarios		
[46, 47]	Direct probabilistic forecast	✓	✗	✗	✗	✗
[48]	Direct probabilistic forecast	✓	✓	✗	✗	✗
[246]	Direct probabilistic forecast	✓	✓	✗	HPO	✗
[49, 247–254]	Direct probabilistic forecast	✓	✓	✓	✗	✗
[28, 224, 225, 255, 256]	Point forecast-based probabilistic forecast	✓	✗	✗	✗	✗
[226]	Point forecast-based probabilistic forecast	✓	✓	✓	✗	✗
[223]	Point forecast-based probabilistic forecast	✓	✓	✗	✗	✗
[32, 61, 66, 77, 105, 236, 257–260]	Point forecast	✗	✗	✗	HPO	✗
[78, 227, 228]	Point forecast	✗	✗	✗	CASH	✗
[229]	Point forecast-based and direct probabilistic forecast	✓	✗	✗	CASH	✗
AutoPQ-default	Point forecast-based probabilistic forecast	✓	✓	✓	HPO	✓
AutoPQ-advanced	Point forecast-based probabilistic forecast	✓	✓	✓	CASH	✓

HyperParameter Optimization (HPO), Combined Algorithm Selection and Hyperparameter optimization (CASH)

of a forecast is represented, direct probabilistic forecasting methods can be categorized into quantile-based forecasts, distribution-based forecasts, and normalizing flow-based forecasts.

Quantile-based forecasts are trained to forecast specific quantiles, allowing to estimate PIs with a given conditional probability of containing the target value. For example, a QRNN [46] is trained with the PL function to forecast a selected quantile, and we may train multiple QRNNs with the different quantiles in the PL to make forecasts for a set of quantiles. A different approach is proposed by González-Ordiano et al. [47] called NNQF that filters the training data to determine a set of quantiles in the data and trains an ML-based method to forecast these quantiles. However, the above methods are limited to forecast quantiles and PIs, respectively, and are not able to forecast the full probability distribution or generate conditional future scenarios.

Distribution-based forecasts are trained to forecast the full PDF or Cumulative Distribution Function (CDF). A well-known representative is DeepAR [48]; an AR-based RNN that learns the parameters of an assumed PDF and makes probabilistic forecasts by sampling from it. A similar approach is proposed by Rangapuram et al. [246] that extends DeepAR to provide coherent probabilistic hierarchical forecasts. Disadvantageously, both require the assumption of the underlying PDF and are unsuitable for generating conditional future scenarios.

Normalizing flow-based probabilistic forecasts overcome this disadvantage by learning a bijective mapping from the unknown PDF to a known and tractable PDF. For example, normalizing flows are combined with QRNNs [247], RNNs [248], Gaussian mixture models

[249], and Bernstein polynomials [49]. The above methods use normalizing flows to enrich existing probabilistic forecasting methods. Alternatively, normalizing flows are used to directly provide probabilistic forecasts of weather variables [250], electrical load [251, 252], load and renewable energy generation [253], and electricity prices [254]. While being effective, these forecasts assume that the underlying learned distribution remains constant and does not always consider exogenous features.

A limitation that applies to all direct probabilistic forecasting methods is that they cannot be applied to generate probabilistic forecasts from existing, well-designed point forecasts, which still form the majority of forecasting methods in the literature [222].

Point forecast-based probabilistic forecasting methods. Probabilistic forecasts based on point forecasts rely on representing the forecast uncertainty with a separate uncertainty quantification method [261]. Most methods rely on generating PIs for point forecasts based on the residuals between the point forecasts and the realized values on a validation data set. These methods differ in the assumptions made about the distribution of the residuals. For example, one can assume the residuals follow a Gaussian distribution [28], an empirical distribution [224], or an empirical non-conformity score distribution [225, 255, 256]. The latter measures how unusual a residual is compared to other residuals in the validation data set, and calculates a critical non-conformity score for the desired significance level of the PI. However, the above methods are designed to create PIs rather than providing the full PDF, which is a limitation.

Similar approaches combine the utilization of residuals with ML. For example, Wang et al. [226] train a Generative Adversarial Network (GAN) with the residuals of a point forecast, which is used to generate residual scenarios. These scenarios are then used to quantify the uncertainty of the point forecast. A different approach is proposed by Camporeale et al. [223], who train an ANN to forecast the standard deviation of residuals. The estimated standard deviation is then used, under the assumption of a Gaussian distribution around the point forecast, to quantify its uncertainty. A limitation of both approaches is that the ML models used to estimate the uncertainty directly depend on the point forecast, as they are trained on the residuals. That is, if considering multiple candidate point forecasting methods and hyperparameter configurations, an additional ML model must be trained for each candidate and each configuration.

Automated time series forecasting

Related work on automating the design process can be classified into HPO and CASH. While a large amount of research exists on two types, almost no work considers automated

probabilistic forecasting [1]. The following analysis of related work summarizes the review [1] and is extended by recently published literature.

Hyperparameter optimization. The HPO of a forecasting method aims to tailor its configuration to the forecasting task and has a long history. For example, grid search-based HPO for point forecasting method of the SM family are proposed for ETS [32], sARIMAX [66], and TBATS [236]. Similarly, grid search is applied for the HPO of point forecasting methods of the ML family, e. g., for the SVR [77, 105]. While effective for small configuration spaces, complex spaces require more efficient HPO algorithms, e. g., random search is used for the HPO of an MLP [61] and a Gradient Boosting Machine (GBM) [257]. For a directed search in complex configuration spaces of DL methods, gradient-based search methods [258], BO [259], and EAs [260] are used. Apart from point forecasting methods, such HPO methods can also be applied to DL-based direct probabilistic forecasting methods as in [246]. While effective, HPO of a single forecasting method neglects the no-free lunch theorem, stating that no method exists that excels in all forecasting tasks, since the forecast quality is sensitive to various model design decisions [20].

Combined algorithm selection and hyperparameter optimization. For the selection of the forecasting method combined with the optimization of hyperparameters, several CASH methods for point forecasting have been proposed in the literature. For example, Rätz et al. [78] consider multiple ML forecasting methods, namely GBM, Least Absolute Shrinkage and Selection Operator (LASSO), MLP, Random Forest (RF), and SVR, and perform the CASH using BO. Shah et al. [227] use a similar approach but additionally consider SM forecasting methods such as AutoRegressive Integrated Moving Average (ARIMA), Triple Exponential Smoothing (TES), and Box-Cox transformation, ARMA errors, Trend, Seasonal components (BATS). A CASH method for DL methods is proposed by Deng et al. [228], considering more complex forecasting methods such as Neural Basis Expansion Analysis for interpretable Time Series forecasting (N-BEATS) and TFT. Besides point forecasting methods, such CASH methods are also applicable to direct probabilistic forecasting methods and point forecast-based probabilistic forecasting methods as in [229]. The authors consider several SM- and ML-based forecasting methods using conformal PIs and direct probabilistic forecasting methods like DeepAR. While the authors contribute to closing the under-researched field of CASH for probabilistic forecasts, the approaches used to quantify forecast uncertainty are subject to the limitations identified above.

Customized probabilistic forecast properties for smart grid applications

Related work on customizing probabilistic properties of forecasts to the specific requirements of the downstream smart grid application is a widely under-researched field [164]. However, recent research shows that forecast quality depends specifically on the downstream application [24]. For example, a robust optimization problem may only be interested in extreme events occurring with low probability. Yet, many probabilistic methods estimate PIs that are too narrow, respectively PDFs that are too sharp, as they do not account for all sources of uncertainty [262]. Although research exists on probabilistic energy time series forecasts – such as electrical load [249, 263], electricity price [254, 264], PV power generation [253, 263], and WP generation [54, 253] – they do not provide customized probabilistic properties.

3.5 Novelty and remaining questions

Given the shortcomings of related work identified in the previous section, the novelty of AutoPQ can be summarized as follows:

1. AutoPQ provides automated and high-quality probabilistic forecasts that are crucial for performance-critical smart grid applications.
2. AutoPQ has an in-built method for saving computing resources and quantifies the model's energy footprint.
3. AutoPQ enables the generation of forecasts with customized probabilistic properties based on non-derivable validation metrics.
4. AutoPQ relies on arbitrary point forecasting methods and enables integrating new methods of future research.

The automation of the design makes AutoPQ an Automation level 2 method under the AutoLVL taxonomy [162, 163] of Chapter 2. In terms of the stated open questions of this thesis in Section 1.5, AutoPQ covers two performance critical sections of the forecasting pipeline – namely HPO and forecasting method selection – and contributes to closing the research gap in automated probabilistic forecasting that is not addressed by the large majority of literature only considering automated point forecasting. Apart from the publications [164, 165], AutoPQ is available as open-source software, as linked in Figure 3.17. However, given the limitations of AutoPQ discussed in Section 3.3.4, new questions may be raised:

- Which validation metric is to be selected for the CASH to tailor the probabilistic properties and maximize the so-called forecast value in the smart grid application?

- Does estimating the forecast value for the smart grid application resolve the previous question, e. g., by the costs of a downstream electricity cost minimization?
- What is the performance improvement potential of considering automated feature selection?

These stated questions must be tackled in future work to improve AutoPQ.

AutoPQ

Template for automated point forecast-based quantile forecasts



Publications

K. Phipps, S. Meisenbacher, B. Heidrich, M. Turowski, R. Mikut, and V. Hagenmeyer, "Loss-customised probabilistic energy time series forecasts using automated hyperparameter optimisation," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy '23, Orlando, USA: Association for Computing Machinery, 2023, pp. 271–286.

S. Meisenbacher, K. Phipps, O. Taubert, M. Weiel, M. Götz, R. Mikut, and V. Hagenmeyer, "AutoPQ: Automating quantile estimation from point forecasts in the context of sustainability", *Applied Energy*, vol. 392, p. 125 931, 2025.

Implementation

<https://github.com/SMEISEN/AutoPQ>

Figure 3.17: The contributions of this chapter with the Automation level 2 template AutoPQ.¹⁶

¹⁶ The icon of the probability distributions is inspired by [219].

4 AutoWP: Template for automated wind power forecasts

Forecasts of the locally distributed Wind Power (WP) generation are necessary for balancing the electrical transmission and distribution grids to avoid line overloads [265]. Automating the design and operation of WP forecasting models is necessary to keep pace with the expansion of the WP generation capacity in future energy systems. In this context, two major challenges exist in the development of WP forecasting models:

First, scalable WP forecasting models are required that achieve good forecast quality. On the one hand, large centralized offshore WP farms are created in a uniform terrain. This allows the WP farm to be modeled holistically, taking into account the influence of the wind direction to implicitly learn the mutual influence of the WP turbines (wake losses), as in [266] using an autoregressive Deep Learning (DL) model. On the other hand, decentralized onshore WP turbines are created in different terrains, such as in open fields with little air turbulence or in environments with higher air turbulence such as in the area of cities, forests, and hills. In this case, DL models have the limitation that they require a high computational training effort in the model design. It is therefore an open question, whether this effort pays off when modeling hundreds of locally-distributed WP turbines with varying characteristics and differing terrains.

Second, regular and irregular interventions in the WP generation capabilities pose challenges in the design and operation of WP forecasting models. While regular shutdowns include time-controlled shutdowns such as maintenance and bat protection [158], redispatch interventions are irregular shutdowns. For redispatch planning, the forecast targets to communicate the availability of power generation and must not forecast these irregular shutdowns. Thus, this forecasting target must be considered when preparing the training data set and designing the model. While numerous methods for cleaning training data exist [267–269], redispatch interventions may affect models during operation while making forecasts if they are based on autoregression. More specifically, such methods consider a horizon of past values of the target time series alongside future covariates such as forecasts of wind speed and direction to make a WP forecast. Since shutdowns due to redispatch interventions also occur in this past horizon, future shutdowns may be forecasted, which is an undesirable effect in forecast-based planning [270].

Therefore, this chapter raises awareness of these two challenges and analyzes state-of-the-art forecasting methods on data sets with irregular shutdowns. We consider three autoregressive DL methods, namely Deep AutoRegression (DeepAR) [48], Neural Hierarchical Interpolation for Time Series (N-HiTS) [43] and the Temporal Fusion Transformer (TFT) [44]. We compare these autoregressive forecasting methods with methods based on WP curve modeling that do not consider past values and are only based on weather forecasts. Specifically, we include the three Machine Learning (ML) methods MultiLayer Perceptron (MLP) [100], Support Vector Regression (SVR) [100] and eXtreme Gradient Boosting (XGB) [238], as well as the Original Equipment Manufacturer (OEM)'s WP curve, and a novel ensemble learning method based on WP curves called AutoWP. AutoWP comprises an ensemble of a diverse set of OEM WP curves from the Open Energy Platform (OEP) wind turbine library [271], making a forecast by the optimally weighted sum of their outputs (Automation level 3 according to the AutoLVL taxonomy [162, 163] in Chapter 2).¹

4.1 Automated design methods

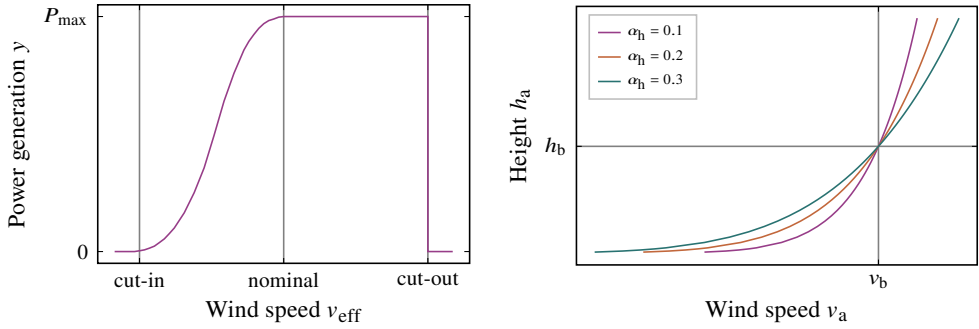
In this section, we first introduce the forecasting methods based on WP curve modeling and DL, followed by the post-processing to include Prior Knowledge (PK) and considered shutdown handling methods.

4.1.1 Forecasting methods based on wind power curve modeling

WP curve modeling aims to establish an empirical (temporally-independent) relationship between WP, wind speed, and possibly, other explanatory variables. In the following, we first introduce the WP curve and describe how it can be used for WP forecasting. Afterward, we present AutoWP before outlining ML methods for WP curve modeling.

Wind power curve. A WP curve $y[k] = f(v_{\text{eff}}[k])$ represents the power generation $y[k]$ as a function of the wind speed at a reference height $v_{\text{eff}}[k]$. As shown in Figure 4.1a, a WP curve consists of four sections, separated by the cut-in, nominal, and cut-out wind speed. For WP curves provided by turbine OEMs, the reference height is the hub height. As wind speed forecasts are commonly provided for 10 m and 100 m, a height correction is required (unless the turbine coincidentally has a hub height of 100 m). The reason for different wind speeds at different heights is the interaction between the wind and the earth's

¹ Note that this chapter is based on [166, 167] and contain identical phrases; a detailed list is given in the Appendix.



(a) The WP curve is the power generation y as a function of the wind speed at a reference height v_{eff} and consists of four sections separated by the cut-in, nominal, and cut-out wind speed.

(b) Height correction using the wind profile power law (4.1) with the known wind speed v_b at height h_b above ground level outlined for different exponents α_h .

Figure 4.1: Using the WP curve to make forecasts requires the wind speed forecast to be corrected to the curve's reference height. The figure is based on [162, 166].

surface, which creates the so-called atmospheric boundary layer [272]. A common approach for height correction is the straightforward wind power law [273]

$$\frac{v_a}{v_b} = \left(\frac{h_a}{h_b} \right)^{\alpha_h}, \quad (4.1)$$

where v_a and v_b are the wind speeds at heights h_a and h_b above ground level, and the empirically-determined exponent α_h depends on the terrain [274]. The influence of α_h is exemplified in Figure 4.1b. Based on (4.1), we perform the height correction of the wind speed forecast in two steps:² First, we assume $\alpha_h = 1/7$ for onshore and $\alpha_h = 1/9$ for offshore WP turbines [275], and estimate the effective hub height

$$h_{\text{eff}} = 100 \text{ m} \cdot \left(\frac{\overline{v_{\text{eff}}}}{\hat{v}_{100}} \right)^{(1/\alpha_h)} \quad (4.2)$$

using the average wind speed forecast at 100 m $\overline{v_{100}}$ and the average measured wind speed at hub height $\overline{v_{\text{eff}}}$ of the training data set. Second, each wind speed forecast sample $\hat{v}_{100}[k]$ can then be corrected to hub height by

$$\hat{v}_{\text{eff}}[k] = \hat{v}_{100}[k] \cdot \left(\frac{h_{\text{eff}}}{100 \text{ m}} \right)^{\alpha_h}. \quad (4.3)$$

This two-step approach is advantageous because the actual hub height must not be known and extrapolations from the 100 m reference are near the hub height of modern WP turbines,

² Note that the height correction is based on the work of Tim Martin [232].

which are typically between 80 m and 140 m above ground level [273]. The corrected wind speed forecast $\hat{v}_{\text{eff}}[k]$ can finally be used as input for the WP curve to make a WP forecast:

$$\hat{y}[k] = f(\hat{v}_{\text{eff}}[k]). \quad (4.4)$$

AutoWP. The underlying idea of AutoWP is to represent a new WP turbine as a convex linear combination of WP curves from a sufficiently diverse ensemble. The method consists of three steps: i) create the ensemble of normalized WP curves, ii) form the normalized ensemble WP curve by the optimally weighted sum of the WP curves in the ensemble, and iii) re-scale the ensemble WP curve with the new WP turbine's peak power rating. These three steps are detailed in the following, and Figure 4.2 provides exemplary results of these three steps.

For the first step, we create the ensemble of OEM WP curves using the wind turbine library of the OEP [271]. That is, we load the database, select all entries that contain a WP curve, resample the value pairs $\{P_n, v_{\text{eff}, n}\}$ to ensure a uniform wind speed sampling rate, and normalize the power values $P_n[v_{\text{eff}, n}]$ with the WP curve's peak power rating $P_{\text{max}, n}$:

$$P_n^*[v_{\text{eff}, n}] = \frac{P_n[v_{\text{eff}, n}]}{P_{\text{max}, n}}. \quad (4.5)$$

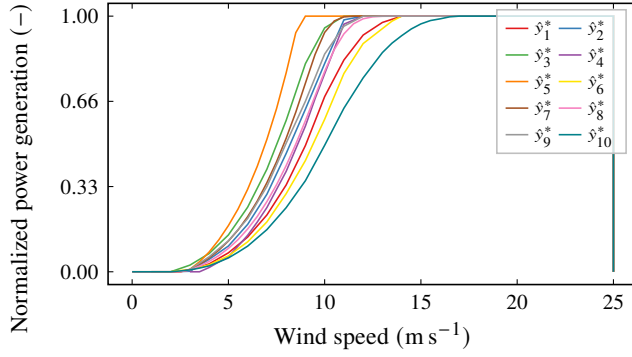
The output of such a normalized WP curve $\hat{y}^*[k] = f(v_{\text{eff}}[k])$ is a function of the wind speed time series $v_{\text{eff}}[k]$, $k \in \mathbb{N}^K$. To limit the size of the ensemble of WP curves without losing diversity, we sort all WP curves by the sum of their normalized power values, keep the first and last WP curves as well as eight WP curves in between at equidistant intervals, i. e., $N_m = 10$.

For the second step, we form the ensemble WP curve by the weighted sum of the normalized WP curves (2.9), which is a convex linear combination with $\hat{y}_n[k]$ and \hat{w}_n , $n \in \mathbb{N}_1^{N_m}$ being the output and weight of the n -th curve in the ensemble pool. To find the optimal weights, we normalize the target time series \mathbf{y} analogously to (4.5), and solve the optimization problem (2.10) using the least squares algorithm of the Python package SciPy [187] with the initial values $\hat{w}_n = 1/N_m$, $\forall n \in \mathbb{N}_1^{N_m}$, normalizing the weights to hold the constraints.³

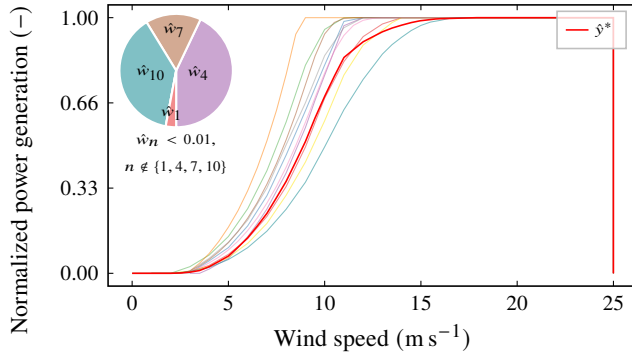
For the third step, we re-scale the ensemble output \hat{y}^* to the peak power rating $P_{\text{max}, \text{new}}$ of the new WP turbine:

$$\hat{y}[k] = \hat{y}^*[k] \cdot P_{\text{max}, \text{new}}. \quad (4.6)$$

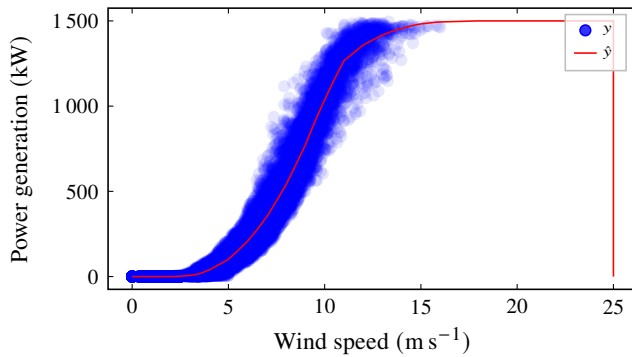
³ Alternatively, one could relax the constraints by allowing weights greater than one and not requiring the sum of weights to be one. This would make the third step obsolete and is suitable if the peak power rating of the WP turbine is unknown.



(a) The first step creates the ensemble pool using $N_m = 10$ normalized WP curves \hat{y}_n^* , $n \in \mathbb{N}_1^{N_m}$ of the OEP wind turbine library [271]. The selection reduces redundancy and preserves diversity.



(b) The second step forms the normalized ensemble WP curve \hat{y}^* as convex linear combination of the pool's curves, with the weights \hat{w}_n , $n \in \mathbb{N}_1^{N_m}$ adapted to optimally fit the new WP turbine.



(c) The third step re-scales the ensemble WP curve \hat{y} to the peak power rating $P_{\max, \text{new}} = 1500 \text{ kW}$ of the new WP turbine. Additionally, the measurement y used to fit the ensemble weights is shown.

Figure 4.2: The three steps of AutoWP's automated design exemplified for the real-world WP turbine no. 1 [166].

Machine learning for wind power curve modeling. Instead of using OEM WP curves that require a height-correction of the wind speed forecast, ML methods can be used to directly learn the relationship between the wind speed forecast at 100 m and the WP turbine's power generation. Since the relationship between wind speed and power generation is non-linear (see Figure 4.1a), we only consider non-linear ML methods. Popular methods that perform well on a large variety of regression tasks are the MLP, the SVR, and Decision Tree (DT)-based methods like XGB. As for the autoregressive DL forecasting methods, we consider the exogenous forecasts of wind speed and direction, air temperature and atmospheric pressure as explanatory variables $\hat{\mathbf{X}}$ in the regression problem

$$\hat{y}[k] = f(\hat{\mathbf{X}}[k], \mathbf{p}). \quad (4.7)$$

That is, the regression model $f(\cdot)$ estimates the WP generation $\hat{y}[k]$ based on the exogenous forecasts $\hat{\mathbf{X}}[k]$, where \mathbf{p} are the model's parameters determined by training. However, unlike the autoregressive DL forecasting methods, the regression model (4.7) is static, i. e., does not consider past WP generation values.

4.1.2 Forecasting methods based on autoregression and deep learning

An autoregressive forecasting model $f(\cdot)$ makes an estimate of future expected values \hat{y} at origin k for the forecast horizon $H \in \mathbb{N}_1$ using current and past values, see Section 1.2.1. For the past horizon H_1 , we consider past values up to one day, i. e., 96 values; for \mathbf{X} , we consider additional explanatory variables for shutdown handling that we detail in Section 4.1.4; and for $\hat{\mathbf{X}}$, we consider the exogenous forecasts of wind speed and direction, air temperature and atmospheric pressure.

State-of-the-art autoregressive forecasting methods are often based on DL, i. e., (deep) Artificial Neural Networks (ANNs) with specialized architectures, such as recurrent layers, attention units, and residual connections. We consider three DL-based methods DeepAR [48], N-HiTS [43], and TFT [44].

4.1.3 Post-processing

To take restrictions into account, we use PK about the WP curve. First, the power generation is limited by the peak power rating of the WP turbine. Second, negative WP generation is

impossible. Third, the WP turbine is shut down when the wind speed \hat{v}_{eff} exceeds the cut-out speed $v_{\text{eff, cut-out}}$. Incorporating these three rules leads to

$$\hat{y}[k] = \begin{cases} \hat{y}[k], & \text{if } \hat{y}[k] > 0 \text{ and } \hat{y}[k] < P_{\text{max}} \text{ and } \hat{v}_{\text{eff}}[k] < v_{\text{eff, cut-out}}, \\ P_{\text{max}}, & \text{if } \hat{y}[k] > P_{\text{max}} \text{ and } \hat{v}_{\text{eff}}[k] < v_{\text{eff, cut-out}}, \\ 0, & \text{otherwise,} \end{cases} \quad (4.8)$$

for post-processing the outputs of the autoregressive DL methods DeepAR, N-HiTS, and TFT, and the outputs of the ML methods MLP, SVR, and XGB in WP curve modeling.⁴

4.1.4 Shutdown handling

We consider different methods for shutdown handling in model training and model operation. For autoregressive methods, we consider the methods *none*, *explanatory variables*, *drop*, *imputation*, and *drop-imputation*.⁵ For methods based on WP curve modeling, we only consider the method *drop*, as these methods operate without a temporal context.

None shutdown handling refers to processing raw data for model training and operation.

Using *explanatory variables* refers to providing additional features. As in [276], we use cyclic features to encode seasonal information. Specifically, we consider the day of year (2.3) and the minute of the day (2.5), which are known in the future horizon. Furthermore, we provide features that are unknown in the future, i. e., they are only available in the past horizon of the autoregressive methods. Specifically, we provide the theoretical power generation according to the turbine's OEM WP curve, and a categorical feature that labels data samples as normal or abnormal using a rule-based and an outlier detection-based approach. Figure 4.3 exemplarily shows the above-introduced additional variables.⁶

The method *drop* refers to dropping training data samples identified as abnormal operational states, and *imputation* refers to replacing these samples with the theoretical power generation according to the turbine's OEM WP curve as in [151], exemplified in Figure 4.4. Finally, *drop-imputation* combines the above principle by dropping abnormally labeled training data samples and replacing abnormally labeled samples during operation. Importantly, the imputation only applies to the autoregressive models' past horizon and not to the data used to calculate the test metrics (test data set).

⁴ In the data set used for evaluation, the measured wind speeds at hub height are below the cut-out wind speed $v_{\text{eff, cut-out}}$.

⁵ Note that the methods *explanatory variables* and *imputation* is based on the work of Mehdi Dado [158].

⁶ Since turbine no. 2 shows shutdowns during night times, we also consider a feature that is one at night and zero during the day.

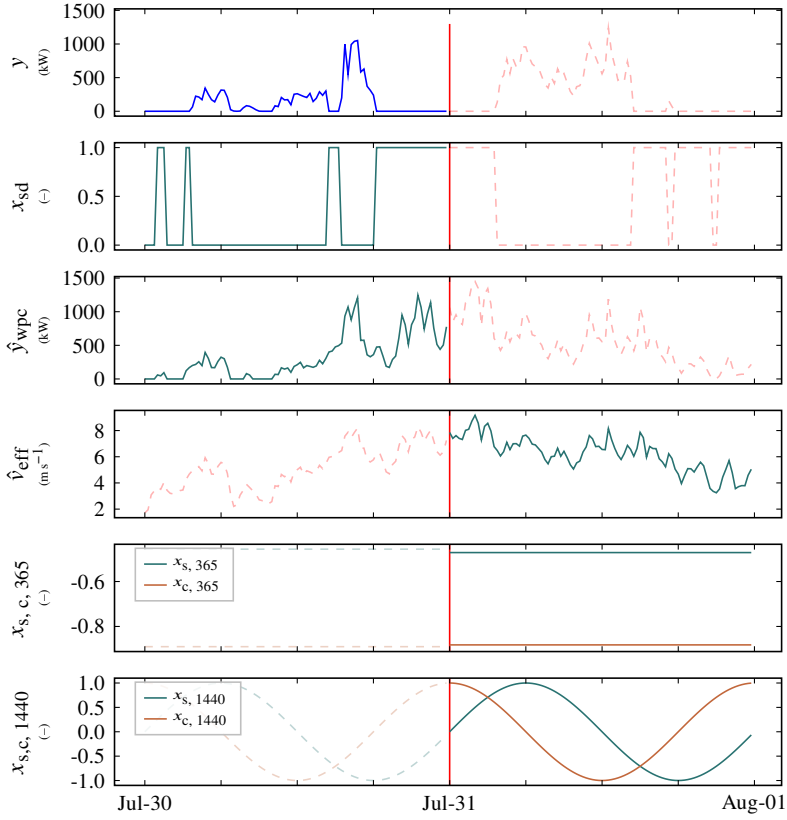


Figure 4.3: Power generation y and explanatory variables available in the forecasting model's future and past horizon. x_{sd} labels identified shutdowns in the past, \hat{y}_{wpc} is the turbine's theoretical power generation according to the OEM WP curve, \hat{v}_{eff} is the wind speed forecast at hub height, and the features $x_{sin, 365}$, $x_{cos, 365}$, $x_{sin, 1440}$, and $x_{cos, 1440}$ encode temporally recurring shutdowns patterns. Dashed lines visualize unavailable future or unused past periods, and the vertical line marks the forecast origin [166].

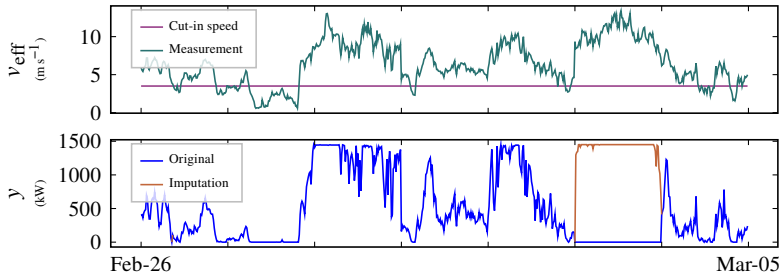


Figure 4.4: Data imputation at shutdowns with the theoretical power generation according to the turbine's OEM WP curve (i. e. the imputed values depend on the wind speed measurement at hub height v_{eff}). During shutdowns, generation y is zero although the wind speed v_{eff} is higher than cut-in speed $v_{eff, cut-in}$ [166].

4.2 Evaluation

In this section, we compare state-of-the-art forecasting methods based on autoregression with methods that do not consider past values and are only based on day-ahead weather forecasts. For this purpose, we first describe the experimental setup for assessing the forecast errors. Afterward, we show the results and provide insights.

4.2.1 Experimental setup

The experimental setup describes the evaluation data, and the evaluation strategy.

Data. The evaluation is performed on a real-world data set containing two years (2019, 2020) of quarter-hourly energy metering (kWh) from two WP turbines; thus, the time resolution is 15 minutes.⁷ We convert the energy metering time series into the mean power generation time series to increase interpretability using (A.5). Additionally, measurements of the average wind speed at hub height are available. Both WP turbines are located in southern Germany, have a peak power rating of 1500 kWp, and are subject to many shutdowns.⁸ To identify abnormal operating states in the data, we use a rule-based and an outlier detection-based approach. For the former, we label all data points at a measured wind speed greater than the cut-in speed with a power generation lower than the cut-in power as shutdowns like in [151], see Figure 4.5.⁹ For the latter, we apply the Local Outlier Factor (LOF) algorithm as in [277] to identify samples during WP turbine stop-to-operation transitions and vice versa. The result of the rule-based and LOF-based identification is shown in Figure 4.6.¹⁰

In the evaluation, the methods described in Section 4.1 use day-ahead weather forecasting data from the European Centre for Medium-Range Weather Forecasts (ECMWF) [278] as additional exogenous features. While the lower temporal and spatial resolution of the weather forecast has an impact on the WP forecast quality, the handling of shutdowns in the autoregressive models' past horizon H_1 is unaffected, as it is performed with the weather measurement data available until the forecast origin.

The data is split into a training data set (2019) and a test data set (2020). The forecasting methods DeepAR, N-HiTS, and TFT additionally hold-out 20 % of the training data set for

⁷ Due to the 15-minute resolution, effects with a dynamic below the resolution such as wind gusts, stop-to-operation transitions and vice versa cannot be modeled.

⁸ While the data set is not publicly available, it provides the possibility to evaluate the effects of regular and irregular WP turbine shutdowns. An exemplary combination of AutoWP with the conditional Invertible Neural Network (cINN) [107, 115] to generate quantile forecasts based on point forecasts (Chapter 3) on an openly available data set is given in Section C.2 in the Appendix.

⁹ As the data set predates the introduction of Redispatch 2.0, the shutdowns are not redispatch interventions.

¹⁰ Note that with *none* shutdown handling all samples are used for model training and operation without filtering.

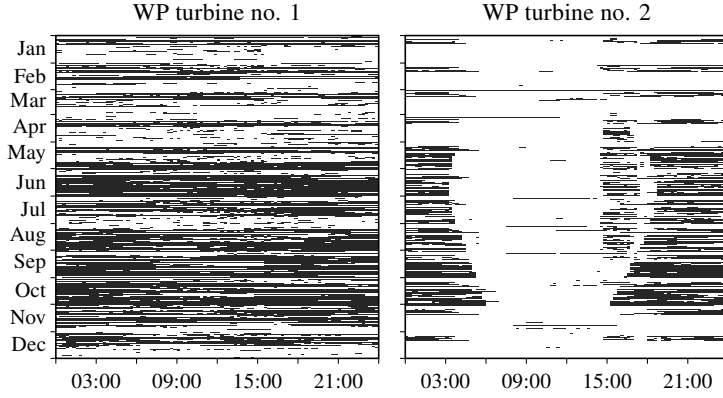


Figure 4.5: Identified WP turbine shutdowns with rule-based filtering. Black fields represent data points at a measured wind speed greater than the cut-in speed with a power generation lower than the cut-in power [167]. The shutdowns for WP turbine no. 1 amount to 49 % and 20 % for no. 2.

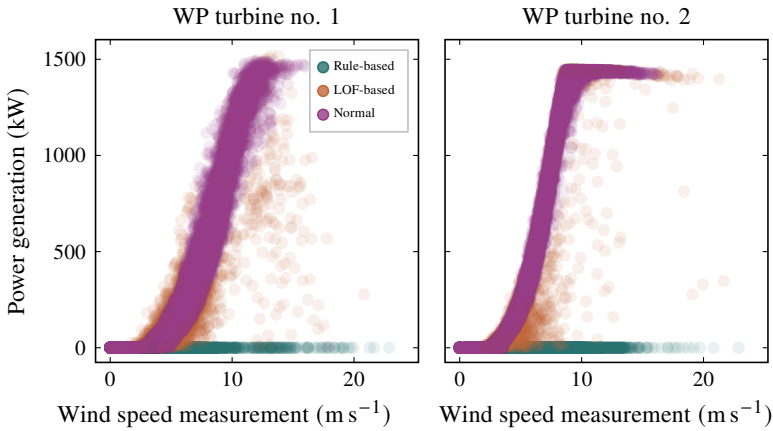


Figure 4.6: Data pre-processing to identify abnormal operational states. Rule-based filtering identifies turbine shutdowns and LOF-based filtering identifies turbine stop-to-operation transitions and vice versa. The samples for which the wind speed at hub height is below the cut-in speed $v_{\text{eff, cut-in}} = 2.5 \text{ m s}^{-1}$ amount to 16.7 %, while the cut-out speed $v_{\text{eff, cut-out}} = 25 \text{ m s}^{-1}$, at which the WP turbine would be shut down for safety reasons, is not reached [167]. Legend of WP turbine no. 1 applies also to WP turbine no. 2.

early stopping the training process when the loss on the hold-out data increases. While the wind speed measurement at hub height is used to identify abnormal operation (Figure 4.5), the wind speed forecasts are used for model training.

Evaluation strategy. We evaluate the forecast errors of the methods described in Section 4.1, comparing two scenarios. The first concerns the day-ahead WP forecast including future shutdowns. The second concerns the use of the day-ahead WP forecast to communicate the availability of the WP turbine for redispatch planning, thus, must not forecast future shutdowns. As test metrics, we use the normalized Mean Absolute Error (nMAE) (A.2) and additionally report the normalized Root Mean Squared Error (nRMSE) (A.4) in the Appendix.

The three DL-based methods DeepAR, N-HiTS, and TFT are implemented with the Python package PyTorch Forecasting [243], SVR and MLP with the Python package scikit-learn [100], and XGB with the authors' [238] Python package. All methods use the default hyperparameter configuration of the respective Python package, and methods that use a stochastic training algorithm (DeepAR, N-HiTS, TFT, and MLP) are run five times.

4.2.2 Results

In the following, we visualize and summarize the results of the two evaluation scenarios. First, we consider day-ahead WP forecasting including shutdowns to evaluate whether future shutdowns can be forecasted. Second, we evaluate day-ahead WP forecasting disregarding identified shutdowns to evaluate the forecasting quality in communicating the availability for redispatch interventions where the times of shutdowns are communicated via non-availability notifications. An interpretation and discussion of these results can be found in the next Section 4.3.

Shutdown handling for methods based on autoregression. The effect of the shutdown handling methods on the forecast errors of the state-of-the-art autoregressive DL methods DeepAR, N-HiTS, and TFT when *considering shutdowns* is shown in Figure 4.7. For WP turbine no. 1, the shutdown handling methods *imputation* and *drop-imputation* perform significantly worse than the others. The reason are the numerous shutdowns in the data set (see Figure 4.5), where the imputation of shutdowns with values from the OEM WP curve in the forecasting model's past horizon leads to forecasts of WP generation while the turbine is actually shut down. For WP turbine no. 2, which shows regular time-dependent shutdowns (see Figure 4.5), the shutdown handling method *explanatory variables* only leads to an improvement for N-HiTS compared to *none*, and performs particularly poorly for DeepAR. One observation that applies to both WP turbines is that the shutdown

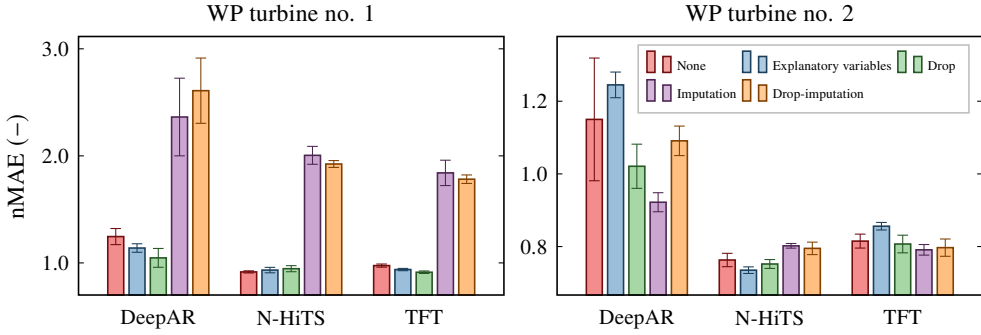


Figure 4.7: The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nMAE when *considering* shutdowns. Legend of WP turbine no. 2 applies also to WP turbine no. 1. The figure is based on [166].

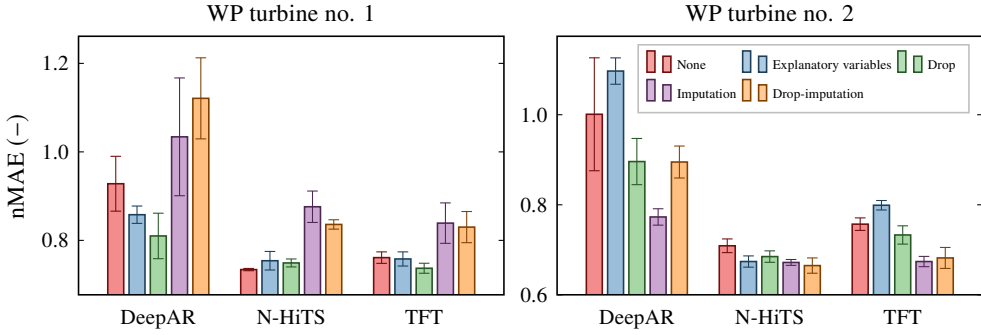


Figure 4.8: The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nMAE when *disregarding* shutdowns. Legend of WP turbine no. 2 applies also to WP turbine no. 1. The figure is based on [166].

handling methods have a significant impact on the forecast errors, but there is no method that consistently performs best across DeepAR, N-HiTS, and TFT.

Comparison of autoregressive and WP curve-based methods. The effect of the shutdown handling methods on the forecast errors of the state-of-the-art autoregressive DL methods DeepAR, N-HiTS, and TFT when *disregarding shutdowns* is shown in Figure 4.8. For WP turbine no. 1, it appears that the shutdown handling method *drop* achieves the lowest forecast errors for DeepAR and TFT, while *none* performs best for N-HiTS. For WP turbine no. 2, the shutdown handling method *imputation* achieves the lowest forecast errors

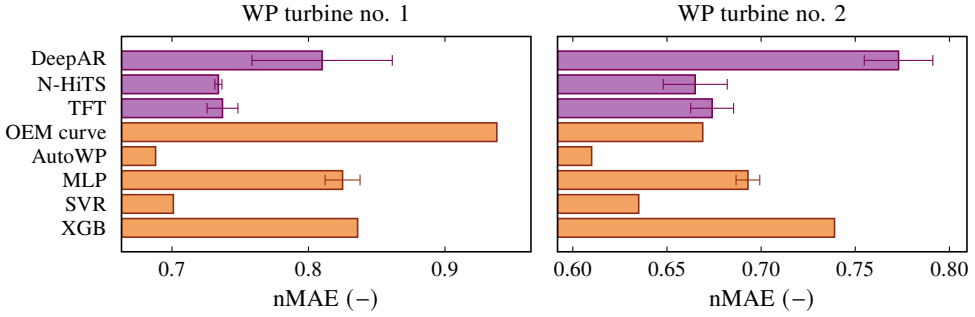


Figure 4.9: Comparison of autoregressive DL forecasting methods (purple) to methods based on WP curve modeling (orange) in terms of the test nMAE when *disregarding* shutdowns. The figure is based on [166].

for DeepAR and TFT, while *drop-imputation* performs best for N-HiTS. Interestingly, the imputation of shutdowns in the past-horizon with values of the OEM WP curve only lead to improvement for WP turbine no. 2. This is because the forecast error of the OEM WP curve is comparably poor for WP turbine no. 1, as can be seen in Figure 4.9. In the figure we compare the forecast errors of the autoregressive DL methods DeepAR, N-HiTS, and TFT (with the respective best-performing shutdown handling method) to forecasting methods based on WP curve modeling. We observe that even if selecting the best individual shutdown handling method, the autoregressive DL forecasting methods perform worse than the WP curve modeling methods. Among these approaches, AutoWP, MLP, and SVR achieve similar forecast errors. While AutoWP performs best for the metric nMAE for both turbines, the SVR performs best for the metric nRMSE for WP turbine no. 1, and the MLP for no. 2, (see Figure C.3 in the Appendix).

4.2.3 Insights

A visual comparison of WP generation forecasts is exemplified in Figure 4.10 for WP turbine no. 2. In addition to the WP generation, we also show the measured wind speed at hub height and the height-corrected wind speed forecast from ECMWF. The exemplary comparison of N-HiTS, AutoWP and MLP shows that the wind speed forecast has a different influence on the resulting WP generation forecast. While AutoWP relies only on the height-corrected wind speed forecast, the MLP is trained based on the forecasts of wind speed and direction, air temperature, and atmospheric pressure, and N-HiTS additionally takes into account past WP generation measurements. This becomes more obvious if we plot the WP generation measurement y and forecast \hat{y} over the wind speed forecast \hat{v}_{eff} in Figure 4.11. More specifically, N-HiTS attempts to compensate for the errors of the wind speed forecast

using WP generation measurements in the past horizon, which does not improve forecast errors for the data sets considered.

4.3 Discussion

This section discusses the results of the two evaluation scenarios, the limitations of the study, and derives recommendations.

Shutdown handling for methods based on autoregression. Regarding the results of the first scenario, we observe that while the shutdown handling methods have a significant impact on forecast errors, no method consistently performs best across the autoregressive DL methods DeepAR, N-HiTS, and TFT. Together with the high computational training effort, this results in an overhead for model design, which limits the scalable model deployment to hundreds of individual WP turbines.

Comparison of autoregressive and WP curve-based methods. Regarding the results of the second scenario, we see that even when choosing the best shutdown handling method, the autoregressive DL methods do not achieve an advantage over methods based on WP curve modeling on the given data set. However, note that the methods based on WP curve modeling are based only on day-ahead weather forecasts and thus cannot forecast regular shutdowns. Regular shutdowns can be taken into account by either using PK of timed shutdowns or by creating a separate classification model that predicts whether the WP turbine is operating or shut down. However, such modeling is subject to the challenge of distinguishing between regular and irregular shutdowns in the data set. For example, although Figure 4.5 shows recurring shutdown patterns, they are not strictly regular and may be interspersed with irregular shutdowns.

Limitations. Although we consider two WP turbines with different shutdown patterns, the data set is relatively small to make generalized conclusions. Nevertheless, we identified major issues for the scalable use of autoregressive DL methods for redispatch planning. For such forecasts, it is advisable to use computationally efficient WP curve modeling approaches. Another limitation concerns possibly untapped potential for forecast error reductions by HyperParameter Optimization (HPO). However, we expect that even with HPO, which further increases the computational effort, no forecast error advantage over methods based on WP curve modeling can be achieved by the autoregressive DL methods. Additionally, improving the performance of the autoregressive DL methods would require labeled data sets to distinguish between regular and irregular shutdowns.

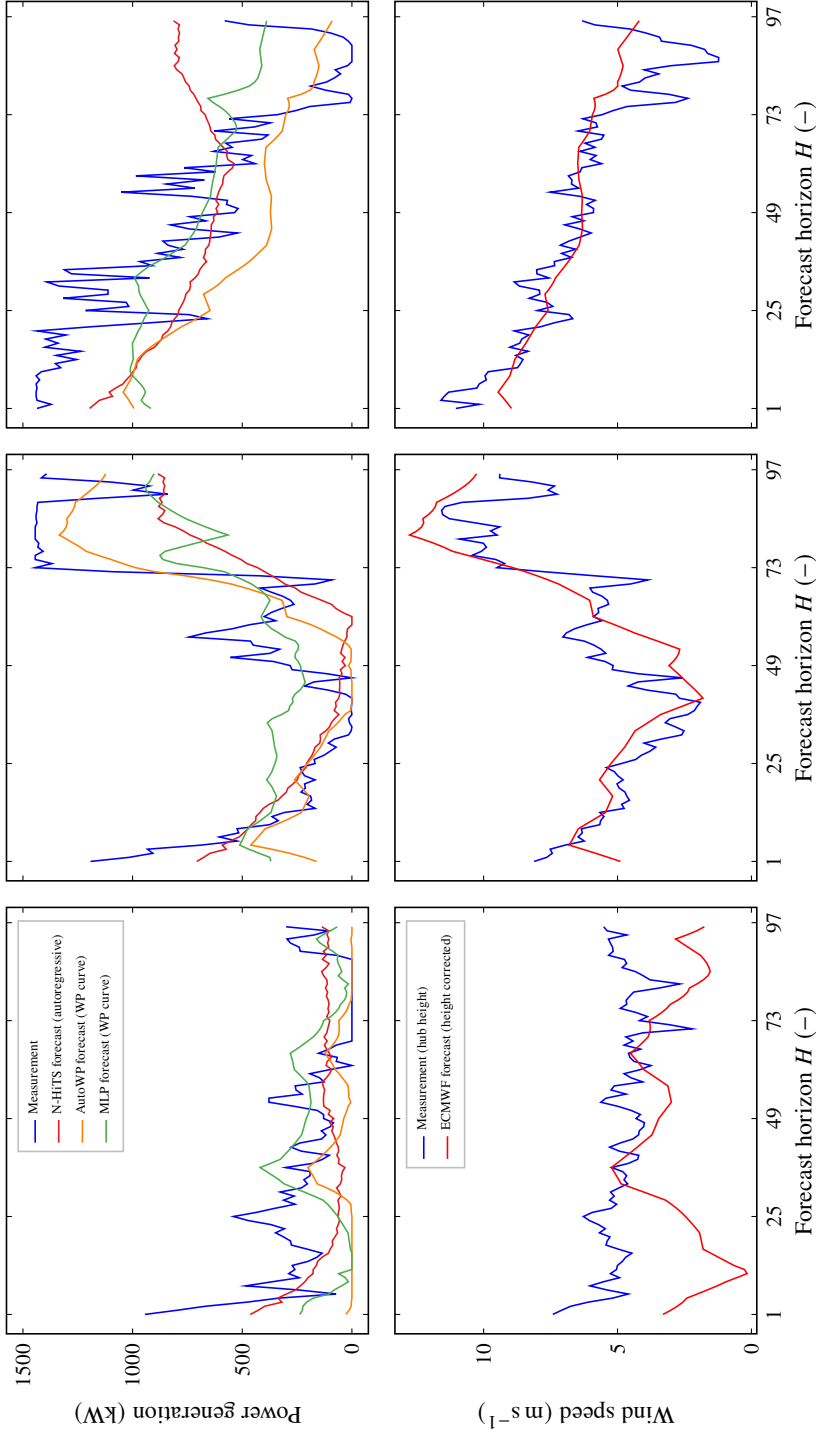


Figure 4.10: Exemplary comparison of WP forecasts for WP turbine no. 2 using N-HITS (autoregressive), AutoWP and MLP (WP curve modeling) on days without shutdowns. While methods based on WP curve modeling are only based on weather forecasts (ECMWF), autoregressive methods also consider past WP generation measurements when making forecasts. Legend of first column of a row applies to all columns in the row. The figure is based on [166].

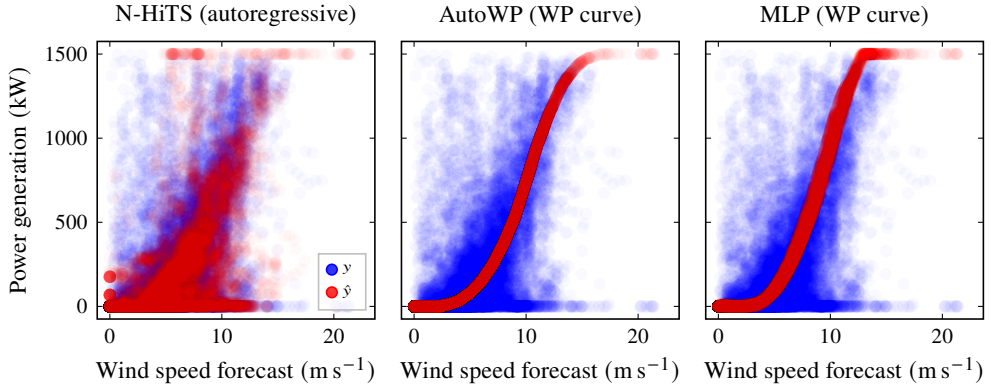


Figure 4.11: Relationship between wind speed forecast (ECMWF) and WP generation forecasts (N-HiTS, AutoWP, MLP) for WP turbine no. 2 when *disregarding* shutdowns. AutoWP only relies on the height-corrected wind speed forecast, the MLP is trained using the forecasts of wind speed and direction, temperature and air pressure, and N-HiTS additionally considers past WP generation measurements. No WP generation despite moderate and high wind speeds are not shutdowns (already filtered out) but result from the error in the weather forecast, i. e., the wind speed forecast is above the cut-in speed while the realized wind speed is underneath. Legend of first column applies to all columns. The figure is based on [166].

