Contents lists available at ScienceDirect

Applied Energy

journal homepage: www.elsevier.com/locate/apen

AutoPQ: Automating quantile estimation from point forecasts in the context of sustainability

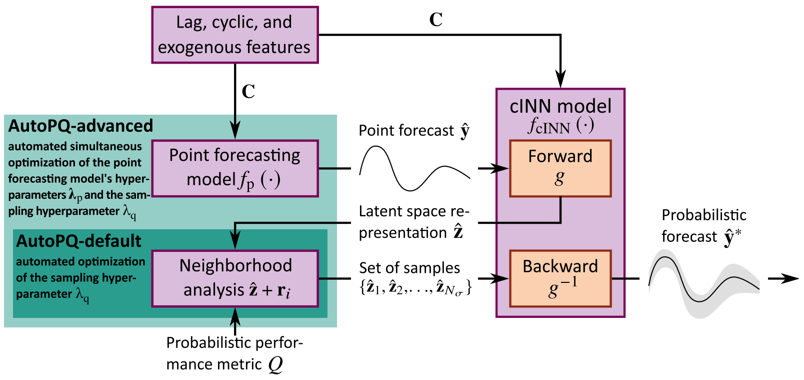
Stefan Meisenbacher[a], Kaleb Phipps[b], Oskar Taubert[b], Marie Weiel[b], Markus Götz[b], Ralf Mikut[a,*], Veit Hagenmeyer[a]

[a] Karlsruhe Institute of Technology, Institute for Automation and Applied Informatics, Eggenstein-Leopoldshafen, 76344, Germany
[b] Karlsruhe Institute of Technology, Scientific Computing Center (SCC), Eggenstein-Leopoldshafen, 76344, Germany

HIGHLIGHTS

- AutoPQ uses a conditional Invertible Neural Network (cINN) to convert point forecasts into quantile forecasts.
- AutoPQ automates model selection and hyperparameter optimization for improving probabilistic performance.
- It offers two configurations making it suitable for a wide range of smart grid applications: AutoPQ-default for general systems and AutoPQ-advanced for HPC systems.
- AutoPQ shows a 15.1 % improvement over direct probabilistic benchmarks and a 9.1 % improvement over point forecast-based probabilistic benchmarks.
- The ablation study shows the effectiveness of applied methods in reducing AutoPQ's electricity consumption.

GRAPHICAL ABSTRACT


Lag, cyclic, and exogenous features
C
C
AutoPQ-advanced
automated simultaneous optimization of the point forecasting model's hyperparameters $\boldsymbol{\lambda}_p$ and the sampling hyperparameter λ_q
Point forecasting model $f_p(\cdot)$
AutoPQ-default
automated optimization of the sampling hyperparameter λ_q
Neighborhood analysis $\hat{\mathbf{z}} + \mathbf{r}_i$
Probabilistic performance metric Q
Point forecast $\hat{\mathbf{y}}$
Latent space representation $\hat{\mathbf{z}}$
Set of samples $\{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_{N_\sigma}\}$
cINN model $f_{\mathrm{cINN}}(\cdot)$
Forward g
Backward g^{-1}
Probabilistic forecast $\hat{\mathbf{y}}^*$

ARTICLE INFO

Keywords:
Probabilistic time series forecasting
Uncertainty quantification
AutoML
Energy consumption

ABSTRACT

Optimizing smart grid operations relies on critical decision-making informed by uncertainty quantification, making probabilistic forecasting a vital tool. However, designing such forecasting models presents three key challenges: achieving accurate and unbiased uncertainty quantification, reducing the workload for data scientists during the design process, and minimizing the environmental impact of model training. In order to address these challenges, we introduce AutoPQ, a novel method that fully automates and optimizes probabilistic forecasting for smart grid applications. AutoPQ enhances forecast uncertainty quantification by generating high-quality quantile forecasts from an existing point forecast by using a conditional Invertible Neural Network (cINN). Furthermore, AutoPQ automates the selection of the optimal point forecasting method and fine-tunes hyperparameters, ensuring the best-possible model and configuration for each application. For flexible adaptation to various performance needs and available computing power, AutoPQ comes with a default and an advanced configuration, making it suitable for a wide range of smart grid applications. We demonstrate that AutoPQ surpasses state-of-the-art

* Corresponding author.
Email address: ralf.mikut@kit.edu (R. Mikut).

https://doi.org/10.1016/j.apenergy.2025.125931
Received 7 December 2024; Received in revised form 19 March 2025; Accepted 10 April 2025
Available online 23 April 2025

probabilistic forecasting methods while effectively limiting computational effort and hence environmental impact. Additionally and in the context of sustainability, `AutoPQ` provides full transparency regarding the electricity consumption required for performance improvements.