Benefits. Since AutoWP is based on an ensemble of different WP curves, it can represent different site conditions characterized by terrain-related air turbulence. Specifically, these conditions affect the wind speed at which the WP turbine’s peak power generation is reached. As the evaluation shows, the site conditions of the target WP turbine can differ significantly from the conditions under which the OEM WP curve was determined, which may result in a high forecast error when applying the OEM WP curve. Although a site-specific WP curve can also be estimated using ML methods, AutoWP is beneficial since physical limitations in WP generation are implicitly considered, requiring only a few data to train the model. This scalability is essential for the deployment to decentral located onshore WP turbines for smart grid applications such as redispatch planning.

4.4 Related work

In the related work, we first focus on renewable energy forecasting. Then, we analyze WP forecasting methods in the context of redispatch interventions. Finally, we review work that considers inconsistent WP data and summarize the research gap.

As outlined in the introduction, with the expansion of renewable energies in the electrical power grid, the need for forecasts is increasing. Here, forecasts with a strong dependency

on the weather are PhotoVoltaic (PV) [168, 279, 280] and WP. In the latter, the methods can be categorized into the following types:

- The first are autoregressive, i. e., the method relies on past WP generation values of the target turbine to make a forecast. Additionally, weather forecasts can be considered as exogenous features.
- The second does not take past values into account and models the so-called WP curve, which is the empirical relationship between WP and wind speed. Additionally, further explanatory variables can be considered.

Regarding the first type, methods based on Statistical Modeling (SM), ML, and DL exist. Classic autoregressive WP forecasting methods include approaches from SM such as models based on the AutoRegressive Moving Average (ARMA) [281] or its integrated variant, the ARIMA [282]. While the above methods neglect the strong dependence of WP on weather, this can be modeled by regression methods based on ML, e. g., the MLP [283], or DT-based methods like XGB [276]. In addition to the weather, further information can be considered. For example in [276], the authors consider cyclic features to encode seasonal relationships, comparing regression-based methods to the Long Short-Term Memory (LSTM), a DL method. Precisely, DL is a sub-set of ML where the methods are based on deep ANNs with specialized architectures. In the literature on WP forecasting, DL architectures are used that include convolutional and recurrent layers [284–287], residual connections [288], as well as attention-based transformer architectures [289–293]. While DL methods can learn complex temporal and covariate relationships, they have limitations in their scalability due to the high computational effort required for training. Furthermore, concerns regarding shutdowns due to redispatch interventions arise for all methods based on autoregression. Specifically, autoregressive methods are at risk to forecast future redispatch interventions, which is not the forecasting target in redispatch planning as outlined in the introduction.

Regarding the second type, parametric and non-parametric modeling approaches for WP curves exist [294]. Parametric approaches rely on the fitting of assumed mathematical modeling expressions, such as polynomial, exponential, and cubic expressions [295]. Here, the WP curve is often separated into four sections, which are separated by the cut-in, nominal and cut-out wind speed [296]. Importantly, such WP curves are only valid for the wind speed at the height from which the measurement data originate. Non-parametric approaches include methods based on ML, e. g., the MLP [151, 152, 297], SVR [151, 298, 299], and DT-based methods [153, 298, 299]. Additionally, ML approaches can also take into account further explanatory variables like wind direction [152, 153, 297], air temperature [152, 154] and atmospheric pressure [154]. A critical challenge for WP curve modeling, however, is inconsistent data, as detailed in the following.

Inconsistent data due to partial load operation or shutdowns pose a challenge for modeling normal WP operation. Specifically, Morrison et al. [277] identify three types of abnormal operation: no power generation whilst above cut-in wind speed, steady power generation at a power less than the turbine’s peak power rating, and power generation by stop-to-operation transitions and vice versa. Therefore, data cleaning is used when modeling WP curves in the literature, such as removing data samples with a power generation that is lower than a threshold near zero [151, 152] or deviates from an already fitted WP curve [151]. Furthermore, expert knowledge can be used to label abnormal operation [299]. Automated approaches that do not require rule or expert knowledge are outlier detection methods, such as calculating a LOF for each sample and labeling all samples that have a substantially lower LOF than their neighbors as outlier [277]. While the pre-processing of inconsistent data is widely used in the modeling of WP curves, its use is still not considered in many state-of-the-art autoregressive methods [276, 286, 288, 290, 292, 293] or is only used for the pre-processing of training data [151, 291]. However, it remains unclear how inconsistent power generation during the operation of autoregressive forecasting models, i. e., models that use past power generation values to make a forecast, impact the forecast error.

4.5 Novelty and remaining questions

Given the shortcomings of related work identified in the previous section, the novelty of AutoWP can be summarized as follows:

1. AutoWP provides a computationally efficient forecasting method for scalable deployment to hundreds of individual WP turbines.
2. AutoWP implicitly includes PK about physical limitations in WP generation, i. e., the power generation is limited by the peak power rating of the WP turbine, negative WP generation is impossible, and the WP turbine is shut down when the wind speed exceeds the cut-out speed.
3. AutoWP uses a rule-based and filter-based approach for training data cleaning, which can be applied in the operation to identify abnormal operation.

The automation of the design together with the operation monitoring makes AutoWP an Automation level 3 method under the AutoLVL taxonomy [162, 163] of Chapter 2. Although designed as a point forecasting method, quantiles can be estimated using the normalizing flow-based cINN (Chapter 3), as exemplified in Section C.2 in the Appendix. In terms of the stated open questions of this thesis in Section 1.5, AutoWP combines automated pre-processing with forecast ensembling to create a computationally efficient WP forecasting


template. Apart from the publications [166, 167], AutoWP is available as open-source software, as linked in Figure 4.12. However, given the limitations of AutoWP discussed in Section 4.3, new questions may be raised:

- How can regular shutdowns be considered in forecasting with AutoWP using PK of timed shutdowns?
- Does designing a separate classification model that predicts whether the WP turbine is operating or shut down resolve the previous question?
- How can regular and irregular shutdowns be reliably distinguished in a data set?
- How can gradual interventions in the WP generation capabilities (e. g. as experienced in [300, 301]) be considered?
- Does relaxing the restriction AutoWP’s ensemble optimization resolve the previous question for permanent peak power curtailments?
- How can timed gradual interventions be taken into account?
- Does designing a separate regression model that predicts the WP turbine’s generation capability factor resolve the previous question?

These stated questions must be tackled in future work to improve AutoWP.

AutoWP

Template for automated wind power forecasts



Publications

S. Meisenbacher et al., “AutoWP: Automated wind power forecasts with limited computing resources using an ensemble of diverse wind power curves”, in *Proceedings of the 34. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2024, pp. 1–20.

S. Meisenbacher et al., *On autoregressive deep learning models for day-ahead wind power forecasts with irregular shutdowns due to redispatching*, 2024. arXiv: 2412.00423.

Implementation

<https://github.com/SMEISEN/AutoWP>

Figure 4.12: The contributions of this chapter with the Automation level 3 template AutoWP.¹¹

¹¹ The icon of the WP turbine is adapted from [14].

5 AutoPV: Template for automated photovoltaic forecasts

Forecasts of the locally distributed PhotoVoltaic (PV) power generation are necessary for balancing the electrical transmission and distribution grids to avoid grid congestion [265]. To keep pace with the expansion of the PV power generation capacity in future energy systems, automating the design and operation of PV forecasting models is necessary. However, three major challenges exist:

First, the mounting configuration of the target PV plant characterized by tilt and azimuth angles $\beta_{T,array}$ and $\theta_{A,array}$ is difficult to obtain as both are often only roughly approximated or incompletely documented. Especially for residential rooftop installations, the PV mounting configuration is difficult to obtain because homeowners generally do not know these technical details [149]. Moreover, since the cost of PV arrays has decreased, it has become more common for the entire available roof area to be utilized, possibly resulting in mixed-oriented PV plants [302]. In many cases, the peak power rating of these PV plants is the only known characteristic. This missing documentation limits the large-scale application of methods that rely on information about the PV mounting configuration (e. g. [303–305]) and approaches based on meta-learning (e. g. [306]).

Second, historical data of the target PV plant for training the model are often not available (cold-start problem). For example, for newly built PV plants, nearly no data are available. Consequently, transfer-learning approaches and data-intensive methods such as Deep Learning (DL) (e. g. [307–309]) are not directly applicable to new PV plants. However, this cold-start capability is essential if the deployment of the forecasting models has to keep pace with the expansion of PV power generation capacity in future energy systems.

Third, exogenous impacts on the PV plant’s generation capabilities, such as PV array soiling, age-related performance degradation, maintenance, or sudden failures of individual arrays may occur [310–312]. These impacts can cause concept drifts, which negatively affect the forecast quality if the model no longer represents the actual PV plant’s characteristics. Therefore, model monitoring during operation is required to detect degrading forecast quality, alert the operator, and initiate model adaption to the new concept. While adaption to the new concept must be quick, it must also be possible to return to the original concept once the PV plant’s performance issue is resolved.

AutoPV addresses these three challenges and combines automated design and automated operation (Automation level 4 according to the AutoLVL taxonomy [162, 163] in Chapter 2).¹

5.1 Automated design methods

The underlying idea of AutoPV is to describe the arbitrary mounting configuration of a new, possibly mixed-oriented PV plant as a convex linear combination of outputs from a sufficiently diverse ensemble pool of PV models of the same region. AutoPV incorporates three steps: i) create the ensemble model pool, ii) form the ensemble output by an optimally weighted sum of the scaled model outputs in the pool, and iii) re-scale the ensemble output with the new PV plant's peak power rating $P_{\max, \text{new}}$. These three steps are detailed in the following, and Figure 5.1 provides exemplary results of these three steps.

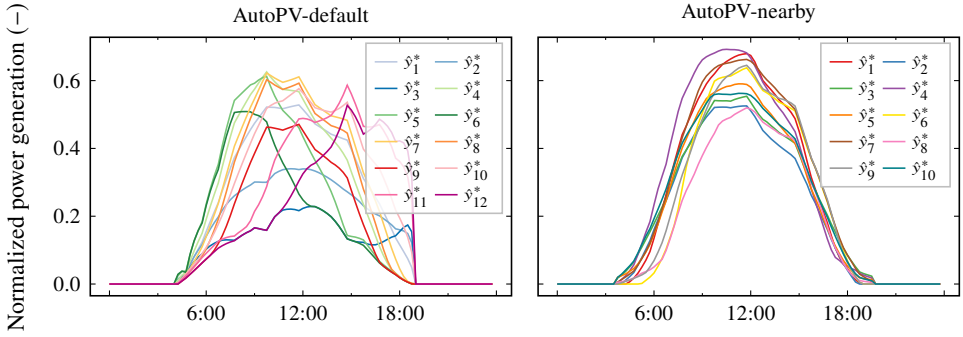
5.1.1 Creation of the ensemble pool models

As mentioned before, the underlying idea of AutoPV requires a sufficiently diverse ensemble model pool of the same region. In the following, we present the design of two ensemble model pools, which we refer to as AutoPV-default and AutoPV-nearby (see also Figure 5.1). AutoPV-default is required when no historical data from nearby PV plants are available and uses Physical-inspired Modeling (PM) to create the ensemble model pool using the new PV plant's location. AutoPV-nearby, on the other hand, uses available historical data from nearby PV plants and Machine Learning (ML) to create the ensemble model pool. Both require the new PV plant's peak power rating $P_{\max, \text{new}}$ for the transformation described in Section 5.1.3.

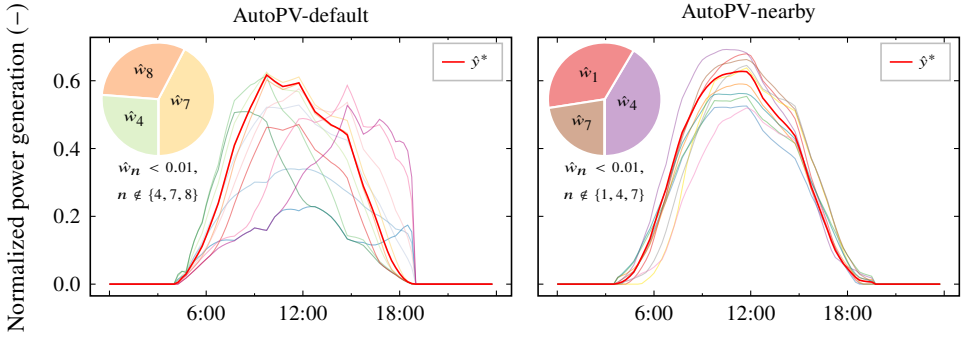
Default model pool for PV plants without data of nearby plants

The case of no available data of nearby PV plants requires building an ensemble pool of models that do not rely on historical data. Therefore, we use 12 PM-based models with different PV mounting configurations (tilt $\beta_{\text{T,array}} \in \{15^\circ, 45^\circ, 75^\circ\}$, azimuth $\theta_{\text{A,array}} \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$). Thus, the resulting default ensemble model pool covers a diverse set of PV mounting configurations. While each model represents a different mounting configuration, we realize each model as an equally designed pipeline based on the two-step process described in [313], which consists of i) estimating Plane On Array (POA) irradiance and ii) estimating power generation. We refer to this pipeline as PM pipeline.

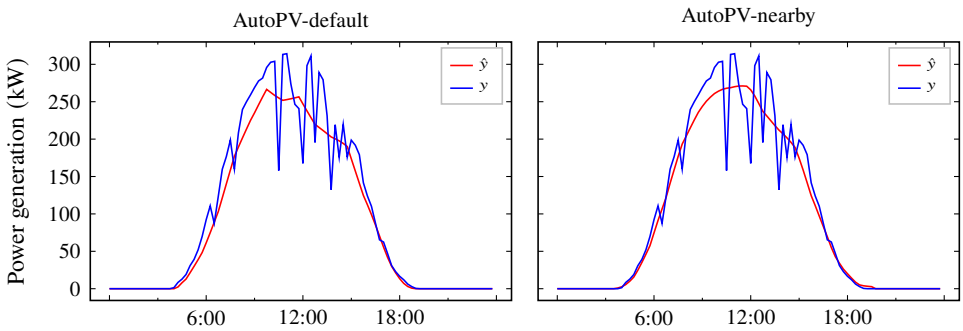
¹ Note that this chapter is based on [168, 169] and contain identical phrases; a detailed list is given in the Appendix.



(a) The result of the first step is the scaled PV model outputs \hat{y}_n^* , $n \in \mathbb{N}_1^{N_m}$ of the N_m models in the ensemble pool. The models are either created using a PM pipeline (AutoPV-default) or using a ML pipeline (AutoPV-nearby).



(b) The result of the second step is the scaled ensemble output \hat{y}^* . The ensemble output results from the convex linear combination of the ensemble pool models' outputs, with the weights \hat{w}_n , $n \in \mathbb{N}_1^{N_m}$ adapted to optimally fit the new PV plant.



(c) The result of the third step is the re-scaled ensemble output \hat{y} . Additionally, the measurement y with cloud overflight between 12:00 and 15:00 is shown. The ensemble output is based on a day-ahead solar irradiance forecast that also considers cloud cover but with lower spatial and temporal resolution.

Figure 5.1: The three steps of AutoPV's automated design exemplified for the real-world PV plant no. 11 [169].

In the following, we first describe the PM pipeline for creating the default ensemble model pool, followed by the implementation.

Physical-inspired modeling pipeline. In the PM pipeline, we first determine the POA irradiance, then estimate the power generated by the PV plant from the POA irradiance, and finally, normalize the estimated power generation, making it transformable to other PV plants with different peak power ratings P_{\max} , as shown in Figure 5.2.

For the first step in PM, we determine the solar position over time for the given location of the PV plant, i.e., the sun's azimuth and zenith angles θ_A and θ_Z . For this, we use the algorithm of the National Renewable Energy Laboratory (NREL) [314]

$$(\theta_A, \theta_Z) [k] = f(t[k], (\text{lat}, \text{long})), \quad (5.1)$$

which is a function of the sampled time $t[k]$, where $k \in \mathbb{N}^K$ is the time series index, and the latitude and longitude (lat, long) coordinates. The subsequent processing step depends on the given pipeline inputs:

- a) If the Diffuse Horizontal Irradiance (DHI) is not provided by the weather data service, we first estimate the Direct Normal Irradiance (DNI) from Global Horizontal Irradiance (GHI). For the estimation, we use the Direct Insolation Simulation Code (DISC) model [315], which combines physical principles with empirical correlation functions [313]:

$$\text{DNI} [k] = f(\text{GHI} [k], t[k], \theta_Z [k]). \quad (5.2)$$

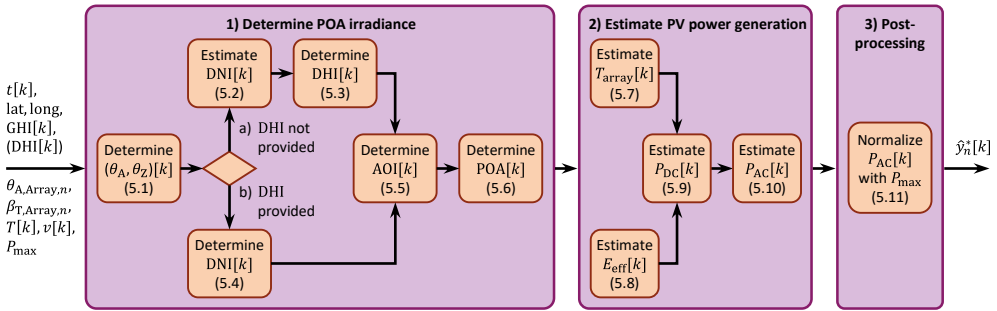


Figure 5.2: The PM pipeline for creating a model for the default model pool, which consists of N_m models. For creating the n -th model, the pipeline first takes the time stamp $t[k]$, the latitude and longitude (lat, long) coordinates, the GHI $[k]$, optionally the DHI $[k]$, and the PV mounting configuration $(\theta_{A, \text{array}, n}, \beta_{T, \text{array}, n})$ to determine the POA irradiance. Then, it estimates the PV power generation $P_{AC} [k]$ based on POA $[k]$, the air temperature $T [k]$, and the wind speed $v [k]$. Finally, it normalizes $P_{AC} [k]$ with the peak power rating P_{\max} , resulting in $\hat{y}_n^* [k]$. The figure is based on [168].

Then, we determine the $\text{DHI}[k]$ with the solar irradiance composition equation

$$\text{DHI}[k] = \text{GHI}[k] - \text{DNI}[k] \cdot \cos(\theta_Z[k]) \quad (5.3)$$

from the provided $\text{GHI}[k]$, and the previously estimated $\text{DNI}[k]$ and $\theta_Z[k]$.

b) Otherwise, we determine the $\text{DNI}[k]$ with the above transposed equation

$$\text{DNI}[k] = \frac{\text{GHI}[k] - \text{DHI}[k]}{\cos(\theta_Z[k])} \quad (5.4)$$

from the provided $\text{GHI}[k]$ and $\text{DHI}[k]$ weather data.

After this input-dependent processing step, we determine the Angle Of Incidence (AOI) of the solar vector on the PV array surface

$$\begin{aligned} \text{AOI}[k] = \arccos & \left(\cos(\theta_Z[k]) \cdot \cos(\beta_{\text{T,array}}) + \right. \\ & \left. \sin(\theta_Z[k]) \cdot \sin(\beta_{\text{T,array}}) \cdot \cos(\theta_A[k] - \theta_{\text{A,array}}) \right), \end{aligned} \quad (5.5)$$

where $\beta_{\text{T,array}}$ and $\theta_{\text{A,array}}$ are the tilt and azimuth angles of the PV array. Finally, we determine the POA irradiance using the model of Hay and Davies [316]

$$\text{POA}[k] = f(\text{AOI}[k], \text{GHI}[k], \text{DHI}[k], \text{DNI}[k], \text{DNI}_{\text{ex}}[k], m_{\text{rel}}[k]), \quad (5.6)$$

with the $\text{DHI}[k]$, the $\text{DNI}[k]$, the extraterrestrial radiation $\text{DNI}_{\text{ex}}[k]$, and the relative air mass $m_{\text{rel}}[k]$. The extraterrestrial radiation $\text{DNI}_{\text{ex}}[k]$ is the intensity of the sun at the edge of the Earth's atmosphere, estimated with the method of Spencer $f(t[k])$ [317]; and the relative air mass $m_{\text{rel}}[k]$ represents the ratio of the absolute air mass through which sunlight travels to the air mass at standard atmospheric conditions, estimated with the method of Kasten and Young $f(\theta_Z[k])$ [318].

In the second step of the PM pipeline, we model the PV array using the Sandia PV Array Performance Model (SAPM) [150]. The model makes three estimations: First, the model estimates the PV array's cell temperature T_{array}

$$T_{\text{array}}[k] = f(\text{POA}[k], T[k], v[k]) \quad (5.7)$$

based on the POA irradiance, the air temperature $T[k]$, and the wind speed $v[k]$. Second, the model estimates the effective irradiance E_{eff}

$$E_{\text{eff}}[k] = f(\text{POA}[k], m_{\text{abs}}[k], \text{AOI}[k]) \quad (5.8)$$

to account for reflections and spectral loss based on the POA irradiance, the absolute air mass $m_{\text{abs}}[k]$, and the $\text{AOI}[k]$. Third, based on the preceding two estimations, the model

estimates the voltage and Direct Current (DC) power from the PV array’s current-voltage (IV) curve:

$$P_{\text{DC}}[k] = f(E_{\text{eff}}[k], T_{\text{array}}[k]). \quad (5.9)$$

With the output of the SAPM, we then estimate the Alternating Current (AC) power using Sandia’s inverter model [319]:

$$P_{\text{AC}}[k] = f(E_{\text{eff}}[k], T_{\text{array}}[k]). \quad (5.10)$$

Finally, we normalize the estimated AC power of the n -th mounting configuration in the default ensemble model pool \hat{y}_n with the peak power rating P_{max} of the PV array

$$\hat{y}_n^*[k] = \frac{\hat{y}_n[k]}{P_{\text{max}}}, \quad n \in \mathbb{N}_1^{N_m} \quad (5.11)$$

with an equal P_{max} for each model. This enables recombining the N_m models to represent a new PV plant with a different peak power rating $P_{\text{max, new}}$, as described in Section 5.1.2 and Section 5.1.3.

Implementation. To realize PM, we implement each model in the ensemble pool as a pipeline. More specifically, we use the Python package `pvlb` [305] to implement the above-mentioned estimation steps. These estimation steps are executed sequentially. The model selection for each estimation step is based on `pvlb`’s user guide², and the PM pipeline’s overall configuration can be found in Table D.1 in the Appendix.

Model pool generated from historical data of nearby PV plants

The case of having historical power generation data of nearby PV plants and corresponding weather measurements allows using ML-based models in the ensemble pool. We realize each model as an equally designed pipeline that is pre-trained on historical data of a separate PV plant. We refer to this pipeline as ML pipeline. Including nearby PV plants that are sufficiently diverse allows the representation of a variety set of PV mounting configurations and shading conditions. To automatically design each model, we design a ML pipeline, which includes data pre-processing, feature engineering, HyperParameter Optimization (HPO), and regression algorithm selection, followed by forecast ensembling (Section 5.1.2). In addition, we leverage a set of rules reflecting Prior Knowledge (PK) for post-processing.

In the following, first, we describe the processing steps in the ML pipeline. Afterward, we describe how we automatically determine the best configuration of the regression algorithm for different PV plants. Finally, we describe how this model is implemented as a pipeline.

² https://pvlb-python.readthedocs.io/en/stable/user_guide/

Machine learning pipeline. The ML pipeline consists of pre-processing, feature engineering, regression estimation, and post-processing to include PK, as shown in Figure 5.3. In the following, we detail each step.

The first step is pre-processing. In the pre-processing, we aim to align the peak power rating P_{\max} of all considered PV plants. This enables recombining models trained on nearby PV plants to represent a new PV plant, regardless of the differing peak power ratings. Thus, for each model $n \in \mathbb{N}_1^{N_m}$, we pre-process the training data by scaling the power generation measurement according to the peak power rating $P_{\max, n}$ of the corresponding PV plant

$$y_n^*[k] = \frac{y_n[k]}{P_{\max, n}}, \quad (5.12)$$

where $k \in \mathbb{N}^K$ is the time series index. In contrast to the PV mounting configurations, we assume that P_{\max} is a parameter of the PV plants that can be reliably determined. However, scaling with P_{\max} does not lead to aligned PV power generation curves (see Figure 5.1a). The reason is that the mounting configuration of a PV plant influences the maximum possible amount of generated power. For example, a PV plant oriented to the south generates more than a PV plant oriented to the west. In addition, the GHI's seasonal and weather-dependent intensity also affects the distance between the PV power output and the plant's peak power rating P_{\max} .

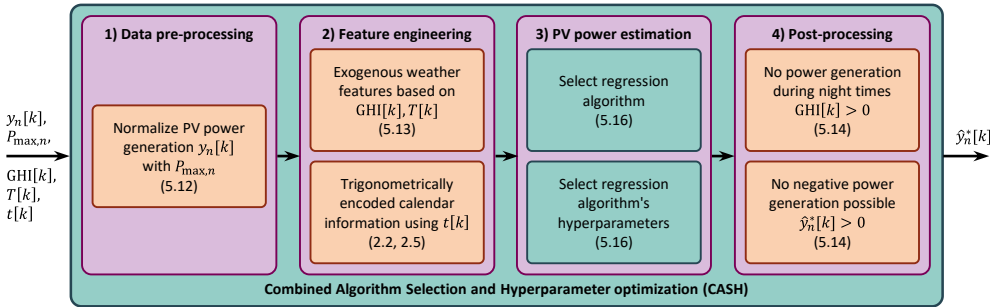


Figure 5.3: The ML pipeline for creating a model for the nearby plants model pool, which consists of N_m models. For creating the n -th model, the pipeline first normalizes the PV power generation measurement $y[k]$ with the peak power rating $P_{\max, n}$. Then, it creates explanatory features based on the GHI $[k]$, the air temperature $T[k]$, and the time stamp $t[k]$. Based on these features, a regression model is trained to estimate the normalized power generation \hat{y}_n^* , which is post-processed to ensure physical limits. Both the selection of the regression algorithm and the optimization of its hyperparameters are automated by CASH. The figure is based on [168].

Thus, the second step in the ML pipeline aims to extract features to describe these deviations. More specifically, we use the exogenous features GHI including cloud cover \hat{G} , air temperature \hat{T} , and corresponding second-order polynomial combinations³

$$\hat{G}^2[k], \hat{G}[k], (\hat{G} \cdot \hat{T})[k], \hat{T}[k], \hat{T}^2[k], \quad (5.13)$$

where the polynomial combinations provide non-linear features to linear regressors available in the configuration space (see next paragraph). Additionally, we use trigonometrically encoded cyclic features of the month (2.2) and minute of the day (2.5) to represent seasonal information. The periodicity of these trigonometric functions establishes similarities between temporally related samples, while the sin-cos pair is necessary because, otherwise, the encoding would be ambiguous at several points.⁴

The third step in the ML pipeline is a regression model that leverages the extracted features to estimate the P_{\max} -scaled power generation.

Finally, in the fourth step, we consider PK for post-processing using two rules: First, no PV power is generated if there is no solar irradiance. Second, negative PV power generation is impossible. Consequently, we drop the night times from the training data and set the model output to zero during these times. Furthermore, the negative values in the model output are set to zero.⁵ Incorporating these two rules leads to

$$\hat{y}_n^*[k] = \begin{cases} \hat{y}_n^*[k], & \text{if } \hat{G}[k] > 0 \text{ and } \hat{y}_n^*[k] > 0, \\ 0 & \text{otherwise,} \end{cases} \quad (5.14)$$

as the final step in the ML pipeline.

The resulting PV model for the n -th plant in the ensemble pool is a function of

$$\hat{y}_n^*[k] = f\left(\hat{G}^2[k], \hat{G}[k], (\hat{G} \cdot \hat{T})[k], \hat{T}[k], \hat{T}^2[k], x_{\sin, 12}[k], x_{\cos, 12}[k], x_{\sin, 1440}[k], x_{\cos, 1440}[k], \mathbf{p}\right), \quad (5.15)$$

where the parameters \mathbf{p} are determined by training the ML pipeline on historical data.

Combined algorithm selection and hyperparameter optimization. For automatically designing the regression model, we define a Combined Algorithm Selection and Hyperparameter optimization (CASH) problem. In the CASH problem, we aim to minimize

³ Note that the feature selection is based on the work of Tim Martin [159].

⁴ We do not consider historical data as lag features. The reason is that the models in the ensemble pool are pre-trained on data of different PV plants. Consequently, using historical data would require getting data from all used PV plants during model operation.

⁵ Slightly negative values may occur in the first data points at sunrise and sunset due to noise.

the ML pipeline's Mean Squared Error (MSE) (1.6) by selecting the (estimated) optimal configuration λ^* , i. e., selecting the optimal regression algorithm together with corresponding optimal hyperparameters simultaneously:

$$\lambda^* = \min_{\lambda \in \Lambda} \text{MSE}(\hat{y}_n^*(\lambda), y_n^*). \quad (5.16)$$

The configuration space Λ (Table D.2 in the Appendix) covers a diverse selection of regression algorithms, including ridge regression, MultiLayer Perceptron (MLP), Gradient Boosting Machine (GBM), Random Forest (RF), and Support Vector Regression (SVR), together with corresponding relevant hyperparameters. Λ includes categorical hyperparameters, and the selection of a regression algorithm conditions the corresponding sub-space. We implement the CASH using Bayesian Optimization (BO) with the Tree Parzen Estimator (TPE) as surrogate model, as reasoned in Section 2.2.2. During exploring the configuration space, each configuration trial is assessed using five-fold Cross Validation (CV) to prevent overfitting. That is, the data is split into five folds, where in each fold 80 % of the data are used for training and 20 % for validation. The CV is stratified to ensure that all months are equally represented in the training and validation split of each fold. Since the model structure (5.15) is static, the temporal ordering of the samples must not be preserved.

Implementation. The ML pipeline is implemented using the Python Workflow Automation Tool for Time Series (pyWATTS) package [171] and the regression algorithms are implemented using the Python package Scikit-learn [100]. For solving the CASH problem, we use the Python package Ray Tune [320] with the hyperopt [182] search algorithm. The CASH is automatically stopped if the MSE plateaus across trials, i. e., the MSE of the top ten trials have a standard deviation of less than 0.001 with a patience of 15 trials.

5.1.2 Optimal weighting of the ensemble pool models

To solve the cold-start problem, we combine ensemble optimization (Section 2.2.4) with pre-trained models (Section 2.3.2). To find the weights of the ensemble, we distinguish the initialization and the operation phase. In the initialization phase, historical power generation data for the new PV plant are not available. Thus, we weight each model in the ensemble pool equally. During the operation, new data are used to successively adapt the ensemble weights in such a way that they optimally fit the new PV plant (see Figure 5.1b). To allow adaption to

drifting power generation capabilities during operation, we modify the optimization problem of (2.11) by introducing the efficiency factor \hat{w}_e :

$$\min_{\hat{w}_e, \hat{\mathbf{w}}} \frac{\hat{w}_e}{K_{\text{ad}}} \sum_{k=1}^{K_{\text{ad}}} (\hat{y}^*(\hat{\mathbf{w}})[k] - y^*[k])^2 \text{ subject to} \quad (5.17)$$

$$\hat{w}_e \in [0, 1], \quad \hat{w}_n \in [0, 1] \quad \forall n \in \mathbb{N}_1^{N_m}, \quad \sum_{n=1}^{N_m} \hat{w}_n = 1.$$

For optimal weighting, we vary the efficiency factor \hat{w}_e and the weights of the ensemble pool models $\hat{\mathbf{w}}$ to minimize the MSE (1.6) over the K_{ad} most recent samples of the P_{max} -scaled data \mathbf{y}^* . We solve the optimization problem (5.17) using the least squares implementation of the Python package SciPy [187] with $\hat{w}_e = 1$, $\hat{w}_n = 1/N_m$, $\forall n \in \mathbb{N}_1^{N_m}$ as initial values, and normalize the weights afterward to hold the constraints.

5.1.3 Transformation of the ensemble output to the new PV plant

The last step yields the final ensemble output (see Figure 5.1c). To this end, it re-scales the weighted average of the ensemble \hat{y}^* according to the peak power rating $P_{\text{max, new}}$ of the new PV plant:

$$\hat{y}[k] = \hat{y}^*[k] \cdot P_{\text{max, new}}. \quad (5.18)$$

5.2 Automated operation methods

Automated adaption during operation is required to ensure that AutoPV remains valid under changing PV power generation capabilities. For AutoPV, cyclic adaption and drift detection-based adaption is applicable.

5.2.1 Cyclic adaption

In the cyclic adaption, AutoPV is adapted in a cyclic routine, regardless of the actual performance. The cycle length C_{ad} together with the number of considered samples K_{ad} must be adjusted to the seasonality of PV power generation. AutoPV can be adapted using the adaption routines Adaptive Fixed Batch (AFB) and Adaptive Increasing Batch (AIB), which are detailed in Section 2.4.1. In AFB with fixed C_{ad} and fixed K_{ad} , the old behavior is forgotten during model adaption. This can be prevented by using AIB, where K_{ad} increases in each adaption i , thereby including the old behavior ($K_{\text{ad}} = i \cdot K_{\text{ad, init}}$, $i = [1, 2, \dots]$). Potential drawbacks of the cyclic adaption concern the computational effort due to unnecessary adaptations and the delay in adapting to suddenly changing behavior.

5.2.2 Drift detection-based adaption

Different from cyclic adaption, drift detection-based adaption aims to adapt AutoPV whenever the actual model performance degrades. Applying the ADaptive WIndowing (ADWIN) drift detection method as detailed in Section 2.4.2 to AutoPV's forecast error appears intuitive since we want to adapt AutoPV when the performance degrades. However, apart from the ensemble weights $\hat{\mathbf{w}}$, the efficiency factor \hat{w}_e , and cyclic features x_{\sin}, x_{\cos} to encode temporal information, AutoPV is a function of the global irradiance \hat{G} and the temperature \hat{T} of an exogenous weather forecast:

$$\hat{y}[k] = f\left(\hat{\mathbf{w}}[k], \hat{w}_e[k], x_{\sin, i}[k], x_{\cos, i}[k], \hat{G}[k], \hat{T}[k]\right), i \in \{12, 1440\}. \quad (5.19)$$

The error of AutoPV consists of two portions: i) the exogenous error Q_{exo} induced by the deviations between the weather forecast and the actual weather measurement $\hat{G}[k]$ and $G[k]$, respectively, $\hat{T}[k]$ and $T[k]$, and ii) the endogenous error Q_{end} of the above function when using actual weather measurement data. Since we can only influence Q_{end} by adapting AutoPV's ensemble weights $\hat{\mathbf{w}}$ and efficiency factor \hat{w}_e , and not Q_{exo} of the exogenous weather forecasting model, we should eliminate Q_{exo} . Moreover, the forecast error can only be evaluated retrospectively once the PV power generation values $y[k]$ have been realized. Therefore, we can use the weather measurement instead of the weather forecast:

$$\hat{y}[k] = f\left(\hat{\mathbf{w}}[k], \hat{w}_e[k], x_{\sin, i}[k], x_{\cos, i}[k], G[k], T[k]\right), i \in \{12, 1440\}. \quad (5.20)$$

In this way, we eliminate Q_{exo} and isolate Q_{end} as modeling error.⁶ Accordingly, we expect a shorter drift detection delay and fewer false alarms when monitoring the modeling error (ADWIN -prediction) than when monitoring the forecast error (ADWIN -forecast). Consequently, we use the term forecast when using weather forecasting data as model input, and the term prediction when using weather measurement data as model input.

We choose the MSE as the metric to be monitored, as it strongly penalizes large deviations and is the basis for adapting the ensemble weights in (5.17). Since the PV power generation is higher in summer than in winter, the MSE is also subject to this seasonality. Therefore, we use the rolling normalized Mean Squared Error (nMSE) (A.3), which is normalized by the average realized power generation and calculated each day using the most recent 672 samples (7 days) of the quarter-hourly time series.

⁶ Note that a further endogenous error arises if the weather is not measured directly at the PV plant but at a nearby weather station.

5.3 Evaluation

In this section, we evaluate AutoPV.⁷ First, we assess the cold-start problem by comparing AutoPV to benchmark methods. Second, we assess the problem of missing information about the PV mounting configuration and investigate the interpretability of AutoPV’s ensemble weights. Third, we assess the adaptability of AutoPV under changing PV power generation capabilities.

5.3.1 Benchmarking

In the following, we first describe the experimental setup for assessing the modeling error and the forecast error in the cold-start problem. Afterward, we show the results and provide insights.

Experimental setup

In the experimental setup, we describe the used data, the evaluation strategy, the considered training and adaption configurations, and the considered benchmarks.

Data. We evaluate AutoPV on a real-world data set containing three years (2018, 2019, 2020) of quarter-hourly energy metering (kWh) from 11 PV plants with unknown mounting configurations and unknown shading conditions. Consequently, the time resolution in all evaluations is 15 minutes.⁸ The PV plants are located in southern Germany and have peak power ratings ranging from 100 kWp to 1000 kWp.⁹ For better interpretability, we convert the energy metering time series into the mean power generation time series using (A.5). To evaluate the modeling error, we use weather measurement data from DWD [321] and to evaluate the forecast error, we use day-ahead weather forecasting data from Meteomedia. Both include air temperature, wind speed, and GHI considering cloud cover.¹⁰ As introduced

⁷ A Python implementation of AutoPV can be found on GitHub: <https://github.com/SMEISEN/AutoPV>

⁸ The models in the nearby plants pool are capable to handle arbitrary resolutions. If using a resolution of less than a minute, the feature minute-of-the-day might need to be changed to second-of-the-day. However, this is merely a feature re-scaling, as it can be assumed that the actual solar irradiance has a greater impact on the short-term forecasts than the time encoding.

⁹ While the data set is not publicly available, it provides the possibility to evaluate the nearby plants model pool, as the 11 PV plants are located close to each other, and historical data of the PV plants’ power generation with corresponding weather measurements and weather forecasts are available. An exemplary combination of AutoPV-default with the conditional Invertible Neural Network (cINN) [107, 115] to generate quantile forecasts based on point forecasts (Chapter 3) on an openly available data set is given in Section D.2 in the Appendix.

¹⁰ Since the weather forecasting data does not include the DHI, AutoPV’s default model pool first estimates the DNI (5.2) and afterward determines the DHI (5.3), see condition a) in Figure 5.2.

before, we use the term forecast when using day-ahead weather forecasting data as model input, and the term prediction when using weather measurement data as model input. The data is split into a training data set (2018, 2019) and a test data set (2020).

Evaluation strategy. We evaluate the performance of AutoPV with the default and the nearby plants ensemble model pool using a plant-wise leave-one-out evaluation. That is, we assess the modeling error and the forecast error of AutoPV on each PV plant while excluding this plant from the nearby plants ensemble model pool. For the default model pool, all models remain in the pool because they are not based on the evaluation data. Since the PV plants in the data set have different peak power ratings and would differently influence the average MAE across plants, we use the normalized Mean Absolute Error (nMAE) (A.2) as test metric. Additionally, we report the normalized Root Mean Squared Error (nRMSE) (A.4) in the Appendix.

We run the evaluation on each data set ten times and report the arithmetic mean and the standard deviation across runs to make stochastic effects during training and optimization transparent.¹¹

Training and adaption configurations. In the evaluation, we compare three different training and adaption configurations: Historical Data Available (HDA) reflects an ideal situation in the sense that we assume that there are two years of historical data available (the training data set) for training. In AFB, every 28 days of the test data set, the most recent 28 days are used for adaption ($C_{ad} = 28$ d, $K_{ad} = 28$ d). AIB uses all data of the test data set that is obtained until the i -th adaption ($C_{ad} = 28$ d, $K_{ad} = i \cdot K_{ad, init}$, $i = [1, 2, \dots]$, $K_{ad, init} = 28$ d). We pre-train the models of AutoPV’s nearby plants ensemble pool with the training data set. In pre-training, the automated regression model design via CASH predominately selects an MLP regressor with two or three hidden layers and the Rectified Linear Unit (ReLU) activation function (see Table D.3 in the Appendix). Less frequently, an SVR is selected, particularly for PV plant no. 4 (seven out of ten runs). One possible reason for this is that the power generation curve of this plant follows the solar irradiance curve very closely, i. e., the time shift of the two curves is very small. The other regression algorithms of the configuration space (Table D.2 in the Appendix) are not selected.

For simulating a cold-start on the test data set, AutoPV is initialized with equal ensemble weights. After 28 days in operation, the weights are adapted using the ensemble optimization (5.17), comparing AFB and the AIB configuration.¹² Since we do not consider degrading

¹¹ Note that the standard deviation of methods based on deterministic training methods is zero.

¹² In addition to ensemble optimization, two other algorithms for online weight adaption are evaluated in [322], namely the Kalman filter and adaptive gradient descent, which does not lead to a significant improvement, however.

PV power generation capabilities in this evaluation, we fix $\hat{w}_e = 1$. Further, we assess the averaging ensemble, which reflects the initialization phase of AutoPV without adapting the weights during operation.¹³

Benchmarks. Apart from the averaging ensemble, we compare AutoPV to two methods: First, we train an Individual Model (IM) for the considered PV plant using the ML pipeline, including automated regressor design using CASH. Second, we fit the SunDance Python implementation [323] for the considered PV plant, which combines PM and ML modeling to estimate a single PV mounting configuration and make a forecast. For both, we assess the training and adaption configurations HDA, AFB, and AIB.

Consequently, we have three method classes in the present evaluation. The first class includes baseline methods that use two years of training data and thus do not correspond to a cold-start (IM-HDA, SunDance-HDA). The second class incorporates adaptive methods that require little training data and are quasi-cold-start capable (IM-AFB, IM-AIB, SunDance-AFB, SunDance-AIB). And the third class includes methods that require no initial training data and are cold-start capable (Averaging, AutoPV-AFB, AutoPV-AIB, with the default and the nearby plants model pool).

Results

In the following, we visualize and summarize the results of the benchmarking. An interpretation and discussion of these results can be found in the next Section 5.4.

In the benchmarking results shown in Figure 5.4, we make three observations that apply to both the modeling error and the forecast error: First, considering the average test metric over all plants \overline{nMAE} , we see that AutoPV using the nearby plants ensemble model pool and the ideal situation IM-HDA perform similarly (0.327 ± 0.065 vs. 0.323 ± 0.051). Second, regarding the cold-start problem, AutoPV outperforms the IM and SunDance regardless of the model pool (compare in Figure 5.4). Third, the nearby plants model pool outperforms the default model pool.

An additional observation concerns the performance of AFB and AIB: AutoPV-AIB performs slightly better than AutoPV-AFB. A finding that applies to all methods is a comparatively high $nMAE$ for PV plant no. 7, which has multiple dips in September and October, as well as a complete plant shutdown in May (see Figure D.2g in the Appendix).

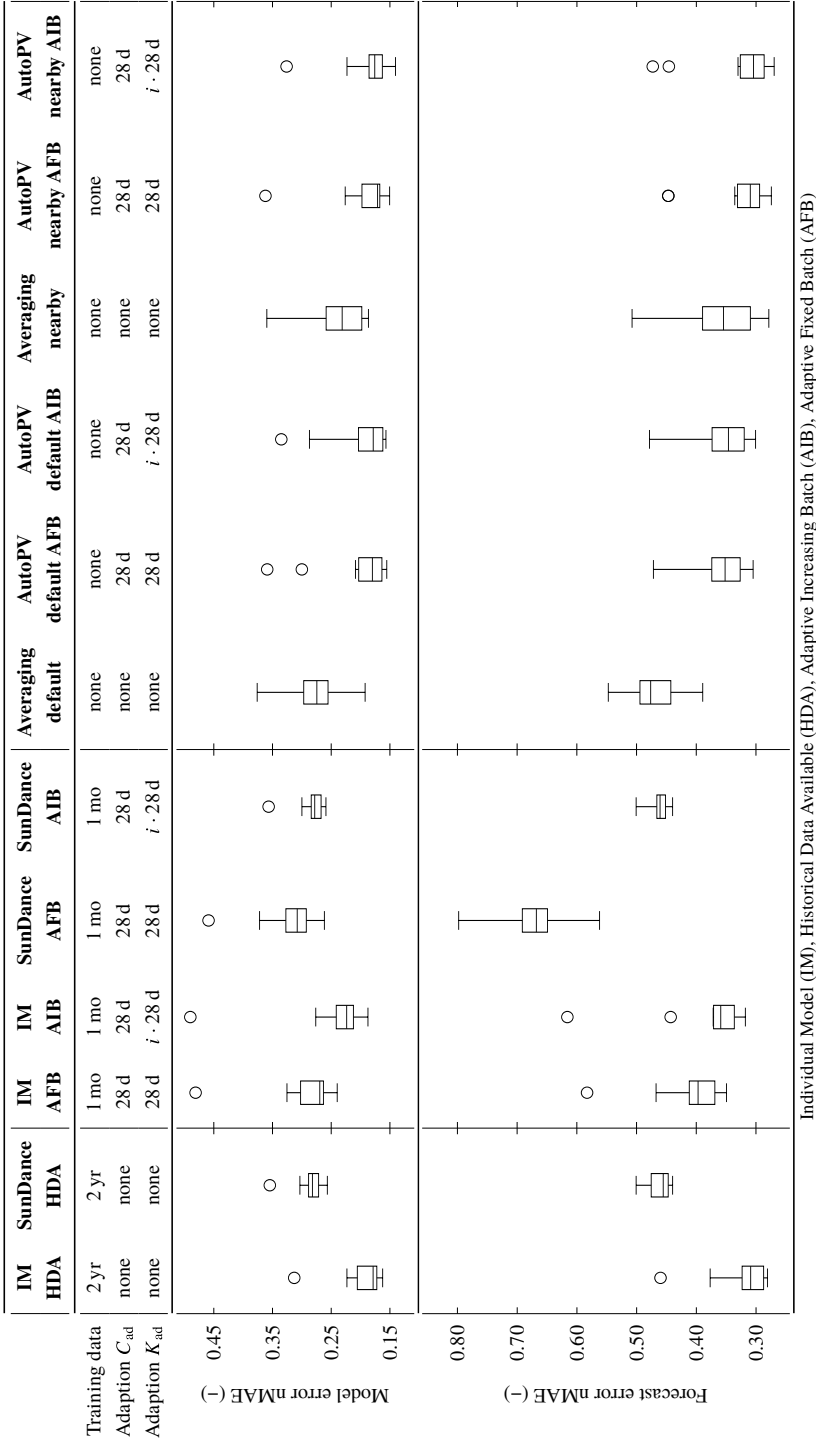


Figure 5.4: The modeling error and the forecast error (nMAE) of models in the plant-wise leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no initial training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The numerical values aggregated in this figure are detailed in Table D.4 and D.6 in the Appendix. The figure is based on [168].

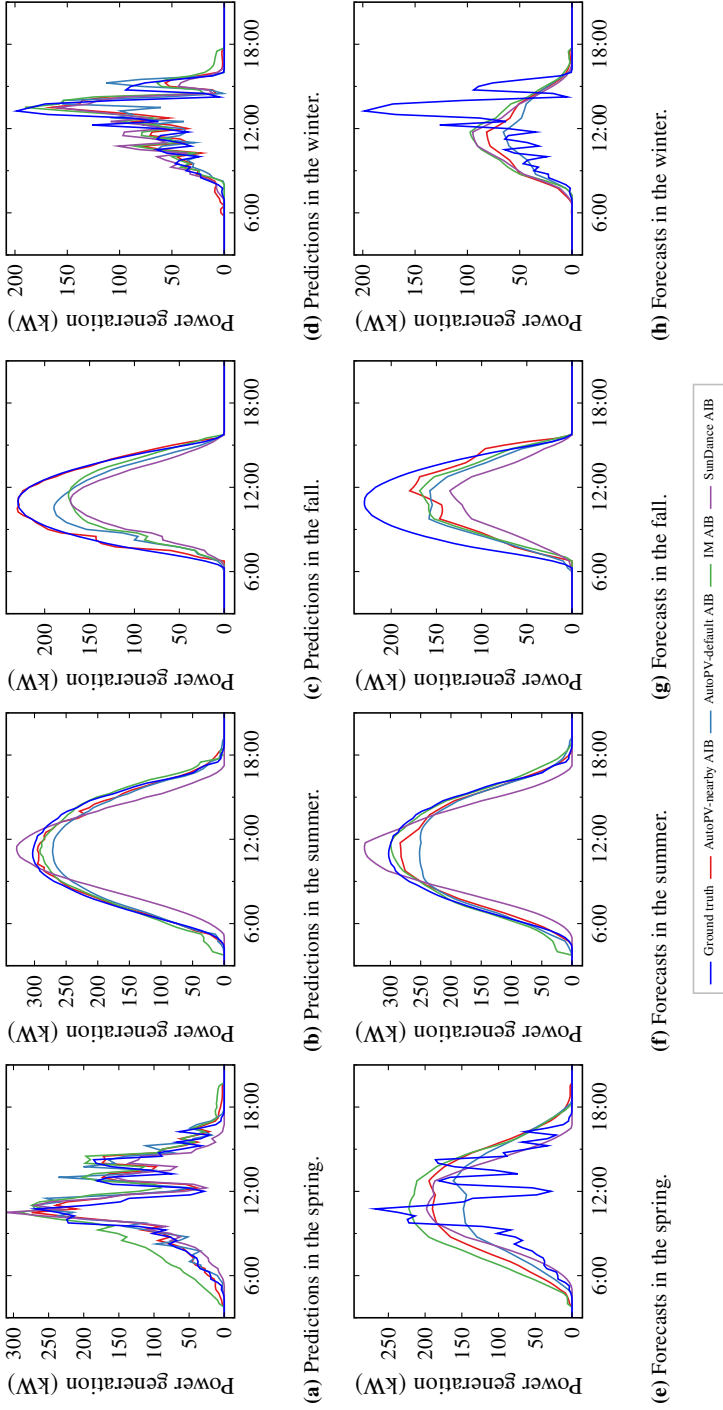


Figure 5.5: Exemplary predictions and forecasts in the four seasons on real-world data. Predictions are generated with weather measurement data as model inputs, and forecasts are generated with day-ahead weather forecasting data as model inputs. The weather measurements are taken from the closest weather station (~ 10 km). That is, there may be a slight time delay between the weather measurement data and the PV generation data when clouds are passing the PV plant. The day-ahead weather forecasts consider cloud cover but with lower spatial and temporal resolution. As a result, sunny days in the summer and in the fall are typically easier to forecast than days with varying cloud cover in the spring and in the winter. The figure is based on [168].

Insights

A visual comparison of predictions and forecasts of AutoPV-default, AutoPV-nearby, the IM, and SunDance is shown in Figure 5.5. While the upper row shows the model predictions based on weather measurement inputs, the lower row shows the model forecasts based on day-ahead weather forecast inputs. It can be seen that the forecasts for changeable and cloudy days do not capture the dips due to cloud overflight accurately. This is because the day-ahead weather forecast considers the cloud coverage in lower temporal and spatial resolution. It is also noticeable that SunDance has a larger deviation in both forecasts and predictions, especially in the morning and evening. This is due to the fact that the underlying equation for the calculation of the POA irradiance can lead to negative values in the morning and in the evening for the given location if the plants are not oriented to the south. Although these negative values are set to 0, the width of the PV curve is not corrected. It is also noticeable that the IM-AIB overestimates PV power generation in the morning or evening in several cases. This may be explained as the online adapted model has not yet learned the entire season and extrapolates.

5.3.2 Interpretability

The second part of the evaluation assesses the problem of missing information about the PV mounting configuration and investigates the interpretability of AutoPV’s ensemble weight optimization using the default model pool. Thus, we first introduce the experimental setup for this evaluation. Afterward, we present the results and insights.

Experimental setup

In the experimental setup, we describe how we create a synthetic data set with known PV mounting configurations and the evaluation strategy.

Data. We create 12 synthetic mixed-oriented PV plants using AutoPV’s PM-based default model pool that represents 12 mounting configurations (tilt $\beta_{T,array} \in \{15^\circ, 45^\circ, 75^\circ\}$, azimuth $\theta_{A,array} \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$) and weather measurement data (2020) from DWD [321].

Generating data of a synthetic mixed-oriented PV plant includes two steps: First, we randomly select 3 of 12 models of the default pool and their shares, e. g., $\mathbf{w} = [0, 0.15, 0, 0, 0.37, 0, 0, 0, 0, 0, 0, 0.48]^\top$. Second, we generate the data with AutoPV using \mathbf{w} as ensemble weights and the weather measurement as input.

¹³ Note that the averaging ensemble is based on the work of Tim Martin [159].

Evaluation strategy. To evaluate the interpretability of AutoPV’s ensemble weight optimization, we use the synthetic mixed-oriented PV plant data of 2020 and corresponding weather forecasting data (instead of the weather measurement data used for data generation). More specifically, we compare AutoPV using the actual weights \mathbf{w} (PK) with AutoPV using the estimated weights $\hat{\mathbf{w}}$ of the ensemble optimization.

Training and adaption configurations. Because we evaluate the interpretability using AutoPV’s default model pool, training is not required. As in Section 5.3.1, AutoPV is initialized with equal ensemble weights and after 28 days in operation, the weights are adapted. Since the adaption configuration AIB performed best in the benchmarking, AFB is omitted. As before, we do not consider degrading PV power generation capabilities in this evaluation and fix $\hat{w}_e = 1$.

Results

In the following, we visualize and summarize the results of the interpretability evaluation. An interpretation and discussion of these results can be found in the next Section 5.4.

In assessing interpretability using the AIB adaption, we make three observations: First, we consider the average forecast error over all mixed-oriented PV plants \overline{nMAE} . Here, AutoPV using the estimated weights $\hat{\mathbf{w}}$ achieves a similar error as AutoPV using the actual weights \mathbf{w} (0.276 ± 0.009 vs. 0.274 ± 0.011 , see Table D.8 in the Appendix), which reflect the mounting configurations and shares used to generate the synthetic mixed-oriented PV plants. Note that the better performance of AutoPV compared to the benchmarking in Section 5.3.1 is because the synthetic data was generated with AutoPV’s default model pool and weather measurement data. Thus, the error of AutoPV using the actual weights \mathbf{w} results solely from the deviation between the weather measurement and the weather forecast. Second, the weights evolve over time and converge toward a steady state (see Figure 5.6). Apparently, the initially not fully-converged weight estimation does not have a significant impact on the forecast error compared to the ideal situation in terms of PK. Consequently, the AIB adaption can be terminated after the weights have converged. Third, the converged ensemble weight optimization results in a similar PV mounting configuration but often represents more shares than are actually present, compare $\hat{\mathbf{w}}$ and \mathbf{w} in Table 5.1. In this table, $\hat{\mathbf{w}}$ represent the estimated weights after one year (the end of December in Figure 5.6), and \mathbf{w} are the actual weights. For example, PV plant no. 1 actually consists of 15 % north-oriented and 45°-tilted, 37 % east-oriented and 45°-tilted, and 48 % west-oriented and 75°-tilted PV arrays. Instead of three shares, the ensemble optimization estimates a configuration consisting of six shares. However, it is noticeable that the estimated shares in sum are likely to result in a similar aggregate behavior. For example, regarding PV plant no. 1, the 12 % estimated north-

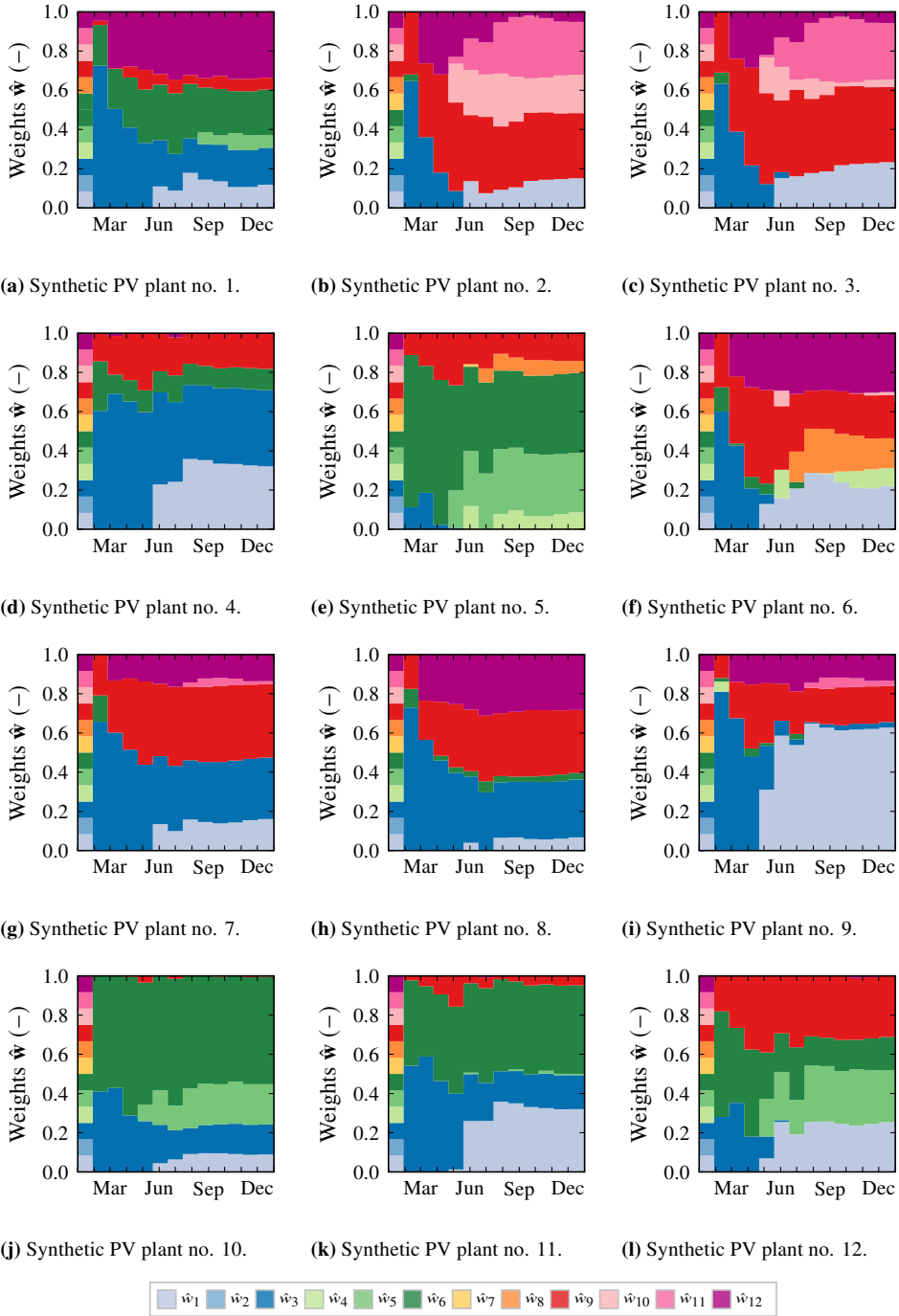


Figure 5.6: Evolution of estimated weights \hat{w} for the synthetic mixed-oriented PV plants; based on [168].

Table 5.1: AutoPV’s estimated weights $\hat{\mathbf{w}}$ after one year compared to the actual weights \mathbf{w} evaluated on synthetic mixed-oriented PV plants. For example, PV plant no. 1 actually consists of 15 % north-oriented and 45°-tilted, 37 % east-oriented and 45°-tilted, and 48 % west-oriented and 75°-tilted PV arrays. The table is based on [168].

	n	1	2	3	4	5	6	7	8	9	10	11	12
		15	45	75	15	45	75	15	45	75	15	45	75
	$\beta_{T,\text{array}}$	0	0	0	90	90	90	180	180	180	270	270	270
	$\theta_{A,\text{array}}$												
Plant	\mathbf{w}	0	0.15	0	0	0.37	0	0	0	0	0	0	0.48
no. 1	$\hat{\mathbf{w}}$	0.12	0	0.19	0	0.07	0.23	0	0	0.06	0	0	0.34
Plant	\mathbf{w}	0	0	0	0	0	0	0.24	0.27	0	0	0.50	0
no. 2	$\hat{\mathbf{w}}$	0.15	0	0	0	0	0	0	0	0.33	0.20	0.26	0.06
Plant	\mathbf{w}	0	0	0	0	0	0	0.36	0	0.19	0	0.45	0
no. 3	$\hat{\mathbf{w}}$	0.23	0	0	0	0	0	0	0	0.38	0.05	0.28	0.06
Plant	\mathbf{w}	0	0.56	0	0.20	0	0	0	0	0.24	0	0	0
no. 4	$\hat{\mathbf{w}}$	0.32	0	0.39	0	0	0.11	0	0	0.18	0	0	0
Plant	\mathbf{w}	0	0	0	0	0.49	0.27	0	0.25	0	0	0	0
no. 5	$\hat{\mathbf{w}}$	0	0	0	0.09	0.30	0.41	0	0.06	0.14	0	0	0
Plant	\mathbf{w}	0	0	0	0	0	0	0.49	0.15	0	0	0	0.36
no. 6	$\hat{\mathbf{w}}$	0.17	0	0	0.13	0	0	0	0.12	0.23	0.05	0	0.30
Plant	\mathbf{w}	0	0	0.40	0	0	0	0	0.39	0	0	0.20	0
no. 7	$\hat{\mathbf{w}}$	0.16	0	0.31	0	0	0	0	0	0.37	0	0.01	0.14
Plant	\mathbf{w}	0	0	0.33	0	0	0	0	0.35	0	0	0	0.32
no. 8	$\hat{\mathbf{w}}$	0.07	0	0.29	0	0	0.03	0	0	0.32	0	0	0.28
Plant	\mathbf{w}	0.32	0	0	0.33	0	0	0	0	0	0	0.35	0
no. 9	$\hat{\mathbf{w}}$	0.63	0	0.03	0	0	0	0	0	0.18	0	0.02	0.14
Plant	\mathbf{w}	0	0.24	0	0	0.44	0.31	0	0	0	0	0	0
no. 10	$\hat{\mathbf{w}}$	0.09	0	0.15	0	0.20	0.55	0	0	0	0	0	0
Plant	\mathbf{w}	0	0.34	0	0.37	0	0.30	0	0	0	0	0	0
no. 11	$\hat{\mathbf{w}}$	0.32	0	0.17	0	0	0.45	0	0	0.05	0	0	0
Plant	\mathbf{w}	0	0	0	0	0.52	0	0	0	0.22	0.26	0	0
no. 12	$\hat{\mathbf{w}}$	0.25	0	0	0	0.26	0.17	0	0	0.31	0	0	0

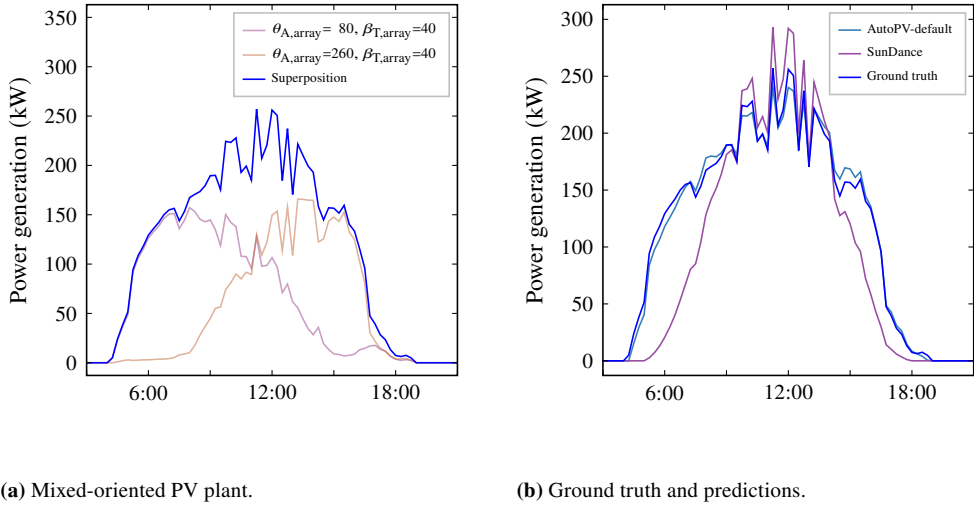


Figure 5.7: Representation of a mixed-oriented PV plant resulting from the superposition of the power generation curves of east- and west-facing arrays (at a ratio of 1:1). While mixed-oriented PV plants can be represented by AutoPV, they cannot be well represented by PM-based methods that estimate a single PV mounting configuration. The predictions are generated using weather measurement data to visualize the modeling error. The figure is based on [168].

15° share together with the 19 % estimated north-75° share are likely to result in similar aggregate behavior as the actual 15 % north-45° share.

Insights

To demonstrate the advantages of AutoPV for mixed-oriented PV plants compared to methods that are limited to estimating a single mounting configuration like SunDance, we consider a practical scenario. Although a south-facing orientation yields the most power in mid to high-latitude regions, many rooftop installations today consider a PV mounting configuration with half of the arrays oriented east and half of the arrays oriented west. The reason for this is that a PV plant with an east-west orientation generates power when it is needed – in the morning and in the afternoon – and the power generation is distributed more evenly throughout the day. Therefore, we evaluate the modeling error of AutoPV-default and SunDance using synthetic data of an east-west oriented PV plant, see Figure 5.7a.

We use $\theta_{A,array} = 80^\circ$ for the east-oriented arrays and $\theta_{A,array} = 260^\circ$ for the west-oriented arrays as azimuth angles. While keeping the ratio of the east- and west-oriented arrays constant at 1:1, we change the tilt angle of the arrays $\beta_{T,array} \in \{20^\circ, 30^\circ, 40^\circ\}$. Note that neither $\theta_{A,array}$ nor $\beta_{T,array}$ is directly represented in the model pool of AutoPV-default. For the three tilt angles, $\beta_{T,array} = 20^\circ$ represents a common configuration for flat roof

installations, and $\beta_{T,\text{array}} > 20^\circ$ represents configurations for gable roofs. Additionally, we evaluate a single-oriented south-facing PV plant with $\theta_{A,\text{array}} = 170^\circ$ with the above three tilt values.

Figure 5.7b visualizes the predictions using weather measurement data to visualize the modeling error. It can be seen that the wide PV power generation curve, which results from the superposition of east- and west-oriented arrays, cannot be well represented by a single configuration. While AutoPV-default can represent the mixed-oriented PV plant by corresponding weights, SunDance represents the mixed-oriented PV plant with the parameters of a south-oriented plant (see Table D.10 in the Appendix). Unfortunately, as also seen in Figure 5.5, the modeling error of SunDance is large for east- and west-oriented PV plants. Thus, the modeling error of SunDance for single- and mixed-oriented PV plants does not differ significantly (see Table D.11 in the Appendix).

5.3.3 Drift detection

The third part of the evaluation assesses the problem of degrading PV power generation capabilities during operation, e. g., due to PV array soiling or sudden failure of individual PV arrays. As before, we first describe the experimental setup, and afterward, we show the results of the evaluation, and provide insights.

Experimental setup

The experimental setup comprises the data description, how concept drifts are synthetically induced, the evaluation strategy, as well as the considered training and adaption configurations.

Data. We evaluate AutoPV using the nearby plants model pool on the real-world data set containing three years (2018, 2019, 2020) of quarter-hourly energy metering (kWh) from 11 PV plants, which we also used in Section 5.3.1. As before, we convert the energy metering time series into the mean power generation time series using (A.5). To evaluate the modeling error, we use weather measurement data from DWD [321] and to evaluate the forecast error, we use day-ahead weather forecasting data from Meteomedia. Both include air temperature, wind speed, and GHI considering cloud cover. Again, the data is split into a training data set (2018, 2019) and a test data set (2020).

To evaluate the adaptability of AutoPV under changing PV power generation capabilities, we insert two types of synthetic drifts into the test data set: The first type is a sudden drift, reflecting the sudden failure of PV arrays shown in Figure 5.8b. During the sudden drift, the peak power rating of the PV plant dips by a random value between 25 and 100 percent of the initial peak power rating. The second type is an incremental drift, reflecting the soiling

of PV arrays shown in Figure 5.8c. For the increase of soiling and simultaneous degradation of the generated PV power, we assume a linear relationship over time and a soiling loss of 0.21 % per day according to the empirical study of Mejia et al. [310]. In this scenario, the peak power rating of the PV plant returns suddenly to its original level after the arrays are cleaned.

For both drift types, the timing of the start and end of the drift is randomized, with a minimum time interval between the start and the end of 75 days and a maximum of 120 days. The resulting configuration of the synthetic drifts for each PV plant is shown in Table 5.2.

Evaluation strategy. We evaluate the performance of AutoPV-nearby using a plant-wise leave-one-out evaluation, i. e., we assess the forecast error on each PV plant while excluding this plant from the ensemble model pool. Because the PV plants in the data set have different peak power ratings, we use the nMAE as the evaluation metric. Otherwise, if using the MAE, the comparison across PV plants would be dominated by plants with high peak power ratings.

Training and adaption configurations. As in Section 5.3.1, we compare different training and adaption configurations: HDA reflects an ideal situation regarding training data in that we assume that two years of historical data (the training data set) are available for training. AFB adapts the model every 28 days of the test data set using the most recent 28 days for adaption ($C_{\text{ad}} = 28 \text{ d}$, $K_{\text{ad}} = 28 \text{ d}$). In AIB, all samples of the test data set that are obtained

Table 5.2: The configuration of the synthetic concept drifts inserted into the test data set; based on [169].

PV plant no.	Synthetic drift insertion			Sudden drift degradation	Incremental drift degradation
	start	end	duration		
1	7-Apr	30-Jun	84 d	-63.9 %	-0.21 % d ⁻¹
2	13-Jul	31-Oct	110 d	-79.1 %	-0.21 % d ⁻¹
3	30-May	5-Sep	98 d	-53.3 %	-0.21 % d ⁻¹
4	12-Jun	29-Aug	78 d	-27.7 %	-0.21 % d ⁻¹
5	12-Jun	29-Sep	109 d	-99.8 %	-0.21 % d ⁻¹
6	2-May	16-Aug	106 d	-31.3 %	-0.21 % d ⁻¹
7	12-Apr	28-Jun	77 d	-80.8 %	-0.21 % d ⁻¹
8	25-Apr	9-Aug	106 d	-26.5 %	-0.21 % d ⁻¹
9	11-Jun	15-Sep	96 d	-42.7 %	-0.21 % d ⁻¹
10	17-May	13-Sep	119 d	-32.2 %	-0.21 % d ⁻¹
11	30-Aug	25-Nov	87 d	-73.8 %	-0.21 % d ⁻¹

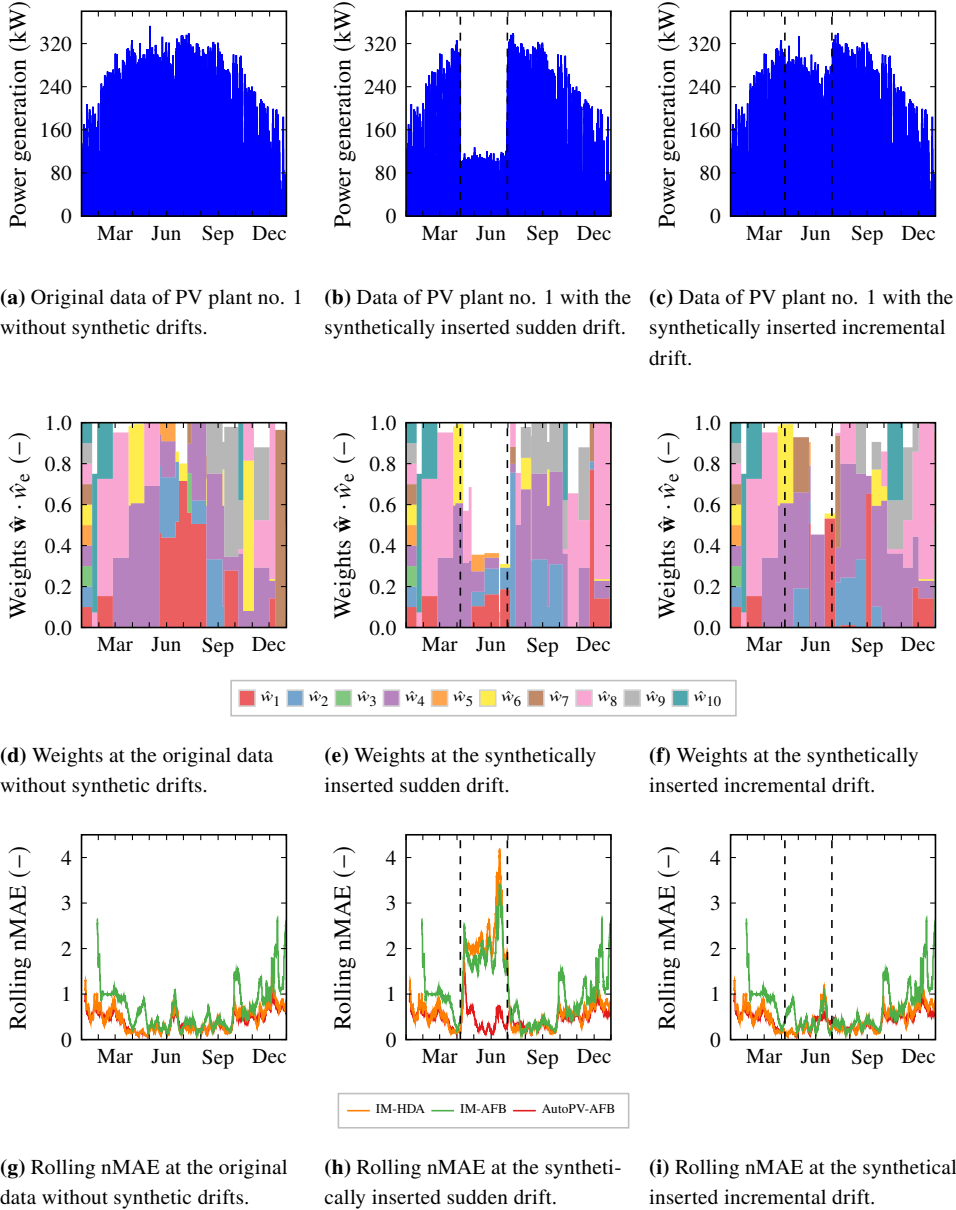


Figure 5.8: Evolution of AutoPV's weights over time and the rolling nMAE with a window size of seven days using ADWIN drift detection [198] based on the modeling error. The figure is based on [169].

until the i -th adaption are used ($C_{\text{ad}} = 28 \text{ d}$, $K_{\text{ad}} = i \cdot K_{\text{ad, init}}$, $i = [1, 2, \dots]$, $K_{\text{ad, init}} = 28 \text{ d}$). In addition to the cyclic adaption with AFB and AIB, we evaluate drift detection to trigger a-cyclic adaptations whenever the actual model performance degrades. A well-known drift detection method that detects the split between the old and the new concept is the ADWIN algorithm [198]. This split can be used in the adaption to consider only the samples of the new concept.

More precisely, we apply ADWIN with $\alpha_t = 0.05$ and compare monitoring the daily forecast error (ADWIN -forecast) with monitoring the daily modeling error (ADWIN -prediction) using the nMSE, see (A.3). Since it is an online drift detection method, ADWIN must not be trained.

We pre-train the models in the nearby plants ensemble pool with the data from 2018 and 2019. In the evaluation on the test data set, AutoPV-nearby is initialized with equal ensemble weights. After 28 days in operation, the ensemble optimization (5.17) with AFB and AIB combined with the above drift detections are compared.¹⁴ We also evaluate the averaging ensemble that reflects the initialization phase of AutoPV-nearby without adapting the weights during operation. In addition to averaging, we compare AutoPV-nearby to an IM for the considered PV plant. For the IM, we compare the training and adaption configurations HDA, AFB, and AIB.

In summary, the evaluation is divided into three classes of methods: The first class includes baseline methods that are not adaptive (IM-HDA, averaging). The second class incorporates adaptive methods that are quasi-cold-start capable since they require little training data (IM-AFB, IM-AIB). And the third class includes methods that are cold-start capable as they do not require initial training data (AutoPV-AFB, AutoPV-AIB).

Results

In the following, we visualize and summarize the results of the drift detection evaluation. An interpretation and discussion of these results can be found in the next Section 5.4.

The results of the plant-wise leave-one-out evaluation are shown in Table 5.3, where we make three key findings: First, regarding the nMAE on the original data without synthetic drifts, we see that AutoPV-nearby and the IM-HDA achieve similar performance regardless of the drift detection (none, ADWIN -forecast or ADWIN -prediction) and the adaption configuration (AFB or AIB). Also, we see that AutoPV-nearby outperforms the IM-AFB and IM-AIB. Second, in the evaluation with synthetic sudden drifts, AutoPV-AFB outperforms all other forecasting methods and achieves the best performance with ADWIN based on the

¹⁴ In addition to ensemble optimization, two other algorithms for online weight adaption are evaluated in [322], namely the Kalman filter and adaptive gradient descent, which does not lead to a significant improvement, however.

Table 5.3: The nMAE averaged over all PV plants in the leave-one-out evaluation on original real-world data and data with synthetically inserted drifts classified into three classes: i) baseline methods that are not adaptive, ii) adaptive methods that are quasi-cold-start capable since they require little training data, and iii) methods that are cold-start capable as they do not require initial training data. The table is based on [169] and the evaluation of the original data with fixed-cycle adaption trigger originate from [168].

Drift type	IM-HDA Averaging <i>no adaption</i>		Drift detection	IM-AFB IM-AIB <i>quasi-cold-start capable</i>	AutoPV-AFB AutoPV-AIB <i>cold-start capable</i>		
None	0.320±0.041	0.362±0.074	None	0.373±0.075	0.376±0.087	0.323±0.057	0.320±0.064
			ADWIN-forecast	0.418±0.069	0.452±0.094	0.322±0.056	0.318±0.042
			ADWIN-prediction	0.473±0.218	0.536±0.383	0.320±0.037	0.314 ±0.039
Sudden	0.596±0.246	0.629±0.217	None	0.639±0.238	0.630±0.208	0.412±0.106	0.491±0.182
			ADWIN-forecast	0.691±0.246	0.668±0.220	0.371±0.084	0.403±0.110
			ADWIN-prediction	0.871±0.529	0.996±0.797	0.362 ±0.052	0.396±0.095
Incremental	0.340±0.040	0.378±0.080	None	0.384±0.064	0.390±0.070	0.326±0.057	0.321±0.056
			ADWIN-forecast	0.432±0.072	0.411±0.066	0.327±0.052	0.318 ±0.034
			ADWIN-prediction	0.490±0.155	0.495±0.167	0.328±0.038	0.324±0.043

modeling nMSE. Additionally, we observe that the differences of ADWIN based on the forecasting and modeling nMSE are small. Third, the evaluation with synthetic incremental drifts also shows that AutoPV-nearby performs best, but the differences between drift detection and adaption configurations (AFB or AIB) are marginal.

The investigation of the drift detection’s performance confirms that the differences between ADWIN -forecast and ADWIN -prediction are marginal. Figure 5.9a shows the drift detection delay of both variants, where we evaluate the delay to detect the start of the drift and the delay to detect the end of the drift (i. e. the return to the original data) together. We consider a delay of more than 14 days as a miss-detection, which corresponds to half of the cyclic adaption cycle $C_{ad} = 28$ d.

Both ADWIN -forecast and ADWIN -prediction have four miss-detections in the evaluation and a median drift detection delay of six days. Regarding the false positives shown in Figure 5.9b, ADWIN -forecast shows a median of six out of 365, and ADWIN -forecast shows a median of eight out of 365 for each plant in the leave-one-out evaluation.

Insights

To gain insights into the drift adaption capabilities of AutoPV, we observe the evolution of AutoPV’s weights and error. Figure 5.8 shows the exemplary time series of PV plant no. 1, the resulting evolution of AutoPV’s weights, and the rolling nMAE with a window size of seven days.

Regarding the original data without synthetic drifts, we see that the sum of weights after the initialization phase of AutoPV-nearby does not always sum up to one to compensate

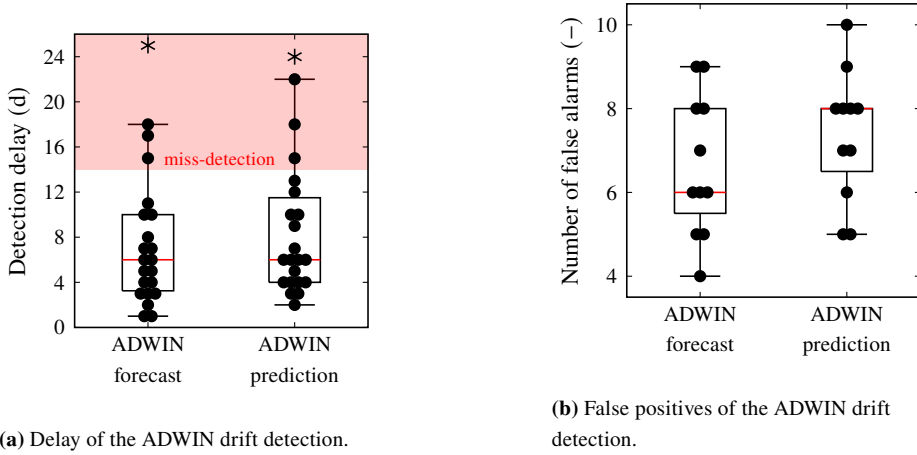


Figure 5.9: Comparison of the ADWIN drift detection method [198] based the forecast error and the modeling error on the synthetic sudden and incremental drifts. The figure is based on [169].

for reduced peak power (Figure 5.8d). For the rolling $n\text{MAE}(7\text{ d})$, AutoPV-nearby and the IM-HDA are comparable, while the IM-AIB has a higher error (compare in Figure 5.8g).

The synthetic sudden drift is clearly visible in the time series. Accordingly, a significant adaption of AutoPV’s weights can be seen in Figure 5.8e. Also, for AutoPV’s rolling $n\text{MAE}(7\text{ d})$, an increase can be seen in Figure 5.8h at the start of the drift, which returns to the initial level after drift detection and weight adaption. We also see that the error of the IM-HDA increases to a high level during the sudden drift and only returns to the initial level after the drift. Also, the IM-AIB is apparently unable to adapt to the sudden drift. In contrast, AutoPV can handle the concept drift and, after a short delay, achieves an error level similar as before the concept drift.

The synthetic incremental drift, in contrast, is not as clearly visible as the sudden drift (compare Figure 5.8a and Figure 5.8c). Accordingly, the effect on the weight adaption and the rolling $n\text{MAE}(7\text{ d})$ is also smaller. We see in Figure 5.8f that the sum of weights dips soon after the drift is detected, and the weights are adapted in four steps before summing back to one after the drift. The rolling $n\text{MAE}(7\text{ d})$ shown in Figure 5.8i illustrates that the differences between IM-HDA, IM-AIB and AutoPV are small, as already shown in Table 5.3.

5.4 Discussion

This section discusses the benchmarking results, AutoPV’s interpretability, the drift detection study, as well as the limitations and benefits.

5.4.1 Benchmarking

With regard to the results of the benchmarking on real-world data with unknown PV mounting configurations and shading conditions to assess the cold-start problem, we discuss two aspects that apply to the modeling error and the forecast error: First, we observe that the cold-start capable AutoPV using nearby plants in the ensemble model pool and the ideal situation of having two years of training data (no cold-start) IM-HDA perform similarly. Thus, we conclude that any PV mounting configuration in the given data set can be represented by the convex linear combination of the ensemble pool models' outputs. This representation also includes the recombination of individual shading conditions from the ensemble pool models. Second, we observe that AutoPV outperforms all quasi-cold-start capable methods (SunDance-AFB, SunDance-AIB, IM-AFB, IM-AIB). This might be explained by the fact that AutoPV's pre-trained ensemble pool models already reflect the entire seasonality, whereas the quasi-cold-start capable methods learn the seasonality with a delay. However, the accuracy of all methods is harmed by dips in the generated PV power since these sudden changes are only captured with a delay by the cyclic adaption.

5.4.2 Interpretability

Regarding the results of assessing the problem of missing information about the PV mounting configuration and the interpretability of AutoPV's ensemble weight optimization, we discuss three aspects: First, the evaluation shows that AutoPV's ensemble weight optimization comes close to the ideal situation of already knowing the actual weights in terms of the forecast error. We suppose that the marginally weaker performance of AutoPV with respect to the ideal situation (0.276 ± 0.009 vs. 0.274 ± 0.011) may be due to the adaptive nature of the ensemble weight optimization. As AutoPV receives more data during operation, the adaption improves, and the weights evolve and stabilize over time, as seen in Figure 5.6. Second, the evaluation shows that AutoPV recombines the outputs of the model pool through weighting rather than selecting the best-performing model from the pool. This convex linear combination allows to represent PV plants with arrays distributed on different roofs with varying angles. More specifically, AutoPV is not limited to represent a single PV mounting configuration. Instead, the mounting configurations of PV arrays distributed on different roofs can be reflected proportionally in the weighting. Third, we notice that AutoPV's ensemble weight optimization estimates a similar PV mounting configuration but often includes more shares than are actually present. The explanation for this is threefold: First, the deviation of the weather forecast from the weather measurement causes noise that makes it challenging to identify the actual weights with the optimization. Second, different mounting configurations might result in similar outputs. Third, the number of weights

greater than zero is not constrained in the optimization. We decide against using such a constraint because for PV plants with limited information, it is unknown whether it is a mixed-oriented plant and how many different mounting configurations this plant includes. However, the fairly competitive performance of AutoPV compared to the ideal situation in the sense of knowing the actual mounting configurations and shares discussed above suggests that AutoPV finds an equivalent configuration. Consequently, we may use AutoPV with the default model pool to estimate an equivalent configuration of the real-world data with unknown PV mounting configurations used for benchmarking (see Figure D.1 in the Appendix). Note that the derived angles given the discretization of the 12 configurations in the pool (tilt $\beta_{T,array} \in \{15^\circ, 45^\circ, 75^\circ\}$, azimuth $\theta_{A,array} \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$) is less accurate than for example with SunDance. Therefore, we consider AutoPV's ensemble weight optimization as a quasi-estimation of the mounting configuration (check mark is in brackets in Table 5.4).

5.4.3 Drift detection

In evaluating the performance of data with synthetic concept drifts, we see that ADWIN is beneficial for detecting drifts early and adapting the weights of AutoPV accordingly. Unlike expected, the difference between monitoring forecasting and modeling errors is not significant. On the one hand, this can be explained by the fact that sudden dips in the PV power generation capabilities are reflected in both forecasting and modeling errors so the drift detection delay is similar. On the other hand, the higher number of false positives in monitoring the modeling error, which results in more frequent weight adaption, does not significantly affect AutoPV's performance. Regarding the drift detection delay of both variants, it can be influenced by adjusting the significance value α_t of ADWIN. However, note that while an increased sensitivity reduces the delay, it also increases the number of false positives, see Section 2.3.3.

5.4.4 Limitations

Regarding the limitations of AutoPV, we discuss implications arising in forecasting at higher levels of aggregation, the creation of the ensemble model pool, and the adaption to drifting PV power generation capabilities.

Forecasting at higher levels of aggregation. While AutoPV is designed to represent locally distributed PV plants, it is not suitable to use it at a greater level of aggregation, e. g., to forecast the total PV power generation of a nation. In this situation, the specific plant

configurations become less important for the actual forecast, as the power generation of all PV plants of the country are aggregated.

Creation of the ensemble model pool. For representing a new PV plant by the convex linear combination of the ensemble pool models' outputs, they must be diverse with respect to the PV mounting configurations and shading conditions. To create such a diverse ensemble model pool, we can use PK or select a diverse set of curves after the P_{\max} -scaling (see Figure 5.1a). However, nearby plants might not be available to create such a diverse ensemble pool. This limitation can be overcome by two approaches: Initially, the default model pool of AutoPV can be used if no PV power generation data from nearby plants are available. As shown in the benchmarking, this default model pool can already achieve a good performance. However, the default model pool does not consider shading. In contrast, the model pool using nearby plants can consider shading. Thus, future work could aim to combine both the default and the nearby plant model pool.

Drifting PV power generation capabilities. The evaluation with synthetic drifts shows that in contrast to sudden drifts, incremental drifts with low degradation rates are difficult to be detected.¹⁵ Further, the proposed adaption based on incremental ensemble optimization (5.17) lags if the drift continues after detection, as shown in Figure 5.8f. Addressing this limitation, therefore, requires continuous adaption. Another limitation regarding drifting PV power generation capabilities concerns AutoPV's adaption to an expansion of the PV plant in terms of the peak power rating by adding new arrays. This limitation can be overcome by relaxing the restriction of the efficiency factor $\hat{w}_e \in [0, 1]$ and allowing values greater than one. However, we decided against it because we consider the peak power rating as a parameter that can be reliably determined and does not require automated adaption. Therefore, we suggest that if the forecast error remains high after an adaption, the power system control room is alerted. Underestimating the actual power generation indicates that the PV plant's peak power rating has increased and requires the data scientist to manually adjust the peak power rating.

5.4.5 Benefits

The automated design and operation, together with the cold-start capability of AutoPV are essential for real-world applications to keep pace with the expansion of PV power generation capacity in future energy systems. Also, the ensemble approach is universally applicable and achieves promising results with only ten representative models in the ensemble pool.

¹⁵ A similar observation is made in the related field of anomaly detection [89].

Since scaling AutoPV to several hundred PV plants still requires only the 10 ensemble pool models to be stored in the computing memory, and furthermore, the weight adaption effort is very low, AutoPV is advantageous for smart grid environments with limited computing capabilities. Moreover, the universality of our approach allows us to use arbitrary complex models in the ensemble pool. On the one hand, this allows to reduce the model complexity if lower accuracy is sufficient, e. g., by using a ridge regression estimator instead of the automated estimator design via CASH. On the other hand, the ensemble approach of AutoPV allows to integrate more complex PV models of future research in the ensemble pool. Another beneficial aspect concerns the interpretability of the ensemble weights. After the adaptive optimization of the ensemble weights has converged, the mounting configuration of the PV plant can be estimated. Further, the adaptive weight optimization can be observed for fault detection. That is, the unconstrained sum of weights decreases in case of a fault, and the unconstrained sum of weights increases if new PV arrays are added.

5.5 Related work

In this section, we evaluate related work regarding the three challenges stated in the introduction of this chapter, which are missing information about the PV mounting configuration, missing historical data to train a forecasting model, and drifting PV power generation capabilities during operation. The related work first focuses on forecasting methods for smart grids. Then, we focus on PV -related literature and analyze a representative selection of methods that address at least one of the three challenges stated in the introduction. Finally, we review literature that is more distantly related to AutoPV.

Forecasting methods for smart grids

As described in Section 1.1, a smart grid requires forecasts of supply and demand at various levels of aggregation. Apart from forecasts on a national level, e. g., [324–326], plant-specific forecasts are required to maintain the stability of transmission and distribution grids. Forecasting of renewable energy plants is highly dependent on weather influences; therefore, forecasting of wind speed [327] and solar irradiance [328] is crucial. In the context of PV -related forecasting, the methods thus can be classified into two types:

- The first does not rely on exogenous weather forecasts: Either, the PV forecast is based entirely on past PV power generation values [329, 330]. Or, two models are designed, one for the weather forecast and one to estimate the PV power generation from this endogenous weather inputs [331–333].

- The second relies on exogenous weather forecasts: A PV model is designed to estimate the power generation using exogenous weather inputs [148, 149, 157, 303, 305–308, 334, 335].

In the first type, autoregressive methods based on Statistical Modeling (SM) [332], ML [331], or DL [333] are used to forecast the solar irradiance. Since free and commercial weather forecasting services with high accuracy are available, we consider the second type – PV modeling – to be more relevant for creating locally distributed PV forecasts. The reason for this is that the weather forecast can be obtained from a central service for any location, while the PV model must be tailored to the particular decentral PV plant to fit its configuration.

Photovoltaic modeling

PV modeling approaches in the literature can be classified into PM, ML, and DL, as well as hybrid approaches. The following analysis is guided by Table 5.4.

Physical-inspired PV modeling. PM approaches use physical laws and equations to model the process of transforming solar irradiance into power. In general, PM involves two steps: i) estimating the solar irradiance reaching the PV array, and ii) estimating the power generated by the PV plant from the incident solar irradiance [313]. Consequently, the PV plant’s location and mounting configuration are required for the first step, and the

Table 5.4: Related work on automated PV power generation forecasting with limited information. If no name was given to the method by the respective authors, only the reference is shown. The table is adapted from [168].

Method name	Modeling approach	Automated design	adaption	Cold-start capable	Estimate mounting configuration	Mixed mounting configurations
pplib [305]	PM	✗	✗	✓	✗	✓
solaR [303]	PM	✗	✗	✓	✗	✓
[306]	ML	✓	✗	✓	✗	✗
[334]	ML	✓	✗	✗	✗	✓
[308]	DL	✓	✗	✗	✗	✓
PV-Net [307]	DL	✓	✗	✗	✗	✓
SolarCast [148, 149]	hybrid(PM & ML)	✓	✓	(✓)	✓	✗
SunDance [157, 335]	hybrid(PM & ML)	✓	✓	(✓)	✓	✗
AutoPV-default	PM	✓	✓	✓	(✓)	✓
AutoPV-nearby	ML	✓	✓	✓	(✓)	✓

Physical-inspired Modeling (PM), Machine Learning (ML), Deep Learning (DL)

PV hardware parameters and peak power rating for the second step. To model this two-step process, the Python package *pvlb* [305] and the R package *solar* [303] are well-known open-source libraries that include simple models with few input parameters to more complex models for extensive design. In both packages, the design process is manual, i. e., the data scientist decides on the model complexity and designs the model. With knowledge about the PV arrays' mounting configurations, the modeling of mixed configurations is possible, e. g., a PV plant with 40 % of the arrays west-oriented and 30°-tilted and 60 % of the arrays east-oriented and 45°-tilted. After the manual design, the model is cold-start capable since no historical data is required to train the model.¹⁶

Machine and deep learning based PV modeling. When PM reaches its limits due to missing information about the PV plants' characteristics, approaches based on ML and DL are powerful alternatives [341]. While ML approaches learn from structured data, DL approaches learn from unstructured or raw data using deep Artificial Neural Networks (ANNs). With both, the performance resulting from the PV hardware configuration and environmental influences, such as shading, can be learned implicitly.

To make forecasts for unobserved PV plants, a cold-start capable ML method using meta-learning is proposed by Bottieau et al. [306]. The meta-model is an automatically designed regressor that uses patterns from neighboring monitored PV power generation profiles and information about the new, unobserved PV plant's location, mounting configuration, and peak power rating. With this information, the model is cold-start capable and requires only weather forecasts during operation. Power generation forecasting for PV plants with mixed mounting configuration is not addressed in the paper but could potentially be achieved by linear superposition as with PM. Zhao et al. [334] propose another ML method that also uses automated design. The authors apply a Genetic Algorithm (GA) for automated feature and model selection, which aims to find the optimal selection for each PV plant, taking into account individual regional differences. The method allows to implicitly represent mixed mounting configurations since an individual model is designed for each PV plant using its historical data. As a result, the method is not cold-start capable, however.

A DL method for day-ahead PV forecasting is proposed by Aslam et al. [308] that combines recurrent layers and an attention mechanism. The DL method PV-Net [307] also uses recurrent layers but combines them with convolutional layers. Both deep ANNs use HPO, i. e., their design process is automated. Abdel-Basset et al. [307] evaluate the accuracy of PV-Net depending on the amount of training data in half-year steps from 0.5 to 5 years.

¹⁶ Such models are also used in software tools for planning PV plants to estimate the power generation potential, e. g., *PV*Sol* [336] used in [337], or *TOP-Energy* [338] used in [339]. As model inputs, time series of a Test Reference Year (TRY) are used to represent an average, but typical weather pattern over the year [340].

Compared to 0.5 years, the accuracy increases after one year by 20 % and increases further up to 33 % after 4 years. Hence, DL methods require a large amount of historical data to leverage their advantages in learning hidden relationships from raw data and thus are not cold-start capable. However, since the model is trained from raw data, no information about the PV plant's hardware configuration or mounting configurations are required for initialization. Learning from raw data also allows representing mixed-oriented PV plants. In all four methods, adaption during operation is not foreseen, and the PV plant's mounting configuration is not estimated.

Hybrid PV modeling. Hybrid approaches combine PM, ML, or DL approaches, aiming to combine their advantages [156]. The hybrid method SolarCast [148, 149] enhances a least-squares regression with a deep ANN. In the initialization phase, the least-squares regression uses the PV plant's location and historical power generation data to estimate the PV mounting configuration and the temperature coefficient. In the operational phase, the deep ANN is continuously adapted to reduce the error by learning dynamic PV plant-specific characteristics like shading and soiling. Only a few days of historical power generation data are required; therefore, the method is quasi-cold-start capable. Chen and Irwin [157] propose a similar approach called SunDance, which combines PM and ML modeling. Like SolarCast, SunDance uses the PV plant's location and historical power generation to estimate the PV mounting configuration and the temperature coefficient in the initialization phase. Since initialization requires at least two data points for power generation, the method is quasi-cold-start capable. To learn PV plant-specific characteristics like shading by nearby buildings and trees, a SVR model corrects the output of SunDance [335]. However, training a correction model that achieves high accuracy requires up to one year of data [323], which hampers the cold-start capability. SolarCast and SunDance both estimate a single PV mounting configuration and are thus not suitable for modeling mixed-oriented PV plants. Since SolarCast and SunDance follow a similar approach and the authors of SunDance provide an official Python implementation [323], we consider SunDance for benchmarking.

Projection of information from reference PV plants

Besides forecasting methods, also other methods exist that project information from nearby PV plants to a new one. For example, Iyengar et al. [147] propose SolarClique to detect anomalies in these data by training a meta-model to learn the relationship between the PV plant to be monitored and other nearby PV plants. Potentially, this approach is adaptable to forecasting if the relationship between forecasts instead of actual power generation measurements is learned. Together with adaptive learning, SolarClique would be (quasi-)cold-start capable and could potentially represent mixed-oriented plants. Another method proposed

by Killinger et al. [342] aims to nowcast the power generation of a region by using reference PV plants. The method uses PM to realize the reference plants. The projection of their power generation to other PV plants is then performed in two steps: first, the reference plant model is inverted to estimate the GHI, which is used in the second step to predict the power generation of the target plant using the target plant's mounting configuration. Thus, the mounting configuration of all PV plants must be known.

Since both methods were not originally proposed for PV forecasting, and require extensive modifications to solve the two problems stated in the introduction, we do not consider them in the benchmarking.

Ensemble-based forecasting

As AutoPV is based on forecast ensembling, we further review ensemble methods for renewable energy forecasting. An ensemble-based method for Wind Power (WP) forecasting is proposed by Hao and Tian [343]. The method consists of two stages: in the first stage, the time series is decomposed into sub-series using variational mode decomposition. For each sub-series, a Multi-Objective Grey Wolf Optimizer (MOGWO)-Extreme Learning Machine (ELM) model is trained, resulting in the initial forecast series. Afterward, they train MOGWO-ELMs to forecast the error sequence between the raw data and the initial forecast series. Finally in the second stage, an MOGWO-ELM meta-model is trained to ensemble the initial forecast and the error forecast series into the final forecast. A related method for PV forecasting is proposed by Abdellatif et al. [344]. The authors train a RF and two different GBMs to create the ensemble model pool. Afterward, they train an Extra Trees Regressor (ETR) meta-model to ensemble the three forecasts into the final forecast. Both methods aim to reduce the forecast error while increasing the robustness. However, unlike AutoPV, the ensemble is not adapted over time. Furthermore, all ensemble pool models represent the target plant instead of reference plants as in AutoPV, which enables AutoPV to solve the cold-start problem.

Automated machine learning

Apart from the analyzed domain-specific methods above, generic frameworks for automatically designing ML models such as TPOT [345] and auto-sklearn [346] exist. However, they are not adapted to PV modeling and thus cannot leverage PK. Further, the automated design process requires historical data of the target, making these generic frameworks not cold-start applicable to new PV plants.

5.6 Novelty and remaining questions

Given the shortcomings of related work identified in the previous section, the novelty of AutoPV can be summarized as follows:

1. AutoPV does not require information about the PV plant's mounting configuration, which are the tilt and azimuth angles $\beta_{T,array}$ and $\theta_{A,array}$, and can represent mixed-oriented PV plants.
2. AutoPV is a cold-start capable method for PV power generation forecasting, i. e., the method does not require historical data of the target PV plant for model training.
3. AutoPV can adapt to changing PV power generation capabilities during operation, e. g., due to PV array soiling, age-related performance degradation or maintenance.

The automation of both design and operation and the cold-start capability make AutoPV an Automation level 4 method under the AutoLVL taxonomy [162, 163] of Chapter 2. Although designed as a point forecasting method, quantiles can be estimated using the normalizing flow-based cINN (Chapter 3), as exemplified in Section D.2 in the Appendix. Regarding the open questions of this thesis stated in Section 1.5, AutoPV considers all five sections of the forecasting pipeline and interconnects automation methods across the pipeline sections. Apart from the publications [168, 169], AutoPV is available as open-source software, as linked in Figure 5.10. However, given the limitations of AutoPV discussed in Section 5.4.4, new questions may be raised:

- Which modifications are required to make AutoPV suitable at greater level of aggregation, e. g., to forecast the total PV power generation of a country?
- How to guarantee that the nearby plants ensemble model pool is diverse enough without using PK?
- Does combining the default and the nearby plants ensemble model pool resolve the previous question, and what rules are necessary to ensure ensemble model pool diversity?
- Can the CASH in creating models for the nearby plants model pool be replaced by a robust hyperparameter configuration that performs well over different PV plants?
- How to detect incremental drifts of the PV power generation capabilities with low degradation rates during operation?
- Does continuous adaption instead of cyclic and drift detection-based adaption resolve the previous question?