Acronyms

ANN	Artificial Neural Network
AR	AutoRegression
ARIMA	AutoRegressive Integrated Moving Average
BATS	Box-Cox transformation, ARMA errors, Trend, Seasonal components
BO	Bayesian Optimization
CASH	Combined Algorithm Selection and Hyperparameter optimization
CDF	Cumulative Distribution Function
cINN	conditional Invertible Neural Network
cPDF	conditional Probability Density Function
CPU	Central Processing Unit
CRPS	Continuous Ranked Probability Score
DeepAR	Deep AutoRegression
DL	Deep Learning
EA	Evolutionary Algorithm
ECMWF	European Centre for Medium-Range Weather Forecasts
ETS	Error Trend Seasonality
GAN	Generative Adversarial Network
GBM	Gradient Boosting Machine
GCP	Grid Connection Point
GEFCom	Global Energy Forecasting Competition
GHI	Global Horizontal Irradiance
GPU	Graphics Processing Unit
HPC	High-Performance Computing
HPO	Hyperparameter Optimization
LASSO	Least Absolute Shrinkage and Selection Operator
ML	Machine Learning
MLP	MultiLayer Perceptron
MRMR	Minimum Redundancy Maximum Relevance
MSE	Mean Squared Error
N-HiTS	Neural Hierarchical Interpolation for Time Series
NNQF	Nearest Neighbor Quantile Filter
NWP	Numerical Weather Prediction
OPSD	Open Power System Data
PDF	Probability Density Function
PI	Prediction Interval
PK	Prior Knowledge
PL	Pinball Loss
PV	PhotoVoltaic
QRNN	Quantile Regression Neural Network
RAM	Random-Access Memory
RF	Random Forest
RNN	Recurrent Neural Network
sARIMAX	seasonal AutoRegressive Integrated Moving Average with eXternal input
SM	Statistical Modeling
SVR	Support Vector Regression
TBATS	Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend, Seasonal components
TES	Triple Exponential Smoothing
TFT	Temporal Fusion Transformer
TPE	Tree Parzen Estimator
WP	Wind Power
XGB	eXtreme Gradient Boosting

1. Introduction

Uncertainty quantification is necessary for making informed decisions in smart grid applications, rendering probabilistic time series forecasts essential [1]. Exemplary downstream applications are stochastic power flow optimizations [2,3], smart charging of stationary battery systems and electric vehicles [4–7], and economic dispatch [8,9]. As the number of applications increases, automating the design process of probabilistic time series forecasting models is necessary to keep pace with the growing demand.

This process involves three major challenges: first, a forecast's inherent uncertainty must be quantified in an unbiased and accurate manner. However, many recently published methods [10–14] only provide point forecasts [15]. Methods to generate a probabilistic forecast from a point forecast exist but have limitations, e.g., they assume a Probability Density Function (PDF) [16,17]), only estimate Prediction Intervals (PIs) [18,19], or rely on the residuals of the point forecast [20] instead of representing the uncertainty of the process itself. Although direct probabilistic methods can partially overcome these limitations, they fail to leverage the many existing and well-established forecasting models already in use [21].

Second, given that the forecast quality is sensitive to different model design decisions [22], no forecasting method can excel in all tasks (no-free-lunch theorem), and the quality requirements depend on the downstream application. That is why automatically selecting the best-performing method is absolutely crucial [21]. In addition, performance-critical smart grid applications often demand further model enhancement by Hyperparameter Optimization (HPO). However, existing Combined Algorithm Selection and Hyperparameter optimization (CASH) methods in time series forecasting are limited to point forecasting [23–25] or are subject to the constraints outlined in the first challenge [26].