- Which modifications are required to make AutoPV adaptable to the expansion of PV plants in terms of the peak power rating by adding new arrays?
- Does relaxing the restriction of the efficiency factor $\hat{w}_e \in [0, 1]$ and allowing values greater than one resolve the previous question, and does it still ensure interpretability?

These stated questions are to be tackled in future work to improve AutoPV.

AutoPV

Template for automated photovoltaic forecasts



Publications

S. Meisenbacher, B. Heidrich, T. Martin, R. Mikut, and V. Hagenmeyer, “AutoPV: Automated photovoltaic forecasts with limited information using an ensemble of pretrained models,” in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy ’23, Orlando, USA: Association for Computing Machinery, 2023, pp. 386–414.

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, pp. 1–8.

Implementation

<https://github.com/SMEISEN/AutoPV>

Figure 5.10: The contributions of this chapter with the Automation level 4 template AutoPV.¹⁷

¹⁷ The icon of the PV plant is adapted from [14].

6 Application

This chapter highlights the application of automated forecasting templates in real-world environments. AutoPV [168, 169] is applied in the forecasting services of the Energy Lab [347, 348] (Section 6.1) and Smart East¹ (Section 6.2). AutoPV is also applied together with AutoWP [166, 167] at the Stadtwerke Karlsruhe Netzservice GmbH (SWKN) (Section 6.3).²

6.1 Forecasting for smart grid applications in the Energy Lab at KIT

The Energy Lab [347, 348] is the real-world testing environment for future energy systems of the Karlsruhe Institute of Technology (KIT). A current research area is the intelligent interconnection of different energy generation, storage, and supply options, also known as sector coupling. For realizing sector coupling, a plant network is available that links electrical, thermal, and chemical energy flows, as well as new information and communication technologies.

Necessity of forecasts. Many applications in sector coupling involve proactive energy distribution, e. g., to counteract grid congestion before they actually occur. To this end, many decentralized optimization problems are solved, which in turn rely on forecasts [12, 13]. Hence, scalable web services are required to systematically orchestrate the data flow between forecasting models and optimization solvers. Deploying such locally-adapted forecasting models to serve so-called downstream applications in a scalable manner calls for a high level of automation.

Communication infrastructure. To establish a forecasting service in the Energy Lab, the automation level taxonomy AutoLVL [162, 163] is used as a guideline to structure the requirements for achieving Automation level 4 and derive a generic service architecture, see Section 2.5. This forecasting service is divided into sub-services where communication in

¹ <https://smart-east-ka.de/>

² Note that this chapter is based on [163, 349, 350] and contain identical phrases; a detailed list is given in the Appendix.

between is established via a cluster network and data is exchanged via a central database.³ Since this forecasting service is valid for all time series forecasting tasks, it can handle different models, e. g., PhotoVoltaic (PV) and electrical load forecasting.

Real-world application. The Automation level 4 forecasting service is deployed in the cloud services ecosystem of the Energy Lab via Kubernetes.⁴ Specifically, the underlying sub-services (Figure 2.19) are deployed as individual pods for scalable resource allocation. Communication between pods is established via a cluster network, see Figure 6.1. Only the management service handles the interaction with the other services, e. g., optimization, visualization, and data collection services. The management service uses an ingress and a REST-API based on FastAPI, where the data flow via REST is based on JSON-defined by Pydantic models.

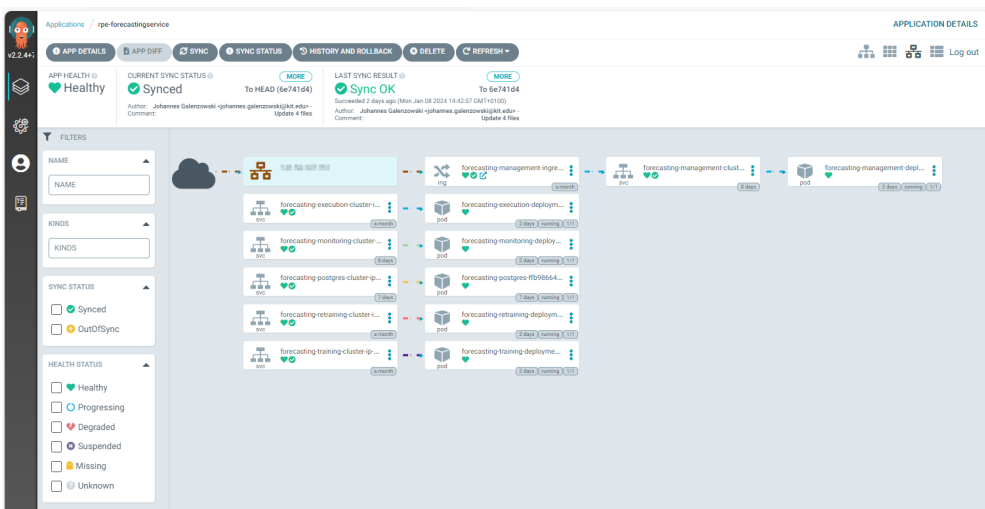


Figure 6.1: The forecasting service consists of sub-services deployed as individual pods where communication between pods is established via a cluster network. Only the management service interacts with the other services of the Energy Lab [347, 348] like optimization, visualization, and data collection services.

Two models are implemented in the forecasting service as part of the present work: AutoPV [168, 169] (Chapter 5) for PV forecasts, and the Profile Neural Network (PNN) [42] for electrical load forecasts. With these two models, the forecasting service currently supplies a visualization service in the Energy Lab with PV and electrical load forecasts. In the future, already running smart grid applications like the Smart Charging Wizard for

³ Note that the communication infrastructure is the work of Johannes Galenzowski [163].

⁴ Note that the deployment is the work of Johannes Galenzowski [163].

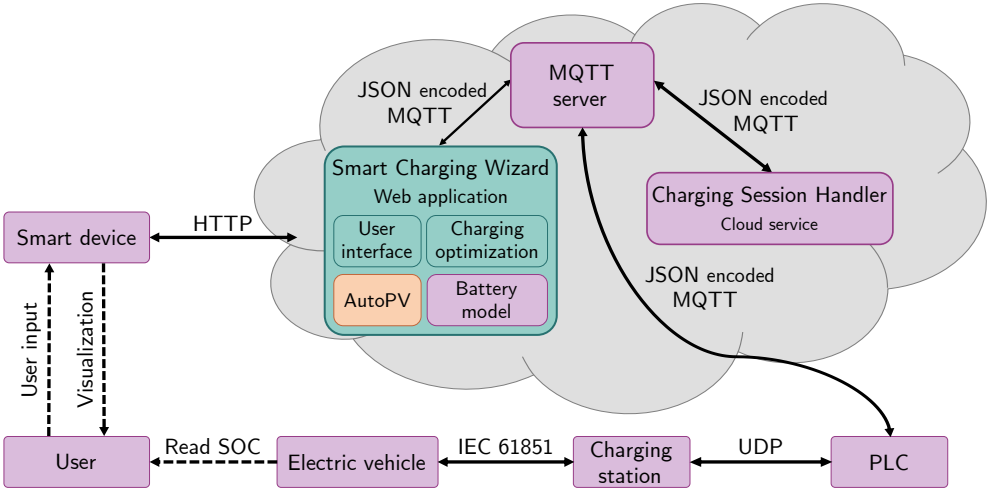


Figure 6.2: An overview of the communication architecture of the Smart Charging Wizard in future work, taking into account the EV’s battery behavior and PV forecasts. New elements are orange (AutoPV), while elements to be modified are green (user interface and charging optimization) and unchanged elements are purple. The illustration is adapted from [350].

(battery) Electric Vehicles (EVs) [349, 350] could integrate forecasts provided by the service, see Figure 6.2. Specifically, the charging optimization that minimizes energy and battery aging costs [351] could integrate PV forecasts to store surplus PV power as simulated in [337]. Such an integration of prosumers into the energy system to provide flexibility is also known as dispatchable feeders [161].

6.2 Forecasting of photovoltaic generation for demand side management in the Smart East environment

Since almost 70 % of the global population is expected to live in cities and towns by 2050 [352], the concentration of energy demand in urban districts offers a particularly high flexibility potential. The Smart East⁵ pilot project aims to explore such flexibility potentials by transforming a mixed residential and commercial area (124 200 m²) into an energy-optimized smart neighborhood. The district can be divided into subdivisions that each have an individual connection to the public electricity grid, the so-called Grid Connection Point (GCP) [353]. All six GCPs with their corresponding PV installations and the EV

⁵ <https://smart-east-ka.de/>

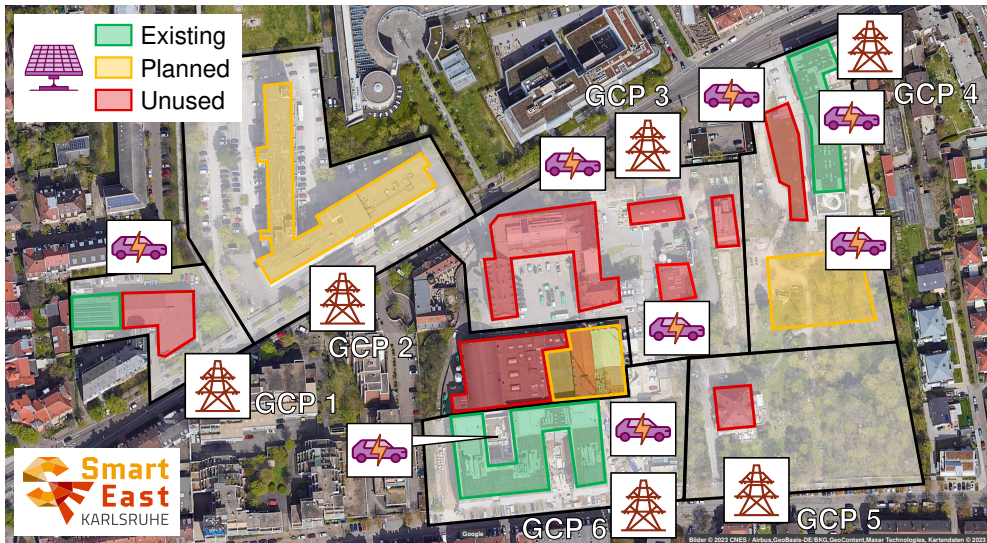


Figure 6.3: Overview of the Smart East district with six GCPs, PV installations and EV charging locations. The smart charging algorithm is currently applied to two GCPs (4 and 6), with the available PV generation capacity expanding rapidly (see planned areas). The figure is based on [163].

charging locations are displayed in Figure 6.3. To this end, the project’s broader aim is to supply the district with low-cost and low-carbon energy.

Necessity of electrical load and PV forecasts. Currently, the flexibility potential of EVs is studied using the smart charging algorithm of InnoCharge GmbH⁶ that performs load shifting, exploiting the intra-day variations of electricity prices during parking (time arbitrage). Additionally, the smart charging algorithm shifts the charging processes such that the electricity drawn from the public grid is minimized and the consumption of locally generated PV electricity is increased. For this purpose, day-ahead forecasts are required for the smart charging algorithm (downstream application), i. e., the PV power generation and the inflexible residual power consumption with a quarter-hourly resolution are required for each PV plant and each GCP.

Communication infrastructure. The PV plants and the EV charging stations connected to the GCPs are part of a District Community Platform (DCP) that incorporates different cloud services for data collection, data stream orchestration, and the district’s energy man-

⁶ <https://www.innocharge.de/>

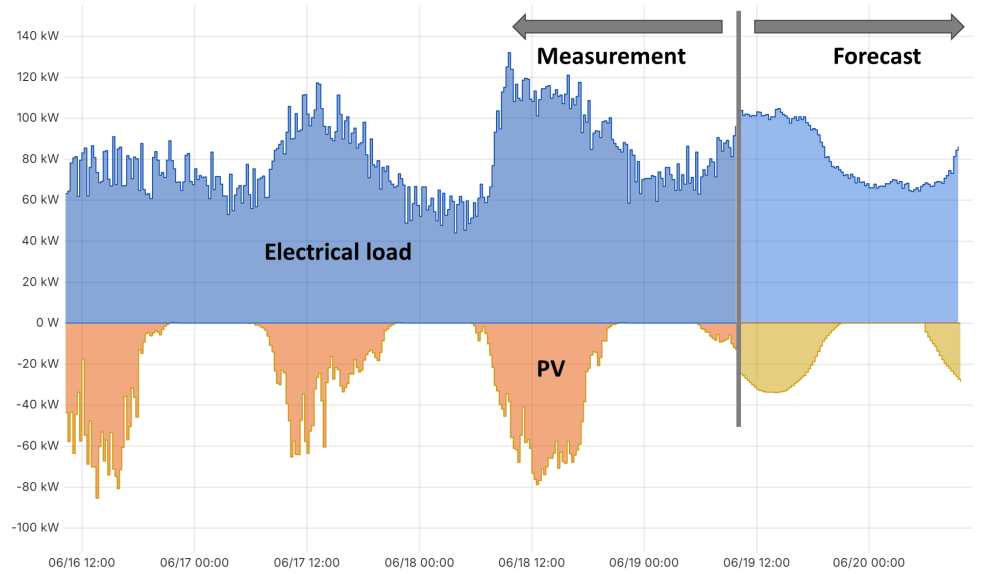


Figure 6.4: Measurement and forecasts of electrical load (positive) and PV generation (negative) for a real-world GCP combined as profiles. It is visible that the installed PV plant does not cover the district's entire electricity demand on the three depicted winter days. However, the PV plant supplies electricity at times of high consumption, reducing the overall peak load [163].

agement [353].⁷ The generic forecasting service introduced in Section 2.5 provides PV and load forecasts for the downstream optimization and control service. To allow scalable resource allocation, the sub-services are deployed as individual pods that communicate via a cluster network with each other and a central database.⁸

Real-world application. To realize the smart charging application, the database is updated in near real-time to provide seamless profiles for both electrical load (positive) and PV power generation (negative) for the past as well as the next 24 h as exemplified in Figure 6.4.

Regarding the PV forecasts, only limited measurement data is available for the connected PV plants, as the measurement data infrastructure and the data collection service was set up during the project period. Furthermore, the PV expansion in the district is rapidly progressing. While only a small 15 kWp PV plant existed in 2021, two PV plants with a total of 219 kWp have been added since then, and further three PV plants with a total of 410 kWp are currently being planned, see Figure 6.3. Another challenging aspect is that most

⁷ In this case, the DCP is a proprietary software platform, but it could also be integrated into open-source solutions such as BEMCom [354].

⁸ Note that the communication infrastructure is the work of Johannes Galenzowski [163].



Figure 6.5: The 119 kWp PV plant with mixed east-west orientation at GCP no. 6.

PV plants have a mixed east-west orientation of PV arrays, see Figure 6.5. These challenges represent a research gap that was closed by AutoPV [168, 169], see Chapter 5.

In terms of the electrical load forecasts, the GCPs are diverse in terms of load profiles and power peaks. Additionally, the electrical loads are subject to concept drifts. For example, during the Covid-19 pandemic, the consumption of some large office buildings varied significantly (as also experienced in [175]). Both challenges represent a research gap that was addressed by the PNN of Heidrich et al. [42, 107].

Two models are implemented in the forecasting service as part of the present work: AutoPV [168, 169] (Chapter 5) for PV forecasts, and the PNN [42] for electrical load forecasts. With these two models, forecasts are provided for the smart charging application in the DCP. Currently, the DCP considers 24 charging points in the Smart East pilot project, for which schedules are calculated and executed. In the future, at least twice as many charging stations will be integrated. Additionally, the charging optimization business model has already been successfully transferred to another district, the RaumFabrik Durlach⁹, comprising 50 charging points.

⁹ <https://www.raumfabrik-durlach.de/>

6.3 Forecasting of photovoltaic and wind power generation for Redispatch 2.0

The balance of generation and demand determined on the electricity market is called dispatch. If the dispatch would lead to grid congestion, transmission system operators, who are responsible for grid stability, must take action using redispatch. While today's redispatch interventions change the generation patterns as visualized in Figure 6.6, future interventions may also change the load patterns by sector coupling [355]. Importantly, redispatch resolves potential line overloads before they actually occur and is planned based on forecasts and load flow calculations.

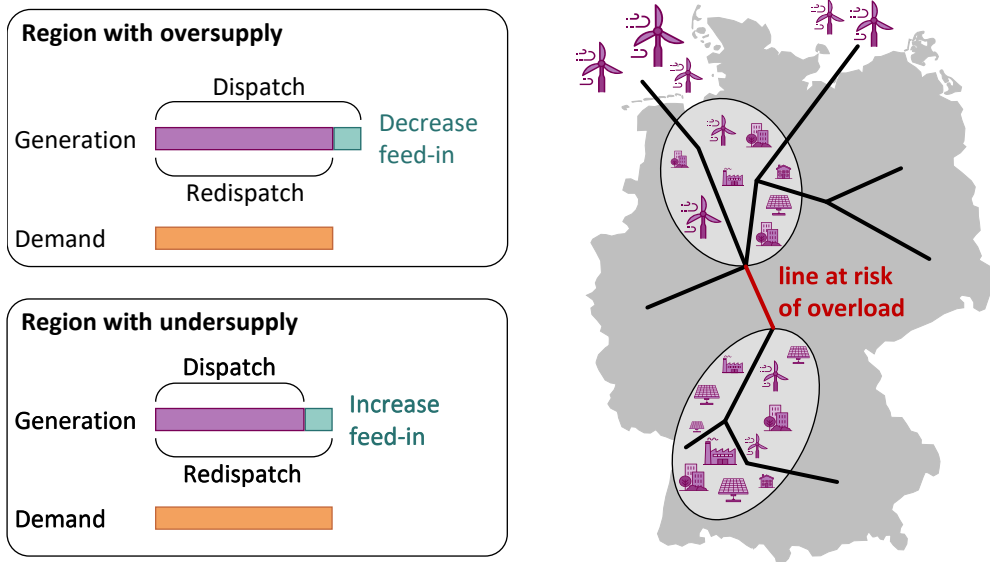


Figure 6.6: Illustration of dispatch and redispatch for a congested line between two regions; simplified, without claiming to correctly represent the power grid in Germany.¹⁰

Necessity of renewable energy forecasts. In the past, conventional power plants were considered for redispatch interventions to avoid line overloads in electrical transmission grids [356]. That is, power plants before the overloaded line are disconnected or instructed to reduce their generation, while power plants after the overloaded line are connected or instructed to increase their generation. In this way, the load on the line is reduced while the total amount of energy remains unchanged. However, with the expansion of renewable

¹⁰ The icons are adapted from [14].

energies, they are also increasingly pushing conventional power plants out of the market due to the low marginal costs of renewable energies (so-called *Merit-order effect*) [357]. This trend accelerates under high commodity prices for coal and gas. To ensure future grid stability under a decreasing share of conventional power plants, renewable energies must also be considered. In Germany, the latest regulatory framework *Redispatch 2.0* takes into account renewable energy plants with a peak power rating of more than 100 kWp. To communicate the availability for redispatch interventions [265], each considered renewable energy power plant must provide a day-ahead forecast with a quarter-hourly resolution to comply with established congestion management processes.

Communication infrastructure. Planning redispatch interventions requires close information exchange between transmission and distribution grid operators. In the state Baden-Württemberg (Germany), redispatch interventions are coordinated across all grid levels with the digital information platform Datenaustausch/Redispatch (DA/RE) [358]. Specifically, distribution grid operators register power plants and transmit forecasts of their power generation. If the load flow calculation predicts a line overload, the most cost-efficient redispatch intervention should be selected, whereby conventional plants are instructed to shutdown first before the renewable energy plants have to reduce their feed-in (merit order with preference for renewable energies).

Real-world application. With the entry into force of *Redispatch 2.0*, the distribution grid operator Stadtwerke Karlsruhe Netzservice GmbH (SWKN) must provide day-ahead power generation forecasts for 65 PV plants and two Wind Power (WP) plants.

Regarding the PV forecasts, measurement data is missing for most PV plants. In addition, further plants are planned that must be taken into account as part of *Redispatch 2.0*, for which no data will be available when they are connected to the grid. Moreover, the actual PV mounting configuration (tilt and azimuth angles) for most PV plants is only roughly approximated or incompletely documented. As detailed in Chapter 5, these challenges represent a research gap that was closed by AutoPV [168, 169].

In terms of the WP forecast, measurement data is available. However, the two WP turbines are often manually shut down, which makes the use of autoregressive methods difficult, see Section 4.2. While the manufacturer and the turbine type is known, the effective hub height is only roughly approximated. These challenges are identified as research gap that was closed by AutoWP [166, 167], see Chapter 4.

Currently, AutoPV and AutoWP are applied by the SWKN using an Excel spreadsheet.¹¹ More specifically, AutoPV-nearby is implemented without weight adaption (averaging),

¹¹ Note that the Excel implementation is the work of Tim Martin [159].

and AutoWP is implemented using the WP curve provided by the Original Equipment Manufacturer (OEM) with the wind speed corrected to the effective hub height. The reason for the static implementation without cyclic or drift-detection-triggered adaption is that the necessary data management infrastructure is not in operation yet. If such an automated model adaption is implemented, redispatch interventions must be documented and taken into account in the process. That is, corresponding periods can simply be dropped from the data batch for adaption because AutoPV and AutoWP are not autoregressive.

7 Conclusion and outlook

The accelerating climate change has forced a paradigm shift in energy systems and requires a transition to a decentralized system with a high share of renewable energy generation. However, this transition to intermittent energy sources such as Wind Power (WP) and PhotoVoltaic (PV) poses electricity storage challenges that require adding flexibility to the power grid by coupling energy-consuming sectors with the power-generating sector. Realizing sector coupling involves solving many decentralized optimization problems that, in turn, rely on forecasts of local electricity demand and supply. As a result, the demand for such locally tailored forecasting models is increasing rapidly, requiring the models' time-consuming design and application processes to be made more efficient by automation.

The research contribution of this thesis supports data scientists when automating forecasting models. To achieve this, fundamentals in times series forecasting are introduced, and existing design automation methods are reviewed (Chapter 1). To fuse such an automated model design with automated model operation, the novel automation level taxonomy for time series forecasting called AutoLVL is defined. These automation levels describe different scopes of automation and act as guideline for selecting automation methods in this thesis (Chapter 2). To automatically tailor models to specific smart grid applications and enable scalable use in the decentralized energy system, three forecasting templates are proposed: AutoPQ, AutoWP, and AutoPV. AutoPQ is an Automation level 2 template for automated point forecast-based quantile forecasts that automates HyperParameter Optimization (HPO) and forecasting method selection (Chapter 3). AutoWP is an Automation level 3 template for automated WP forecasts that automates the model design process and enables monitoring WP turbine operation (Chapter 4). AutoPV is an Automation level 4 template for automated PV forecasts that automates the model design and model operation, providing cold-start capable forecasts and automated adaption to changing power generation capabilities (Chapter 5). Finally, using these templates in three real-world smart grid applications is demonstrated (Chapter 6). In summary, the key contributions of the present work are as follows:

The review on automated forecasting pipelines [1] systematically categorizes design automation methods regarding the pipeline sections i) pre-processing, ii) feature engineering, iii) HPO, iv) forecasting method selection, and v) forecast ensembling in Section 1.3 and gives a detailed overview of forecasting pipelines in the literature.

Furthermore, the five pipeline sections serve as a framework for automating the design process of forecasting pipelines.

The automation level taxonomy AutoLVL [162, 163] goes beyond automated design and also includes automated operation to consider model monitoring, model adaption, and cold-start capabilities. The automation levels range from Automation level 0, which is manual design and manual operation, to Automation level 5, which is an autonomous system, and contribute to categorizing different scopes of automation. In this way, AutoLVL can be used to classify existing literature and serve as a guideline for developing future forecasting templates and services, as demonstrated in this thesis.

The Automation level 2 template AutoPQ [164, 165] automates the design process to provide high-quality and customized probabilistic forecasts for different tasks such as electrical load, electricity price, mobility, PV, and WP forecasting. The core idea of AutoPQ is to automatically make the appropriate design decisions to improve the probabilistic performance of the preliminary work in [115], i. e., generating probabilistic forecast based on an arbitrary point forecast using a conditional Invertible Neural Network (cINN) [107]. Two variants are available: AutoPQ-default for general-purpose computing systems achieving competitive forecasting performance based on [115, 164], and AutoPQ-advanced [165] requiring High-Performance Computing (HPC) systems to further increase forecasting performance for smart grid applications with high decision costs. The benchmarking shows that AutoPQ-default already achieves competitive performance against direct probabilistic benchmarks and other point forecast-based probabilistic benchmarks, while AutoPQ-advanced achieves an additional improvement of up to 6.8 %.

The Automation level 3 template AutoWP [166, 167] is specifically designed for the scalable deployment to decentralized onshore WP turbines subject to regular and irregular interventions in WP generation capacity. AutoWP is based on an ensemble of different WP curves that represent a diverse set of turbine types. Since physical limitations in WP generation are implicitly considered, only few data samples are required to train the model. The training data is cleaned using rule-based and filter-based anomaly detection to identify regular and irregular WP turbine shutdowns. Additionally, such anomaly detection can be applied in the model operation. To this end, anomaly detection is applied in the operation of state-of-the-art autoregressive Deep Learning (DL) benchmarks, and different shutdown handling methods are evaluated. While no compensation method consistently performs best across the three considered autoregressive DL benchmarks, even the comparison of the best-performing combination does not achieve a performance advantage over AutoWP.

The Automation level 4 template AutoPV [168, 169] is tailored to provide PV forecasts with limited information, which is an urgent issue in the rapid expansion of PV capacity. More specifically, AutoPV addresses the challenge of i) missing meta-information about the PV mounting configuration characterized by tilt and azimuth angles, ii) missing historical data of the target PV plant to train the model, and iii) exogenous impacts on the PV plant's generation capabilities occurring during operation. The underlying idea of AutoPV is to describe the arbitrary mounting configuration of a new, possibly mixed-oriented PV plant by an ensemble of PV models representing a diverse set of mounting configurations. Two variants are available: AutoPV-nearby for situations where data of nearby PV plants is available and AutoPV-default if not. The benchmarking shows that both AutoPV variants perform competitively in a cold-start situation, with AutoPV-nearby achieving similar performance as having two years of training data available to train an individual model. It was also shown that the ensemble weights can be interpreted to infer the mounting configuration of the target PV plant. Furthermore, drift detection allows rapid adaption to degrading power generation capabilities during operation, e. g., due to age-related degradation and sudden failures.

The application in real-world smart grid environments demonstrates the utility of the developed automated forecasting templates for downstream applications. To this end, the automation level taxonomy AutoLVL is used as a guideline to identify the requirements of downstream applications and to design suitable implementations. As a result, AutoWP and AutoPV are applied at the Stadtwerke Karlsruhe Netzservice GmbH (SWKN) to communicate the availability of 65 PV plants and two WP turbines for redispatch interventions under the German regulatory framework *Redispatch 2.0*. AutoPV is also applied in the forecasting service of the Energy Lab [347, 348], the real-world testing environment for future energy systems of the Karlsruhe Institute of Technology (KIT), and in the District Community Platform (DCP) of Smart East [353] to serve a downstream smart charging application.

Besides various specific insights on benefits and limitations, as well as a comparison against related work of each contribution that are discussed throughout the corresponding chapters, three future research directions are summarized in the following.

The first research direction concerns increasing the automation level of AutoPQ and AutoWP. For AutoPQ, one could adapt the hyperparameter for sampling in the cINN's latent space during operation to quantify increasing or decreasing uncertainty of the forecast during operation, e. g., caused by concept drifts. Such a cyclic adaption could also be combined with drift detection, which triggers retraining of the cINN, the point forecasting model, or both [359]. To this end, it would be necessary to observe whether the underlying

distribution changes, whether the accuracy of the point forecasting model decreases, or both. For AutoWP, one could design a separate classification model that predicts whether the WP turbine is operating or shut down to consider regular shutdowns during operation. Alternatively, time-controlled shutdowns can also be integrated using Prior Knowledge (PK), as realized by SWKN in the context of Redispatch 2.0 with unavailability notifications.

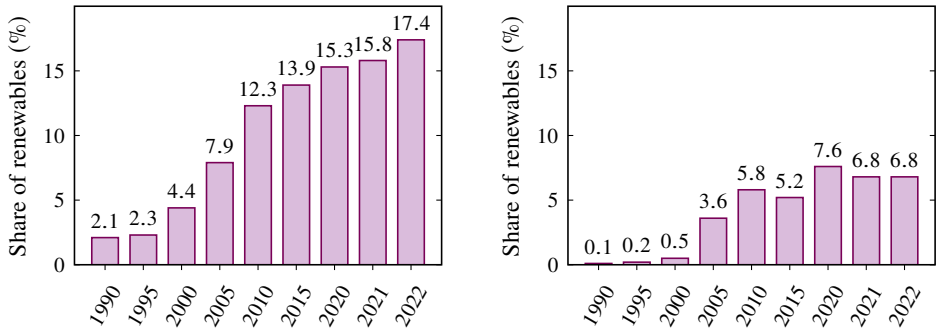
The second research direction considers the development of further automated forecasting templates (AutoXX). For example, a template for modeling the thermal dynamics of buildings could be created to automate the design in [360–362]. Such models are necessary for model predictive heating control in demand side management using buildings’ heat pumps [363]. Another promising example for an automated template is modeling the state of (battery) Electric Vehicles (EVs) [186, 351] and stationary batteries. Integrating such a battery model together with AutoPV into a grid-serving charging scheme like [337] could store surplus PV power and increase the building’s self-sufficiency.

The third research direction is about the use of Generative Pre-trained Transformers (GPTs) or other foundation models to create automated forecasting templates. More specifically, a GPT could be trained on a large data set covering various energy-related time series – such as electrical load, electricity price, mobility, PV, and WP – to achieve cold-start capability on new tasks. Once task-specific data is available, the GPT could be fine-tuned to increase its forecast quality. Initially, cyclic model adaption with an increasing batch for fine-tuning could be applied. Furthermore, it could be interesting to investigate whether a GPT can handle concept drifts implicitly, given the large number of situations it has seen during training.

Appendix

A Introduction

A.1 Supplementary figures



(a) Renewable energy share in heating and cooling.

(b) Renewable energy share in transport.

Figure A.1: Evolution of the renewable energy share in the German energy consumption in the heating and cooling sector and the transport sector [15].

A.2 Further metrics

The RMSE measures the average difference between forecasts \hat{y} and realized values y . Mathematically, it is the standard deviation of the residuals:

$$\text{RMSE} = \sqrt{\frac{1}{K} \sum_{k=1}^K (y[k] - \hat{y}[k])^2}. \quad (\text{A.1})$$

To enable comparing error measures across data sets with different magnitudes, we can normalize the MAE, MSE, and RMSE with the Arithmetic mean of the realized values:

$$\text{nMAE} = \frac{\sum_{k=1}^K |\hat{y}[k] - y[k]|}{\sum_{k=1}^K y[k]}, \quad (\text{A.2})$$

$$\text{nMSE} = \frac{\sum_{k=1}^K (\hat{y}[k] - y[k])^2}{\sum_{k=1}^K y[k]}, \quad (\text{A.3})$$

$$\text{nRMSE} = \frac{\sqrt{\frac{1}{K} \sum_{k=1}^K (\hat{y}[k] - y[k])^2}}{\frac{1}{K} \sum_{k=1}^K y[k]}. \quad (\text{A.4})$$

Normalization with the Arithmetic mean is only valid for strictly positive time series, which we often encounter in energy generation and consumption time series. For time series with negative values (e. g. for prosumers that both draw and feed in energy) normalization with the range between minimum and maximum time series values is suitable.

A.3 Energy time series transformation

For better interpretability, an energy metering time series can be converted into a mean power generation time series

$$\bar{P}[k] = \frac{\Delta E[k]}{t_k}, \quad (\text{A.5})$$

where $\Delta E[k]$ (kWh) is the energy generated in the sample period t_k (h).

B AutoPQ

B.1 Supplementary tables and figures

Table B.1: The configuration spaces Λ_p for the SM-based point forecasting methods in sktime naming convention [364] with the seasonal period $s_p = 24$. The table is based on [165].

Forecasting method	Hyperparameter	Value range	Default value
SARIMAX [66]	p	[0, 5]	1
	d	[0, 2]	0
	q	[0, 5]	0
	P	[0, 2]	0
	D	[0, 1]	0
	Q	[0, 2]	0
ExponentialSmoothing [32]	error	{add, mul}	add
	trend	{add, mul, None}	None
	seasonal	{add, mul, None}	None
	damped_trend	{True, False}	False
TBATS [236]	use_trigonometric_seasonality	{True, False}	True
	use_box_cox	{True, False}	False
	use_arma_errors	{True, False}	False
	use_trend	{True, False}	False
	use_damped_trend	{True, False}	False

Table B.2: The configuration spaces Λ_p for the ML-based point forecasting methods in the respective naming conventions. The table is based on [165].

Forecasting method	Hyperparameter	Value range	Default value
MLPRegressor [100]	activation	{logistic, tanh, relu}	relu
	batch_size	{32, 64, 128}	min(200, n_samples)
	hidden_layer_sizes	{([10, 100]), ([10, 100], [10, 100]), ([10, 100], [10, 100], [10, 100])}	(100)
MSVR [237]	C	[0.01, 100]	1.0
	epsilon	[0.001, 1]	0.1
	kernel	{rbf, laplacian, sigmoid}	rbf
XGBRegressor [238]	learning_rate	[0.01, 1]	0.3
	max_depth	[1, 18]	6
	n_estimators	[10, 300]	100
	sub_sample	[0.5, 1.0]	1.0

Table B.3: The configuration spaces Λ_p for the DL-based (point) forecasting methods in PyTorch Forecasting naming convention [243]. The table is based on [165].

Forecasting method	Hyperparameter	Value range	Default value
DeepAR [48]	batch_size	{32, 64, 128}	64
	cell_type	{LSTM, GRU}	LSTM
	dropout	[0.0, 0.2]	0.1
	hidden_size	[10, 100]	10
	rnn_layers	[1, 3]	2
NHITS [43]	batch_size	{32, 64, 128}	64
	dropout	[0.0, 0.2]	0.0
	hidden_size	[8, 1024]	512
	n_blocks	{[1], [1, 1], [1, 1, 1]}	[1, 1, 1]
	n_layers	[1, 3]	2
	shared_weights	{True, False}	True
TemporalFusion - Transformer [44]	batch_size	{32, 64, 128}	64
	dropout	[0.0, 0.2]	0.1
	hidden_continuous_size	[8, 64]	8
	hidden_size	[16, 256]	16
	lstm_layers	[1, 3]	1

DeepAR may also provide probabilistic forecasts.

Table B.4: The configuration space Λ_q for the cINN [107, 115], which transforms a point forecasting model into a probabilistic one, in the respective naming convention. The table is based on [165].

Hyperparameter	Value range	Default value
sampling_std	[0.01, 3.0]	0.1

Table B.5: The configuration of the Propulate EA [181] for HPO, in the respective naming convention. The table is based on [165].

Module	Hyperparameter	Value
Islands	generations	-1
	num_isles	2
	migration_probability	0.7
	pollination	True
Compose	pop_size	8
	mate_prob	0.7
	mut_prob	0.4
	random_prob	0.2
	sigma_factor	0.05

Table B.6: Overview of the training, validation, and test data sets with hourly resolution used for evaluation, as well as the selected features based on [115]. The names of the target variables and the features refer to the column names in the data sets. The table is based on [165].

Data set	Target variable	Features	Training sub-set	Validation sub-set	Test sub-set
Load-BW	load_power_statistics	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1)	[0,4904] (204.3 d)	[4905,7007] (87.6 d)	[7008,8760] (73.0 d)
Load-GCP	MT_158	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1)	[0,14716] (613.2 d)	[14717,21023] (262.8 d)	[21024,26280] (219.0 d)
Mobility	cnt	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1) temp hum windspeed weathersit	[0,9824] (409.3 d)	[9825,14034] (175.4 d)	[14035,17544] (146.2 d)
Price	Zonal Price	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1) Forecast Total Load Forecast Zonal Load	[0,14541] (605.9 d)	[14542,20773] (259.6 d)	[20774,25968] (216.4 d)
PV	POWER	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1) SSRD TCC	[0,11033] (459.7 d)	[11034,15762] (197.0 d)	[15763,19704] (164.2 d)
WP	TARGETVAR	Cyclic features (2.2), (2.4), (2.6) Lag features (2.1) U100 V100 Speed100	[0,9400] (391.7 d)	[9401,13430] (167.9 d)	[13431,16789] (139.9 d)

cnt: count; SSRD: Surface Short-wave (solar) Radiation Downwards; TCC: Total Cloud Cover; U100, V100, Speed100: wind speeds at a height of 100 meters above ground, with the eastward component (U), the northward component (V), and the total wind speed (Speed).

Table B.7: The CRPS evaluated on the hold-out test data sets with methods categorized into three classes: i) direct probabilistic benchmark methods, ii) point forecast-based probabilistic benchmark methods, and iii) AutoPQ with the default and the advanced configuration. The results of the benchmarks and AutoPQ-default for the data sets Load-GCP, Mobility, Price, and PV originate from [115]. The table is based on [165].

	DeepAR	QRNNs	NNQF	Gaussian PIs	Empirical PIs	Conformal PIs	AutoPQ default	AutoPQ advanced
Load-BW	0.192 ±0.008	0.145 ±0.002	0.208 ±0.004	0.156 ⁽¹⁾ ±0.000	0.143 ⁽¹⁾ ±0.000	0.143 ⁽¹⁾ ±0.000	0.147 ⁽¹⁾ ±0.001	0.138 ⁽¹⁾ ±0.001
Load-GCP	0.312 ±0.009	0.287 ±0.002	0.263 ±0.001	0.299 ⁽¹⁾ ±0.000	0.234 ⁽¹⁾ ±0.000	0.234 ⁽¹⁾ ±0.000	0.234 ⁽¹⁾ ±0.002	0.218 ⁽¹⁾ ±0.001
Mobility	0.299 ±0.007	0.443 ±0.006	0.542 ±0.004	0.360 ⁽²⁾ ±0.021	0.268 ⁽²⁾ ±0.015	0.268 ⁽²⁾ ±0.015	0.263 ⁽²⁾ ±0.004	0.258 ⁽²⁾ ±0.005
Price	0.158 ±0.005	0.157 ±0.002	0.183 ±0.003	0.279 ⁽²⁾ ±0.008	0.161 ⁽²⁾ ±0.005	0.161 ⁽²⁾ ±0.005	0.140 ⁽²⁾ ±0.006	0.131 ⁽³⁾ ±0.001
PV	0.151 ±0.013	0.101 ±0.001	0.119 ±0.000	0.208 ⁽¹⁾ ±0.000	0.125 ⁽¹⁾ ±0.000	0.125 ⁽¹⁾ ±0.000	0.106 ⁽¹⁾ ±0.001	0.101 ⁽¹⁾ ±0.001
WP	0.618 ±0.028	0.374 ±0.002	0.394 ±0.004	0.419 ⁽¹⁾ ±0.000	0.373 ⁽¹⁾ ±0.000	0.373 ⁽¹⁾ ±0.000	0.377 ⁽¹⁾ ±0.001	0.362 ⁽¹⁾ ±0.001
Median	0.245	0.222	0.236	0.289	0.197	0.197	0.191	0.178
Mean	0.288	0.251	0.285	0.287	0.217	0.217	0.211	0.201

best-performing base point forecasting method: (1) XGB, (2) TFT, (3) N-HITS

Table B.8: The p-values of the one-tailed t-test with the null hypothesis that the test CRPS of AutoPQ-advanced and the comparison method is equal and the alternative hypothesis that the test CRPS of AutoPQ-advanced is less. The values are colored green if the p-value is smaller than the significance level 0.05 and colored red otherwise.

	DeepAR	QRNNs	NNQF	Gaussian PIs	Empirical PIs	Conformal PIs	AutoPQ default
Load-BW	0.0002	0.0023	<0.0001	<0.0001 ⁽¹⁾	0.0010 ⁽¹⁾	0.0013 ⁽¹⁾	0.0003 ⁽¹⁾
Load-GCP	<0.0001	<0.0001	<0.0001	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	0.0001 ⁽¹⁾
Mobility	0.0004	<0.0001	<0.0001	0.0004 ⁽²⁾	0.2237 ⁽²⁾	0.2276 ⁽²⁾	0.1620 ⁽²⁾
Price	0.0003	<0.0001	<0.0001	<0.0001 ⁽²⁾	0.0002 ⁽²⁾	0.0002 ⁽²⁾	0.0237 ⁽²⁾
PV	0.0009	0.9413	<0.0001	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	0.0007 ⁽¹⁾
WP	<0.0001	0.0003	<0.0001	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾	<0.0001 ⁽¹⁾

best-performing base point forecasting method: (1) XGB, (2) TFT

Table B.9: The CRPS evaluated on the hold-out test data sets before and after post-processing. In post-processing, negative values in the quantile forecasts are set to zero to maintain given limits (mobility indicator, PV generation, and WP generation are greater than or equal to zero). For the other three data sets (Load-BW, Load-GCP, and Price), either no negative values exist in the quantile forecasts, respectively, negative values are valid. The table is based on [165].

	AutoPQ-default	AutoPQ-advanced	
Mobility	0.2641 ±0.0138	0.2585 ±0.0051	before post-processing
	0.2627 ±0.0037	0.2581 ±0.0047	after post-processing
PV	0.1072 ±0.0003	0.1018 ±0.001	before post-processing
	0.1060 ±0.0009	0.1013 ±0.0007	after post-processing
WP	0.3798 ±0.0018	0.3625 ±0.0017	before post-processing
	0.3775 ±0.0010	0.3621 ±0.0012	after post-processing

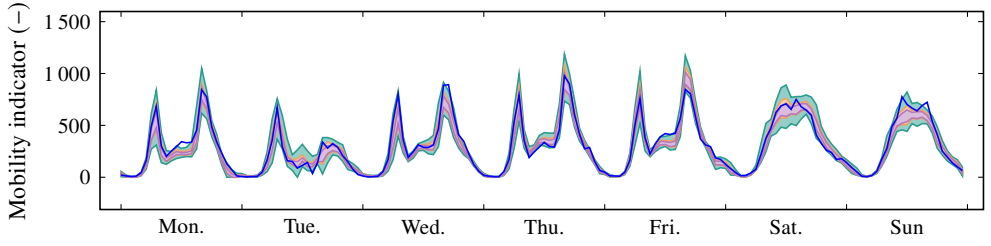
Table B.10: The validation CRPS compared for three trial schedulers (random search, BO-TPE, and BO-TPE-PK) in the inner loop of Algorithm 2. Across the six data sets and the nine forecasting methods, only minor differences occur. The table is based on [165].

	ETS	sARIMAX	TBATS	MLP	SVR	XGB	DeepAR	N-HiTS	TFT	
Load-BW	0.357 ±0.018	0.266 ±0.003	0.259 ±0.004	0.099 ±0.008	0.100 ±0.001	0.079 ±0.001	0.134 ±0.01	0.090 ±0.004	0.107 ±0.006	random search
	0.357 ±0.018	0.266 ±0.003	0.259 ±0.004	0.103 ±0.011	0.100 ±0.001	0.079 ±0.001	0.133 ±0.01	0.090 ±0.004	0.107 ±0.006	BO TPE
	0.357 ±0.018	0.277 ±0.014	0.259 ±0.004	0.100 ±0.008	0.101 ±0.001	0.079 ±0.001	0.133 ±0.01	0.089 ±0.002	0.107 ±0.006	BO TPE PK
Load-GCP	0.576 ±0.024	0.491 ±0.005	0.405 ±0.008	0.219 ±0.002	0.215 ±0.002	0.197 ±0.001	0.247 ±0.007	0.230 ±0.004	0.212 ±0.002	random search
	0.576 ±0.024	0.491 ±0.005	0.405 ±0.008	0.219 ±0.002	0.215 ±0.002	0.197 ±0.001	0.247 ±0.007	0.230 ±0.004	0.212 ±0.002	BO TPE
	0.576 ±0.024	0.491 ±0.005	0.405 ±0.008	0.219 ±0.002	0.215 ±0.002	0.197 ±0.001	0.247 ±0.007	0.230 ±0.004	0.212 ±0.002	BO TPE PK
Mobility	0.630 ±0.007	0.611 ±0.01	0.446 ±0.001	0.377 ±0.014	0.416 ±0.004	0.397 ±0.005	0.280 ±0.01	0.295 ±0.05	0.261 ±0.008	random search
	0.630 ±0.008	0.611 ±0.01	0.446 ±0.002	0.376 ±0.006	0.416 ±0.004	0.397 ±0.005	0.279 ±0.011	0.294 ±0.048	0.261 ±0.008	BO TPE
	0.630 ±0.007	0.611 ±0.01	0.445 ±0.002	0.374 ±0.006	0.416 ±0.005	0.397 ±0.005	0.280 ±0.01	0.288 ±0.036	0.261 ±0.007	BO TPE PK
Price	0.551 ±0.012	0.252 ±0.006	0.285 ±0.003	0.194 ±0.002	0.205 ±0.001	0.206 ±0.003	0.189 ±0.01	0.164 ±0.01	0.194 ±0.008	random search
	0.550 ±0.011	0.252 ±0.006	0.285 ±0.003	0.196 ±0.004	0.205 ±0.001	0.206 ±0.003	0.190 ±0.01	0.162 ±0.007	0.194 ±0.008	BO TPE
	0.550 ±0.012	0.252 ±0.006	0.285 ±0.003	0.194 ±0.005	0.205 ±0.001	0.206 ±0.003	0.191 ±0.003	0.162 ±0.007	0.194 ±0.008	BO TPE PK
PV	0.303 ±0.018	0.696 ±0.196	0.165 ±0.001	0.127 ±0.002	0.114 ±0.001	0.107 ±0.001	0.158 ±0.004	0.161 ±0.025	0.131 ±0.001	random search
	0.303 ±0.018	0.695 ±0.196	0.165 ±0.001	0.127 ±0.002	0.114 ±0	0.107 ±0.001	0.158 ±0.004	0.168 ±0.043	0.131 ±0.001	BO TPE
	0.303 ±0.017	0.695 ±0.196	0.165 ±0.001	0.127 ±0.002	0.114 ±0	0.107 ±0.001	0.158 ±0.005	0.159 ±0.023	0.131 ±0.001	BO TPE PK
WP	0.549 ±0.004	0.497 ±0.008	0.547 ±0.011	0.320 ±0.009	0.303 ±0.003	0.292 ±0.001	0.412 ±0.003	0.389 ±0.011	0.407 ±0.017	random search
	0.549 ±0.004	0.496 ±0.008	0.547 ±0.012	0.321 ±0.013	0.303 ±0.003	0.292 ±0.001	0.412 ±0.003	0.389 ±0.011	0.408 ±0.017	BO TPE
	0.549 ±0.003	0.502 ±0.019	0.545 ±0.013	0.319 ±0.005	0.303 ±0.003	0.292 ±0.001	0.412 ±0.004	0.390 ±0.01	0.407 ±0.017	BO TPE PK

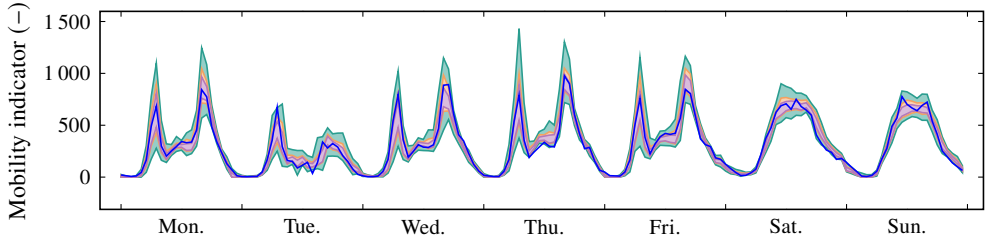
Table B.11: The p-values of the one-tailed t-test with the null hypothesis that the validation CRPS of the optimized configuration λ^* and the default configuration λ_{default} is equal and the alternative hypothesis that the validation CRPS of the optimized configuration is less. The values are colored green if the p-value is smaller than the significance level 0.05 and colored red otherwise.

	ETS	sARIMAX	TBATS	MLP	SVR	XGB	DeepAR	N-HiTS	TFT
Load-BW	<0.0001	0.0002	0.5532	0.0017 ⁽³⁾	0.5244	0.0009 ⁽¹⁾	0.0295	<0.0001 ⁽²⁾	0.0172
Load-GCP	>0.9999	>0.9999	0.0930	0.0016	0.0019 ⁽³⁾	0.0001 ⁽¹⁾	0.0016	0.0045	0.0683 ⁽²⁾
Mobility	<0.0001	<0.0001	0.0007	0.0087	0.0006	0.0002	0.0042 ⁽²⁾	0.0004 ⁽³⁾	0.3970 ⁽¹⁾
Price	0.3692	<0.0001	>0.9999	0.0032 ⁽²⁾	0.0304	0.0223	0.0367	0.0086 ⁽¹⁾	0.0803 ⁽³⁾
PV	0.0007	0.2385	<0.0001	<0.0001 ⁽³⁾	0.0001 ⁽²⁾	0.0003 ⁽¹⁾	0.0243	0.0001	0.0548
WP	>0.9999	>0.9999	>0.9999	0.0007 ⁽³⁾	0.0038 ⁽²⁾	<0.0001 ⁽¹⁾	0.0103	0.0001	0.0533

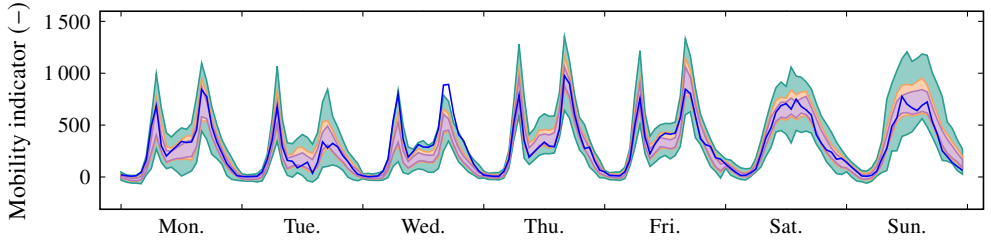
(1) successive halving rank no. 1, (2) successive halving rank no. 2, (3) successive halving rank no. 3



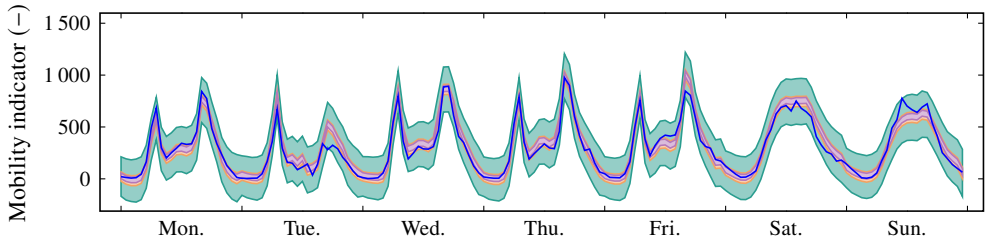
(a) AutoPQ-default.



(b) AutoPQ-advanced.



(c) DeepAR.



(d) Conformal PIs with TFT.



Figure B.1: Exemplary 40 %, 70 %, and 98 % PIs of seven day-ahead forecasts with origin at 00:00 each for the Mobility data set, comparing AutoPQ-default and AutoPQ-advanced with the best-performing direct (DeepAR) and point-based probabilistic benchmark (Conformal PIs with TFT).

B.2 Algorithm for the ablation study

Algorithm 3 HPO algorithm for optimizing the hyperparameters of the point forecasting method λ_p and the sampling hyperparameter λ_q of the cINN [107, 115] simultaneously without consideration of the significantly lower computational effort of the latter. The algorithm is based on [165].

Input: $\Lambda_p, \Lambda_q, B_i, B_t, \mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}, f_{\text{cINN}}(\cdot)$

```

1: trial_scheduler( $\Lambda_p$ ) ▷ Initialize Propulate [181] (EA)
2:  $b_t \leftarrow 0$  s
3: while  $b_t < B_t$  do
4:    $t \leftarrow \text{get\_current\_time}()$ 
5:    $\lambda_p, \lambda_q \leftarrow \text{trial\_scheduler.get\_configuration}(\Lambda_p, \Lambda_q)$ 
6:    $f_p(\cdot) \leftarrow \text{train\_p\_model}(\lambda_p, \mathbf{X}_{\text{train}}, \mathbf{y}_{\text{train}})$  ▷ High computational effort
7:    $f_q(\cdot) \leftarrow \text{generate\_q\_model}(f_p(\cdot), f_{\text{cINN}}(\cdot), \lambda_q)$  ▷ Low computational effort
8:    $Q \leftarrow \text{assess\_prob\_performance}(f_q(\mathbf{X}_{\text{val}}, \mathbf{y}_{\text{val}}))$  ▷ Low computational effort
9:   trial_scheduler.update( $\lambda_p, \lambda_q, Q$ )
10:   $b_t \leftarrow b_t + (\text{get\_current\_time}() - t)$ 
11: end while
12:  $\lambda_p^*, \lambda_q^* \leftarrow \text{trial\_scheduler.get\_best\_configuration}(\Lambda_p, \Lambda_q)$ 

```

Output: λ_p^*, λ_q^*

C AutoWP

C.1 Supplementary figures

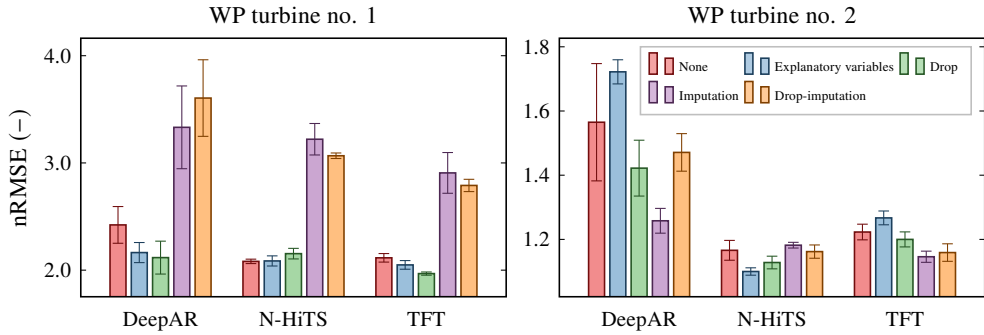


Figure C.1: The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nRMSE when *considering* shutdowns. Legend of WP turbine no. 2 applies also to no. 1. The figure is based on [166].

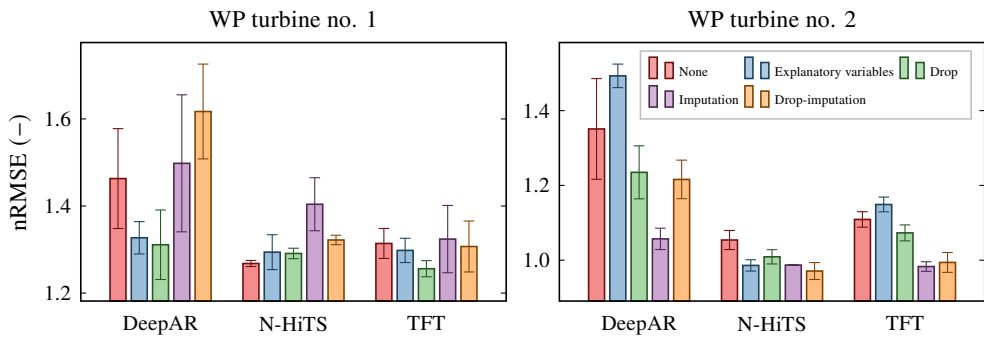


Figure C.2: The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nRMSE when *disregarding* shutdowns. Legend of WP turbine no. 2 applies also to no. 1. The figure is based on [166].

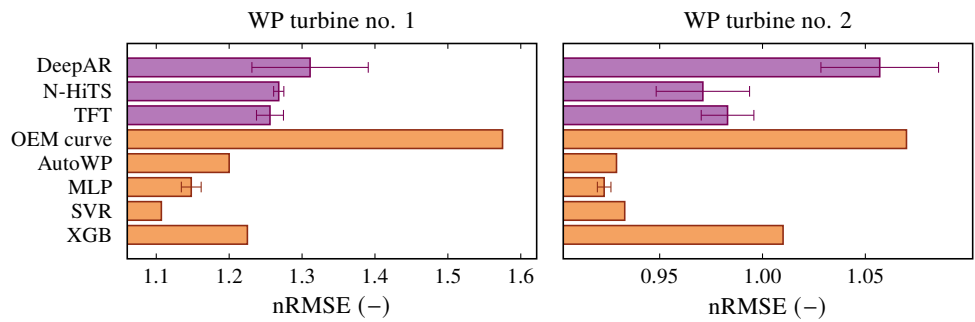


Figure C.3: Comparison of autoregressive DL forecasting methods (purple) to methods based on WP curve modeling (orange) in terms of the test nRMSE when *disregarding* shutdowns. The figure is based on [166].

C.2 Generating probabilistic WP forecasts with AutoWP using normalizing flows

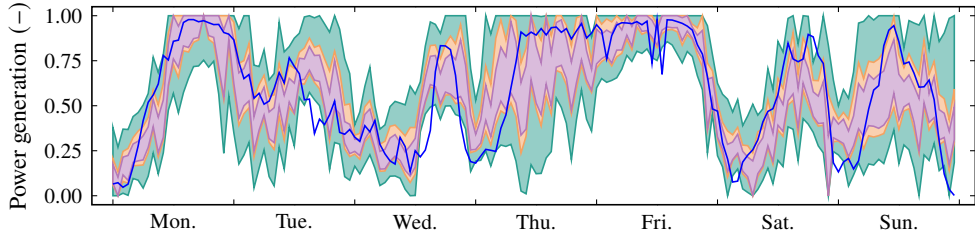
In this additional evaluation, we combine AutoWP [166, 167] (Chapter 4) and AutoPQ [164, 165] (Chapter 3). First, we make a point forecast with AutoWP and then, we generate a probabilistic forecast using the normalizing flow-based cINN [107, 115]. For the evaluation, we use the open-source WP data set of the GEFCOM 2014 [241] with the training, validation and test data sets as defined in Table B.6.

Table C.1 shows the CRPS achieved on the hold-out test data set by AutoWP compared to the nine point forecasting methods considered by AutoPQ. In this ranking, AutoWP shows competitive performance and takes the 2nd place. A visual comparison of the resulting PIs is exemplified in Figure C.4.

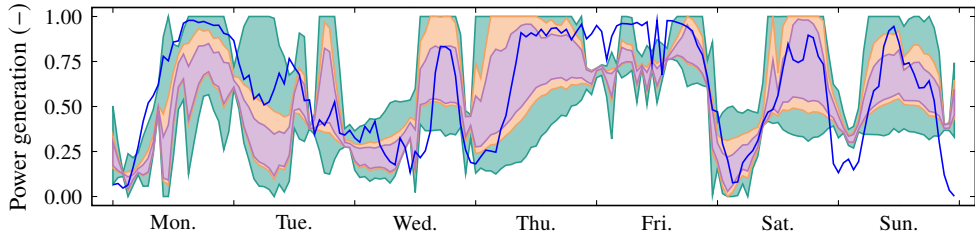
Table C.1: The probabilistic forecast error (CRPS) of AutoWP compared to the forecasting methods considered by AutoPQ evaluated on the test data set of the WP data set [241]. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table B.1, B.2, and B.3).

Forecasting method	CRPS
XGB-cINN	0.377±0.001
AutoWP-cINN	0.436±0.005
SVR-cINN	0.440±0.006
MLP-cINN	0.494±0.011
TFT-cINN	0.546±0.015
N-HITS-cINN	0.556±0.008
sARIMAX-cINN	0.631±0.02
DeepAR-cINN	0.674±0.046
TBATS-cINN	0.692±0.031
ETS-cINN	0.803±0.022

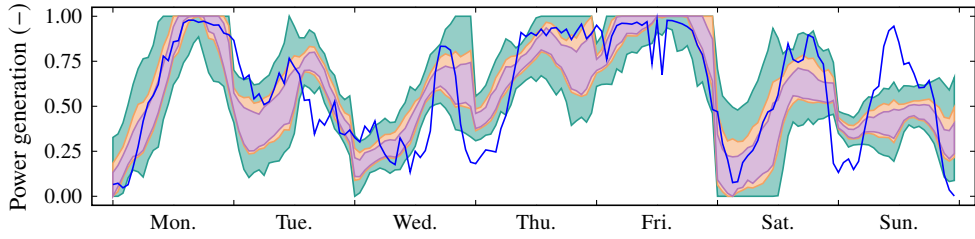
From the results of this additional evaluation, we conclude that a probabilistic forecast can be generated from AutoWP’s point forecast using the normalizing flow-based cINN [107, 115]. In this way, AutoWP achieves a competitive probabilistic performance compared to other state-of-the-art forecasting methods.



(a) XGB-cINN.



(b) AutoWP-cINN.



(c) SVR-cINN.



Figure C.4: Exemplary 40 %, 70 %, and 98 % PIs for the WP data set, comparing AutoWP to XGB and SVR. For all methods, we apply the cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use the default configuration (Table B.2). The day-ahead forecasts originate every day from 00:00.

D AutoPV

D.1 Supplementary tables and figures

Table D.1: The configuration of the PM pipeline in pvlib [305] naming convention. The table is based on [168].

Estimation	Module	Hyperparameter
Solar position	solarposition .get_solarposition	method="nrel_numpy"
DNI	irradiance .disc	pressure=101325.0
Extraterrestrial radiation	irradiance .get_extra_radiation	method="spencer"
Relative air mass	atmosphere .get_relative_airmass	model="kastenyoung1989"
Absolute air mass	atmosphere .get_absolute_airmass	pressure=101325.0
AOI	irradiance .aoi	
POA irradiance	irradiance .get_total_irradiance	model="haydavies"
PV array temperature	temperature .sapm_cell	a=-3.47, b=-0.0594, deltaT=3
PV effective irradiance	pvsystem .sapm_effective_irradiance	module="Canadian_Solar_CS5P_220M___2009_"
PV array DC power	pvsystem .sapm	module="Canadian_Solar_CS5P_220M___2009_"
PV inverter AC power	inverter .sandia	inverter="ABB_MICRO_0_25_I_OUTD_US_208__208V_"

Table D.2: The configuration space Λ for the automated regression estimator design using CASH in Scikit-learn [100] naming convention. The table is based on [168].

Regression estimator	Hyperparameter	Value range
Ridge	alpha	[0.05, 1]
MLPRegressor	activation	{logistic, tanh, relu}
	batch_size	[16, 1024]
	hidden_layer_sizes	{([10, 100]), ([10, 100], [10, 100]), ([10, 100], [10, 100], [10, 100])}
GradientBoosting - Regressor	learning_rate	[0.01, 1]
	max_depth	[1, 10]
	n_estimators	[10, 300]
RandomForest - Regressor	max_depth	[1, 10]
	n_estimators	[10, 300]
SVR	C	[0.01, 10]
	epsilon	[0.001, 1]

Table D.3: The selected (estimated) optimal configuration $\hat{\lambda}^*$, i. e., the optimal regression algorithm and corresponding optimal hyperparameters, in the 10 runs of the CASH on real-world data. The selected configuration is used for creating AutoPV’s nearby-plants model pool (leave-one-out) and the IM. The table is based on [168].

	Run no. 1	Run no. 2	Run no. 3	Run no. 4	Run no. 5	Run no. 6	Run no. 7	Run no. 8	Run no. 9	Run no. 10
Plant no. 1	MPRegressor(batch_size=32, hidden_layer_sizes=(20, 10), activation="relu")	MPRegressor(batch_size=260, hidden_layer_sizes=(10, 10), activation="relu")	MPRegressor(batch_size=125, hidden_layer_sizes=(10, 10), activation="relu")	MPRegressor(batch_size=215, hidden_layer_sizes=(70, 40), activation="relu")	MPRegressor(batch_size=61, hidden_layer_sizes=(60, 30), activation="relu")	MPRegressor(batch_size=171, hidden_layer_sizes=(64, 30), activation="relu")	MPRegressor(batch_size=385, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=153, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=107, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=254, hidden_layer_sizes=(127, 70), activation="relu")
Plant no. 2	MPRegressor(batch_size=5, hidden_layer_sizes=(1, 1), activation="relu")	MPRegressor(batch_size=99, hidden_layer_sizes=(88, 80, 45), activation="relu")	MPRegressor(batch_size=288, hidden_layer_sizes=(104, 71, 67), activation="relu")	MPRegressor(batch_size=268, hidden_layer_sizes=(102, 38, 49), activation="relu")	MPRegressor(batch_size=308, hidden_layer_sizes=(104, 71, 67), activation="relu")	MPRegressor(batch_size=215, hidden_layer_sizes=(62, 42), activation="relu")	MPRegressor(batch_size=56, hidden_layer_sizes=(141, 86, 43), activation="relu")	MPRegressor(batch_size=105, hidden_layer_sizes=(160, 90), activation="relu")	MPRegressor(batch_size=192, hidden_layer_sizes=(159, 85, 45), activation="relu")	MPRegressor(batch_size=99, hidden_layer_sizes=(32, 62), activation="relu")
Plant no. 3	MPRegressor(batch_size=11, hidden_layer_sizes=(1, 1), activation="relu")	MPRegressor(batch_size=154, hidden_layer_sizes=(39, 41, 88), activation="relu")	MPRegressor(batch_size=96, hidden_layer_sizes=(79, 64), activation="relu")	MPRegressor(batch_size=442, hidden_layer_sizes=(84, 91, 43), activation="relu")	MPRegressor(batch_size=153, hidden_layer_sizes=(136, 61), activation="relu")	MPRegressor(batch_size=315, hidden_layer_sizes=(85, 27, 67), activation="relu")	MPRegressor(batch_size=26, hidden_layer_sizes=(88, 59, 24), activation="relu")	MPRegressor(batch_size=284, hidden_layer_sizes=(79, 46, 26), activation="relu")	MPRegressor(batch_size=123, hidden_layer_sizes=(81), activation="relu")	MPRegressor(batch_size=180, hidden_layer_sizes=(48, 57, 39), activation="relu")
Plant no. 4	MPRegressor(batch_size=182, hidden_layer_sizes=(97, 65), activation="relu")	MPRegressor(C=40.4675, epsilon=0.0184)	MPRegressor(C=20.5841, epsilon=0.0172)	MPRegressor(C=7.7864, epsilon=0.0279)	MPRegressor(batch_size=48, hidden_layer_sizes=(70, 63, 49), activation="relu")	MPRegressor(C=37.1585, epsilon=0.0287)	MPRegressor(C=31.3894, epsilon=0.0171)	MPRegressor(C=11.5300, epsilon=0.0213)	MPRegressor(C=46.6174, epsilon=0.0208)	MPRegressor(batch_size=453, hidden_layer_sizes=(27, 75, 32), activation="relu")
Plant no. 5	MPRegressor(batch_size=67, hidden_layer_sizes=(53, 99), activation="relu")	MPRegressor(C=84.1588, epsilon=0.0374)	MPRegressor(batch_size=69, hidden_layer_sizes=(104, 71, 67), activation="relu")	MPRegressor(batch_size=86, hidden_layer_sizes=(102, 38, 49), activation="relu")	MPRegressor(batch_size=111, hidden_layer_sizes=(104, 71, 67), activation="relu")	MPRegressor(batch_size=81, hidden_layer_sizes=(104, 71, 67), activation="relu")	MPRegressor(batch_size=151, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=139, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=267, hidden_layer_sizes=(180, 100), activation="relu")	MPRegressor(batch_size=286, hidden_layer_sizes=(180, 100), activation="relu")
Plant no. 6	MPRegressor(batch_size=15, hidden_layer_sizes=(76, 69), activation="relu")	MPRegressor(batch_size=50, hidden_layer_sizes=(53, 99), activation="relu")	MPRegressor(batch_size=50, hidden_layer_sizes=(91, 94, 23), activation="relu")	MPRegressor(batch_size=75, hidden_layer_sizes=(90, 90), activation="relu")	MPRegressor(batch_size=33, hidden_layer_sizes=(74, 61, 17), activation="relu")	MPRegressor(batch_size=51, hidden_layer_sizes=(97, 19, 20), activation="relu")	MPRegressor(batch_size=85, hidden_layer_sizes=(77, 94, 20), activation="relu")	MPRegressor(batch_size=152, hidden_layer_sizes=(84, 12, 51), activation="relu")	MPRegressor(batch_size=122, hidden_layer_sizes=(77, 99, 53), activation="relu")	MPRegressor(batch_size=57, hidden_layer_sizes=(99, 84, 48), activation="relu")
Plant no. 7	MPRegressor(batch_size=74, hidden_layer_sizes=(34, 99, 63), activation="relu")	MPRegressor(batch_size=273, hidden_layer_sizes=(88, 86, 10), activation="relu")	MPRegressor(batch_size=116, hidden_layer_sizes=(64, 57, 39), activation="relu")	MPRegressor(batch_size=170, hidden_layer_sizes=(69, 66, 59), activation="relu")	MPRegressor(batch_size=33, hidden_layer_sizes=(95, 80, 73), activation="relu")	MPRegressor(batch_size=61, hidden_layer_sizes=(97, 19, 20), activation="relu")	MPRegressor(batch_size=8, hidden_layer_sizes=(27, 66, 14), activation="relu")	MPRegressor(batch_size=40, hidden_layer_sizes=(85, 47), activation="relu")	MPRegressor(batch_size=287, hidden_layer_sizes=(61, 71, 42), activation="relu")	MPRegressor(batch_size=64, hidden_layer_sizes=(38, 89), activation="relu")
Plant no. 8	MPRegressor(C=91.9907, epsilon=0.0348)	MPRegressor(batch_size=119, hidden_layer_sizes=(46, 90), activation="relu")	MPRegressor(batch_size=137, hidden_layer_sizes=(53, 70, 50), activation="relu")	MPRegressor(batch_size=129, hidden_layer_sizes=(47, 50, 42), activation="relu")	MPRegressor(batch_size=213, hidden_layer_sizes=(88, 77), activation="relu")	MPRegressor(batch_size=276, hidden_layer_sizes=(154, 39, 18), activation="relu")	MPRegressor(batch_size=31, hidden_layer_sizes=(97, 77), activation="relu")	MPRegressor(batch_size=104, hidden_layer_sizes=(60, 87, 14), activation="relu")	MPRegressor(batch_size=166, hidden_layer_sizes=(89, 94, 28), activation="relu")	MPRegressor(batch_size=35, hidden_layer_sizes=(93), activation="relu")
Plant no. 9	MPRegressor(batch_size=160, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=42, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=27, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=42, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=45, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=203, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=30, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=63, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=127, hidden_layer_sizes=(18, 9, 11), activation="relu")	MPRegressor(batch_size=176, hidden_layer_sizes=(18, 9, 11), activation="relu")
Plant no. 10	MPRegressor(batch_size=355, hidden_layer_sizes=(67, 73, 87), activation="relu")	MPRegressor(batch_size=813, hidden_layer_sizes=(90, 80, 340), activation="relu")	MPRegressor(batch_size=38, hidden_layer_sizes=(90, 80, 340), activation="relu")	MPRegressor(batch_size=123, hidden_layer_sizes=(65, 77, 61), activation="relu")	MPRegressor(batch_size=102, hidden_layer_sizes=(62, 85, 30), activation="relu")	MPRegressor(batch_size=109, hidden_layer_sizes=(61, 75, 39), activation="relu")	MPRegressor(batch_size=48, hidden_layer_sizes=(74, 45, 49), activation="relu")	MPRegressor(batch_size=213, hidden_layer_sizes=(58, 31, 36), activation="relu")	MPRegressor(batch_size=127, hidden_layer_sizes=(64, 42, 21), activation="relu")	MPRegressor(batch_size=280, hidden_layer_sizes=(57, 21, 63), activation="relu")
Plant no. 11	MPRegressor(batch_size=355, hidden_layer_sizes=(81, 89), activation="relu")	MPRegressor(batch_size=127, hidden_layer_sizes=(76, 99, 24), activation="relu")	MPRegressor(batch_size=38, hidden_layer_sizes=(44, 21, 55), activation="relu")	MPRegressor(batch_size=123, hidden_layer_sizes=(62, 89), activation="relu")	MPRegressor(batch_size=102, hidden_layer_sizes=(76, 42, 61), activation="relu")	MPRegressor(batch_size=109, hidden_layer_sizes=(86, 79, 17), activation="relu")	MPRegressor(batch_size=48, hidden_layer_sizes=(88, 27, 87), activation="relu")	MPRegressor(batch_size=213, hidden_layer_sizes=(73, 61), activation="relu")	MPRegressor(batch_size=127, hidden_layer_sizes=(57, 21, 63), activation="relu")	MPRegressor(batch_size=280, hidden_layer_sizes=(64, 99, 20), activation="relu")

Table D.4: The forecast error (nMAE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].

	IM		SunDance		IM		SunDance		SunDance		Averaging		AutoPV		Averaging		AutoPV		AutoPV	
	HDA	HDA	HDA	HDA	AFB	AFB	AFB	AFB	AFB	AFB	default	nearby	default	nearby	default	nearby	default	nearby	nearby	AIB
Training data	2 yr	2 yr	2 yr	2 yr	1 mo	1 mo	1 mo	1 mo	1 mo	1 mo	none	none	none	none	none	none	none	none	none	none
	none	none	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
Adaption C_{ad}	none	none	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
Adaption K_{ad}	none	none	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
nMAE ₁	0.284 ±0.013	0.501 ±0.000	0.501 ±0.000	0.501 ±0.000	0.371 ±0.020	0.318 ±0.016	0.708 ±0.000	0.501 ±0.000	0.501 ±0.000	0.501 ±0.000	0.398 ±0.000	0.398 ±0.000	0.305 ±0.000	0.301 ±0.000	0.380 ±0.009	0.301 ±0.011	0.301 ±0.008	0.301 ±0.008	0.301 ±0.008	0.301 ±0.008
nMAE ₂	0.309 ±0.007	0.446 ±0.000	0.446 ±0.000	0.446 ±0.000	0.413 ±0.031	0.359 ±0.013	0.668 ±0.000	0.455 ±0.000	0.455 ±0.000	0.455 ±0.000	0.476 ±0.000	0.476 ±0.000	0.352 ±0.000	0.346 ±0.000	0.303 ±0.002	0.300 ±0.001	0.300 ±0.001	0.300 ±0.001	0.300 ±0.001	0.300 ±0.001
nMAE ₃	0.282 ±0.009	0.448 ±0.000	0.448 ±0.000	0.448 ±0.000	0.367 ±0.053	0.322 ±0.028	0.675 ±0.000	0.448 ±0.000	0.448 ±0.000	0.448 ±0.000	0.420 ±0.000	0.420 ±0.000	0.310 ±0.000	0.304 ±0.000	0.316 ±0.007	0.274 ±0.003	0.274 ±0.003	0.274 ±0.003	0.274 ±0.003	0.274 ±0.003
nMAE ₄	0.291 ±0.010	0.487 ±0.000	0.487 ±0.000	0.487 ±0.000	0.410 ±0.065	0.353 ±0.028	0.798 ±0.000	0.465 ±0.000	0.465 ±0.000	0.465 ±0.000	0.389 ±0.000	0.389 ±0.000	0.311 ±0.000	0.303 ±0.000	0.399 ±0.009	0.310 ±0.007	0.310 ±0.007	0.310 ±0.007	0.310 ±0.007	0.310 ±0.006
nMAE ₅	0.330 ±0.022	0.451 ±0.000	0.451 ±0.000	0.451 ±0.000	0.397 ±0.035	0.372 ±0.019	0.660 ±0.000	0.468 ±0.000	0.468 ±0.000	0.468 ±0.000	0.498 ±0.000	0.498 ±0.000	0.372 ±0.000	0.375 ±0.000	0.354 ±0.002	0.336 ±0.003	0.336 ±0.003	0.336 ±0.003	0.336 ±0.003	0.330 ±0.003
nMAE ₆	0.308 ±0.006	0.455 ±0.000	0.455 ±0.000	0.455 ±0.000	0.350 ±0.047	0.342 ±0.019	0.639 ±0.000	0.455 ±0.000	0.455 ±0.000	0.455 ±0.000	0.491 ±0.000	0.491 ±0.000	0.367 ±0.000	0.360 ±0.000	0.302 ±0.002	0.288 ±0.002	0.288 ±0.002	0.288 ±0.002	0.288 ±0.002	0.281 ±0.003
nMAE ₇	0.460 ±0.015	0.495 ±0.000	0.495 ±0.000	0.495 ±0.000	0.583 ±0.040	0.616 ±0.025	0.730 ±0.000	0.496 ±0.000	0.496 ±0.000	0.496 ±0.000	0.547 ±0.000	0.547 ±0.000	0.472 ±0.000	0.478 ±0.000	0.488 ±0.005	0.447 ±0.004	0.447 ±0.004	0.447 ±0.004	0.447 ±0.004	0.473 ±0.007
nMAE ₈	0.281 ±0.007	0.440 ±0.000	0.440 ±0.000	0.440 ±0.000	0.367 ±0.034	0.331 ±0.022	0.599 ±0.000	0.440 ±0.000	0.440 ±0.000	0.440 ±0.000	0.470 ±0.000	0.470 ±0.000	0.345 ±0.000	0.336 ±0.000	0.279 ±0.002	0.276 ±0.001	0.276 ±0.001	0.276 ±0.001	0.276 ±0.001	0.270 ±0.003
nMAE ₉	0.377 ±0.014	0.464 ±0.000	0.464 ±0.000	0.464 ±0.000	0.467 ±0.028	0.443 ±0.033	0.562 ±0.000	0.464 ±0.000	0.464 ±0.000	0.464 ±0.000	0.478 ±0.000	0.478 ±0.000	0.380 ±0.000	0.380 ±0.000	0.508 ±0.008	0.447 ±0.009	0.447 ±0.009	0.447 ±0.009	0.447 ±0.009	0.446 ±0.006
nMAE ₁₀	0.316 ±0.006	0.442 ±0.000	0.442 ±0.000	0.442 ±0.000	0.411 ±0.038	0.370 ±0.016	0.670 ±0.000	0.442 ±0.000	0.442 ±0.000	0.442 ±0.000	0.465 ±0.000	0.465 ±0.000	0.342 ±0.000	0.345 ±0.000	0.368 ±0.005	0.326 ±0.006	0.326 ±0.006	0.326 ±0.006	0.326 ±0.006	0.322 ±0.004
nMAE ₁₁	0.317 ±0.007	0.461 ±0.000	0.461 ±0.000	0.461 ±0.000	0.386 ±0.019	0.368 ±0.009	0.665 ±0.000	0.461 ±0.000	0.461 ±0.000	0.461 ±0.000	0.504 ±0.000	0.504 ±0.000	0.376 ±0.000	0.372 ±0.000	0.323 ±0.002	0.312 ±0.003	0.312 ±0.003	0.312 ±0.003	0.312 ±0.003	0.304 ±0.003
Median	0.309	0.455	0.455	0.455	0.397	0.359	0.668	0.461	0.461	0.461	0.476	0.476	0.352	0.346	0.354	0.354	0.354	0.354	0.354	0.304
Mean	0.323	0.463	0.463	0.463	0.411	0.381	0.670	0.463	0.463	0.463	0.467	0.467	0.357	0.355	0.365	0.365	0.365	0.365	0.365	0.327

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB), Adaptive Fixed Batch (AFB)

Table D.5: The forecast error (nRMSE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].

	IM		SunDance		IM		SunDance		SunDance		Averaging		AutoPV		Averaging		AutoPV		AutoPV	
	HDA	HDA	2 yr	2 yr	IM	IM	SunDance	SunDance	SunDance	SunDance	default	nearby	default	nearby	default	nearby	default	nearby	nearby	nearby
Training data	HDA	HDA	2 yr	2 yr	IM	IM	SunDance	SunDance	SunDance	SunDance	Averaging	nearby	AutoPV	AutoPV	AutoPV	nearby	AutoPV	AutoPV	nearby	AutoPV
Adaption C_{ad}	none	none	none	none	1 mo	28 d	1 mo	28 d	1 mo	28 d	none	none	28 d	28 d	none	none	28 d	28 d	none	28 d
Adaption K_{ad}	none	none	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	28 d	28 d	28 d	28 d	28 d	28 d
nRMSE ₁	0.606	0.938	0.938	0.938	0.722	0.651	1.266	1.266	0.937	0.937	1.002	0.765	0.725	0.732	0.732	0.765	0.623	0.626	0.623	0.626
nRMSE ₂	0.640	0.918	0.918	0.918	0.815	0.725	1.276	1.276	0.930	0.930	0.959	0.607	0.673	0.682	0.682	0.607	0.613	0.604	0.613	0.604
nRMSE ₃	0.581	0.937	0.937	0.937	0.714	0.656	1.289	1.289	0.937	0.937	0.824	0.649	0.603	0.613	0.613	0.649	0.573	0.566	0.573	0.566
nRMSE ₄	0.607	0.995	0.995	0.995	0.818	0.711	1.540	1.540	0.963	0.963	0.764	0.802	0.603	0.612	0.612	0.802	0.639	0.638	0.639	0.638
nRMSE ₅	0.690	0.933	0.933	0.933	0.773	0.740	1.266	1.266	0.959	0.959	1.014	0.677	0.756	0.755	0.755	0.677	0.663	0.651	0.663	0.651
nRMSE ₆	0.608	0.929	0.929	0.929	0.711	0.692	1.216	1.216	0.928	0.928	0.991	0.598	0.697	0.709	0.709	0.598	0.597	0.586	0.597	0.586
nRMSE ₇	1.029	1.150	1.150	1.150	1.285	1.341	1.519	1.519	1.150	1.150	1.117	0.998	1.030	0.988	0.988	0.998	0.963	1.002	0.963	1.002
nRMSE ₈	0.580	0.911	0.911	0.911	0.713	0.675	1.138	1.138	0.911	0.911	0.934	0.567	0.646	0.664	0.664	0.567	0.570	0.562	0.570	0.562
nRMSE ₉	0.782	1.004	1.004	1.004	0.950	0.910	1.149	1.149	1.003	1.003	0.946	0.966	0.776	0.780	0.780	0.966	0.877	0.883	0.877	0.883
nRMSE ₁₀	0.679	0.955	0.955	0.955	0.813	0.762	1.324	1.324	0.955	0.955	0.943	0.717	0.694	0.696	0.696	0.717	0.668	0.664	0.668	0.664
nRMSE ₁₁	0.649	1.025	1.025	1.025	0.756	0.729	1.357	1.357	1.024	1.024	0.768	0.625	0.602	0.600	0.600	0.625	0.627	0.613	0.627	0.613
Median	0.640	0.938	0.938	0.938	0.773	0.725	1.276	1.276	0.955	0.955	0.946	0.677	0.694	0.696	0.696	0.677	0.627	0.626	0.627	0.626
Mean	0.677	0.972	0.972	0.972	0.825	0.781	1.304	1.304	0.972	0.972	0.933	0.725	0.710	0.712	0.712	0.725	0.674	0.672	0.674	0.672

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB), Adaptive Fixed Batch (AFB)

Table D.6: The modeling error (nMAE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].

	IM		IM		SunDance		SunDance		Averaging		AutoPV		Averaging		AutoPV		AutoPV	
	HDA	HDA	AFB	AIB	1 mo	28 d	1 mo	28 d	default	nearby	default	28 d	default	nearby	default	28 d	nearby	AIB
Training data	2 yr	2 yr	1 mo	1 mo	1 mo	28 d	1 mo	28 d	none	none	none	28 d	none	none	none	28 d	none	none
	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
Adaption C_{ad}	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
Adaption K_{ad}	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	none	none	28 d	28 d	28 d	28 d
nMAE ₁	0.165 ±0.007	0.279 ±0.000	0.240 ±0.024	0.187 ±0.010	0.320 ±0.000	0.278 ±0.000	0.320 ±0.000	0.278 ±0.000	0.192 ±0.000	0.231 ±0.006	0.175 ±0.000	0.168 ±0.000	0.231 ±0.006	0.150 ±0.008	0.141 ±0.007	0.150 ±0.008	0.141 ±0.007	0.141 ±0.007
nMAE ₂	0.162 ±0.004	0.258 ±0.000	0.289 ±0.029	0.223 ±0.021	0.308 ±0.000	0.259 ±0.000	0.308 ±0.000	0.259 ±0.000	0.261 ±0.000	0.189 ±0.001	0.155 ±0.000	0.157 ±0.000	0.189 ±0.001	0.157 ±0.002	0.152 ±0.002	0.157 ±0.002	0.152 ±0.002	0.152 ±0.002
nMAE ₃	0.174 ±0.007	0.271 ±0.000	0.264 ±0.065	0.203 ±0.036	0.284 ±0.000	0.264 ±0.000	0.284 ±0.000	0.264 ±0.000	0.230 ±0.000	0.187 ±0.002	0.158 ±0.000	0.157 ±0.000	0.187 ±0.002	0.170 ±0.002	0.167 ±0.001	0.170 ±0.002	0.167 ±0.001	0.167 ±0.001
nMAE ₄	0.194 ±0.005	0.284 ±0.000	0.315 ±0.059	0.253 ±0.037	0.326 ±0.000	0.284 ±0.000	0.326 ±0.000	0.284 ±0.000	0.249 ±0.000	0.263 ±0.004	0.198 ±0.000	0.197 ±0.000	0.263 ±0.004	0.210 ±0.003	0.194 ±0.004	0.210 ±0.003	0.194 ±0.004	0.194 ±0.004
nMAE ₅	0.189 ±0.007	0.282 ±0.000	0.269 ±0.042	0.230 ±0.015	0.287 ±0.000	0.279 ±0.000	0.287 ±0.000	0.279 ±0.000	0.305 ±0.000	0.255 ±0.002	0.209 ±0.000	0.210 ±0.000	0.255 ±0.002	0.178 ±0.006	0.177 ±0.006	0.178 ±0.006	0.177 ±0.006	0.177 ±0.006
nMAE ₆	0.223 ±0.007	0.293 ±0.000	0.252 ±0.052	0.207 ±0.021	0.262 ±0.000	0.261 ±0.000	0.262 ±0.000	0.261 ±0.000	0.288 ±0.000	0.204 ±0.002	0.160 ±0.000	0.159 ±0.000	0.204 ±0.002	0.166 ±0.002	0.159 ±0.002	0.166 ±0.002	0.159 ±0.002	0.159 ±0.002
nMAE ₇	0.313 ±0.012	0.355 ±0.000	0.481 ±0.056	0.490 ±0.021	0.459 ±0.000	0.357 ±0.000	0.459 ±0.000	0.357 ±0.000	0.376 ±0.000	0.360 ±0.003	0.359 ±0.000	0.335 ±0.000	0.360 ±0.003	0.362 ±0.006	0.326 ±0.005	0.362 ±0.006	0.326 ±0.005	0.326 ±0.005
nMAE ₈	0.177 ±0.005	0.272 ±0.000	0.263 ±0.048	0.217 ±0.030	0.297 ±0.000	0.273 ±0.000	0.297 ±0.000	0.273 ±0.000	0.271 ±0.000	0.192 ±0.003	0.168 ±0.000	0.165 ±0.000	0.192 ±0.003	0.171 ±0.003	0.168 ±0.002	0.171 ±0.003	0.168 ±0.002	0.168 ±0.002
nMAE ₉	0.217 ±0.012	0.303 ±0.000	0.325 ±0.018	0.276 ±0.029	0.372 ±0.000	0.300 ±0.000	0.372 ±0.000	0.300 ±0.000	0.327 ±0.000	0.347 ±0.005	0.300 ±0.000	0.287 ±0.000	0.347 ±0.005	0.226 ±0.010	0.223 ±0.007	0.226 ±0.010	0.223 ±0.007	0.223 ±0.007
nMAE ₁₀	0.170 ±0.009	0.257 ±0.000	0.288 ±0.050	0.228 ±0.021	0.298 ±0.000	0.270 ±0.000	0.298 ±0.000	0.270 ±0.000	0.274 ±0.000	0.236 ±0.003	0.181 ±0.000	0.179 ±0.000	0.236 ±0.003	0.185 ±0.005	0.176 ±0.003	0.185 ±0.005	0.176 ±0.003	0.176 ±0.003
nMAE ₁₁	0.179 ±0.006	0.284 ±0.000	0.267 ±0.041	0.224 ±0.028	0.328 ±0.000	0.284 ±0.000	0.328 ±0.000	0.284 ±0.000	0.288 ±0.000	0.227 ±0.001	0.180 ±0.000	0.178 ±0.000	0.227 ±0.001	0.169 ±0.002	0.176 ±0.002	0.169 ±0.002	0.176 ±0.002	0.176 ±0.002
Median	0.179	0.282	0.269	0.224	0.308	0.278	0.308	0.278	0.274	0.231	0.180	0.178	0.231	0.171	0.176	0.171	0.176	0.176
Mean	0.197	0.285	0.296	0.249	0.322	0.283	0.322	0.283	0.278	0.244	0.204	0.199	0.244	0.195	0.187	0.195	0.187	0.187

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB), Adaptive Fixed Batch (AFB)

Table D.7: The modeling error (nRMSE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].

	IM		SunDance		IM		SunDance		Averaging		AutoPV		Averaging		AutoPV		AutoPV	
	HDA	HDA	AFB	AIB	AFB	AIB	AFB	AIB	default	nearby	default	AIB	nearby	nearby	nearby	nearby	AIB	
Training data	2 yr	2 yr	1 mo	1 mo	1 mo	1 mo	1 mo	1 mo	none	none	none	none	none	none	none	none	none	
	none	none	28 d	28 d	28 d	28 d	28 d	28 d	none	none	28 d	28 d	28 d	none	28 d	28 d	28 d	
	none	none	28 d	$i \cdot 28 \text{ d}$	28 d	$i \cdot 28 \text{ d}$	$i \cdot 28 \text{ d}$	$i \cdot 28 \text{ d}$	none	none	28 d	$i \cdot 28 \text{ d}$	28 d	none	28 d	$i \cdot 28 \text{ d}$	$i \cdot 28 \text{ d}$	
Adaption C_{ad} Adaption K_{ad}	nRMSE ₁	0.378 ±0.008	0.561 ±0.000	0.491 ±0.037	0.413 ±0.019	0.703 ±0.000	0.568 ±0.000	0.568 ±0.000	0.414 ±0.000	0.390 ±0.000	0.384 ±0.000	0.494 ±0.007	0.349 ±0.007	0.341 ±0.006				
	nRMSE ₂	0.372 ±0.003	0.519 ±0.000	0.570 ±0.061	0.475 ±0.047	0.649 ±0.000	0.519 ±0.000	0.519 ±0.000	0.545 ±0.000	0.363 ±0.000	0.363 ±0.000	0.398 ±0.002	0.398 ±0.002	0.384 ±0.001				
	nRMSE ₃	0.387 ±0.009	0.559 ±0.000	0.541 ±0.109	0.444 ±0.059	0.597 ±0.000	0.550 ±0.000	0.550 ±0.000	0.478 ±0.000	0.374 ±0.000	0.375 ±0.000	0.419 ±0.003	0.413 ±0.001	0.404 ±0.001				
	nRMSE ₄	0.435 ±0.004	0.590 ±0.000	0.628 ±0.097	0.534 ±0.057	0.686 ±0.000	0.589 ±0.000	0.589 ±0.000	0.523 ±0.000	0.449 ±0.000	0.448 ±0.000	0.576 ±0.006	0.469 ±0.003	0.448 ±0.004				
	nRMSE ₅	0.424 ±0.015	0.577 ±0.000	0.555 ±0.075	0.484 ±0.026	0.613 ±0.000	0.570 ±0.000	0.570 ±0.000	0.625 ±0.000	0.441 ±0.000	0.445 ±0.000	0.526 ±0.005	0.424 ±0.005	0.417 ±0.005				
	nRMSE ₆	0.469 ±0.013	0.582 ±0.000	0.523 ±0.084	0.443 ±0.025	0.529 ±0.000	0.529 ±0.000	0.529 ±0.000	0.595 ±0.000	0.381 ±0.000	0.381 ±0.000	0.440 ±0.004	0.410 ±0.001	0.397 ±0.001				
	nRMSE ₇	0.831 ±0.019	0.896 ±0.000	1.167 ±0.080	1.196 ±0.043	1.028 ±0.000	0.865 ±0.000	0.865 ±0.000	0.839 ±0.000	0.882 ±0.000	0.848 ±0.000	0.834 ±0.002	0.954 ±0.006	0.943 ±0.005				
	nRMSE ₈	0.392 ±0.005	0.551 ±0.000	0.528 ±0.081	0.460 ±0.049	0.656 ±0.000	0.549 ±0.000	0.549 ±0.000	0.554 ±0.000	0.387 ±0.000	0.385 ±0.000	0.416 ±0.005	0.423 ±0.003	0.411 ±0.003				
	nRMSE ₉	0.488 ±0.013	0.608 ±0.000	0.679 ±0.030	0.581 ±0.052	0.806 ±0.000	0.605 ±0.000	0.605 ±0.000	0.669 ±0.000	0.611 ±0.000	0.590 ±0.000	0.683 ±0.005	0.522 ±0.010	0.516 ±0.009				
	nRMSE ₁₀	0.402 ±0.009	0.534 ±0.000	0.580 ±0.083	0.487 ±0.038	0.689 ±0.000	0.559 ±0.000	0.559 ±0.000	0.570 ±0.000	0.412 ±0.000	0.410 ±0.000	0.476 ±0.003	0.451 ±0.004	0.436 ±0.003				
	nRMSE ₁₁	0.399 ±0.006	0.569 ±0.000	0.542 ±0.074	0.474 ±0.061	0.683 ±0.000	0.561 ±0.000	0.561 ±0.000	0.598 ±0.000	0.403 ±0.000	0.402 ±0.000	0.472 ±0.004	0.427 ±0.002	0.426 ±0.002				
Median	0.402	0.569	0.555	0.475	0.683	0.561	0.561	0.570	0.403	0.402	0.476	0.424	0.417					
Mean	0.452	0.595	0.618	0.545	0.694	0.588	0.588	0.583	0.463	0.457	0.521	0.476	0.466					

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB), Adaptive Fixed Batch (AFB)

Table D.8: The forecast error (nMAE) of models in the evaluation on synthetic mixed-oriented PV plants. AutoPV’s ensemble weight optimization $\hat{\mathbf{w}}$ is compared to AutoPV using the actual weights \mathbf{w} of each plant (prior knowledge), which reflect the mounting configurations and shares used to generate the synthetic data. The table is based on [168].

	IM HDA	SunDance HDA	IM AIB	SunDance AIB	AutoPV \mathbf{w}	AutoPV $\hat{\mathbf{w}}$
Training data	2 yr	2 yr	1 mo	1 mo	none	none
Adaption C_{ad}	none	none	28 d	28 d	none	28 d
Adaption K_{ad}	none	none	$i \cdot 28 \text{ d}$	$i \cdot 28 \text{ d}$	none	$i \cdot 28 \text{ d}$
nMAE ₁	0.319 ± 0.010	0.542 ± 0.000	0.359 ± 0.015	0.542 ± 0.000	0.272 ± 0.000	0.275 ± 0.000
nMAE ₂	0.337 ± 0.008	0.474 ± 0.000	0.381 ± 0.011	0.473 ± 0.000	0.281 ± 0.000	0.276 ± 0.000
nMAE ₃	0.336 ± 0.007	0.471 ± 0.000	0.397 ± 0.010	0.467 ± 0.000	0.279 ± 0.000	0.276 ± 0.000
nMAE ₄	0.277 ± 0.008	0.485 ± 0.000	0.351 ± 0.018	0.485 ± 0.000	0.271 ± 0.000	0.273 ± 0.000
nMAE ₅	0.320 ± 0.004	0.561 ± 0.000	0.378 ± 0.016	0.561 ± 0.000	0.274 ± 0.000	0.277 ± 0.000
nMAE ₆	0.331 ± 0.008	0.461 ± 0.000	0.384 ± 0.013	0.457 ± 0.000	0.269 ± 0.000	0.274 ± 0.000
nMAE ₇	0.313 ± 0.010	0.456 ± 0.000	0.365 ± 0.011	0.456 ± 0.000	0.270 ± 0.000	0.266 ± 0.000
nMAE ₈	0.328 ± 0.009	0.505 ± 0.000	0.387 ± 0.026	0.492 ± 0.000	0.276 ± 0.000	0.274 ± 0.000
nMAE ₉	0.281 ± 0.006	0.487 ± 0.000	0.323 ± 0.008	0.487 ± 0.000	0.260 ± 0.000	0.274 ± 0.000
nMAE ₁₀	0.334 ± 0.005	0.624 ± 0.000	0.386 ± 0.013	0.633 ± 0.000	0.303 ± 0.000	0.302 ± 0.000
nMAE ₁₁	0.302 ± 0.036	0.551 ± 0.000	0.355 ± 0.031	0.552 ± 0.000	0.276 ± 0.000	0.285 ± 0.000
nMAE ₁₂	0.306 ± 0.007	0.490 ± 0.000	0.362 ± 0.010	0.513 ± 0.000	0.257 ± 0.000	0.263 ± 0.000
Median	0.320	0.488	0.372	0.490	0.273	0.275
Mean	0.315	0.509	0.369	0.510	0.274	0.276

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB)

Table D.9: The forecast error (nRMSE) of models in the evaluation on synthetic mixed-oriented PV plants. AutoPV’s ensemble weight optimization $\hat{\mathbf{w}}$ is compared to AutoPV using the actual weights \mathbf{w} of each plant (prior knowledge), which reflect the mounting configurations and shares used to generate the synthetic data. The table is based on [168].

	IM HDA	SunDance HDA	IM AIB	SunDance AIB	AutoPV \mathbf{w}	AutoPV $\hat{\mathbf{w}}$
Training data	2 yr	2 yr	1 mo	1 mo	none	none
Adaption C_{ad}	none	none	28 d	28 d	none	28 d
Adaption K_{ad}	none	none	$i \cdot 28 \text{ d}$	$i \cdot 28 \text{ d}$	none	$i \cdot 28 \text{ d}$
nRMSE ₁	0.618 ± 0.020	1.102 ± 0.000	0.691 ± 0.028	1.102 ± 0.000	0.567 ± 0.000	0.550 ± 0.000
nRMSE ₂	0.697 ± 0.013	1.043 ± 0.000	0.777 ± 0.023	1.033 ± 0.000	0.604 ± 0.000	0.588 ± 0.000
nRMSE ₃	0.692 ± 0.010	1.042 ± 0.000	0.808 ± 0.028	1.019 ± 0.000	0.600 ± 0.000	0.586 ± 0.000
nRMSE ₄	0.559 ± 0.014	1.051 ± 0.000	0.692 ± 0.038	1.050 ± 0.000	0.581 ± 0.000	0.562 ± 0.000
nRMSE ₅	0.695 ± 0.005	1.215 ± 0.000	0.795 ± 0.033	1.214 ± 0.000	0.591 ± 0.000	0.585 ± 0.000
nRMSE ₆	0.672 ± 0.010	1.016 ± 0.000	0.771 ± 0.024	1.002 ± 0.000	0.583 ± 0.000	0.570 ± 0.000
nRMSE ₇	0.631 ± 0.005	1.008 ± 0.000	0.725 ± 0.017	1.008 ± 0.000	0.575 ± 0.000	0.551 ± 0.000
nRMSE ₈	0.667 ± 0.014	1.079 ± 0.000	0.788 ± 0.065	1.051 ± 0.000	0.584 ± 0.000	0.559 ± 0.000
nRMSE ₉	0.573 ± 0.008	1.042 ± 0.000	0.651 ± 0.012	1.041 ± 0.000	0.564 ± 0.000	0.571 ± 0.000
nRMSE ₁₀	0.724 ± 0.008	1.302 ± 0.000	0.806 ± 0.020	1.330 ± 0.000	0.631 ± 0.000	0.616 ± 0.000
nRMSE ₁₁	0.618 ± 0.043	1.140 ± 0.000	0.709 ± 0.039	1.151 ± 0.000	0.583 ± 0.000	0.576 ± 0.000
nRMSE ₁₂	0.620 ± 0.009	1.052 ± 0.000	0.728 ± 0.027	1.091 ± 0.000	0.563 ± 0.000	0.559 ± 0.000
Median	0.649	1.051	0.749	1.051	0.583	0.570
Mean	0.647	1.091	0.745	1.091	0.586	0.573

Individual Model (IM), Historical Data Available (HDA), Adaptive Increasing Batch (AIB)

Table D.10: The estimated SunDance parameters for single- and mixed-oriented PV plants based on weather measurement data. The table is based on [168].

	PV size and efficiency	Tilt $\hat{\beta}_{T,array}$	Azimuth $\hat{\theta}_{A,array}$
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 20^\circ$	242	49.6	174
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 30^\circ$	262	58.6	174
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 40^\circ$	280	65.6	173
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 20^\circ$	215	30.4	123
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 30^\circ$	222	36.7	109
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 40^\circ$	224	42.3	102
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 20^\circ$	218	35.6	222
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 30^\circ$	226	42.4	232
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 40^\circ$	231	47.4	239
mixed $\beta_{T,array} = 20^\circ$	389	29.1	178
mixed $\beta_{T,array} = 30^\circ$	365	29.1	178
mixed $\beta_{T,array} = 40^\circ$	336	29.1	179

Table D.11: The modeling errors (nMAE and nRMSE) of SunDance and AutoPV-default on single- and mixed-oriented PV plants. The table is based on [168].