Third, methodological advances in probabilistic forecasting must quantify the environmental impact of employing such an advanced model in a standardized, quantifiable, and comparable manner [27]. Specifically, smart grid applications utilizing forecasts must ensure that the electricity consumption required for forecasting model design remains at a reasonable level. This is crucial because excessive electricity consumption, particularly for computationally intensive modern Deep Learning (DL) methods, can undermine sustainability efforts by increasing the overall carbon footprint. Despite this need, most existing probabilistic forecasting methods only specify the computing hardware employed [26,28,29] but fail to report the associated electricity consumption.

To tackle these challenges, we introduce `AutoPQ`, a novel method for automated and electricity consumption-aware quantile forecasts from point forecasts. Our key contributions include:

- `AutoPQ` generates quantile forecasts without requiring prior information about the underlying distribution by (i) leveraging established point forecasting methods (unknown conditional distribution), (ii) mapping the point forecast into the latent space of a conditional Invertible Neural Network (cINN) [30] (known and tractable conditional distribution), and (iii) efficiently sampling in the neighborhood of the point forecast's latent space representation [21].

Table 1
Overview of existing time series forecasting methods in terms of uncertainty quantification, automated design, and reporting the electricity consumption for the model design.

Reference	Category	Uncertainty quantification			Automated design	Electricity consumption
		Quantiles/PIs	PDF	Scenarios		
[31–39]	Direct probabilistic forecast	✓	✗	✗	✗	✗
[40–42]	Direct probabilistic forecast	✓	✗	✗	HPO	✗
[28,43]	Direct probabilistic forecast	✓	✓	✗	✗	✗
[29]	Direct probabilistic forecast	✓	✓	✗	HPO	✗
[44]	Direct probabilistic forecast	✓	✓	✗	✗	✓
[45–53]	Direct probabilistic forecast	✓	✓	✓	✗	✗
[16,18,19,54,55]	Point forecast-based probabilistic forecast	✓	✗	✗	✗	✗
[20]	Point forecast-based probabilistic forecast	✓	✓	✓	✗	✗
[17]	Point forecast-based probabilistic forecast	✓	✓	✗	✗	✗
[20]	Point forecast-based probabilistic forecast	✓	✓	✓	✗	✗
[56]	Point forecast-based probabilistic forecast	✓	✓	✗	HPO	✗
[57–66]	Point forecast	✗	✗	✗	HPO	✗
[23–25]	Point forecast	✗	✗	✗	CASH	✗
[26]	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).

- `AutoPQ` enhances [21] and fully automates task-dependent model design decisions by (*a*) optimizing the variance for sampling in the latent space, (*b*) selecting the best-performing point forecasting method, and (*c*) fine-tuning its hyperparameters for optimal performance.
- In order to accommodate different computing systems and performance requirements, we introduce `AutoPQ-default` (automating *a* and *b*) for general-purpose computing systems to provide high-quality probabilistic forecasts, and `AutoPQ-advanced` (automating *a*, *b*, and *c*) for High-Performance Computing (HPC) systems to increase the forecasting quality further, as required by smart grid applications with high decision costs.
- For optimal performance, we address environmental considerations by systematically reporting the electricity consumption of `AutoPQ-default` and `AutoPQ-advanced` in relation to their achievable performance.

The remainder of the paper is structured as follows: Section 2 provides an overview of relevant related work. In Section 3 we then present `AutoPQ`, which we further evaluate in Section 4. The results of our evaluation are discussed in Section 5. Finally, Section 6 gives a conclusion and an outlook.

2. Related work

This section analyzes related work with a particular focus on the three challenges stated above, i.e., uncertainty quantification in time series forecasting, design process automation, and quantification of energy consumption required for performance improvements. Table 1 gives an overview of the analyzed literature.

2.1. Probabilistic time series forecasting

Probabilistic time series forecasting methods can be categorized into direct probabilistic methods and probabilistic methods based on point forecasts.

Direct probabilistic forecasting methods. Direct probabilistic forecasting methods aim to learn the uncertainty of the forecast during model training. These methods have gained significant attention in recent years due to their ability to quantify forecast uncertainty directly. Depending on how uncertainty is represented, direct probabilistic forecasting methods can be categorized into quantile-based, distribution-based, and normalizing flow-based forecasts.