	SunDance nMAE	SunDance nRMSE	AutoPV-default nMAE	AutoPV-default nRMSE
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 20^\circ$	0.210	0.411	0.045	0.090
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 30^\circ$	0.225	0.434	0.050	0.107
$\theta_{A,array} = 170^\circ, \beta_{T,array} = 40^\circ$	0.238	0.456	0.062	0.135
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 20^\circ$	0.210	0.462	0.064	0.141
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 30^\circ$	0.245	0.548	0.084	0.192
$\theta_{A,array} = 80^\circ, \beta_{T,array} = 40^\circ$	0.295	0.657	0.088	0.206
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 20^\circ$	0.212	0.464	0.054	0.121
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 30^\circ$	0.238	0.537	0.071	0.168
$\theta_{A,array} = 260^\circ, \beta_{T,array} = 40^\circ$	0.275	0.620	0.079	0.184
mixed $\beta_{T,array} = 20^\circ$	0.197	0.406	0.043	0.085
mixed $\beta_{T,array} = 30^\circ$	0.222	0.475	0.046	0.090
mixed $\beta_{T,array} = 40^\circ$	0.267	0.584	0.051	0.104

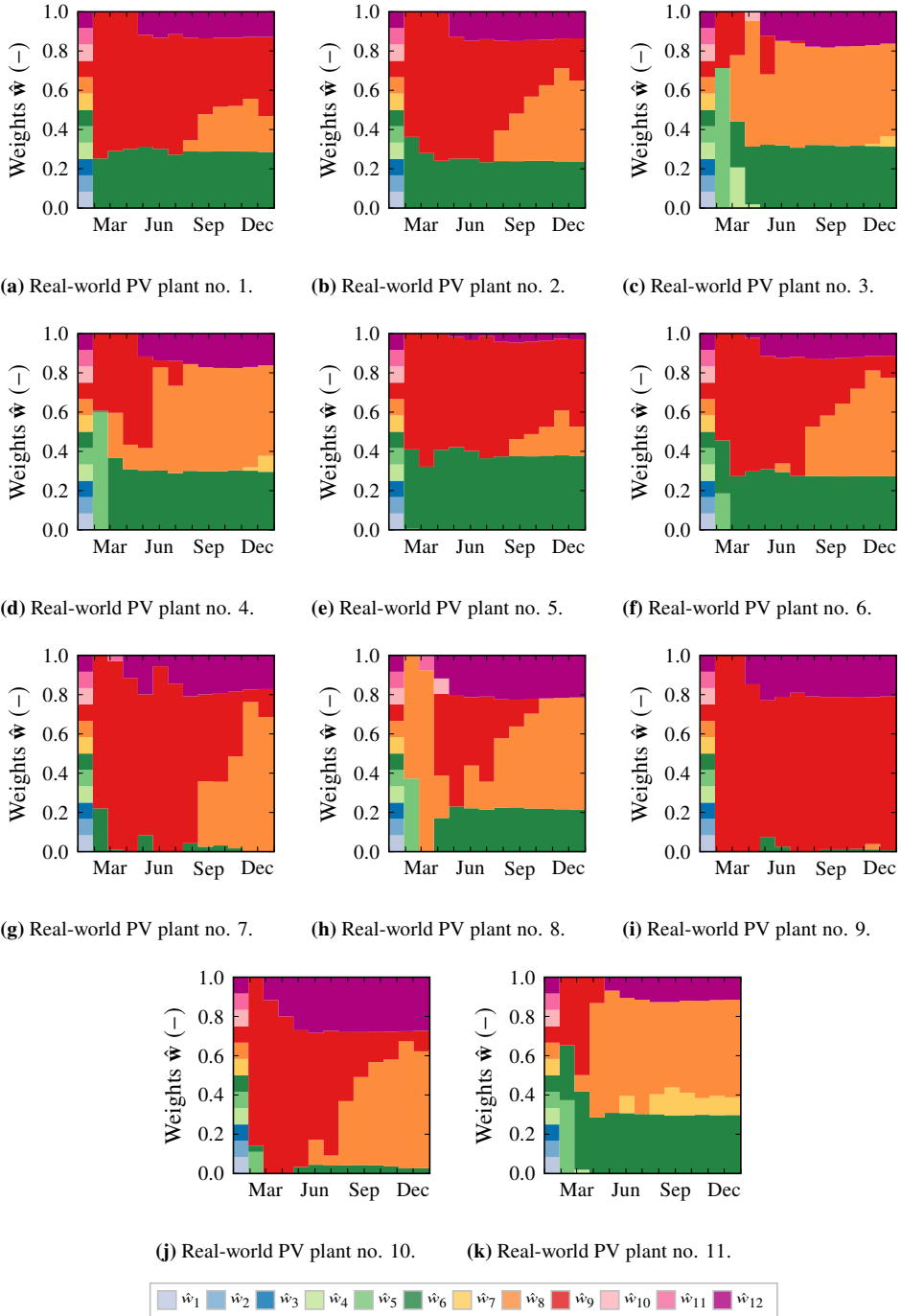


Figure D.1: Evolution of estimated weights \hat{w} for the real-world PV plants [168].

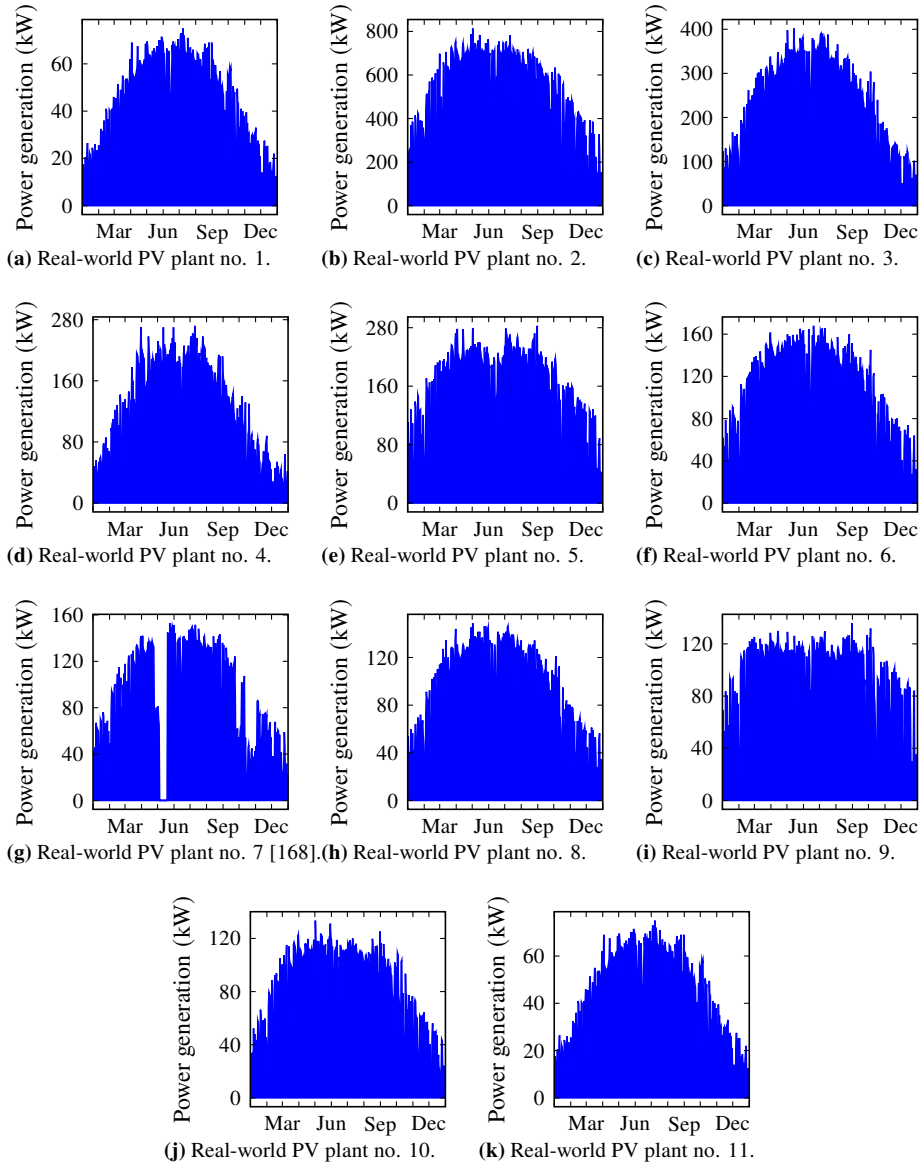


Figure D.2: Daily power generation maximum of the real-world PV plants.

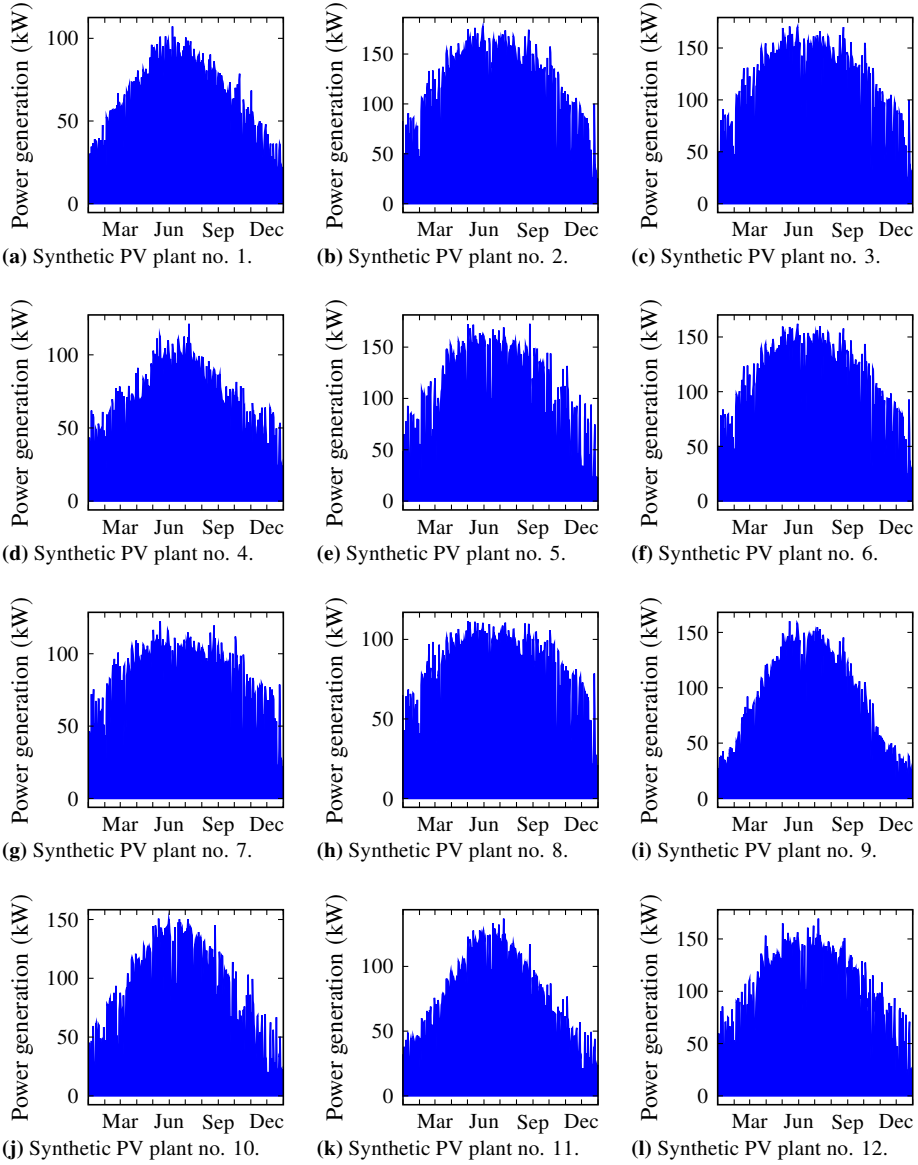


Figure D.3: Daily power generation maximum of the synthetic mixed-oriented PV plants.

D.2 Generating probabilistic PV forecasts with AutoPV using normalizing flows

In this additional evaluation, we combine AutoPV [168, 169] (Chapter 5) and AutoPQ [164, 165] (Chapter 3). First, we make a point forecast with AutoPV and then, we generate a probabilistic forecast using the normalizing flow-based cINN [107, 115]. For the evaluation, we use the open-source PV data set of the GEFCom 2014 [241] with the training, validation and test data sets as defined in Table B.6.

The data set consists of three adjacent PV plants in Australia, whose exact geographical location is unknown. Using AutoPV-nearby for this data set is not practical for two reasons: First, the data set only includes three plants, and the ensemble would only consist of two plants in a leave-one-out evaluation. Second, the evaluation in Section 3.2.1 is based on the aggregate power generation of all three PV plants, and the results would not be comparable to the leave-one-out evaluation. Therefore, we only consider AutoPV-default in the additional evaluation. AutoPV-default, however, requires the geographical location of the PV plants. Since it is unknown where the plants are located in Australia, we estimate the coordinates using an HPO. More specifically, we minimize the CRPS as in Section 3.1.3 with the objective function (3.6). As configuration space $\Lambda = \Lambda_p \cup \Lambda_q$, we consider the latitude and longitude coordinates that enclose Australia for Λ_p (specified in Table D.12), and the cINN's sampling hyperparameter for Λ_q (specified in Table B.4).

Table D.12: The configuration space Λ_p for estimating the coordinates in decimal degrees of the PV plant using HPO.

Hyperparameter	Value range
lat	[−44.0, −10.0]
long	[113.0, 154.0]

The result of the HPO is shown in Figure D.4, which visualizes the CRPS assessed on the validation data set in dependence on the latitude and longitude coordinates. We consider the estimated geographical location as the point for which the validation CRPS becomes minimal. This point lies in the state of Victoria, north-west of Melbourne, with the coordinates in decimal degrees $\{\text{lat}=-36.2\pm0.1, \text{long}=-148.2\pm0.2\}$, averaged over five runs. Table D.13 shows the CRPS achieved on the hold-out test data set by AutoPV-default compared to the nine point forecasting methods considered by AutoPQ. In this ranking, AutoPV-default shows competitive performance and takes the 3rd place. A visual comparison of the resulting PIs is exemplified in Figure D.5.

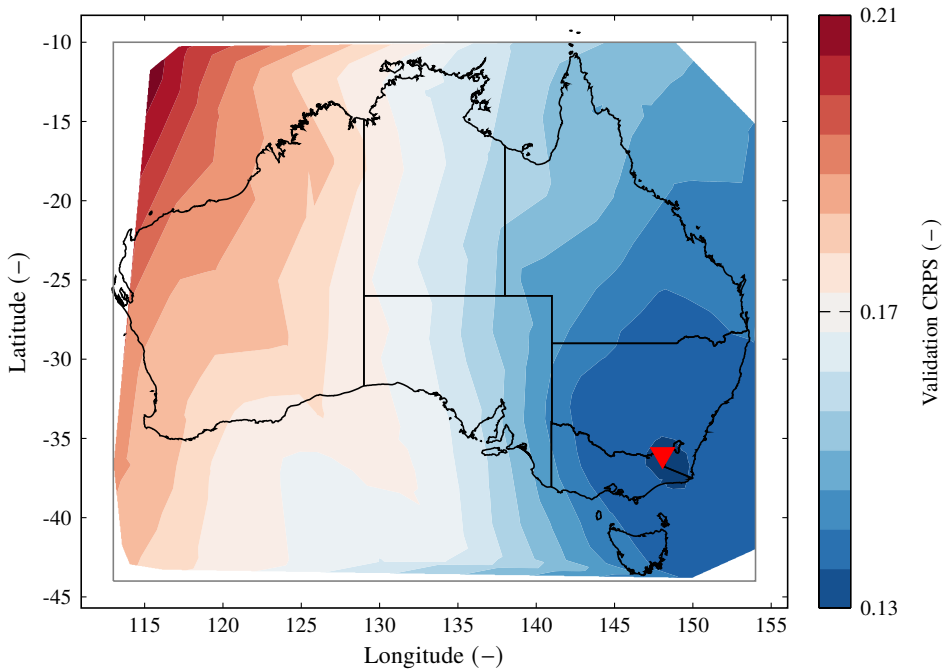


Figure D.4: The estimated coordinates of the PV plant in decimal degrees for which the CRPS is minimal, which is assessed in the HPO on the validation data set. The limits of the configuration space (Table D.12) are visualized as gray lines, the color contour is calculated by linear interpolation on the triangular grid spanned by these dots, and the shape of Australia is visualized as black lines, which originate from [365].

Table D.13: The probabilistic forecast error (CRPS) of AutoPV-default compared to the forecasting methods considered by AutoPQ evaluated on the test data set of the PV data set [241]. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table D.1, B.1, B.2, and B.3).

Forecasting method	CRPS
XGB-cINN	0.106±0.001
SVR-cINN	0.114±0.001
AutoPV-default-cINN	0.118±0.001
TFT-cINN	0.123±0.004
MLP-cINN	0.140±0.002
DeepAR-cINN	0.165±0.010
N-HiTS-cINN	0.171±0.008
TBATS-cINN	0.376±0.016
ETS-cINN	0.386±0.022
sARIMAX-cINN	1.002±0.045

From the results of this additional evaluation, we can draw the following two conclusions: First, a probabilistic forecast can be generated from AutoPV’s point forecast using the normalizing flow-based cINN [107, 115]. In this way, AutoPV-default achieves a competitive probabilistic performance compared to other state-of-the-art forecasting methods. Second, based on the evaluation in Section 5.3.1, the performance of AutoPV-nearby can be expected to improve over the performance of AutoPV-default. That is, if data of nearby PV plants are available and the ensemble pool models are created with a ML-based forecasting method like XGB, AutoPV-nearby may achieve similar performance to this method.

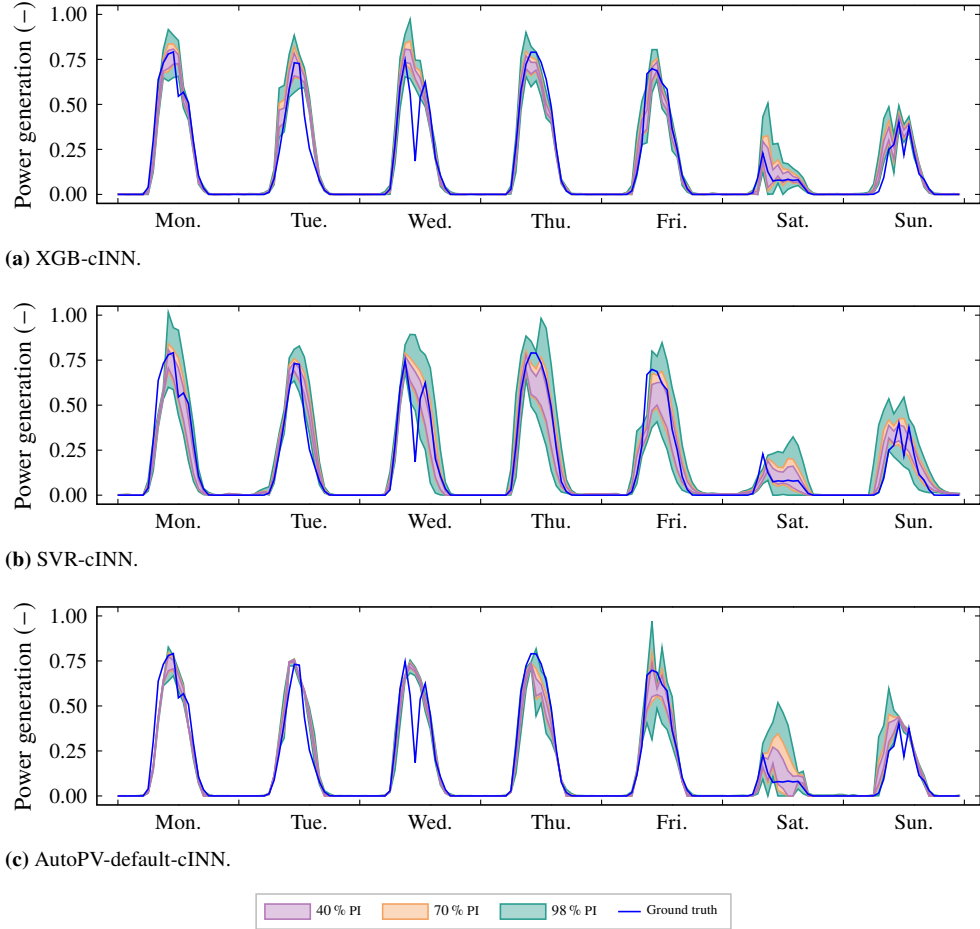


Figure D.5: Exemplary 40 %, 70 %, and 98 % PIs for the PV data set, comparing AutoPV-default to XGB and SVR. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table D.1 and B.2). The day-ahead forecasts originate every day from 00:00.

List of abbreviations

aggreg.	aggregation
avg.	averaging
AC	Alternating Current
ACF	AutoCorrelation Function
ADF	Augmented Dickey-Fuller
ADWIN	ADaptive WINdowing
AFB	Adaptive Fixed Batch
AIB	Adaptive Increasing Batch
AIC	Akaike Information Criterion
ANN	Artificial Neural Network
AOI	Angle Of Incidence
API	Application Programming Interface
AR	AutoRegression
ARMA	AutoRegressive Moving Average
ARMAX	AutoRegressive Moving Average with eXternal input
ARIMA	AutoRegressive Integrated Moving Average
AutoML	Automated Machine Learning
BATS	Box-Cox transformation, ARMA errors, Trend, Seasonal components
BIC	Bayesian Information Criterion
BO	Bayesian Optimization
cyc.	cyclic
CASH	Combined Algorithm Selection and Hyperparameter optimization
CDF	Cumulative Distribution Function
cINN	conditional Invertible Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CRPS	Continuous Ranked Probability Score
CV	Cross Validation
dyn.	dynamic
DA/RE	Datenaustausch/Redispatch
DC	Direct Current

DCP	District Community Platform
DEA	Differential Evolution Algorithm
DES	Double Exponential Smoothing
DeepAR	Deep AutoRegression
DHI	Diffuse Horizontal Irradiance
DISC	Direct Insolation Simulation Code
DL	Deep Learning
DNI	Direct Normal Irradiance
dRW	Random Walk considering drift
DT	Decision Tree
exo.	exogenous
EA	Evolutionary Algorithm
ECMWF	European Centre for Medium-Range Weather Forecasts
EDA	Estimation Distribution Algorithm
ELM	Extreme Learning Machine
ES	Exponential Smoothing
ETR	Extra Trees Regressor
ETS	Error Trend Seasonality
EU	European Union
EV	(battery) Electric Vehicle
FCNN	Fully Connected Neural Network
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GBM	Gradient Boosting Machine
GCP	Grid Connection Point
GEFCom	Global Energy Forecasting Competition
GHI	Global Horizontal Irradiance
GOF	Goodness Of Fit
GP	Gaussian Process
GPT	Generative Pre-trained Transformer
GPU	Graphics Processing Unit
GRNN	General Regression Neural Network
HDA	Historical Data Available
HPC	High-Performance Computing
HPO	HyperParameter Optimization
IC	Information Criteria
IM	Individual Model

INF	Iterative Neural Filter
IQR	Inter-Quartile Range
JSON	JavaScript Object Notation
KIT	Karlsruhe Institute of Technology
kNN	k-Nearest Neighbors
KPSS	Kwiatkowski-Phillips-Schmidt-Shin
KS	Kolmogorov-Smirnov
LASSO	Least Absolute Shrinkage and Selection Operator
LDA	Linear Discriminant Analysis
LOF	Local Outlier Factor
LogR	Logistic Regression
LR	Linear Regression
LSTM	Long Short-Term Memory
MA	Moving Average
MAD	Median Absolute Deviations
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MAQD	Mean Absolute Quantile Deviation
MIQP	Mixed Integer Quadratic Programming
MKL	Multiple Kernel Learning
ML	Machine Learning
MLOps	Machine Learning Operations
MLP	MultiLayer Perceptron
MOGWO	Multi-Objective Grey Wolf Optimizer
MRMR	Minimum Redundancy Maximum Relevance
MSE	Mean Squared Error
n. d.	not defined
NARX	Non-linear Auto-Regressive with eXternal Input
N-BEATS	Neural Basis Expansion Analysis for interpretable Time Series forecasting
N-HITS	Neural Hierarchical Interpolation for Time Series
NLP	Non-Linear Programming
nMAE	normalized Mean Absolute Error
nMSE	normalized Mean Squared Error
nRMSE	normalized Root Mean Squared Error
NNQF	Nearest Neighbor Quantile Filter

NREL	National Renewable Energy Laboratory
NWP	Numerical Weather Prediction
OCSB	Osborn-Chui-Smith-Birchenhall
OEM	Original Equipment Manufacturer
OEP	Open Energy Platform
OPSD	Open Power System Data
PACF	Partial AutoCorrelation Function
PCA	Principal Component Analysis
PDF	Probability Density Function
PI	Prediction Interval
PK	Prior Knowledge
PL	Pinball Loss
PM	Physical-inspired Modeling
PNN	Profile Neural Network
POA	Plane On Array
PSO	Particle Swarm Optimization
PV	PhotoVoltaic
pyWATTS	Python Workflow Automation Tool for Time Series
QD	Quantile Deviation
QRNN	Quantile Regression Neural Network
RAM	Random-Access Memory
ReLU	Rectified Linear Unit
REST	REpresentational State Transfer
RF	Random Forest
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RW	Random Walk
SAPM	Sandia PV Array Performance Model
sARIMA	seasonal AutoRegressive Integrated Moving Average
sARIMAX	seasonal AutoRegressive Integrated Moving Average with eXternal input
SES	Simple Exponential Smoothing
SETARMA	Self Exciting Threshold AutoRegressive Moving Average
SM	Statistical Modeling
sMAPE	symmetric Mean Absolute Percentage Error
sRW	Random Walk considering seasonality
STL	Seasonal and Trend decomposition using LOESS

SVM	Support Vector Machine
SVR	Support Vector Regression
SWKN	Stadtwerke Karlsruhe Netzservice GmbH
tran.	transformation
TBATS	Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend, Seasonal components
TES	Triple Exponential Smoothing
TFT	Temporal Fusion Transformer
TimeGPT	Time Generative Pre-trained Transformer
TPE	Tree Parzen Estimator
TRY	Test Reference Year
VAR	Vector AutoRegression
wgt.	weighting
WP	Wind Power
XGB	eXtreme Gradient Boosting

List of symbols

This section summarizes the notation used throughout the present thesis inspired by the formulation given in [366]. The following general rules of notation apply:

- The symbol \hat{x} indicates an estimate of x .
- x^* is a transformation of x .
- x^\star is the optimal value of x .
- Δx shows that it is an increment or a gradient of x , respectively.
- \bar{x} indicates that it is the arithmetic mean value of x .
- The symbols in round brackets after a symbol $x(a, b)$ point to the dependencies of this symbol.
- Upper case bold face letters represent matrices (e. g. \mathbf{X}), lower case bold face letters represent vectors (e. g. \mathbf{x} for column vectors and \mathbf{x}^\top for row vectors).

Numerical symbols

$\mathbb{1}$	indicator function	(-)
--------------	--------------------	-----

Latin symbols

b_i	exhausted budget (iterations)	(-)
B_i	available computation budget (iterations)	(-)
b_t	exhausted budget (time)	(s)
B_t	available computation budget (time)	(s)
C_{SV}	regularization parameter of SVM and SVR methods	(-)
C_{ad}	cycle length of adaption	(s)
d_1	order of differencing	(-)

D_I	order of seasonal differencing	(-)
E	energy	(J)
E_{eff}	effective irradiance	(W m ⁻²)
\mathbb{E}	expected value	(-)
f	function	(-)
F	Càdlàg function	(-)
g	function	(-)
G	GHI including TCC	(W m ⁻²)
h	time point on the forecast horizon H	(-)
H	forecast horizon	(-)
H_1	horizon of past values	(-)
h_a	height above ground level (unknown variable in the wind power law)	(m)
h_b	height above ground level (known variable in the wind power law)	(m)
h_{eff}	effective height in the wind power curve	(m)
i	running index	(-)
j	running index	(-)
J	Jacobian matrix	(-)
k_{sparse}	sparsity parameter of the MIQP method	(-)
k_{NN}	number of nearest neighbors of the kNN method	(-)
k_{best}	number of best elements considered	(-)
k	time point	(-)
K	number of time points	(-)
K_{ad}	number of samples for adaption	(-)
\mathcal{L}	likelihood function	(-)
L	number of features	(-)
lat	latitude in decimal degrees; pos. north of equator, neg. to south	(-)
long	longitude in decimal degrees; pos. east of prime meridian, neg. to west	(-)
m	sample mean	(-)
m_{abs}	absolute air mass	(-)
m_{rel}	relative air mass	(-)
n	running index	(-)
N	terminal value of running index n	(-)
N_m	number of models in the ensemble pool	(-)
N_p	number of considered point forecasting methods	(-)
N_h	number of hidden neurons	(-)
N_i	number of input neurons	(-)
N_p	number of parameters	(-)

N_λ	number of hyperparameters	(-)
N_σ	number of random samples	(-)
\mathcal{N}	normal distribution	(-)
\mathbb{N}	natural numbers	(-)
\mathbb{N}_1	natural numbers without zero	(-)
p_{AR}	order of autoregression	(-)
P_{AR}	order of seasonal autoregression	(-)
p	model parameter scalar	(-)
\mathbf{p}	model parameters vector	(-)
P	power	(kW)
P_{\max}	peak power rating	(kWp)
q	quantile	(-)
\mathbf{q}	set of quantiles	(-)
q_{MA}	order of moving average	(-)
Q_{MA}	order of seasonal moving average	(-)
Q	performance measure	(-)
Q_{exo}	exogenous error	(-)
Q_{end}	endogenous error	(-)
Q	objective function	(-)
r	residual scalar	(-)
\mathbf{r}	residual vector	(-)
\mathbf{r}_σ	random noise vector	(-)
\mathbb{R}	real numbers	(-)
R^2	coefficient of determination	(-)
s	sample standard deviation	(-)
s_p	seasonal period	(-)
s_{init}	initialization seed parameter	(-)
t	time stamp	((YYYY-MM-DD hh:mm:ss))
t_k	sample period	(h)
TCC	total cloud cover	(-)
T	air temperature	(°C)
T_{array}	cell temperature of the PV array	(°C)
\mathcal{U}	uniform distribution	(-)
v	wind speed	(m s ⁻¹)
v_{eff}	effective wind speed in the wind power curve	(m s ⁻¹)
v_a	wind speed at height h_a (unknown variable in the wind power law)	(m s ⁻¹)
v_b	wind speed at height h_b (known variable in the wind power law)	(m s ⁻¹)

w	ensemble weight scalar	(-)
\mathbf{w}	ensemble weight vector	(-)
w_e	efficiency factor	(-)
x	feature scalar	(-)
x_{lag}	lag feature scalar	(-)
x_{sin}	sin-encoded cyclic feature scalar	(-)
x_{cos}	cos-encoded cyclic feature scalar	(-)
x_{ord}	ordinally-encoded cyclic feature scalar	(-)
x_{Mon}	one-hot-encoded cyclic feature scalar (Monday)	(-)
x_{Tue}	one-hot-encoded cyclic feature scalar (Tuesday)	(-)
x_{Wed}	one-hot-encoded cyclic feature scalar (Wednesday)	(-)
x_{Thu}	one-hot-encoded cyclic feature scalar (Thursday)	(-)
x_{Fri}	one-hot-encoded cyclic feature scalar (Friday)	(-)
x_{Sat}	one-hot-encoded cyclic feature scalar (Saturday)	(-)
x	exogenous input value scalar	(-)
\mathbf{x}	exogenous input values vector (K rows)	(-)
\mathbf{X}	exogenous input values matrix (K rows, L columns)	(-)
y	output value scalar	(-)
\mathbf{y}	output values vector (K rows)	(-)
\mathbb{Y}	realization space	(-)
z	latent scalar	(-)
\mathbf{z}	latent vector	(-)
\mathbb{Z}	latent space	(-)

Greek symbols

α	learning rate	(-)
α_{ES}	level parameter of ES methods	(-)
α_t	threshold of a statistical test	(-)
α_h	Hellmann exponent of the wind power law	(-)
β_{ES}	trend parameter of ES methods	(-)
$\beta_{\text{T,array}}$	tilt angle of the PV array in decimal degrees	(-)
γ_{SV}	kernel coefficient of SVM and SVR methods	(-)
γ_{ES}	seasonal parameter of ES methods	(-)
ε_{SV}	penalty margin of SVM and SVR methods	(-)
θ_*	transformation parameter of the Theta method	(-)

θ_Z	zenith angle of the sun over time in decimal degrees	(-)
θ_A	azimuth angle of the sun over time in decimal degrees	(-)
$\theta_{A,array}$	azimuth angle of the PV array in decimal degrees	(-)
λ_s	shrinkage factor of the LASSO method	(-)
$\lambda_{\mathcal{L}}$	regularization parameter in the maximum likelihood loss function	(-)
λ	hyperparameter configuration scalar	(-)
$\boldsymbol{\lambda}$	hyperparameter configuration vector	(-)
$\boldsymbol{\Lambda}$	hyperparameter configuration space	(-)
μ	population mean	(-)
σ	population standard deviation	(-)
φ_{ES}	damped trend parameter of ES methods	(-)
φ_h	relative air humidity	(-)

List of Figures

1.1	The transformation of the energy system from a primarily fossil-fueled power plant mix to renewable electricity generation; simplified, without claiming to correctly depict the electricity mix in Germany. ¹	2
1.2	The development of renewable energy share in the German gross electricity consumption [15] and the targeted share for 2030 [16]. Explicit sectors are shown in Figure A.1 in the Appendix.	3
1.3	The forecasting pipeline systematizes the design process for time series forecasting using five pipeline sections. The figure is based on [1].	3
1.4	Comparison of a grid search and a random search for minimizing a function with one important and one unimportant parameter [1]. The figure is based on [112, 123].	21
1.5	Two exemplary iterations of a BO on a 1D function. The BO minimizes the predicted objective function $\hat{Q}(\lambda)$ (purple line) by maximizing the acquisition function (light orange surface). The acquisition value is high where the value of $\hat{Q}(\lambda)$ is low, and its predictive uncertainty (light purple surface) is high. The true objective function (dashed line) might lie outside of the predicted uncertainty interval [1]. The figure is based on [112]	23
1.6	The contributions of this chapter with the review on automated forecasting pipelines.	35
2.1	The six levels of the automation level taxonomy AutoLVL for time series forecasting [162], inspired by the SAE standard for autonomous driving of vehicles [170]. The purple bars indicate manual steps and the orange bars automated steps in the workflow.	37
2.2	Example of a graphical forecasting pipeline. An option for aggregating the outputs of several estimators is ensemble optimization shown in Example 5.	41
2.3	Example of HPO of the estimator within a forecasting pipeline. While this example focuses on HPO, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1).	42

2.4 The assessed trials on the validation data set in the HPO. The limits of the configuration space (Table 2.1) are visualized as gray lines, the assessed configurations are represented by black dots, and the color contour is calculated by linear interpolation on the triangular grid spanned by these dots. In this example, HPO stops after 100 iterations, leaving unexplored areas, where BO-TPE chose to exploit areas near already-evaluated high-performing configurations rather than to explore. 43

2.5 Example of CASH within a forecasting pipeline. While this example focuses on CASH, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1). 44

2.6 The assessed trials on the validation data set in the CASH using BO-TPE. The limits of the configuration space (Table 2.3) are visualized as gray lines, the assessed configurations are represented by black dots, and the color contour is calculated by linear interpolation on the triangular grid spanned by these dots. In this example, CASH stops after 100 iterations, leaving unexplored areas, where BO-TPE chose to exploit areas near already-evaluated high-performing configurations rather than to explore. 45

2.7 Example of CASH using the successive halving strategy within a forecasting pipeline. While this example focuses on successive halving, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1). 48

2.8 Evolution of the considered estimators’ validation performances in the successive halving-based CASH with the performance at 0 iterations reflecting the default configurations, shown as a reference and therefore not considered within the iteration budget B_i . The solid lines with dot markers show the performance of active methods, while the dotted lines with cross markers show the performance of inactive methods as it would have evolved without pruning. The progress in the successive halving pruning rounds is assumed to be linear for simplicity’s sake. 49

2.9 Example of ensemble optimization within a forecasting pipeline. While this example focuses on ensemble optimization, the pre-processing includes data scaling (Section 1.3.1), and the post-processing considers the use of Prior Knowledge (PK) (Section 2.3.1). 50

2.10 Exemplary 24 h-ahead forecast using ensemble optimization. Both MLP models receive high weights, whereas the weights of the GBM models are below 1 %. The forecasts of the ensemble pool models $\hat{y}_n, n \in [1, 2, 3, 4]$ are shown opaquely. 51

2.11	Example of applying PK within a forecasting pipeline. While this example focuses on PK, the pre-processing includes data scaling (Section 1.3.1), and the MLP estimator's configuration may be optimized using HPO (Section 2.2.1).	53
2.12	Correlation of air temperature \hat{T} , relative humidity $\hat{\varphi}_h$, and wind speed \hat{v} of an exogenous forecast $\hat{\mathbf{X}}$ to the target time series \mathbf{y} and exemplary 24 h-ahead forecast using these features together with post-processing.	53
2.13	Exemplary 24 h-ahead forecast using pre-trained models in the ensemble pool.	55
2.14	Iterations of the cut check procedure with two sub-windows of the ADWIN algorithm [198]; figure is based on [199]. The sequence shown is exemplary and represents the values of an arbitrary time series [169].	55
2.15	Example of an incremental and a sudden MSE drift at $k = 850$ and corresponding p-values of the ADWIN drift detection.	56
2.16	Detection delay and false positives depending on the significance level α_t	57
2.17	Illustration of the cycle length C_{ad} and the number of considered samples K_{ad} in the cyclic model adaption for a time series with the resolution of 15 minutes (96 samples per day). For AIB, K_{ad} increases over time for the number of adaptations $i = 1$ (purple), $i = 2$ (orange), and $i = 3$ (green), whereas it remains unchanged for AFB.	58
2.18	Illustration of the parameter K_{ad} in the drift detection-based adaption. Once a drift is detected (red vertical line), the model is adapted with the samples collected after the identified split (green vertical line). While these samples can be used for quick adaption after drift detection (purple), the model can be adapted once more data is collected by combining drift detection-based adaption with the cyclic adaption AIB (orange and green).	59
2.19	Architecture concept for the forecasting service to achieve Automation level 1, 2, 3, and 4 [163].	60
2.20	The contributions of this chapter with the proposed automation level taxonomy AutoLVL.	65
2.21	The three forecasting templates contributed in this thesis, achieving Automation level 2, 3, and 4. ²	67

3.1 Overview of AutoPQ: Lag features, cyclic features, and exogenous features are selected and used as inputs by a point forecasting model to create a point forecast in an unknown distribution. This point forecast and the features are combined in a cINN [107, 115], which creates a representation of the forecast in a known and tractable distribution. The neighborhood of this representation is analyzed to determine how to include uncertainty information. Finally, with the backward pass through the cINN, the uncertainty is mapped back to the unknown distribution to create the probabilistic forecast. Novel automation methods are highlighted in green [165]. 71

3.2 Comparison of the average computational effort in terms of runtime across the six data sets used in the evaluation: effort for training a point forecasting model with configuration λ_p and effort for generating a quantile forecast based on the point forecast using the cINN [107, 115] with λ_q . Due to the significantly smaller computational effort of generating the quantile forecast, it is worthwhile to evaluate several λ_q for one λ_p to properly balance the efforts. Note that for ETS, SVR, and XGB the standard deviation is higher than the mean value. The figure is based on [165]. 76

3.3 Estimation of the BO prior distribution and its utilization in the HPO of the sampling hyperparameter λ_q . The figure is based on [165]. 81

3.4 AutoPQ-advanced with the successive halving pruning strategy exemplified for the Load-BW data set. Starting with the nine forecasting methods, underperforming methods successively drop out, while promising methods receive increased time budget for the HPO. The markers at $B_t = 0$ represent the performance of the respective default configuration, shown as a reference and therefore not considered within the time budget B_t 82

3.5 The CRPS evaluated on the hold-out test data sets with methods categorized into three classes: i) direct probabilistic benchmark methods, ii) point forecast-based probabilistic benchmark methods, and iii) AutoPQ with the default and the advanced configuration. The numerical values aggregated in this figure are detailed in Table B.7 in the Appendix. 86

3.6	Exemplary 40 %, 70 %, and 98 % PIs of a day-ahead forecast with origin at 00:00 for the Mobility data set. The probabilistic forecasts are generated by using AutoPQ-advanced with the nine considered point forecasting methods, specifically, three of each for the method families SM (sARIMAX, ETS, TBATS), ML (MLP, SVR, XGB), and DL (DeepAR, N-HiTS, TFT). The configuration of the point forecasting method λ_p and the sampling hyperparameter λ_q are optimized jointly to minimize the CRPS. The figure is based on [165].	88
3.7	Exemplary 40 %, 70 %, and 98 % PIs of a day-ahead forecast with origin at 00:00 for the Mobility data set. The probabilistic benchmarks can be classified into direct probabilistic methods (DeepAR, NNQF, QRNNs) and point forecast-based probabilistic methods (Gaussian PIs, Empirical PIs, Conformal PIs based on a TFT forecaster). The figure is based on [165].	89
3.8	Comparison of the MSE (point forecast) with the CRPS (probabilistic forecast) on the six data sets in the HPO of AutoPQ-advanced. Each dot represents the performance of a trial configuration in the HPO and legend of (a) applies to all sub-plots.	93
3.9	Comparison of the MSE (point forecast) with the MAQD (probabilistic forecast) on the six data sets in the HPO of AutoPQ-advanced. Each dot represents the performance of a trial configuration in the HPO and legend of (a) applies to all sub-plots.	93
3.10	Exemplary 40 %, 70 %, and 98 % PIs of seven day-ahead forecasts with origin at 00:00 each for the Load-GCP data set. The probabilistic forecasts are generated by using AutoPQ-advanced with the MLP and N-HiTS as point forecasting methods, respectively. The configuration of the point forecasting method λ_p and the sampling hyperparameter λ_q are optimized jointly to minimize the CRPS and the MAQD, respectively.	94
3.11	Comparison of the convergence of Algorithm 2 and 3 (ablation) for the HPO of MLP on the six data sets. The thick solid line represents the mean value, and the opaque area is the standard deviation over five runs [165]. Legend of (a) applies to all sub-plots.	96
3.12	Comparison of the convergence of Algorithm 2 and 3 (ablation) for the HPO of N-HiTS on the six data sets. The thick solid line represents the mean value, and the opaque area is the standard deviation over five runs [165]. Legend of (a) applies to all sub-plots.	97

3.13	The number of iterations required for the inner loop of Algorithm 2 for optimizing the sampling hyperparameter λ_q until the early stopping criteria is fulfilled compared for three trial schedulers: random search, BO-TPE, and BO-TPE-PK. Each box plot considers all nine forecasting methods and five runs. The figure is based on [165].	98
3.14	Evolution of the considered forecasting methods' validation performances in the successive halving-based CASH of AutoPQ-advanced with the performance at 0 h reflecting AutoPQ-default, shown as a reference and therefore not considered within the time budget B_t . The solid lines with dot markers show the performance of active methods, while the dotted lines with cross markers show the performance of inactive methods as it would have evolved without pruning. The progress in the successive halving pruning rounds is assumed to be linear for simplicity's sake [165].	100
3.15	The HPO of the sampling hyperparameter λ_q with the inner loop of Algorithm 2 using BO-TPE and BO-TPE-PK. The validation performance (CRPS) over the trial configuration (λ_q) is shown as a dot, and the TPE prior distribution is shown as a filled area normalized by the maximum of the respective PDF. The plots exemplarily show the HPO for one XGB configuration but are representative in terms of the other point forecasting methods [165].	101
3.16	Comparison of the computational effort for a run of AutoPQ-default and AutoPQ-advanced for different numbers of point forecasting methods N_p in the CASH and different time budgets B_t in the successive halving pruning strategy. Since AutoPQ-advanced assesses four trial configurations in parallel while AutoPQ-default runs sequentially, we multiply the total computing time of AutoPQ-advanced by four. The total electricity consumption was recorded by the HoreKa HPC. The figure is based on [165].	104
3.17	The contributions of this chapter with the Automation level 2 template AutoPQ. ³	111
4.1	Using the WP curve to make forecasts requires the wind speed forecast to be corrected to the curve's reference height. The figure is based on [162, 166].	115
4.2	The three steps of AutoWP's automated design exemplified for the real-world WP turbine no. 1 [166].	117

- 4.3 Power generation y and explanatory variables available in the forecasting model's future and past horizon. x_{sd} labels identified shutdowns in the past, \hat{y}_{wpc} is the turbine's theoretical power generation according to the OEM WP curve, \hat{v}_{eff} is the wind speed forecast at hub height, and the features $x_{sin, 365}$, $x_{cos, 365}$, $x_{sin, 1440}$, and $x_{cos, 1440}$ encode temporally recurring shutdowns patterns. Dashed lines visualize unavailable future or unused past periods, and the vertical line marks the forecast origin [166]. 120
- 4.4 Data imputation at shutdowns with the theoretical power generation according to the turbine's OEM WP curve (i. e. the imputed values depend on the wind speed measurement at hub height v_{eff}). During shutdowns, generation y is zero although the wind speed v_{eff} is higher than cut-in speed $v_{eff, cut-in}$ [166]. 120
- 4.5 Identified WP turbine shutdowns with rule-based filtering. Black fields represent data points at a measured wind speed greater than the cut-in speed with a power generation lower than the cut-in power [167]. The shutdowns for WP turbine no. 1 amount to 49 % and 20 % for no. 2. 122
- 4.6 Data pre-processing to identify abnormal operational states. Rule-based filtering identifies turbine shutdowns and LOF-based filtering identifies turbine stop-to-operation transitions and vice versa. The samples for which the wind speed at hub height is below the cut-in speed $v_{eff, cut-in} = 2.5 \text{ m s}^{-1}$ amount to 16.7 %, while the cut-out speed $v_{eff, cut-out} = 25 \text{ m s}^{-1}$, at which the WP turbine would be shut down for safety reasons, is not reached [167]. Legend of WP turbine no. 1 applies also to WP turbine no. 2. 122
- 4.7 The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nMAE when *considering* shutdowns. Legend of WP turbine no. 2 applies also to WP turbine no. 1. The figure is based on [166]. 124
- 4.8 The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nMAE when *disregarding* shutdowns. Legend of WP turbine no. 2 applies also to WP turbine no. 1. The figure is based on [166]. 124
- 4.9 Comparison of autoregressive DL forecasting methods (purple) to methods based on WP curve modeling (orange) in terms of the test nMAE when *disregarding* shutdowns. The figure is based on [166]. 125

4.10	Exemplary comparison of WP forecasts for WP turbine no. 2 using N-HiTS (autoregressive), AutoWP and MLP (WP curve modeling) on days without shutdowns. While methods based on WP curve modeling are only based on weather forecasts (ECMWF), autoregressive methods also consider past WP generation measurements when making forecasts. Legend of first column of a row applies to all columns in the row. The figure is based on [166]. . . .	127
4.11	Relationship between wind speed forecast (ECMWF) and WP generation forecasts (N-HiTS, AutoWP, MLP) for WP turbine no. 2 when <i>disregarding</i> shutdowns. AutoWP only relies on the height-corrected wind speed forecast, the MLP is trained using the forecasts of wind speed and direction, temperature and air pressure, and N-HiTS additionally considers past WP generation measurements. No WP generation despite moderate and high wind speeds are not shutdowns (already filtered out) but result from the error in the weather forecast, i. e., the wind speed forecast is above the cut-in speed while the realized wind speed is underneath. Legend of first column applies to all columns. The figure is based on [166].	128
4.12	The contributions of this chapter with the Automation level 3 template AutoWP. ⁴	131
5.1	The three steps of AutoPV's automated design exemplified for the real-world PV plant no. 11 [169].	135
5.2	The PM pipeline for creating a model for the default model pool, which consists of N_m models. For creating the n -th model, the pipeline first takes the time stamp $t[k]$, the latitude and longitude (lat, long) coordinates, the GHI $[k]$, optionally the DHI $[k]$, and the PV mounting configuration $(\theta_{A,array,n}, \beta_{T,array,n})$ to determine the POA irradiance. Then, it estimates the PV power generation $P_{AC}[k]$ based on POA $[k]$, the air temperature $T[k]$, and the wind speed $v[k]$. Finally, it normalizes $P_{AC}[k]$ with the peak power rating P_{max} , resulting in $\hat{y}_n^*[k]$. The figure is based on [168].	136
5.3	The ML pipeline for creating a model for the nearby plants model pool, which consists of N_m models. For creating the n -th model, the pipeline first normalizes the PV power generation measurement $y[k]$ with the peak power rating $P_{max,n}$. Then, it creates explanatory features based on the GHI $[k]$, the air temperature $T[k]$, and the time stamp $t[k]$. Based on these features, a regression model is trained to estimate the normalized power generation \hat{y}_n^* , which is post-processed to ensure physical limits. Both the selection of the regression algorithm and the optimization of its hyperparameters are automated by CASH. The figure is based on [168].	139

5.4	The modeling error and the forecast error (nMAE) of models in the plant-wise leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no initial training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The numerical values aggregated in this figure are detailed in Table D.4 and D.6 in the Appendix. The figure is based on [168].	147
5.5	Exemplary predictions and forecasts in the four seasons on real-world data. Predictions are generated with weather measurement data as model inputs, and forecasts are generated with day-ahead weather forecasting data as model inputs. The weather measurements are taken from the closest weather station (~ 10 km). That is, there may be a slight time delay between the weather measurement data and the PV generation data when clouds are passing the PV plant. The day-ahead weather forecasts consider cloud cover but with lower spatial and temporal resolution. As a result, sunny days in the summer and in the fall are typically easier to forecast than days with varying cloud cover in the spring and in the winter. The figure is based on [168].	148
5.6	Evolution of estimated weights $\hat{\mathbf{w}}$ for the synthetic mixed-oriented PV plants; based on [168].	151
5.7	Representation of a mixed-oriented PV plant resulting from the superposition of the power generation curves of east- and west-facing arrays (at a ratio of 1:1). While mixed-oriented PV plants can be represented by AutoPV, they cannot be well represented by PM-based methods that estimate a single PV mounting configuration. The predictions are generated using weather measurement data to visualize the modeling error. The figure is based on [168].	153
5.8	Evolution of AutoPV's weights over time and the rolling nMAE with a window size of seven days using ADWIN drift detection [198] based on the modeling error. The figure is based on [169].	156
5.9	Comparison of the ADWIN drift detection method [198] based the forecast error and the modeling error on the synthetic sudden and incremental drifts. The figure is based on [169].	159
5.10	The contributions of this chapter with the Automation level 4 template AutoPV. ⁵	169

6.1	The forecasting service consists of sub-services deployed as individual pods where communication between pods is established via a cluster network. Only the management service interacts with the other services of the Energy Lab [347, 348] like optimization, visualization, and data collection services.	172
6.2	An overview of the communication architecture of the Smart Charging Wizard in future work, taking into account the EV's battery behavior and PV forecasts. New elements are orange (AutoPV), while elements to be modified are green (user interface and charging optimization) and unchanged elements are purple. The illustration is adapted from [350].	173
6.3	Overview of the Smart East district with six GCPs, PV installations and EV charging locations. The smart charging algorithm is currently applied to two GCPs (4 and 6), with the available PV generation capacity expanding rapidly (see planed areas). The figure is based on [163].	174
6.4	Measurement and forecasts of electrical load (positive) and PV generation (negative) for a real-world GCP combined as profiles. It is visible that the installed PV plant does not cover the district's entire electricity demand on the three depicted winter days. However, the PV plant supplies electricity at times of high consumption, reducing the overall peak load [163].	175
6.5	The 119 kWp PV plant with mixed east-west orientation at GCP no. 6.	176
6.6	Illustration of dispatch and redispatch for a congested line between two regions; simplified, without claiming to correctly represent the power grid in Germany. ⁶	177
A.1	Evolution of the renewable energy share in the German energy consumption in the heating and cooling sector and the transport sector [15].	185
B.1	Exemplary 40 %, 70 %, and 98 % PIs of seven day-ahead forecasts with origin at 00:00 each for the Mobility data set, comparing AutoPQ-default and AutoPQ-advanced with the best-performing direct (DeepAR) and point-based probabilistic benchmark (Conformal PIs with TFT).	195
C.1	The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nRMSE when <i>considering</i> shutdowns. Legend of WP turbine no. 2 applies also to no. 1. The figure is based on [166].	197
C.2	The impact of different shutdown handling methods applied to the autoregressive DL forecasting methods DeepAR, N-HiTS, and TFT on the test nRMSE when <i>disregarding</i> shutdowns. Legend of WP turbine no. 2 applies also to no. 1. The figure is based on [166].	197

C.3	Comparison of autoregressive DL forecasting methods (purple) to methods based on WP curve modeling (orange) in terms of the test nRMSE when <i>disregarding</i> shutdowns. The figure is based on [166].	198
C.4	Exemplary 40 %, 70 %, and 98 % PIs for the WP data set, comparing AutoWP to XGB and SVR. For all methods, we apply the cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use the default configuration (Table B.2). The day-ahead forecasts originate every day from 00:00.	200
D.1	Evolution of estimated weights $\hat{\mathbf{w}}$ for the real-world PV plants [168].	211
D.2	Daily power generation maximum of the real-world PV plants.	212
D.3	Daily power generation maximum of the synthetic mixed-oriented PV plants.	213
D.4	The estimated coordinates of the PV plant in decimal degrees for which the CRPS is minimal, which is assessed in the HPO on the validation data set. The limits of the configuration space (Table D.12) are visualized as gray lines, the color contour is calculated by linear interpolation on the triangular grid spanned by these dots, and the shape of Australia is visualized as black lines, which originate from [365].	215
D.5	Exemplary 40 %, 70 %, and 98 % PIs for the PV data set, comparing AutoPV-default to XGB and SVR. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table D.1 and B.2). The day-ahead forecasts originate every day from 00:00.	216

List of Tables

1.1	Summary of automated data pre-processing methods in the literature on time series forecasting pipelines; adapted from [1].	9
1.2	Summary of automated feature engineering methods in the literature on time series forecasting pipelines; adapted from [1].	13
1.3	Exemplification of cyclic encoding methods for the day of the week of a time series; adapted from [1].	15
1.4	Summary of HPO methods in the literature on time series forecasting pipelines; adapted from [1].	21
1.5	Summary of automated method selection and ensembling methods in the literature on time series forecasting pipelines; adapted from [1].	27
2.1	HPO configuration space Λ of the GBM forecasting method in Scikit-learn [100] naming convention.	42
2.2	The forecast error of the GBM's optimized configuration $\lambda^{\hat{\star}}$ compared its default configuration λ_{default} . The error metrics are calculated on the hold-out test data set and are averaged over five HPO runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).	43
2.3	CASH configuration space Λ of the GBM and the MLP in Scikit-learn [100] naming convention. The batch size of the MLP estimator is chosen between $2^i, i \in \{3, 4, \dots, 9\}$	44
2.4	The forecast error of the optimized configuration $\lambda^{\hat{\star}}$ compared to the default configurations λ_{default} of the GBM and the MLP. The error metrics are calculated on the hold-out test data set and are averaged over five CASH runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).	45
2.5	CASH configuration space Λ in Scikit-learn [100] naming convention. The batch size of the MLP estimator is chosen between $2^i, i \in \{3, 4, \dots, 9\}$	48
2.6	Hyperparameter configurations λ of the two GBM and the two MLP estimators in the ensemble model pool in Scikit-learn [100] naming convention. . .	50

2.7	The forecast error of the ensemble forecast compared to the default configurations λ_{default} of the GBM and the MLP. The error metrics are calculated on the hold-out test data set and are averaged over five ensemble optimization runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).	51
2.8	The forecast error of the MLP forecaster using PK for feature engineering and post-processing compared to not using PK. The error metrics are calculated on the hold-out test data set and are averaged over five training runs. The improvement is significant if the p-value of the one-sided t-test is smaller than 0.05 (colored green).	53
2.9	Taxonomies related to automated time series forecasting [163].	62
2.10	The six levels of the AutoLVL taxonomy [162] as a guideline for developing automated forecasting templates [163]. A check mark signifies that the step is automated, and a cross denotes a manual step.	66
3.1	The percentage improvement of AutoPQ-advanced over the comparison methods in terms of the CRPS evaluated on the hold-out test data sets. AutoPQ-advanced uses HPO to optimize both the configuration of the point forecasting method λ_p and the cINN's sampling hyperparameter λ_q , while AutoPQ-default only optimizes λ_q . The values are marked with an asterisk if the improvement is significant (p-value < 0.05). The table is based on [165]. The actual p-values of the one-tailed t-test are detailed in Table B.8 in the Appendix.	86
3.2	The impact of using the CRPS and the MAQD as validation metrics in the HPO of AutoPQ-advanced with the MLP point forecasting method on different performance metrics evaluated on the hold-out test data sets and averaged across five runs. The best average value for each data set is highlighted in bold.	91
3.3	The impact of using the CRPS and the MAQD as validation metrics in the HPO of AutoPQ-advanced with the N-HiTS point forecasting method on different performance metrics evaluated on the hold-out test data sets and averaged across five runs. The best average value for each data set is highlighted in bold.	91

3.4	The percentage improvement of each optimized configuration over the default configuration in the successive halving ablation study without pruning in terms of the CRPS evaluated on the validation data. The values are marked with an asterisk if the improvement is significant (p -value < 0.05). The actual p -values of the one-tailed t -test are detailed in Table B.11 in the Appendix. The table is based on [165].	101
3.5	Related work on time series forecasting in terms of uncertainty quantification, automated design, and customizable properties. The table is based on [165].	107
5.1	AutoPV's estimated weights $\hat{\mathbf{w}}$ after one year compared to the actual weights \mathbf{w} evaluated on synthetic mixed-oriented PV plants. For example, PV plant no. 1 actually consists of 15 % north-oriented and 45°-tilted, 37 % east-oriented and 45°-tilted, and 48 % west-oriented and 75°-tilted PV arrays. The table is based on [168].	152
5.2	The configuration of the synthetic concept drifts inserted into the test data set; based on [169].	155
5.3	The nMAE averaged over all PV plants in the leave-one-out evaluation on original real-world data and data with synthetically inserted drifts classified into three classes: i) baseline methods that are not adaptive, ii) adaptive methods that are quasi-cold-start capable since they require little training data, and iii) methods that are cold-start capable as they do not require initial training data. The table is based on [169] and the evaluation of the original data with fixed-cycle adaption trigger originate from [168].	158
5.4	Related work on automated PV power generation forecasting with limited information. If no name was given to the method by the respective authors, only the reference is shown. The table is adapted from [168].	164
B.1	The configuration spaces Λ_p for the SM-based point forecasting methods in sktime naming convention [364] with the seasonal period $s_p = 24$. The table is based on [165].	187
B.2	The configuration spaces Λ_p for the ML-based point forecasting methods in the respective naming conventions. The table is based on [165].	187
B.3	The configuration spaces Λ_p for the DL-based (point) forecasting methods in PyTorch Forecasting naming convention [243]. The table is based on [165].	188
B.4	The configuration space Λ_q for the cINN [107, 115], which transforms a point forecasting model into a probabilistic one, in the respective naming convention. The table is based on [165].	188

B.5	The configuration of the Propulate EA [181] for HPO, in the respective naming convention. The table is based on [165].	189
B.6	Overview of the training, validation, and test data sets with hourly resolution used for evaluation, as well as the selected features based on [115]. The names of the target variables and the features refer to the column names in the data sets. The table is based on [165].	190
B.7	The CRPS evaluated on the hold-out test data sets with methods categorized into three classes: i) direct probabilistic benchmark methods, ii) point forecast-based probabilistic benchmark methods, and iii) AutoPQ with the default and the advanced configuration. The results of the benchmarks and AutoPQ-default for the data sets Load-GCP, Mobility, Price, and PV originate from [115]. The table is based on [165].	191
B.8	The p-values of the one-tailed t-test with the null hypothesis that the test CRPS of AutoPQ-advanced and the comparison method is equal and the alternative hypothesis that the test CRPS of AutoPQ-advanced is less. The values are colored green if the p-value is smaller than the significance level 0.05 and colored red otherwise.	191
B.9	The CRPS evaluated on the hold-out test data sets before and after post-processing. In post-processing, negative values in the quantile forecasts are set to zero to maintain given limits (mobility indicator, PV generation, and WP generation are greater than or equal to zero). For the other three data sets (Load-BW, Load-GCP, and Price), either no negative values exist in the quantile forecasts, respectively, negative values are valid. The table is based on [165].	192
B.10	The validation CRPS compared for three trial schedulers (random search, BO-TPE, and BO-TPE-PK) in the inner loop of Algorithm 2. Across the six data sets and the nine forecasting methods, only minor differences occur. The table is based on [165].	193
B.11	The p-values of the one-tailed t-test with the null hypothesis that the validation CRPS of the optimized configuration $\hat{\lambda}^*$ and the default configuration λ_{default} is equal and the alternative hypothesis that the validation CRPS of the optimized configuration is less. The values are colored green if the p-value is smaller than the significance level 0.05 and colored red otherwise.	194

C.1	The probabilistic forecast error (CRPS) of AutoWP compared to the forecasting methods considered by AutoPQ evaluated on the test data set of the WP data set [241]. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table B.1, B.2, and B.3).	199
D.1	The configuration of the PM pipeline in pvlib [305] naming convention. The table is based on [168].	201
D.2	The configuration space Λ for the automated regression estimator design using CASH in Scikit-learn [100] naming convention. The table is based on [168].	202
D.3	The selected (estimated) optimal configuration λ^* , i. e., the optimal regression algorithm and corresponding optimal hyperparameters, in the 10 runs of the CASH on real-world data. The selected configuration is used for creating AutoPV's nearby-plants model pool (leave-one-out) and the IM. The table is based on [168].	203
D.4	The forecast error (nMAE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].	204
D.5	The forecast error (nRMSE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].	205
D.6	The modeling error (nMAE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].	206

D.7	The modeling error (nRMSE) of models in the leave-one-out evaluation on real-world data classified into three classes: i) baseline methods that use two years of training data and are not cold-start capable, ii) adaptive methods that require little training data and are quasi-cold-start capable, and iii) methods that require no training data and are cold-start capable. AutoPV is evaluated using the default model pool and the nearby plants model pool. The table is based on [168].	207
D.8	The forecast error (nMAE) of models in the evaluation on synthetic mixed-oriented PV plants. AutoPV's ensemble weight optimization $\hat{\mathbf{w}}$ is compared to AutoPV using the actual weights \mathbf{w} of each plant (prior knowledge), which reflect the mounting configurations and shares used to generate the synthetic data. The table is based on [168].	208
D.9	The forecast error (nRMSE) of models in the evaluation on synthetic mixed-oriented PV plants. AutoPV's ensemble weight optimization $\hat{\mathbf{w}}$ is compared to AutoPV using the actual weights \mathbf{w} of each plant (prior knowledge), which reflect the mounting configurations and shares used to generate the synthetic data. The table is based on [168].	209
D.10	The estimated SunDance parameters for single- and mixed-oriented PV plants based on weather measurement data. The table is based on [168]. . .	210
D.11	The modeling errors (nMAE and nRMSE) of SunDance and AutoPV-default on single- and mixed-oriented PV plants. The table is based on [168]. . . .	210
D.12	The configuration space Λ_p for estimating the coordinates in decimal degrees of the PV plant using HPO.	214
D.13	The probabilistic forecast error (CRPS) of AutoPV-default compared to the forecasting methods considered by AutoPQ evaluated on the test data set of the PV data set [241]. For all methods, we apply the normalizing flow-based cINN [107, 115] to generate a probabilistic forecast based on the respective point forecast and use their default configurations (Table D.1, B.1, B.2, and B.3).	215

List of supervised student theses

The following chronological list contains all student theses supervised by the author:

- [367] R. Grewal, *AutoML: Automated machine learning for short term electrical load forecasting*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2021
- [360] E. Klumpp, *Data-driven building modelling: A comparison of different grey-box models, identified with real-world data*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2021
- [159] T. Martin, *Redispatch 2.0: Prognose der Einspeisemenge erneuerbarer Energien mithilfe von Machine Learning Ansätzen*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2021
- [337] E. Kampmann, *Optimized charging of an electric vehicle considering a battery energy storage system and a photovoltaic system*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2022
- [368] N. Jung, *Development and implementation of a cost-sensitive adaptive thermal building model using incremental and ensemble learning*, master's thesis, Karlsruhe Institute of Technology, Germany, 2023
- [339] A. Wohnig, *Estimation of PV system size, tilt and azimuth from power time series: Machine learning and optimisation approaches*, master's thesis, Karlsruhe Institute of Technology, Germany, 2023
- [369] J. M. Pinter, *Anomaly detection in end-of-line testing at ZF Wind Power*, master's thesis, Karlsruhe Institute of Technology, Germany, 2023
- [322] O. Schwärmer and F. Cevik, "Implementation and testing of the gradient descent algorithm and Kalman filter for the day ahead PV-forecasting model AutoPV", Karlsruhe Institute of Technology, Karlsruhe, Germany, Smart Energy System Lab (SESL) internship report, 2023
- [158] M. Dado, *Analyse des Potenzials von Physik-informierter Künstlicher Intelligenz zur Windleistungsvorhersage*, master's thesis, Karlsruhe Institute of Technology, Germany, 2023
- [98] J. B. Kolar, *Time series forecasting of the differential balancing group of Stadtwerke Karlsruhe Netzservice GmbH*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2024

- [359] M. Kopp, *Comparison of online adaptation methods for probabilistic time series forecasting*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2024

List of own publications

The following, chronologically ordered list presents all publications to which the author of this thesis contributed as first author:

- [162] S. Meisenbacher, J. Pinter, T. Martin, V. Hagenmeyer, and R. Mikut, “Concepts for automated machine learning in smart grid applications”, in *Proceedings of the 31. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2021, pp. 11–35
- [349] S. Meisenbacher, K. Schwenk, J. Galenzowski, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “A lightweight user interface for smart charging of electric vehicles: A real-world application”, in *2021 9th International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*, Virtual Event, 2021, pp. 57–61
- [350] S. Meisenbacher, K. Schwenk, J. Galenzowski, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “Smart charging of electric vehicles with cloud-based optimization and a lightweight user interface: A real-world application in the Energy Lab 2.0”, in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, ser. e-Energy ’21, Virtual Event: Association for Computing Machinery, 2021, pp. 284–285
- [1] S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut, “Review of automated time series forecasting pipelines”, *WIREs Data Mining and Knowledge Discovery*, vol. 12, no. 6, e1475, 2022
- [168] 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*, ser. e-Energy ’23, Orlando, USA: Association for Computing Machinery, 2023, pp. 386–414
- [169] 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, pp. 1–8

- [163] S. Meisenbacher, J. Galenzowski, K. Förderer, W. Suess, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “Automation level taxonomy for time series forecasting services: Guideline for real-world smart grid applications”, in *Energy Informatics*, B. N. Jørgensen, Z. G. Ma, F. D. Wijaya, R. Irnawan, and S. Sarjiya, Eds., Cham, Switzerland: Springer Nature Switzerland, 2025, pp. 277–297
- [167] S. Meisenbacher *et al.*, “AutoWP: Automated wind power forecasts with limited computing resources using an ensemble of diverse wind power curves”, in *Proceedings of the 34. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2024, pp. 1–20
- [166] S. Meisenbacher *et al.*, *On autoregressive deep learning models for day-ahead wind power forecasts with irregular shutdowns due to redispatching*, 2024. arXiv: 2412.00423
- [165] S. Meisenbacher, K. Phipps, O. Taubert, M. Weiel, M. Götz, R. Mikut, and V. Hagenmeyer, “AutoPQ: Automating quantile estimation from point forecasts in the context of sustainability”, *Applied Energy*, vol. 392, p. 125 931, 2025

The following, chronologically ordered list presents all further publications to which the author of this thesis contributed:

- [351] K. Schwenk, S. Meisenbacher, B. Briegel, T. Harr, V. Hagenmeyer, and R. Mikut, “Integrating battery aging in the optimization for bidirectional charging of electric vehicles”, *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5135–5145, 2021
- [171] B. Heidrich *et al.*, “pyWATTS: Python workflow automation tool for time series”, 2021. arXiv: 2106.10157
- [361] M. Frahm, E. Klumpp, S. Meisenbacher, J. Matthes, R. Mikut, and V. Hagenmeyer, “Development and validation of grey-box multi-zone thermal building models”, in *Proceedings of BauSim Conference 2022: 9th Conference of IBPSA-Germany and Austria*, ser. BauSim Conference, vol. 9, Weimar, Germany: IBPSA-Germany and Austria, 2022
- [362] M. Frahm, S. Meisenbacher, E. Klumpp, R. Mikut, J. Matthes, and V. Hagenmeyer, “Multi-zone grey-box thermal building identification with real occupants”, in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys ’22, Boston, USA: Association for Computing Machinery, 2022, pp. 484–487

- [164] K. Phipps, S. Meisenbacher, B. Heidrich, M. Turowski, R. Mikut, and V. Hagenmeyer, “Loss-customised probabilistic energy time series forecasts using automated hyperparameter optimisation”, in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy '23, Orlando, USA: Association for Computing Machinery, 2023, pp. 271–286
- [353] J. Galenzowski, S. Waczowicz, S. Meisenbacher, R. Mikut, and V. Hagenmeyer, “A real-world district community platform as a cyber-physical-social infrastructure systems in the energy domain”, in *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '23, New York, USA: Association for Computing Machinery, 2023, pp. 434–441

This work is based on the above publications and contains identical phrases. A list showing which publication has contributed to which chapter is given below:

- [1] Chapter 1, Chapter 2
- [162] Chapter 2
- [164] Chapter 3
- [165] Chapter 3
- [166] Chapter 4
- [163] Chapter 4
- [168] Chapter 5
- [169] Chapter 5
- [349] Chapter 6
- [350] Chapter 6

The author's Curriculum Vitae

Stefan Meisenbacher

born 10th of May 1992 in Stuttgart, Germany

Education

- | | |
|-------------------|---|
| 09/2020 – 09/2024 | Doctoral Program Mechanical Engineering
Karlsruhe Institute of Technology
<i>Karlsruhe, Germany</i> |
| 03/2017 – 04/2020 | Master's Program General Mechanical Engineering
Karlsruhe Institute of Technology
<i>Karlsruhe, Germany</i> |
| 10/2013 – 02/2017 | Bachelor's Program Mechanical Engineering
Esslingen University of Applied Sciences
<i>Esslingen, Germany</i> |
| 09/2012 – 07/2013 | University Entrance Qualification (FHR)
Gottlieb-Daimler-Schule 2
<i>Sindelfingen, Germany</i> |
| 09/2008 – 06/2011 | Apprenticeship as an Industrial Mechanic
Robert Bosch Power Tools GmbH
<i>Leinfelden-Echterdingen, Germany</i> |
| 07/2008 | Intermediate Secondary School
Johann-Bruecker-Realschule
<i>Schönaich, Germany</i> |

Working experience

09/2024 – now	Specialist Integration of Distributed Flexibility TransnetBW GmbH <i>Stuttgart, Germany</i>
09/2020 – 08/2024	Doctoral Researcher “Automated Machine Learning for Time Series Forecasting in Smart Grid Applications” Karlsruhe Institute of Technology <i>Karlsruhe, Germany</i>
03/2023	Visiting Researcher Drift Detection in Energy Time Series University of Birmingham <i>Birmingham, UK</i>
10/2019 – 04/2020	Master Thesis Student “Bi-directional Charging Optimization for Electric Vehicles considering current Energy Prices and Battery Aging” Mercedes-Benz AG <i>Sindelfingen, Germany</i>
04/2019 – 09/2019	Working Student Finite element analysis in machine dynamics TRUMPF SE + Co. KG <i>Ditzingen, Germany</i>
04/2019 – 09/2019	Engineering Intern Concept improvements of a new-gen drill driver gearbox Robert Bosch Power Tools Sdn Bhd <i>Penang, Malaysia</i>
04/2019 – 09/2019	Working Student Data mining on durability tests and design of experiments for production line release procedures Robert Bosch GmbH <i>Schwieberdingen, Germany</i>

- 08/2016 – 01/2017 **Bachelor Thesis Student**
 “Comparison and extension of design and release concepts for
 laser-welded components”
 Robert Bosch GmbH
Sindelfingen, Germany
- 08/2015 – 02/2016 **Engineering Intern**
 Development of an algorithm for buckling stability calculations
 Voith Hydro GmbH & Co. KG
Heidenheim, Germany
- 08/2015 – 02/2016 **Machine Operator**
 CNC milling machines
 Robert Bosch Power Tools GmbH
Leinfelden-Echterdingen, Germany

Bibliography

- [1] S. Meisenbacher, M. Turowski, K. Phipps, M. Rätz, D. Müller, V. Hagenmeyer, and R. Mikut, “Review of automated time series forecasting pipelines”, *WIREs Data Mining and Knowledge Discovery*, vol. 12, no. 6, e1475, 2022.
- [2] UNFCCC, “Paris Agreement”, in *United Nations Treaty Collection*, vol. XXVII 7. d, Paris, France, 2015.
- [3] Bundesministerium für Wirtschaft und Technologie (BMWi) and Bundesministerium für Umwelt, Naturschutz und Reaktorsicherheit (BMU), *Energiekonzept für eine umweltschonende, zuverlässige und bezahlbare Energieversorgung*, Berlin, Germany, 2010.
- [4] J. Markard, “The next phase of the energy transition and its implications for research and policy”, *Nature Energy*, vol. 3, no. 8, pp. 628–633, 2018.
- [5] Bundesnetzagentur (BNetzA), *Smart Grid und Smart Market: Eckpunktepapier der Bundesnetzagentur zu den Aspekten des sich verändernden Energieversorgungssystems*, Bonn, Germany, 2011.
- [6] X. Fang, S. Misra, G. Xue, and D. Yang, “Smart grid – The new and improved power grid: A survey”, *IEEE Communications Surveys & Tutorials*, vol. 14, no. 4, pp. 944–980, 2012.
- [7] G. Fridgen, R. Keller, M.-F. Körner, and M. Schöpf, “A holistic view on sector coupling”, *Energy Policy*, vol. 147, p. 111 913, 2020.
- [8] S. Waczowicz, “Konzept zur datengetriebenen Analyse und Modellierung des preisbeeinflussten Verbrauchsverhaltens”, Karlsruher Institut für Technologie, Germany, Ph.D. dissertation, 2018.
- [9] R. R. Appino, “Scheduling of energy storage using probabilistic forecasts and energy-based aggregated models”, Karlsruher Institut für Technologie, Germany, Ph.D. dissertation, 2019.
- [10] K. Schwenk, “A smart charging assistant for electric vehicles considering battery degradation, power grid and user constraints”, Karlsruher Institut für Technologie, Germany, Ph.D. dissertation, 2022.

- [11] P. Kohlhepp, H. Harb, H. Wolisz, S. Waczowicz, D. Müller, and V. Hagenmeyer, “Large-scale grid integration of residential thermal energy storages as demand-side flexibility resource: A review of international field studies”, *Renewable and Sustainable Energy Reviews*, vol. 101, pp. 527–547, 2019.
- [12] L. Dannecker, *Energy time series forecasting*. Wiesbaden, Germany: Springer Fachmedien, 2015.
- [13] S. Haben, M. Voss, and W. Holderbaum, *Core concepts and methods in load forecasting: With applications in distribution networks*. Cham, Switzerland: Springer International Publishing, 2023.
- [14] European Commission, *Factsheet: EU energy system integration strategy*, Brussels, Belgium, 2020.
- [15] Bundesministerium für Wirtschaft und Klimaschutz (BMWK) and Arbeitsgruppe Erneuerbare Energien-Statistik (AGEE-Stat), *Zeitreihen zur Entwicklung der erneuerbaren Energien in Deutschland*, Berlin, Germany, 2023.
- [16] Bundesministerium der Justiz (BMJ), *Gesetz für den Ausbau erneuerbarer Energien (Erneuerbare-Energien-Gesetz - EEG 2023)*, § 1 Ziel des Gesetzes, Berlin, Germany, 2023.
- [17] X. Wang and C. Wang, “Time series data cleaning: A survey”, *IEEE Access*, vol. 8, pp. 1866–1881, 2020.
- [18] K. Shaukat, T. M. Alam, S. Luo, S. Shabbir, I. A. Hameed, J. Li, S. K. Abbas, and U. Javed, “A review of time series anomaly detection techniques: A step to future perspectives”, in *Advances in Information and Communication*, Virtual Event, 2021, pp. 865–877.
- [19] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, “A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction”, *Journal of Applied Science and Technology Trends*, vol. 1, no. 2, pp. 56–70, 2020.
- [20] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: Methods, systems, challenges* (The Springer Series on Challenges in Machine Learning). Cham, Switzerland: Springer International Publishing, 2019.
- [21] M.-A. Zöller and M. F. Huber, “Benchmark and survey of automated machine learning frameworks”, *Journal of Artificial Intelligence Research*, vol. 70, pp. 409–472, 2021.

- [22] Z. Hajirahimi and M. Khashei, “Hybrid structures in time series modeling and forecasting: A review”, *Engineering Applications of Artificial Intelligence*, vol. 86, pp. 83–106, 2019.
- [23] D. Shaub, “Fast and accurate yearly time series forecasting with forecast combinations”, *International Journal of Forecasting*, vol. 36, no. 1, pp. 116–120, 2020.
- [24] D. Werling, M. Beichter, B. Heidrich, K. Phipps, R. Mikut, and V. Hagenmeyer, “The impact of forecast characteristics on the forecast value for the dispatchable feeder”, in *Companion Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy ’23 Companion, Orlando, USA: Association for Computing Machinery, 2023, pp. 59–71.
- [25] L. Tuggenier, M. Amirian, K. Rombach, S. Lorwald, A. Varlet, C. Westermann, and T. Stadelmann, “Automated machine learning in practice: State of the art and recent results”, in *2019 6th Swiss Conference on Data Science (SDS)*, Bern, Switzerland, 2019, pp. 31–36.
- [26] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting* (Springer Texts in Statistics), Third edition. Cham, Switzerland: Springer International Publishing, 2016.
- [27] J. Á. González Ordiano, S. Waczowicz, V. Hagenmeyer, and R. Mikut, “Energy forecasting tools and services”, *WIREs Data Mining and Knowledge Discovery*, vol. 8, no. 2, e1235, 2018.
- [28] R. J. Hyndman and G. Athanasopoulos, *Forecasting: principles and practice*, Third edition. Melbourne, Australia: OTexts, 2021.
- [29] J. G. De Gooijer and R. J. Hyndman, “25 years of time series forecasting”, *International Journal of Forecasting*, vol. 22, no. 3, pp. 443–473, 2006.
- [30] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: Forecasting and control* (Wiley Series in Probability and Statistics), Fifth edition. Hoboken, USA: Wiley, 2016.
- [31] C. Cheng, A. Sa-Ngasoongsong, O. Beyca, T. Le, H. Yang, Z. Kong, and S. T. Bukkapatnam, “Time series forecasting for nonlinear and non-stationary processes: A review and comparative study”, *IIE Transactions*, vol. 47, no. 10, pp. 1053–1071, 2015.
- [32] R. J. Hyndman, A. B. Koehler, R. D. Snyder, and S. Grose, “A state space framework for automatic forecasting using exponential smoothing methods”, *International Journal of Forecasting*, vol. 18, no. 3, pp. 439–454, 2002.

- [33] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, “A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition”, *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012.
- [34] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, “Statistical and machine learning forecasting methods: Concerns and ways forward”, *PLOS ONE*, vol. 13, no. 3, pp. 1–26, 2018.
- [35] 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, pp. 1–8.
- [36] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [37] S. Hochreiter, “The vanishing gradient problem during learning recurrent neural nets and problem solutions”, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, 1998.
- [38] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
- [39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need”, in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30, Curran Associates, Inc., 2017.
- [40] M. Hertel, S. Ott, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer, *Transformer neural networks for building load forecasting*, Poster: Tackling Climate Change with Machine Learning Workshop at NeurIPS 2022, 2022.
- [41] M. Hertel, M. Beichter, B. Heidrich, O. Neumann, B. Schäfer, R. Mikut, and V. Hagenmeyer, “Transformer training strategies for forecasting multiple load time series”, *Energy Informatics*, vol. 6, no. 1, p. 20, 2023.
- [42] 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*, ser. e-Energy ’20, Virtual Event: Association for Computing Machinery, 2020, pp. 220–230.
- [43] C. Challú, K. G. Olivares, B. N. Oreshkin, F. Garza Ramirez, M. Mergenthaler Canseco, and A. Dubrawski, “N-HiTS: Neural hierarchical interpolation for time series forecasting”, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2023, pp. 6989–6997.

- [44] B. Lim, S. Ö. Arik, 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.
- [45] A. Garza and M. Mergenthaler-Canseco, *TimeGPT-1*, 2023. arXiv: 2310.03589.
- [46] A. J. Cannon, “Quantile regression neural networks: Implementation in R and application to precipitation downscaling”, *Computers & Geosciences*, vol. 37, no. 9, pp. 1277–1284, 2011.
- [47] J. Á. González 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.
- [48] 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.
- [49] M. Arpogaus, M. Voss, B. Sick, M. Nigge-Uricher, and O. Dürr, “Short-term density forecasting of low-voltage load using Bernstein-polynomial normalizing flows”, *IEEE Transactions on Smart Grid*, vol. 14, no. 6, pp. 4902–4911, 2023.
- [50] P. Pinson and H. Madsen, “Ensemble-based probabilistic forecasting at Horns Rev”, *Wind Energy*, vol. 12, no. 2, pp. 137–155, 2009.
- [51] T. Gneiting and A. E. Raftery, “Weather forecasting with ensemble methods”, *Science*, vol. 310, no. 5746, pp. 248–249, 2005.
- [52] J. Bröcker and L. A. Smith, “From ensemble forecasts to predictive distribution functions”, *Tellus A: Dynamic Meteorology and Oceanography*, vol. 60, no. 4, pp. 663–678, 2008.
- [53] S. Rasp and S. Lerch, “Neural networks for postprocessing ensemble weather forecasts”, *Monthly Weather Review*, vol. 146, no. 11, pp. 3885–3900, 2018.
- [54] K. Phipps, S. Lerch, M. Andersson, R. Mikut, V. Hagenmeyer, and N. Ludwig, “Evaluating ensemble post-processing for wind power forecasts”, *Wind Energy*, vol. 25, no. 8, pp. 1379–1405, 2022.
- [55] L. Buzna, P. De Falco, G. Ferruzzi, S. Khormali, D. Proto, N. Refa, M. Straka, and G. van der Poel, “An ensemble methodology for hierarchical probabilistic electric vehicle load forecasting at regular charging stations”, *Applied Energy*, vol. 283, p. 116337, 2021.
- [56] Z. Han, J. Zhao, H. Leung, K. F. Ma, and W. Wang, “A review of deep learning models for time series prediction”, *IEEE Sensors Journal*, vol. 21, no. 6, pp. 7833–7848, 2019.

- [57] S. Liu, S. Gu, and T. Bao, “An automatic forecasting method for time series”, *Chinese Journal of Electronics*, vol. 26, no. 3, pp. 445–452, 2017.
- [58] W. Yan, “Toward automatic time series forecasting using neural networks”, *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1028–1039, 2012.
- [59] F. Martínez, M. P. Frías, M. D. Pérez, and A. J. Rivera, “A methodology for applying k-nearest neighbor to time series forecasting”, *Artificial Intelligence Review*, vol. 52, no. 3, pp. 2019–2037, 2019.
- [60] A. Widodo, I. Budi, and B. Widjaja, “Automatic lag selection in time series forecasting using multiple kernel learning”, *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 1, pp. 95–110, 2016.
- [61] S. Fan, X. Qin, Z. Jia, X. Qi, and M. Lin, “ELM-based improved layered ensemble architecture for time series forecasting”, *IEEE Access*, vol. 7, pp. 97 827–97 837, 2019.
- [62] A. Maravall, R. López-Pavón, and D. Pérez-Cañete, “Reliability of the automatic identification of ARIMA models in program TRAMO”, in *Empirical Economic and Financial Research*, ser. Advanced Studies in Theoretical and Applied Econometrics, J. Beran, Y. Feng, and H. Hebbel, Eds., vol. 48, Springer International Publishing, 2015, pp. 105–122.
- [63] M. Züfle and S. Kounev, “A framework for time series pre-processing and history-based forecasting method recommendation”, in *Proceedings of the 2020 Federated Conference on Computer Science and Information Systems*, Sofia, Bulgaria, 2020, pp. 141–144.
- [64] N. Tran and D. A. Reed, “Automatic ARIMA time series modeling for adaptive I/O prefetching”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 4, pp. 362–377, 2004.
- [65] M. Alzyout, M. Alsmirat, and M. I. Al-Saleh, “Automated ARIMA model construction for dynamic vehicle GPS location prediction”, in *2019 Sixth International Conference on Internet of Things: Systems, Management and Security (IOTSMS)*, Granada, Spain, 2019, pp. 380–386.
- [66] R. J. Hyndman and Y. Khandakar, “Automatic time series forecasting: The forecast package for R”, *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, 2008.
- [67] Y. Lu and S. M. AbouRizk, “Automated Box-Jenkins forecasting modelling”, *Automation in Construction*, vol. 18, no. 5, pp. 547–558, 2009.

- [68] S. Anvari, S. Tuna, M. Canci, and M. Turkay, “Automated Box-Jenkins forecasting tool with an application for passenger demand in urban rail systems”, *Journal of Advanced Transportation*, vol. 50, no. 1, pp. 25–49, 2016.
- [69] A. Amin, L. Grunske, and A. Colman, “An automated approach to forecasting QoS attributes based on linear and non-linear time series modeling”, in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, Essen, Germany, 2012, pp. 130–139.
- [70] R. Fildes and F. Petropoulos, “Simple versus complex selection rules for forecasting many time series”, *Journal of Business Research*, vol. 68, no. 8, pp. 1692–1701, 2015.
- [71] N. C. Sekma, A. Elleuch, and N. Dridi, “Automated forecasting approach minimizing prediction errors of CPU availability in distributed computing systems”, *International Journal of Intelligent Systems and Applications*, vol. 8, no. 9, pp. 8–21, 2016.
- [72] A. Bauer, M. Züfle, J. Grohmann, N. Schmitt, N. Herbst, and S. Kounev, “An automated forecasting framework based on method recommendation for seasonal time series”, in *Proceedings of the ACM/SPEC International Conference on Performance Engineering*, Edmonton, Canada, 2020, pp. 48–55.
- [73] S. F. Crone and N. Kourentzes, “Feature selection for time series prediction: A combined filter and wrapper approach for neural networks”, *Neurocomputing*, vol. 73, no. 10, pp. 1923–1936, 2010.
- [74] N. Kourentzes and S. F. Crone, “Frequency independent automatic input variable selection for neural networks for forecasting”, in *The 2010 International Joint Conference on Neural Networks (IJCNN)*, Barcelona, Spain, 2010, pp. 1–8.
- [75] K. Bandara, C. Bergmeir, and S. Smyl, “Forecasting across time series databases using recurrent neural networks on groups of similar series: A clustering approach”, *Expert Systems with Applications*, vol. 140, p. 112 896, 2020.
- [76] Y. R. Sagaert, E.-H. Aghezzaf, N. Kourentzes, and B. Desmet, “Tactical sales forecasting using a very large set of macroeconomic indicators”, *European Journal of Operational Research*, vol. 264, no. 2, pp. 558–569, 2018.
- [77] S. Maldonado, A. González, and S. Crone, “Automatic time series analysis for electric load forecasting via support vector regression”, *Applied Soft Computing*, vol. 83, p. 105 616, 2019.
- [78] M. Rätz, A. P. Javadi, M. Baranski, K. Finkbeiner, and D. Müller, “Automated data-driven modeling of building energy systems via machine learning algorithms”, *Energy and Buildings*, vol. 202, p. 109 384, 2019.

- [79] J. P. Donate and P. Cortez, “Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting”, *Applied Soft Computing*, vol. 23, pp. 432–443, 2014.
- [80] J. P. Donate, X. Li, G. G. Sánchez, and A. S. de Miguel, “Time series forecasting by evolving artificial neural networks with genetic algorithms, differential evolution and estimation of distribution algorithm”, *Neural Computing and Applications*, vol. 22, no. 1, pp. 11–20, 2013.
- [81] A. T. Sergio, T. P. de Lima, and T. B. Ludermir, “Dynamic selection of forecast combiners”, *Neurocomputing*, vol. 218, no. C, pp. 37–50, 2016.
- [82] F. Martínez, F. Charte, A. J. Rivera, and M. P. Frías, “Automatic time series forecasting with GRNN: A comparison with other models”, in *Advances in Computational Intelligence*, ser. Lecture Notes in Computer Science, I. Rojas, G. Joya, and A. Catala, Eds., vol. 11506, Springer International Publishing, 2019, pp. 198–209.
- [83] S. Panigrahi and H. S. Behera, “Time series forecasting using differential evolution-based ANN modelling scheme”, *Arabian Journal for Science and Engineering*, vol. 45, no. 12, pp. 11 129–11 146, 2020.
- [84] S. Ma and R. Fildes, “Retail sales forecasting with meta-learning”, *European Journal of Operational Research*, vol. 288, no. 1, pp. 111–128, 2021.
- [85] M. Züfle, A. Bauer, V. Lesch, C. Krupitzer, N. Herbst, S. Kounev, and V. Curtef, “Autonomic forecasting method selection: Examination and ways ahead”, in *2019 IEEE International Conference on Autonomic Computing (ICAC)*, Umeå, Sweden, 2019, pp. 167–176.
- [86] A. Widodo and I. Budi, “Feature enhancement for model selection in time series forecasting”, in *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, Bali, Indonesia, 2013, pp. 367–373.
- [87] G. Dellino, T. Laudadio, R. Mari, N. Mastronardi, and C. Meloni, “Microforecasting methods for fresh food supply chain management: A computational study”, *Mathematics and Computers in Simulation*, vol. 147, pp. 100–120, 2018.
- [88] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey”, *ACM Computing Surveys*, vol. 41, no. 3, 2009.
- [89] M. Turowski, O. Neumann, L. Mannsperger, K. Kraus, K. Layer, R. Mikut, and V. Hagenmeyer, “Managing anomalies in energy time series for automated forecasting”, in *Energy Informatics*, B. N. Jørgensen, L. C. P. da Silva, and Z. Ma, Eds., Cham, Switzerland: Springer Nature Switzerland, 2024, pp. 3–29.

- [90] A. Bauer, M. Züfle, N. Herbst, S. Kounev, and V. Curtef, “Telescope: An automatic feature extraction and transformation approach for time series forecasting on a level-playing field”, in *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, Dallas, USA, 2020, pp. 1902–1905.
- [91] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, “Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?”, *Journal of Econometrics*, vol. 54, no. 1, pp. 159–178, 1992.
- [92] D. R. Cox and A. Stuart, “Some quick sign tests for trend in location and dispersion”, *Biometrika*, vol. 42, no. 1/2, pp. 80–95, 1955.
- [93] Y.-W. Cheung and K. S. Lai, “Lag order and critical values of the Augmented Dickey-Fuller test”, *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 277–280, 1995.
- [94] F. Canova and B. E. Hansen, “Are seasonal patterns constant over time? A test for seasonal stability”, *Journal of Business & Economic Statistics*, vol. 13, no. 3, pp. 237–252, 1995.
- [95] D. R. Osborn, A. P. L. Chui, J. P. Smith, and C. R. Birchenhall, “Seasonality and the order of integration for consumption”, *Oxford Bulletin of Economics and Statistics*, vol. 50, no. 4, pp. 361–377, 1988.
- [96] H. W. Lilliefors, “On the Kolmogorov-Smirnov test for normality with mean and variance unknown”, *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.
- [97] G. E. P. Box and D. R. Cox, “An analysis of transformations”, *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 26, no. 2, pp. 211–243, 1964.
- [98] J. B. Kolar, *Time series forecasting of the differential balancing group of Stadtwerke Karlsruhe Netzservice GmbH*, bachelor’s thesis, Karlsruhe Institute of Technology, Germany, 2024.
- [99] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, “STL: A seasonal-trend decomposition procedure based on LOESS”, *Journal of Official Statistics*, vol. 6, no. 1, pp. 3–73, 1990.
- [100] F. Pedregosa *et al.*, “Scikit-learn: Machine learning in Python”, *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.
- [101] D. Chakrabarti and C. Faloutsos, “F4: Large-scale automated forecasting using fractals”, in *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, McLean, USA, 2002, pp. 2–9.

- [102] S. D. Balkin and J. Ord, “Automatic neural network modeling for univariate time series”, *International Journal of Forecasting*, vol. 16, no. 4, pp. 509–515, 2000.
- [103] A. P. Lowther, P. Fearnhead, M. A. Nunes, and K. Jensen, “Semi-automated simultaneous predictor selection for regression-sARIMA models”, *Statistics and Computing*, vol. 30, no. 6, pp. 1759–1778, 2020.
- [104] H. Son and C. Kim, “Forecasting short-term electricity demand in residential sector based on support vector regression and fuzzy-rough feature selection with particle swarm optimization”, *Procedia Engineering*, vol. 118, pp. 1162–1168, 2015.
- [105] J. M. Valente and S. Maldonado, “SVR-FFS: A novel forward feature selection approach for high-frequency time series forecasting using support vector regression”, *Expert Systems with Applications*, vol. 160, p. 113 729, 2020.
- [106] G. Dellino, T. Laudadio, R. Mari, N. Mastronardi, and C. Meloni, “A reliable decision support system for fresh food supply chain management”, *International Journal of Production Research*, vol. 56, no. 4, pp. 1458–1485, 2018.
- [107] B. Heidrich, M. Turowski, K. Phipps, K. Schmieder, W. Süß, R. Mikut, and V. Hagenmeyer, “Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks”, *Applied Intelligence*, vol. 53, no. 8, pp. 8826–8843, 2023.
- [108] V. Cerqueira, N. Moniz, and C. Soares, “VEST: Automatic feature engineering for forecasting”, *Machine Learning*, 2021.
- [109] A. Jović, K. Brkić, and N. Bogunović, “A review of feature selection methods with applications”, in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, Opatija, Croatia, 2015, pp. 1200–1205.
- [110] D. Bertsimas, A. King, and R. Mazumder, “Best subset selection via a modern optimization lens”, *The Annals of Statistics*, vol. 44, no. 2, pp. 813–852, 2016.
- [111] K. Yang and C. Shahabi, “On the stationarity of multivariate time series for correlation-based data analysis”, in *Fifth IEEE International Conference on Data Mining (ICDM '05)*, Houston, USA, 2005, pp. 805–808.
- [112] M. Feurer and F. Hutter, “Hyperparameter optimization”, in *Automated Machine Learning*, ser. The Springer Series on Challenges in Machine Learning, F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., Cham, Switzerland: Springer International Publishing, 2019, pp. 3–33.
- [113] T. Gneiting and M. Katzfuss, “Probabilistic forecasting”, *Annual Review of Statistics and Its Application*, vol. 1, no. 1, pp. 125–151, 2014.

- [114] T. Gneiting, F. Balabdaoui, and A. E. Raftery, “Probabilistic forecasts, calibration and sharpness”, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 69, no. 2, pp. 243–268, 2007.
- [115] K. Phipps, B. Heidrich, M. Turowski, M. Wittig, R. Mikut, and V. Hagenmeyer, “Generating probabilistic forecasts from arbitrary point forecasts using a conditional invertible neural network”, *Applied Intelligence*, vol. 54, no. 8, pp. 6354–6382, 2024.
- [116] I. Svetunkov and F. Petropoulos, “Old dog, new tricks: A modelling view of simple moving averages”, *International Journal of Production Research*, vol. 56, no. 18, pp. 6034–6047, 2018.
- [117] D. J. Pedregal, “Time series analysis and forecasting with ECOTOOL”, *PLOS ONE*, vol. 14, no. 10, pp. 1–23, 2019.
- [118] S. Hwang, M. Park, H.-S. Lee, and H. Kim, “Automated time series cost forecasting system for construction materials”, *Journal of Construction Engineering and Management*, vol. 138, no. 11, pp. 1259–1269, 2012.
- [119] J. Arlt and P. Trcka, “Automatic sARIMA modeling and forecast accuracy”, *Communications in Statistics - Simulation and Computation*, vol. 50, no. 10, pp. 2949–2970, 2021.
- [120] J. D. Bermúdez, J. V. Segura, and E. Vercher, “SIOPRED performance in a forecasting blind competition”, in *2012 IEEE Conference on Evolving and Adaptive Intelligent Systems*, Madrid, Spain, 2012, pp. 192–197.
- [121] E. Spiliotis, V. Assimakopoulos, and S. Makridakis, “Generalizing the Theta method for automatic forecasting”, *European Journal of Operational Research*, vol. 284, no. 2, pp. 550–558, 2020.
- [122] T. M. Fischer, F. Bauer, S. A. Selzer, and P. Bretschneider, “Genetische Algorithmen zur Hyperparameteroptimierung künstlicher neuronaler Netze für die Energiezeitreihenprognose”, in *Proceedings of the 31. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2021, pp. 11–35.
- [123] J. Bergstra and Y. Bengio, “Random search for hyperparameter optimization”, *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.
- [124] M. S. Bazaraa, H. D. Sherali, and S. C. M., *Nonlinear programming: Theory and algorithms*, Third edition. Hoboken, USA: John Wiley & Sons, 2006.
- [125] J. Pearl, *Heuristics: Intelligent search strategies for computer problem solving*, First edition. Boston, USA: Addison Wesley Publishing Company, 1984.

- [126] T. Stützle, “Local search algorithms for combinatorial problems: Analysis, improvements, and new applications”, Darmstadt University of Technology, Germany, Ph.D. dissertation, 1999.
- [127] T. Bäck, *Evolutionary algorithms in theory and practice: Evolution strategies, evolutionary programming, genetic algorithms*, First edition. Oxford, UK: Oxford university press, 1996.
- [128] X. Cui, W. Zhang, Z. Tüske, and M. Picheny, “Evolutionary stochastic gradient descent for optimization of deep neural networks”, in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31, Curran Associates, Inc., 2018.
- [129] M. Shcherbakov, V. Kamaev, and N. Shcherbakova, “Automated electric energy consumption forecasting system based on decision tree approach”, *IFAC Proceedings Volumes*, vol. 46, no. 9, pp. 1027–1032, 2013.
- [130] M. Pawlikowski and A. Chorowska, “Weighted ensemble of statistical models”, *International Journal of Forecasting*, vol. 36, no. 1, pp. 93–97, 2020.
- [131] P. Pereira, J. Araujo, R. Matos, N. Pregoica, and P. Maciel, “Software rejuvenation in computer systems: An automatic forecasting approach based on time series”, in *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, Orlando, USA, 2018, pp. 1–8.
- [132] N. Kourentzes, D. Barrow, and F. Petropoulos, “Another look at forecast selection and combination: Evidence from forecast pooling”, *International Journal of Production Economics*, vol. 209, pp. 226–235, 2019.
- [133] J. Shetty and G. Shobha, “An ensemble of automatic algorithms for forecasting resource utilization in cloud”, in *2016 Future Technologies Conference (FTC)*, San Francisco, USA, 2016, pp. 301–306.
- [134] Z. Wu, X. Xia, L. Xiao, and Y. Liu, “Combined model with secondary decomposition-model selection and sample selection for multi-step wind power forecasting”, *Applied Energy*, vol. 261, p. 114 345, 2020.
- [135] S. Taghiyeh, D. C. Lengacher, and R. B. Handfield, “Forecasting model selection using intermediate classification: Application to MonarchFx corporation”, *Expert Systems with Applications*, vol. 151, p. 113 371, 2020.
- [136] M. Kück, S. F. Crone, and M. Freitag, “Meta-learning with neural networks and landmarking for forecasting model selection an empirical evaluation of different feature sets applied to industry data”, in *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, Canada, 2016, pp. 1499–1506.

- [137] B. Scholz-Reiter, M. Kück, and D. Lappe, “Prediction of customer demands for production planning: Automated selection and configuration of suitable prediction methods”, *CIRP Annals*, vol. 63, no. 1, pp. 417–420, 2014.
- [138] K. E. Baddour, A. Ghasemi, and H. Rutagemwa, “Spectrum occupancy prediction for land mobile radio bands using a recommender system”, in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, Chicago, USA, 2018, pp. 1–6.
- [139] S. Shahoud, H. Khalloof, C. Duepmeier, and V. Hagenmeyer, “Incorporating unsupervised deep learning into meta learning for energy time series forecasting”, in *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*, vol. 1288, Vancouver, Canada, 2020, pp. 326–345.
- [140] C. Cui, T. Wu, M. Hu, J. D. Weir, and X. Li, “Short-term building energy model recommendation system: A meta-learning approach”, *Applied Energy*, vol. 172, pp. 251–263, 2016.
- [141] P. Montero-Manso, G. Athanasopoulos, R. J. Hyndman, and T. S. Talagala, “FFORMA: Feature-based forecast model averaging”, *International Journal of Forecasting*, vol. 36, no. 1, pp. 86–92, 2020.
- [142] X. Li, Y. Kang, and F. Li, “Forecasting with time series imaging”, *Expert Systems with Applications*, vol. 160, p. 113 680, 2020.
- [143] M. A. Villegas, D. J. Pedregal, and J. R. Trapero, “A support vector machine for model selection in demand forecasting applications”, *Computers & Industrial Engineering*, vol. 121, pp. 1–7, 2018.
- [144] X. Wang, K. Smith-Miles, and R. J. Hyndman, “Rule induction for forecasting method selection: Meta-learning the characteristics of univariate time series”, *Neurocomputing*, vol. 72, no. 10, pp. 2581–2594, 2009.
- [145] S. Makridakis and M. Hibon, “The M3 Competition: Results, conclusions and implications”, *International Journal of Forecasting*, vol. 16, no. 4, pp. 451–476, 2000.
- [146] 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.
- [147] S. Iyengar, S. Lee, D. Sheldon, and P. Shenoy, “SolarClique: Detecting anomalies in residential solar arrays”, in *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, ser. COMPASS ’18, Menlo Park and San Jose, USA: Association for Computing Machinery, 2018.