Quantile-based forecasting methods are trained to represent certain quantiles of the target distribution. These quantiles can then be applied to derive PIs with a specific conditional probability that the target value lies within the interval. One widely used method is Quantile Regression Neural Networks (QRNNs) [31], which utilize the Pinball Loss (PL) function to learn conditional quantiles. To obtain a full set of quantiles, the network has multiple outputs representing different quantiles learned during training. This approach is widely used in the literature with different neural architectures consisting of convolutional layers [33,41], recurrent [32,34,35,67] and residual connections [33,41,67], as well as transformer architectures based on the attention-mechanism [41,67]. Apart from Artificial Neural Networks (ANNs), other Machine Learning (ML) methods are applied for quantile regression such as tree-based methods [36,40,42] and Support Vector Regression (SVR) [37,38].[1] A different approach is the Nearest Neighbor Quantile Filter (NNQF) [39], which modifies the training dataset based on similarity in the target variable to determine a set of quantiles. Afterward, the method trains an ML-based model on the modified data to forecast this set of quantiles.

However, both QRNNs and the NNQF only provide quantiles, from which PIs can be derived, but do not provide the probability distribution, limiting their interpretability and flexibility in some applications [69].

The full PDF or Cumulative Distribution Function (CDF) can be obtained from distribution-based forecasts, which aim to learn a parametric representation of the uncertainty associated with forecasts. A well-known example is Deep AutoRegression (DeepAR) [28], an AutoRegression (AR)-based Recurrent Neural Network (RNN) trained to learn the parameters of a given PDF and generate probabilistic forecasts by sampling from it. This approach has been extended in [29] to provide probabilistic hierarchical forecasts. A more recent approach in [44] follows a similar approach to learn the parameters of a given PDF but proposes a more parameter-efficient temporal convolutional network to improve computational efficiency and reduce training efforts, making it more scalable for large datasets. Despite their advantages, all three methods require assuming the underlying parametric PDF, such as a Gaussian or a negative binomial distribution. This assumption can be restrictive when the data exhibits multimodality, skewness, or other complex distributional characteristics that do not conform to standard parametric forms.

To address the limitations, recent research has explored alternative approaches, such as flexible mixture density networks [43], which model the target distribution as a mixture of multiple distributions, and normalizing flows, which allow for non-parametric density estimation

[1] For more details on quantile regression, refer to [68].

without the need to assume a fixed parametric form. More specifically, normalizing flow-based models learn a bijective mapping from an unknown PDF to a known, tractable distribution, such as a Gaussian.[2] Typically, normalizing flows are used to enhance existing probabilistic forecasting methods, such as QRNNs [45], RNNs [46], Gaussian mixture models [47], and Bernstein polynomials [48]. They can also be applied to directly generate probabilistic forecasts in various domains, including atmospheric variables [49], renewable energy generation [52], electrical load [50–52], and electricity prices [53]. These methods provide a high degree of flexibility in modeling complex and multimodal distributions. However, one common limitation is that many normalizing flow-based approaches do not explicitly incorporate exogenous features, such as weather variables, economic indicators, or other external factors that may condition the PDF. As a result, these methods often assume that the learned distribution remains constant under varying external conditions, potentially limiting their adaptability and quality in dynamic environments.

All direct probabilistic forecasting methods are limited by their inability to generate probabilistic forecasts from available, well-designed point forecasts [15].

Point forecast-based probabilistic forecasting methods. Point forecast-based probabilistic forecasting methods take a different approach quantifying forecast uncertainty through a separate uncertainty estimation procedure [70]. In this context, most uncertainty quantification methods rely on the residuals between point forecasts and observed values in a validation dataset to estimate PIs. The distributional assumptions about these residuals differ between methods. Classical methods often assume a Gaussian distribution [16] or an empirical distribution [18], while more recent methods assume empirical non-conformity score distributions [19,54,55]. The latter calculates a critical non-conformity score for the desired significance level of the PIs by evaluating how unusual a residual is compared to other residuals in the validation dataset.[3] These methods are widely used due to their simplicity for estimating PIs of forecasting methods that are originally designed as a point forecasting method [10–14], e.g., in the Python libraries `GluonTS` [72] and `NeuralForecast` [73]. However, a fundamental limitation is that conformal prediction is limited to PIs rather than providing a full PDF, making it less suitable for applications requiring a complete probabilistic representation of forecast uncertainty [69].