- [148] S. Iyengar, N. Sharma, D. Irwin, P. Shenoy, and K. Ramamritham, “SolarCast: A cloud-based black box solar predictor for smart homes”, in *Proceedings of the 1st ACM Conference on Embedded Systems for Energy-Efficient Buildings*, ser. BuildSys ’14, Memphis, USA: Association for Computing Machinery, 2014, pp. 40–49.
- [149] S. Iyengar, N. Sharma, D. Irwin, P. Shenoy, and K. Ramamritham, “A cloud-based black-box solar predictor for smart homes”, *ACM Trans. Cyber-Phys. Syst.*, vol. 1, no. 4, 2017.
- [150] D. L. King, J. A. Kratochvil, and W. E. Boyson, “Photovoltaic array performance model”, Sandia National Laboratories (SNL), Albuquerque, NM (USA), Tech. Rep. 3535, 2004, pp. 1–43.
- [151] Y. Wang, Q. Hu, D. Srinivasan, and Z. Wang, “Wind power curve modeling and wind power forecasting with inconsistent data”, *IEEE Transactions on Sustainable Energy*, vol. 10, no. 1, pp. 16–25, 2019.
- [152] M. Schlechtingen, I. F. Santos, and S. Achiche, “Using data-mining approaches for wind turbine power curve monitoring: A comparative study”, *IEEE Transactions on Sustainable Energy*, vol. 4, no. 3, pp. 671–679, 2013.
- [153] U. Singh, M. Rizwan, M. Alaraj, and I. Alsaïdan, “A machine learning-based gradient boosting regression approach for wind power production forecasting: A step towards smart grid environments”, *Energies*, vol. 14, no. 16, 2021.
- [154] M. Á. Rodríguez-López, E. Cerdá, and P. d. Rio, “Modeling wind-turbine power curves: Effects of environmental temperature on wind energy generation”, *Energies*, vol. 13, no. 18, 2020.
- [155] 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*, vol. n/a, no. n/a, 2024.
- [156] D. Chen and D. Irwin, “Black-box solar performance modeling: Comparing physical, machine learning, and hybrid approaches”, *SIGMETRICS Perform. Eval. Rev.*, vol. 45, no. 2, pp. 79–84, 2017.
- [157] D. Chen and D. Irwin, “SunDance: Black-box behind-the-meter solar disaggregation”, in *Proceedings of the Eighth International Conference on Future Energy Systems*, ser. e-Energy ’17, Shatin, Hong Kong: Association for Computing Machinery, 2017, pp. 45–55.
- [158] M. Dado, *Analyse des Potenzials von Physik-informierter Künstlicher Intelligenz zur Windleistungsvorhersage*, master’s thesis, Karlsruhe Institute of Technology, Germany, 2023.

- [159] T. Martin, *Redispatch 2.0: Prognose der Einspeisemenge erneuerbarer Energien mithilfe von Machine Learning Ansätzen*, bachelor's thesis, Karlsruhe Institute of Technology, Germany, 2021.
- [160] A. H. Gambier, *Control of large wind energy systems: Theory and methods for the user* (Advances in Industrial Control), First edition. Cham, Switzerland: Springer International Publishing, 2022.
- [161] S. Beichter *et al.*, "Towards a real-world dispatchable feeder", in *2023 8th IEEE Workshop on the Electronic Grid (eGRID)*, 2023, pp. 1–6.
- [162] S. Meisenbacher, J. Pinter, T. Martin, V. Hagenmeyer, and R. Mikut, "Concepts for automated machine learning in smart grid applications", in *Proceedings of the 31. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2021, pp. 11–35.
- [163] S. Meisenbacher, J. Galenzowski, K. Förderer, W. Suess, S. Waczowicz, R. Mikut, and V. Hagenmeyer, "Automation level taxonomy for time series forecasting services: Guideline for real-world smart grid applications", in *Energy Informatics*, B. N. Jørgensen, Z. G. Ma, F. D. Wijaya, R. Irnawan, and S. Sarjiya, Eds., Cham, Switzerland: Springer Nature Switzerland, 2025, pp. 277–297.
- [164] K. Phipps, S. Meisenbacher, B. Heidrich, M. Turowski, R. Mikut, and V. Hagenmeyer, "Loss-customised probabilistic energy time series forecasts using automated hyperparameter optimisation", in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy '23, Orlando, USA: Association for Computing Machinery, 2023, pp. 271–286.
- [165] S. Meisenbacher, K. Phipps, O. Taubert, M. Weiel, M. Götz, R. Mikut, and V. Hagenmeyer, "AutoPQ: Automating quantile estimation from point forecasts in the context of sustainability", *Applied Energy*, vol. 392, p. 125 931, 2025.
- [166] S. Meisenbacher *et al.*, *On autoregressive deep learning models for day-ahead wind power forecasts with irregular shutdowns due to redispatching*, 2024. arXiv: 2412.00423.
- [167] S. Meisenbacher *et al.*, "AutoWP: Automated wind power forecasts with limited computing resources using an ensemble of diverse wind power curves", in *Proceedings of the 34. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2024, pp. 1–20.

- [168] 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*, ser. e-Energy ’23, Orlando, USA: Association for Computing Machinery, 2023, pp. 386–414.
- [169] S. Meisenbacher, T. Martin, B. Heidrich, R. Mikut, and V. Hagenmeyer, “Automating day-ahead forecasting of photovoltaic power generation: Model design, monitoring, and adaptation”, in *ETG Congress 2023*, 2023, pp. 1–8.
- [170] Society of Automotive Engineers, *SAE J3016: Levels of driving automation*, 2021.
- [171] B. Heidrich *et al.*, “pyWATTS: Python workflow automation tool for time series”, 2021. arXiv: 2106.10157.
- [172] E. Giacomazzi, F. Haag, and K. Hopf, “Short-term electricity load forecasting using the Temporal Fusion Transformer: Effect of grid hierarchies and data sources”, in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, ser. e-Energy ’23, Orlando, USA: Association for Computing Machinery, 2023, pp. 353–360.
- [173] L. Richter, F. Bauer, S. Klaiber, and P. Bretschneider, “Day-ahead electricity load prediction based on calendar features and temporal convolutional networks”, in *Theory and Applications of Time Series Analysis and Forecasting*, O. Valenzuela, F. Rojas, L. J. Herrera, H. Pomares, and I. Rojas, Eds., Cham, Switzerland: Springer International Publishing, 2023, pp. 243–253.
- [174] S. A. Selzer, F. Bauer, S. Bohm, P. Bretschneider, and E. Runge, “Physik-geführte NARXnets (PGNARXnets) zur Zeitreihenvorhersage”, in *Proceedings of the 31. Workshop Computational Intelligence*, Berlin, Germany: KIT Scientific Publishing, 2021, pp. 235–261.
- [175] S. Chetty, H. Wang, and S. Goodwin, “Visualising the effect of COVID-19 on electricity consumption in Victoria, Australia”, in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, ser. e-Energy ’21, Virtual Event: Association for Computing Machinery, 2021, pp. 367–371.
- [176] H. Fanaee-T, *Bike sharing dataset*, UCI Machine Learning Repository, 2013.
- [177] N. K. Ahmed, A. F. Atiya, N. El Gayar, and H. El-Shishiny, “An empirical comparison of machine learning models for time series forecasting”, *Econometric Reviews*, vol. 29, no. 5–6, pp. 594–621, 2010.

- [178] R. P. Masini, M. C. Medeiros, and E. F. Mendes, “Machine learning advances for time series forecasting”, *Journal of Economic Surveys*, vol. 37, no. 1, pp. 76–111, 2023.
- [179] F. Chollet *et al.*, *Keras*, <https://keras.io>, 2015.
- [180] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library”, in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, USA: Curran Associates Inc., 2019.
- [181] O. Taubert, M. Weiel, D. Coquelin, A. Farshian, C. Debus, A. Schug, A. Streit, and M. Götz, “Massively parallel genetic optimization through asynchronous propagation of populations”, in *High Performance Computing*, A. Bhatele, J. Hammond, M. Baboulin, and C. Kruse, Eds., Cham, Switzerland: Springer Nature, 2023, pp. 106–124.
- [182] J. Bergstra, D. Yamins, and D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures”, in *Proceedings of the 30th International Conference on Machine Learning*, ser. ICML ’13, Proceedings of Machine Learning Research, PMLR, 2013, pp. 115–123.
- [183] B. Bischl *et al.*, “Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges”, *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 2, e1484, 2023.
- [184] Y. Song, J. Lu, H. Lu, and G. Zhang, “Fuzzy clustering-based adaptive regression for drifting data streams”, *IEEE Transactions on Fuzzy Systems*, vol. 28, no. 3, pp. 544–557, 2020.
- [185] M. Böhlend, W. Doneit, L. Gröll, R. Mikut, and M. Reischl, “Automated design process for hybrid regression modeling with a one-class SVM”, *at - Automatisierungstechnik*, vol. 67, no. 10, pp. 843–852, 2019.
- [186] S. Meisenbacher, *Bi-directional charging optimization for electric vehicles considering energy prices and battery aging*, master’s thesis, Karlsruhe Institute of Technology, Germany, 2020.
- [187] P. Virtanen *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in Python”, *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [188] S. A. Selzer, F. Bauer, S. Bohm, E. Runge, and P. Bretschneider, “Physics-guided machine learning techniques for improving temperature calculations of high-voltage transmission lines”, in *ETG Congress 2023*, 2023, pp. 1–8.

- [189] L. Richter, T. Bender, S. Lenk, and P. Bretschneider, “Generating synthetic electricity load time series at district scale using probabilistic forecasts”, *Energies*, vol. 17, no. 7, 2024.
- [190] B. Heidrich, L. Mannsperger, M. Turowski, K. Phipps, B. Schäfer, R. Mikut, and V. Hagenmeyer, “Boost short-term load forecasts with synthetic data from transferred latent space information”, *Energy Informatics*, vol. 5, no. 1, p. 20, 2022.
- [191] A. Kummerow, M. Dirbas, C. Monsalve, S. Nicolai, and P. Bretschneider, “Robust disturbance classification in power transmission systems with denoising recurrent autoencoders”, *Sustainable Energy, Grids and Networks*, vol. 32, p. 100 803, 2022.
- [192] A. Kummerow, M. Dirbas, C. Monsalve, and P. Bretschneider, “Siamese sigmoid networks for the open classification of grid disturbances in power transmission systems”, *IET Smart Grid*, vol. 6, no. 2, pp. 136–146, 2023.
- [193] A. Kummerow, M. Henneke, P. Bachmann, S. Krackruegge, J. Laessig, and S. Nicolai, “Cyber-security platform for the transparent cyber-attack detection in energy supply infrastructures”, in *ETG Congress 2023*, 2023, pp. 1–7.
- [194] A. Kummerow, K. Schaefer, C. Monsalve, M. Alramlawi, S. Nicolai, and P. Bretschneider, “A cyber-security testbed for the dynamic operation of transmission power systems”, in *ETG Congress 2023*, 2023, pp. 1–6.
- [195] F. Rippstein, S. Lenk, A. Kummerow, L. Richter, S. Klaiber, and P. Bretschneider, “Anomaly detection algorithm using a hybrid modelling approach for energy consumption time series”, in *Theory and Applications of Time Series Analysis and Forecasting*, O. Valenzuela, F. Rojas, L. J. Herrera, H. Pomares, and I. Rojas, Eds., Cham, Switzerland: Springer International Publishing, 2023, pp. 19–30.
- [196] M. Turowski, B. Heidrich, K. Phipps, K. Schmieder, O. Neumann, R. Mikut, and V. Hagenmeyer, “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*, ser. e-Energy ’22, Virtual Event: Association for Computing Machinery, 2022, pp. 208–227.
- [197] M. Weber, M. Turowski, H. K. Çakmak, R. Mikut, U. Kühnapfel, and V. Hagenmeyer, “Data-driven copy-paste imputation for energy time series”, *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5409–5419, 2021.
- [198] A. Bifet and R. Gavaldà, “Learning from time-changing data with adaptive windowing”, in *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pp. 443–448.

- [199] P. M. Grulich, R. Saitenmacher, J. Traub, S. Breß, T. Rabl, and V. Markl, “Scalable detection of concept drifts on data streams with parallel adaptive windowing.”, in *Proceedings of the 21st International Conference on Extending Database Technology (EDBT)*, 2018, pp. 477–480.
- [200] D. Wölfle, K. Förderer, T. Riedel, L. Landwich, R. Mikut, V. Hagenmeyer, and H. Schmeck, *Open Energy Services: Forecasting and optimization as a service for energy management applications at scale*, 2024. arXiv: 2402.15230.
- [201] K. Chauhan, S. Jani, D. Thakkar, R. Dave, J. Bhatia, S. Tanwar, and M. S. Obaidat, “Automated machine learning: The new wave of machine learning”, in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, 2020, pp. 205–212.
- [202] G. Mengi, S. K. Singh, S. Kumar, D. Mahto, and A. Sharma, “Automated machine learning (AutoML): The future of computational intelligence”, in *International Conference on Cyber Security, Privacy and Networking (ICSPN 2022)*, N. Nedjah, G. Martínez Pérez, and B. B. Gupta, Eds., Cham, Switzerland: Springer International Publishing, 2023, pp. 309–317.
- [203] S. K. Karmaker (“Santu”), M. M. Hassan, M. J. Smith, L. Xu, C. Zhai, and K. Veeramachaneni, “AutoML to date and beyond: Challenges and opportunities”, *ACM Comput. Surv.*, vol. 54, no. 8, 2021.
- [204] R. Zheng, L. Qu, B. Cui, Y. Shi, and H. Yin, “AutoML for deep recommender systems: A survey”, *ACM Trans. Inf. Syst.*, vol. 41, no. 4, 2023.
- [205] Q. Yao, M. Wang, Y. Chen, W. Dai, Y.-F. Li, W.-W. Tu, Q. Yang, and Y. Yu, *Taking human out of learning applications: A survey on automated machine learning*, 2019. arXiv: 1810.13306.
- [206] M. Bahri, F. Salutari, A. Putina, and M. Sozio, “AutoML: State of the art with a focus on anomaly detection, challenges, and research directions”, *International Journal of Data Science and Analytics*, vol. 14, no. 2, pp. 113–126, 2022.
- [207] R. Barbudo, S. Ventura, and J. R. Romero, “Eight years of AutoML: Categorisation, review and trends”, *Knowledge and Information Systems*, vol. 65, no. 12, pp. 5097–5149, 2023.
- [208] C. Wang, Z. Chen, and M. Zhou, “AutoML from software engineering perspective: Landscapes and challenges”, in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 2023, pp. 39–51.

- [209] J. Klaise, A. Van Looveren, C. Cox, G. Vacanti, and C. Alexandru, *Monitoring and explainability of models in production*, Abstract: Workshop on Challenges in Deploying and Monitoring Machine Learning Systems (ICML 2020), 2020. arXiv: 2007.06299.
- [210] M. Arnold, J. Boston, M. Desmond, E. Duesterwald, B. Elder, A. Murthi, J. Navratil, and D. Reimer, “Towards automating the AI operations lifecycle”, Poster: MLOps Workshop at MLSys 2020, 2020. arXiv: 2003.12808.
- [211] B. Eck, D. Kabakci-Zorlu, Y. Chen, F. Savard, and X. Bao, “A monitoring framework for deployed machine learning models with supply chain examples”, in *2022 IEEE International Conference on Big Data (Big Data)*, Los Alamitos, USA: IEEE Computer Society, 2022, pp. 2231–2238.
- [212] L. E. Lwakatare, A. Raj, J. Bosch, H. H. Olsson, and I. Crnkovic, “A taxonomy of software engineering challenges for machine learning systems: An empirical investigation”, in *Agile Processes in Software Engineering and Extreme Programming*, P. Kruchten, S. Fraser, and F. Coallier, Eds., Cham, Switzerland: Springer International Publishing, 2019, pp. 227–243.
- [213] L. E. Lwakatare, I. Crnkovic, E. Rånge, and J. Bosch, “From a data science driven process to a continuous delivery process for machine learning systems”, in *Product-Focused Software Process Improvement*, M. Morisio, M. Torchiano, and A. Jedlitschka, Eds., Cham, Switzerland: Springer International Publishing, 2020, pp. 185–201.
- [214] Google LLC. “MLOps: Continuous delivery and automation pipelines in machine learning”. (2023), [Online]. Available: <https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning> (visited on 11/10/2023).
- [215] Microsoft Corporation. “Machine learning operations maturity model”. (2023), [Online]. Available: <https://learn.microsoft.com/en-us/azure/architecture/ai-ml/guide/mlops-maturity-model> (visited on 11/10/2023).
- [216] D. Kreuzberger, N. Köhl, and S. Hirschl, “Machine learning operations (MLOps): Overview, definition, and architecture”, *IEEE Access*, vol. 11, pp. 31 866–31 879, 2023.
- [217] M. Testi, M. Ballabio, E. Frontoni, G. Iannello, S. Moccia, P. Soda, and G. Vessio, “MLOps: A taxonomy and a methodology”, *IEEE Access*, vol. 10, pp. 63 606–63 618, 2022.

- [218] M. Simmler and R. Frischknecht, “A taxonomy of human-machine collaboration: Capturing automation and technical autonomy”, *AI & SOCIETY*, vol. 36, no. 1, pp. 239–250, 2021.
- [219] M. Clark. “Bayesian basics: A hands-on example”. (2022), [Online]. Available: <https://m-clark.github.io/bayesian-basics/example.html> (visited on 06/06/2024).
- [220] H. Khajeh and H. Laaksonen, “Applications of probabilistic forecasting in smart grids: A review”, *Applied Sciences*, vol. 12, no. 4, 2022.
- [221] B. N. Oreshkin, D. Carпов, N. Chapados, and Y. Bengio, “N-BEATS: Neural basis expansion analysis for interpretable time series forecasting”, in *International Conference on Learning Representations*, 2020.
- [222] F. Petropoulos *et al.*, “Forecasting: Theory and practice”, *International Journal of Forecasting*, vol. 38, no. 3, pp. 705–871, 2022.
- [223] E. Camporeale, X. Chu, O. V. Agapitov, and J. Bortnik, “On the generation of probabilistic forecasts from deterministic models”, *Space Weather*, vol. 17, no. 3, pp. 455–475, 2019.
- [224] W. H. Williams and M. L. Goodman, “A simple method for the construction of empirical confidence limits for economic forecasts”, *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 752–754, 1971.
- [225] K. Stankeviciute, A. M. Alaa, and M. van der Schaar, “Conformal time series forecasting”, in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 6216–6228.
- [226] Y. Wang, G. Hug, Z. Liu, and N. Zhang, “Modeling load forecast uncertainty using generative adversarial networks”, *Electric Power Systems Research*, vol. 189, p. 106732, 2020.
- [227] S. Y. Shah *et al.*, “AutoAI-TS: AutoAI for time series forecasting”, in *Proceedings of the 2021 International Conference on Management of Data*, ser. SIGMOD ’21, Virtual Event: Association for Computing Machinery, 2021, pp. 2584–2596.
- [228] D. Deng, F. Karl, F. Hutter, B. Bischl, and M. Lindauer, “Efficient automated deep learning for time series forecasting”, in *Machine Learning and Knowledge Discovery in Databases*, M.-R. Amini, S. Canu, A. Fischer, T. Guns, P. Kralj Novak, and G. Tsoumakas, Eds., Cham, Switzerland: Springer Nature, 2023, pp. 664–680.

- [229] O. Shchur, A. C. Turkmen, N. Erickson, H. Shen, A. Shirkov, T. Hu, and B. Wang, “AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting”, in *International Conference on Automated Machine Learning*, PMLR, 2023, pp. 9–1.
- [230] P. Pinson, H. A. Nielsen, J. K. Møller, H. Madsen, and G. N. Kariniotakis, “Non-parametric probabilistic forecasts of wind power: Required properties and evaluation”, *Wind Energy*, vol. 10, no. 6, pp. 497–516, 2007.
- [231] C. Sweeney, R. J. Bessa, J. Browell, and P. Pinson, “The future of forecasting for renewable energy”, *WIREs Energy and Environment*, vol. 9, no. 2, e365, 2020.
- [232] G. M. Martin, R. Loaiza-Maya, W. Maneesoonthorn, D. T. Frazier, and A. Ramírez-Hassan, “Optimal probabilistic forecasts: When do they work?”, *International Journal of Forecasting*, vol. 38, no. 1, pp. 384–406, 2022.
- [233] K. Phipps, K. Schwenk, B. Briegel, R. Mikut, and V. Hagenmeyer, “Customized uncertainty quantification of parking duration predictions for EV smart charging”, *IEEE Internet of Things Journal*, vol. 10, no. 23, pp. 20 649–20 661, 2023.
- [234] Y. Chen, Y. Kang, Y. Chen, and Z. Wang, “Probabilistic forecasting with temporal convolutional neural network”, *Neurocomputing*, vol. 399, pp. 491–501, 2020.
- [235] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, *Guided image generation with conditional invertible neural networks*, 2019. arXiv: 1907.02392.
- [236] A. M. De Livera, R. J. Hyndman, and R. D. Snyder, “Forecasting time series with complex seasonal patterns using exponential smoothing”, *Journal of the American Statistical Association*, vol. 106, no. 496, pp. 1513–1527, 2011.
- [237] Y. Bao, T. Xiong, and Z. Hu, “Multi-step-ahead time series prediction using multiple-output support vector regression”, *Neurocomputing*, vol. 129, pp. 482–493, 2014.
- [238] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system”, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD ’16, San Francisco, USA: Association for Computing Machinery, 2016, pp. 785–794.
- [239] F. Wiese *et al.*, “Open Power System Data: Frictionless data for electricity system modelling”, *Applied Energy*, vol. 236, pp. 401–409, 2019.
- [240] A. Trindade, *Electricity load diagrams 2011-2014*, UCI Machine Learning Repository, 2015.
- [241] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, “Probabilistic energy forecasting: Global Energy Forecasting Competition 2014 and beyond”, *International Journal of Forecasting*, vol. 32, no. 3, pp. 896–913, 2016.

- [242] C. Debus, M. Piraud, A. Streit, F. Theis, and M. Götz, “Reporting electricity consumption is essential for sustainable AI”, *Nature Machine Intelligence*, vol. 5, no. 11, pp. 1176–1178, 2023.
- [243] J. Beitner. “PyTorch Forecasting”. (2020), [Online]. Available: <https://towardsdatascience.com/introducing-pytorch-forecasting-64de99b9ef46> (visited on 01/02/2021).
- [244] Amazon Web Services, Inc. “Amazon EC2 G4 instances”. (2024), [Online]. Available: <https://aws.amazon.com/ec2/instance-types/g4/> (visited on 05/01/2024).
- [245] C. Ding and H. Peng, “Minimum redundancy feature selection from microarray gene expression data”, in *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003*, 2003, pp. 523–528.
- [246] S. S. Rangapuram, L. D. Werner, K. Benidis, P. Mercado, J. Gasthaus, and T. Januschowski, “End-to-end learning of coherent probabilistic forecasts for hierarchical time series”, in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 8832–8843.
- [247] R. Wen and T. Kari, “Deep generative quantile-copula models for probabilistic forecasting”, in *Proceedings of the 36th International Conference on Machine Learning*, Paper: Time Series Workshop at ICML 2019, 2019.
- [248] K. Rasul, A.-S. Sheikh, I. Schuster, U. M. Bergmann, and R. Vollgraf, “Multivariate probabilistic time series forecasting via conditioned normalizing flows”, in *International Conference on Learning Representations*, 2021.
- [249] A. Jamgochian, D. Wu, K. Menda, S. Jung, and M. J. Kochenderfer, *Conditional approximate normalizing flows for joint multi-step probabilistic forecasting with application to electricity demand*, 2022. arXiv: 2201.02753.
- [250] A. Fanfarillo, B. Roozitalab, W. Hu, and G. Cervone, “Probabilistic forecasting using deep generative models”, *GeoInformatica*, vol. 25, no. 1, pp. 127–147, 2021.
- [251] L. Zhang and B. Zhang, “Scenario forecasting of residential load profiles”, *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 1, pp. 84–95, 2020.
- [252] 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*, vol. 8, pp. 77 587–77 597, 2020.

- [253] J. Dumas, A. Wehenkel, D. Lanaspeze, B. Cornélusse, and A. Sutura, “A deep generative model for probabilistic energy forecasting in power systems: Normalizing flows”, *Applied Energy*, vol. 305, p. 117 871, 2022.
- [254] E. Cramer, D. Witthaut, A. Mitsos, and M. Dahmen, “Multivariate probabilistic forecasting of intraday electricity prices using normalizing flows”, *Applied Energy*, vol. 346, p. 121 370, 2023.
- [255] V. Chernozhukov, K. Wüthrich, and Y. Zhu, “Distributional conformal prediction”, *Proceedings of the National Academy of Sciences*, vol. 118, no. 48, e2107794118, 2021.
- [256] M. Zaffran, O. Féron, Y. Goude, J. Josse, and A. Dieuleveut, “Adaptive conformal predictions for time series”, in *International Conference on Machine Learning*, PMLR, 2022, pp. 25 834–25 866.
- [257] F. S. Barros, V. Cerqueira, and C. Soares, “Empirical study on the impact of different sets of parameters of gradient boosting algorithms for time-series forecasting with LightGBM”, in *PRICAI 2021: Trends in Artificial Intelligence*, D. N. Pham, T. Theeramunkong, G. Governatori, and F. Liu, Eds., Cham, Switzerland: Springer International Publishing, 2021, pp. 454–465.
- [258] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, “AutoCTS: Automated correlated time series forecasting”, *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 971–983, 2021.
- [259] W. Kong, Z. Y. Dong, F. Luo, K. Meng, W. Zhang, F. Wang, and X. Zhao, “Effect of automatic hyperparameter tuning for residential load forecasting via deep learning”, in *2017 Australasian Universities Power Engineering Conference (AUPEC)*, 2017, pp. 1–6.
- [260] A. Al Mamun, M. Hoq, E. Hossain, and R. Bayindir, “A hybrid deep learning model with evolutionary algorithm for short-term load forecasting”, in *2019 8th International Conference on Renewable Energy Research and Applications (ICRERA)*, 2019, pp. 886–891.
- [261] R. C. Smith, *Uncertainty quantification: Theory, implementation, and applications*, First edition. Philadelphia, USA: Society for Industrial and Applied Mathematics, 2013.
- [262] R. J. Hyndman. “Prediction intervals too narrow”. (2014), [Online]. Available: <https://robjhyndman.com/hyndsight/narrow-pi/> (visited on 02/24/2024).

- [263] M. Jurado, M. Samper, and R. Rosés, “An improved encoder-decoder-based CNN model for probabilistic short-term load and PV forecasting”, *Electric Power Systems Research*, vol. 217, p. 109 153, 2023.
- [264] G. Marcjasz, M. Narajewski, R. Weron, and F. Ziel, “Distributional neural networks for electricity price forecasting”, *Energy Economics*, vol. 125, p. 106 843, 2023.
- [265] F. Salm, M. Oettmeier, and P. Rönsch, “The new redispatch and the impact on energy management in districts”, in *2022 18th International Conference on the European Energy Market (EEM)*, 2022, pp. 1–4.
- [266] R. Meka, A. Alaeddini, and K. Bhaganagar, “A robust deep learning framework for short-term wind power forecast of a full-scale wind farm using atmospheric variables”, *Energy*, vol. 221, p. 119 759, 2021.
- [267] Z. Wang, L. Wang, and C. Huang, “A fast abnormal data cleaning algorithm for performance evaluation of wind turbine”, *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.
- [268] Y. Su, F. Chen, G. Liang, X. Wu, and Y. Gan, “Wind power curve data cleaning algorithm via image thresholding”, in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 1198–1203.
- [269] Z. Luo, C. Fang, C. Liu, and S. Liu, “Method for cleaning abnormal data of wind turbine power curve based on density clustering and boundary extraction”, *IEEE Transactions on Sustainable Energy*, vol. 13, no. 2, pp. 1147–1159, 2022.
- [270] S. Klaiber, P. Bretschneider, S. Waczowicz, R. Mikut, I. Konotop, and D. Westermann, “A contribution to the load forecast of price elastic consumption behaviour”, in *2015 IEEE Eindhoven PowerTech*, 2015, pp. 1–6.
- [271] M. Petersen, J. Huber, and C. Hofmann, *Wind turbine library*, further contributors: Ludee, jh-RLI, 2019.
- [272] D. Etling, “Die atmosphärische Grenzschicht”, in *Theoretische Meteorologie: Eine Einführung*. Berlin, Heidelberg, Germany: Springer-Verlag, 2004, ch. 21, pp. 297–340.
- [273] C. Jung and D. Schindler, “The role of the power law exponent in wind energy assessment: A global analysis”, *International Journal of Energy Research*, vol. 45, no. 6, pp. 8484–8496, 2021.
- [274] G. M. Masters, “Wind power systems”, in *Renewable and efficient electric power systems*. Chichester, UK: John Wiley & Sons, Ltd, 2004, ch. 6, pp. 307–383.

- [275] S. A. Hsu, E. A. Meindl, and D. B. Gilhousen, “Determining the power-law wind-profile exponent under near-neutral stability conditions at sea”, *Journal of Applied Meteorology and Climatology*, vol. 33, no. 6, pp. 757–765, 1994.
- [276] R. A. Sobolewski, M. T. Tchakorom, and R. Couturier, “Gradient boosting-based approach for short- and medium-term wind turbine output power prediction”, *Renewable Energy*, vol. 203, pp. 142–160, 2023.
- [277] R. Morrison, X. L. Liu, and Z. Lin, “Anomaly detection in wind turbine SCADA data for power curve cleaning”, *Renewable Energy*, vol. 184, pp. 473–486, 2022.
- [278] European Centre for Medium-Range Weather Forecasts (ECMWF). “Meteorological Archival and Retrieval System (MARS)”. (2023), [Online]. Available: <https://confluence.ecmwf.int/display/CEMS/MARS> (visited on 06/03/2024).
- [279] E. Sharma, “Energy forecasting based on predictive data mining techniques in smart energy grids”, *Energy Informatics*, vol. 1, no. 1, p. 44, 2018.
- [280] M. López Santos, X. García-Santiago, F. Echevarría Camarero, G. Blázquez Gil, and P. Carrasco Ortega, “Application of Temporal Fusion Transformer for day-ahead PV power forecasting”, *Energies*, vol. 15, no. 14, 2022.
- [281] S. Rajagopalan and S. Santoso, “Wind power forecasting and error analysis using the autoregressive moving average modeling”, in *2009 IEEE Power & Energy Society General Meeting*, 2009, pp. 1–6.
- [282] B.-M. Hodge, A. Zeiler, D. Brooks, G. Blau, J. Pekny, and G. Reklatis, “Improved wind power forecasting with ARIMA models”, in *21st European Symposium on Computer Aided Process Engineering*, ser. Computer Aided Chemical Engineering, E. Pistikopoulos, M. Georgiadis, and A. Kokossis, Eds., vol. 29, Elsevier, 2011, pp. 1789–1793.
- [283] A. Lahouar and J. Ben Hadj Slama, “Hour-ahead wind power forecast based on random forests”, *Renewable Energy*, vol. 109, pp. 529–541, 2017.
- [284] Y. Liu, L. Guan, C. Hou, H. Han, Z. Liu, Y. Sun, and M. Zheng, “Wind power short-term prediction based on LSTM and discrete wavelet transform”, *Applied Sciences*, vol. 9, no. 6, 2019.
- [285] B. Zhou, X. Ma, Y. Luo, and D. Yang, “Wind power prediction based on LSTM networks and nonparametric kernel density estimation”, *IEEE Access*, vol. 7, pp. 165 279–165 292, 2019.
- [286] F. Shahid, A. Zameer, and M. Muneeb, “A novel genetic LSTM model for wind power forecast”, *Energy*, vol. 223, p. 120 069, 2021.

- [287] P. Arora, S. M. J. Jalali, S. Ahmadian, B. K. Panigrahi, P. N. Suganthan, and A. Khosravi, "Probabilistic wind power forecasting using optimized deep auto-regressive recurrent neural networks", *IEEE Transactions on Industrial Informatics*, vol. 19, no. 3, pp. 2814–2825, 2023.
- [288] R. Zhu, W. Liao, and Y. Wang, "Short-term prediction for wind power based on temporal convolutional network", *Energy Reports*, vol. 6, pp. 424–429, 2020, 2020 The 7th International Conference on Power and Energy Systems Engineering.
- [289] S. Sun, Y. Liu, Q. Li, T. Wang, and F. Chu, "Short-term multi-step wind power forecasting based on spatio-temporal correlations and transformer neural networks", *Energy Conversion and Management*, vol. 283, p. 116916, 2023.
- [290] L. van Heerden, C. van Staden, and H. Vermeulen, "Temporal Fusion Transformer for day-ahead wind power forecasting in the South African context", in *2023 IEEE International Conference on Environment and Electrical Engineering and 2023 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 2023, pp. 1–5.
- [291] Y. Hu, H. Liu, S. Wu, Y. Zhao, Z. Wang, and X. Liu, "Temporal collaborative attention for wind power forecasting", *Applied Energy*, vol. 357, p. 122502, 2024.
- [292] S. Xu, Y. Wang, X. Xu, G. Shi, Y. Zheng, H. Huang, and C. Hong, "A multi-step wind power group forecasting seq2seq architecture with spatial-temporal feature fusion and numerical weather prediction correction", *Energy*, vol. 291, p. 130352, 2024.
- [293] S. Mo, H. Wang, B. Li, Z. Xue, S. Fan, and X. Liu, "Powerformer: A temporal-based transformer model for wind power forecasting", *Energy Reports*, vol. 11, pp. 736–744, 2024.
- [294] M. Lydia, S. S. Kumar, A. I. Selvakumar, and G. E. Prem Kumar, "A comprehensive review on wind turbine power curve modeling techniques", *Renewable and Sustainable Energy Reviews*, vol. 30, pp. 452–460, 2014.
- [295] C. Carrillo, A. Obando Montaña, J. Cidrás, and E. Díaz-Dorado, "Review of power curve modelling for wind turbines", *Renewable and Sustainable Energy Reviews*, vol. 21, pp. 572–581, 2013.
- [296] M. Lydia, A. I. Selvakumar, S. S. Kumar, and G. E. P. Kumar, "Advanced algorithms for wind turbine power curve modeling", *IEEE Transactions on Sustainable Energy*, vol. 4, no. 3, pp. 827–835, 2013.

- [297] S. Li, D. Wunsch, E. O'Hair, and M. Giesselmann, "Using neural networks to estimate wind turbine power generation", *IEEE Transactions on Energy Conversion*, vol. 16, no. 3, pp. 276–282, 2001.
- [298] R. K. Pandit, D. Infield, and A. Kolios, "Comparison of advanced non-parametric models for wind turbine power curves", *IET Renewable Power Generation*, vol. 13, no. 9, pp. 1503–1510, 2019.
- [299] S. R. Moreno, L. d. S. Coelho, H. V. Ayala, and V. C. Mariani, "Wind turbines anomaly detection based on power curves and ensemble learning", *IET Renewable Power Generation*, vol. 14, no. 19, pp. 4086–4093, 2020.
- [300] A. Arnold, S. König, R. Mikut, and P. Bretschneider, "Application of data mining methods for power forecast of wind power plants", in *Proc. of the 9th Internat. Workshop on Large-Scale Integration of Wind Power into Power Systems as well as on Transmission Networks for Offshore Wind Power Plants*, U. Betancourt, Ed., Quebec City, Canada: Energynautics GmbH, 2010, pp. 655–60.
- [301] A. Arnoldt, P. Bretschneider, S. König, and R. Mikut, "Einsatz von Data-Mining-Verfahren zur Einspeisungsprognose für Windenergieanlagen", in *Automation 2010: Der 11. Branchentreff der Mess- und Automatisierungstechnik (VDI-Berichte; 2092)*, Düsseldorf, Germany: VDI Verlag GmbH, 2010, pp. 99–102.
- [302] W. Muchleisen *et al.*, "Energy yield measurement of an elevated PV system on a white flat roof and a performance comparison of monofacial and bifacial modules", *Renewable Energy*, vol. 170, pp. 613–619, 2021.
- [303] O. Perpiñán Lamigueiro, "solaR: Solar radiation and photovoltaic systems with R", *Journal of Statistical Software*, vol. 50, no. 9, pp. 1–32, 2012.
- [304] N. A. Engerer and F. P. Mills, "KPV: A clear-sky index for photovoltaics", *Solar Energy*, vol. 105, pp. 679–693, 2014.
- [305] W. F. Holmgren, C. W. Hansen, and M. A. Mikofski, "pvlib Python: A Python package for modeling solar energy systems", *Journal of Open Source Software*, vol. 3, no. 29, p. 884, 2018.
- [306] J. Bottieau, Z. De Grève, T. Piraux, A. Dubois, F. Vallée, and J.-F. Toubeau, "A cross-learning approach for cold-start forecasting of residential photovoltaic generation", *Electric Power Systems Research*, vol. 212, p. 108415, 2022.
- [307] M. Abdel-Basset, H. Hawash, R. K. Chakraborty, and M. Ryan, "PV-Net: An innovative deep learning approach for efficient forecasting of short-term photovoltaic energy production", *Journal of Cleaner Production*, vol. 303, p. 127037, 2021.

- [308] M. Aslam, S.-J. Lee, S.-H. Khang, and S. Hong, “Two-stage attention over LSTM with Bayesian optimization for day-ahead solar power forecasting”, *IEEE Access*, vol. 9, pp. 107 387–107 398, 2021.
- [309] K. Wang, X. Qi, and H. Liu, “A comparison of day-ahead photovoltaic power forecasting models based on deep learning neural network”, *Applied Energy*, vol. 251, no. C, p. 113 315, 2019.
- [310] F. Mejia, J. Kleissl, and J. L. Bosch, “The effect of dust on solar photovoltaic systems”, in *Proceedings of the SolarPACES 2013 International Conference*, vol. 49, 2014, pp. 2370–2376.
- [311] M. Aghaei *et al.*, “Review of degradation and failure phenomena in photovoltaic modules”, *Renewable and Sustainable Energy Reviews*, vol. 159, p. 112 160, 2022.
- [312] K. Osmani, A. Haddad, T. Lemenand, B. Castanier, and M. Ramadan, “A review on maintenance strategies for PV systems”, *Science of The Total Environment*, vol. 746, p. 141 753, 2020.
- [313] J. J. Roberts, A. A. Mendiburu Zevallos, and A. M. Cassula, “Assessment of photovoltaic performance models for system simulation”, *Renewable and Sustainable Energy Reviews*, vol. 72, pp. 1104–1123, 2017.
- [314] I. Reda and A. Andreas, “Solar position algorithm for solar radiation applications”, *Solar Energy*, vol. 76, no. 5, pp. 577–589, 2004.
- [315] E. L. Maxwell, “A quasi-physical model for converting hourly global horizontal to direct normal insolation”, Solar Energy Research Inst., Golden, CO (USA), Tech. Rep., 1987.
- [316] J. E. Hay and J. A. Davies, “Calculation of the solar radiation incident on an inclined surface”, in *Proceedings of First Canadian Solar Radiation Data Workshop*, Downsview, Canada: Canadian Atmospheric Environment Service, 1980, pp. 59–72.
- [317] J. W. Spencer, “Fourier series representation of the position of the sun”, *Search*, vol. 2, pp. 172+, 1971.
- [318] F. Kasten and A. T. Young, “Revised optical air mass tables and approximation formula”, *Applied Optics*, vol. 28, no. 22, pp. 4735–4738, 1989.
- [319] D. L. King, S. Gonzalez, G. M. Galbraith, and W. E. Boyson, “Performance model for grid-connected photovoltaic inverters”, Sandia National Laboratories (SNL), Albuquerque, NM (USA), Tech. Rep. 5036, 2007, pp. 1–47.
- [320] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, “Tune: A research platform for distributed model selection and training”, 2018. arXiv: 1807 .05118.

- [321] Deutscher Wetterdienst. “Historical 10-minute station observations of solar incoming radiation, longwave downward radiation, pressure, air temperature, and mean wind speed for Germany”. (2023), [Online]. Available: https://opendata.dwd.de/climate_environment/CDC/observations_germany/climate/10_minutes (visited on 01/02/2023).
- [322] O. Schwärmer and F. Cevik, “Implementation and testing of the gradient descent algorithm and Kalman filter for the day ahead PV-forecasting model AutoPV”, Karlsruhe Institute of Technology, Karlsruhe, Germany, Smart Energy System Lab (SESL) internship report, 2023.
- [323] N. Bashir, D. Chen, D. Irwin, and P. Shenoy, “Solar-TK: A data-driven toolkit for solar PV performance modeling and forecasting”, in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Institute of Electrical and Electronics Engineers, 2019, pp. 456–466.
- [324] R. Basmadjian, A. Shaafieyoun, and S. Julka, “Day-ahead forecasting of the percentage of renewables based on time series statistical methods”, *Energies*, vol. 14, no. 21, p. 7443, 2021.
- [325] R. Basmadjian and A. Shaafieyoun, “ARIMA-based forecasts for the share of renewable energy sources: The case study of Germany”, in *2022 3rd International Conference on Smart Grid and Renewable Energy (SGRE)*, Institute of Electrical and Electronics Engineers, 2022, pp. 1–6.
- [326] M. López, C. Sans, S. Valero, and C. Senabre, “Empirical comparison of neural network and auto-regressive models in short-term load forecasting”, *Energies*, vol. 11, no. 8, p. 2080, 2018.
- [327] Aasim, S. Singh, and A. Mohapatra, “Repeated wavelet transform based ARIMA model for very short-term wind speed forecasting”, *Renewable Energy*, vol. 136, pp. 758–768, 2019.
- [328] K. Yan, H. Shen, L. Wang, H. Zhou, M. Xu, and Y. Mo, “Short-term solar irradiance forecasting based on a hybrid deep learning methodology”, *Information*, vol. 11, no. 1, p. 32, 2020.
- [329] Y. Li, Y. Su, and L. Shu, “An ARMAX model for forecasting the power output of a grid connected photovoltaic system”, *Renewable Energy*, vol. 66, pp. 78–89, 2014.
- [330] J. Á. González Ordiano, S. Waczowicz, M. Reischl, R. Mikut, and V. Hagenmeyer, “Photovoltaic power forecasting using simple data-driven models without weather data”, *Computer Science - Research and Development*, vol. 32, no. 1, pp. 237–246, 2017.

- [331] A. Mellit and A. M. Pavan, “A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected PV plant at Trieste, Italy”, *Solar Energy*, vol. 84, no. 5, pp. 807–821, 2010.
- [332] S. Das, “Short-term forecasting of solar radiation and power output of 89.6kwp solar PV power plant”, *Materials Today: Proceedings*, vol. 39, pp. 1959–1969, 2021, 3rd International Conference on Solar Energy Photovoltaics.
- [333] G. Etxegarai, A. López, N. Aginako, and F. Rodríguez, “An analysis of different deep learning neural networks for intra-hour solar irradiation forecasting to compute solar photovoltaic generators’ energy production”, *Energy for Sustainable Development*, vol. 68, pp. 1–17, 2022.
- [334] W. Zhao, H. Zhang, J. Zheng, Y. Dai, L. Huang, W. Shang, and Y. Liang, “A point prediction method based automatic machine learning for day-ahead power output of multi-region photovoltaic plants”, *Energy*, vol. 223, p. 120 026, 2021.
- [335] D. Chen, J. Breda, and D. Irwin, “Staring at the sun: A physical black-box solar performance model”, in *Proceedings of the 5th Conference on Systems for Built Environments*, ser. BuildSys ’18, Shenzhen, China: Association for Computing Machinery, 2018, pp. 53–62.
- [336] Valentin Software GmbH, *PV*SOL*, 2024.
- [337] E. Kampmann, *Optimized charging of an electric vehicle considering a battery energy storage system and a photovoltaic system*, bachelor’s thesis, Karlsruhe Institute of Technology, Germany, 2022.
- [338] Gesellschaft zur Förderung angewandter Informatik e.V. (GfAI), *TOP-Energy®*, 2024.
- [339] A. Wohnig, *Estimation of PV system size, tilt and azimuth from power time series: Machine learning and optimisation approaches*, master’s thesis, Karlsruhe Institute of Technology, Germany, 2023.
- [340] Deutscher Wetterdienst. “Testreferenzjahre (TRY)”. (2024), [Online]. Available: <https://www.dwd.de/DE/leistungen/testreferenzjahre/testreferenzjahre.html> (visited on 01/12/2024).
- [341] D. Markovics and M. J. Mayer, “Comparison of machine learning methods for photovoltaic power forecasting based on numerical weather prediction”, *Renewable and Sustainable Energy Reviews*, vol. 161, p. 112 364, 2022.
- [342] S. Killinger, F. Braam, B. Müller, B. Wille-Haussmann, and R. McKenna, “Projection of power generation between differently-oriented PV systems”, *Solar Energy*, vol. 136, pp. 153–165, 2016.

- [343] Y. Hao and C. Tian, “A novel two-stage forecasting model based on error factor and ensemble method for multi-step wind power forecasting”, *Applied Energy*, vol. 238, pp. 368–383, 2019.
- [344] A. Abdellatif *et al.*, “Forecasting photovoltaic power generation with a stacking ensemble model”, *Sustainability*, vol. 14, no. 17, p. 11 083, 2022.
- [345] R. S. Olson, N. Bartley, R. J. Urbanowicz, and J. H. Moore, “Evaluation of a tree-based pipeline optimization tool for automating data science”, in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, ser. GECCO ’16, Denver, USA: Association for Computing Machinery, 2016, pp. 485–492.
- [346] M. Feurer, A. Klein, K. Eggenberger, J. Springenberg, M. Blum, and F. Hutter, “Efficient and robust automated machine learning”, in *Advances in Neural Information Processing Systems 28 (2015)*, Springer International Publishing, 2015, pp. 2962–2970.
- [347] V. Hagenmeyer *et al.*, “Information and communication technology in Energy Lab 2.0: Smart energies system simulation and control center with an open-street-map-based power flow simulation example”, *Energy Technology*, vol. 4, no. 1, pp. 145–162, 2016.
- [348] F. Wiegel, J. Wachter, M. Kyesswa, R. Mikut, S. Waczowicz, and V. Hagenmeyer, “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.
- [349] S. Meisenbacher, K. Schwenk, J. Galenzowski, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “A lightweight user interface for smart charging of electric vehicles: A real-world application”, in *2021 9th International Conference on Smart Grid and Clean Energy Technologies (ICSGCE)*, Virtual Event, 2021, pp. 57–61.
- [350] S. Meisenbacher, K. Schwenk, J. Galenzowski, S. Waczowicz, R. Mikut, and V. Hagenmeyer, “Smart charging of electric vehicles with cloud-based optimization and a lightweight user interface: A real-world application in the Energy Lab 2.0”, in *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, ser. e-Energy ’21, Virtual Event: Association for Computing Machinery, 2021, pp. 284–285.
- [351] K. Schwenk, S. Meisenbacher, B. Briegel, T. Harr, V. Hagenmeyer, and R. Mikut, “Integrating battery aging in the optimization for bidirectional charging of electric vehicles”, *IEEE Transactions on Smart Grid*, vol. 12, no. 6, pp. 5135–5145, 2021.

- [352] International Energy Agency (IEA), “World energy outlook 2022”, Paris, Tech. Rep., 2022.
- [353] J. Galenzowski, S. Waczowicz, S. Meisenbacher, R. Mikut, and V. Hagenmeyer, “A real-world district community platform as a cyber-physical-social infrastructure systems in the energy domain”, in *Proceedings of the 10th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys '23, New York, USA: Association for Computing Machinery, 2023, pp. 434–441.
- [354] D. Wölfle, M. Lösch, and H. Schmeck, “BEMCom: A framework for the efficient creation of hardware abstraction layers for building energy management”, *SIGENERGY Energy Inform. Rev.*, vol. 2, no. 1, pp. 20–25, 2022.
- [355] C. Krueger, M. Otte, S. Holly, S. Rathjen, A. Wellssow, and S. Lehnhoff, “Redispatch 3.0 – Congestion management for German power grids – Considering controllable resources in low-voltage grids”, in *ETG Congress 2023*, 2023, pp. 1–7.
- [356] C. Linnemann, D. Echternacht, C. Breuer, and A. Moser, “Modeling optimal re-dispatch for the European transmission grid”, in *2011 IEEE Trondheim PowerTech*, 2011, pp. 1–8.
- [357] N. C. Figueiredo and P. P. da Silva, “The merit-order effect of wind and solar power: Volatility and determinants”, *Renewable and Sustainable Energy Reviews*, vol. 102, pp. 54–62, 2019.
- [358] TransnetBW GmbH. “Datenaustausch/Redispatch (DA/RE)”. (2024), [Online]. Available: <https://www.dare-plattform.de/> (visited on 05/03/2024).
- [359] M. Kopp, *Comparison of online adaptation methods for probabilistic time series forecasting*, bachelor’s thesis, Karlsruhe Institute of Technology, Germany, 2024.
- [360] E. Klumpp, *Data-driven building modelling: A comparison of different grey-box models, identified with real-world data*, bachelor’s thesis, Karlsruhe Institute of Technology, Germany, 2021.
- [361] M. Frahm, E. Klumpp, S. Meisenbacher, J. Matthes, R. Mikut, and V. Hagenmeyer, “Development and validation of grey-box multi-zone thermal building models”, in *Proceedings of BauSim Conference 2022: 9th Conference of IBPSA-Germany and Austria*, ser. BauSim Conference, vol. 9, Weimar, Germany: IBPSA-Germany and Austria, 2022.

- [362] M. Frahm, S. Meisenbacher, E. Klumpp, R. Mikut, J. Matthes, and V. Hagenmeyer, “Multi-zone grey-box thermal building identification with real occupants”, in *Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ser. BuildSys ’22, Boston, USA: Association for Computing Machinery, 2022, pp. 484–487.
- [363] M. Frahm, T. Dengiz, P. Zwickel, H. Maaß, J. Matthes, and V. Hagenmeyer, “Occupant-oriented demand response with multi-zone thermal building control”, *Applied Energy*, vol. 347, p. 121 454, 2023.
- [364] M. Löning, A. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király, “sktime: A unified interface for machine learning with time series”, in *2019 Workshop on Systems for ML at NeurIPS*, Vancouver, Canada, 2019.
- [365] Pareto Software LLC, *Australia shapefile*, copyright: ©2024 Pareto Software LLC. This work is licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0)., Cincinnati, USA, 2024.
- [366] R. Mikut, *Data Mining in der Medizin und Medizintechnik* (Schriftenreihe des Instituts für Angewandte Informatik / Automatisierungstechnik). Karlsruhe, Germany: KIT Scientific Publishing, 2008, vol. 22.
- [367] R. Grewal, *AutoML: Automated machine learning for short term electrical load forecasting*, bachelor’s thesis, Karlsruhe Institute of Technology, Germany, 2021.
- [368] N. Jung, *Development and implementation of a cost-sensitive adaptive thermal building model using incremental and ensemble learning*, master’s thesis, Karlsruhe Institute of Technology, Germany, 2023.
- [369] J. M. Pinter, *Anomaly detection in end-of-line testing at ZF Wind Power*, master’s thesis, Karlsruhe Institute of Technology, Germany, 2023.

Climate change has forced a paradigm shift in energy systems and requires increasing the share of renewable energy generation. However, this transition to intermittent energy sources such as Wind Power (WP) and PhotoVoltaic (PV) poses electricity storage challenges, which requires a flexibilization of the electricity grid through sector coupling. Coupling energy-consuming sectors with the power-generating sector requires solving many decentralized optimization problems, which in turn are based on forecasts of local electricity demand and supply. As a result, the demand for such locally customized forecasting models is increasing rapidly, forcing the time-consuming design and application processes to be automated. This thesis therefore presents a novel concept for automating time series forecasting in smart grid applications.



Increasing the automation level
of time series forecasting with
design and operation templates