Beyond the above approaches, there are enhancements in residual-based uncertainty quantification that leverage ML techniques. In [17], an ANN is trained to forecast the standard deviation of these residuals under the assumption of a Gaussian distribution. The estimated standard deviation is then used to quantify the point forecast's uncertainty. Wang et al. [20] enhances flexibility by capturing complex and non-Gaussian residual distributions. The authors train a Generative Adversarial Network (GAN) on the residuals of a point forecast to generate residual scenarios, which are then used to quantify the point forecast's uncertainty. A more recent approach in [56] integrates transformer networks for point forecasting with a Gaussian process-based uncertainty quantification model via residual learning. While enhancing uncertainty quantification using ML, these methods introduce challenges for HPO. As the ML-based uncertainty estimation models are trained on the residuals of a specific point forecast, they are inherently dependent on the point forecasting model. Consequently, when comparing multiple candidate point forecasting methods and hyperparameter configurations, a separate uncertainty estimation model must be trained for each of them, increasing computational overhead and model selection complexity.

2.2. Automated time series forecasting

Automated time series forecasting methods can be classified into two main categories: Hyperparameter Optimization (HPO) and Combined Algorithm Selection and Hyperparameter optimization (CASH). While automated point forecasting utilizing HPO [57–66] and CASH [23–25] has been extensively researched, there is a significant gap in studies that focus on automated probabilistic forecasting [74].

Hyperparameter optimization. The purpose of HPO is to find a forecasting method's configuration that delivers the best-possible performance for a given forecasting task.

Simple grid search-based HPO has been proposed for point forecasting methods within the Statistical Modeling (SM) family, such as Error Trend Seasonality (ETS) [57], seasonal AutoRegressive Integrated Moving Average with eXternal input (sARIMAX) [58], and Trigonometric seasonality, Box-Cox transformation, ARMA errors, Trend, Seasonal components (TBATS) [59]. Grid search is also utilized within the ML family, e.g., for HPO of point forecasting methods based on SVR [60,61]. While effective for small configuration spaces, higher-dimensional spaces require HPO algorithms that explore the configuration space more efficiently. For example, random search is more efficient than grid search [75] and is widely applied for HPO of ML-based point forecasting methods such as the Gradient Boosting Machine (GBM) [63] and the MultiLayer Perceptron (MLP) [62]. For a directed search of complex configuration spaces, such as for methods of the DL family, gradient-based search methods [64], Bayesian Optimization (BO) [65], and Evolutionary Algorithms (EAs) [66] are used. Aside from point forecasting methods, these HPO techniques can also be applied to direct [29,40–42] and point-forecast-based [56] probabilistic forecasting methods. Although HPO of a single forecasting method is effective, it neglects the no-free lunch theorem, stating that no single forecasting method excels at all forecasting tasks.

Combined algorithm selection and hyperparameter optimization. To select the best-performing forecasting method with optimized hyperparameters, several CASH methods are available for point forecasting. In [23], the authors take different ML-based forecasting methods into account, including Least Absolute Shrinkage and Selection Operator (LASSO), Random Forest (RF), eXtreme Gradient Boosting (XGB), MLP, and SVR, and use BO for solving the CASH problem. Similar approaches exist in [24,25], additionally considering point forecasting methods of the SM family (e.g., AutoRegressive Integrated Moving Average (ARIMA), Triple Exponential Smoothing (TES), and Box-Cox transformation, ARMA errors, Trend, Seasonal components (BATS)) and the DL family (e.g., Neural Hierarchical Interpolation for Time Series (N-HiTS) and Temporal Fusion Transformer (TFT)). Apart from the application to point forecasting, CASH methods are also used for direct and point forecast-based probabilistic forecasting [26]. In their CASH problem, the authors consider several direct probabilistic forecasting methods, like DeepAR, as well as point forecasting methods of the SM and ML family using conformal PIs to derive probabilistic forecasts. Despite their contribution to addressing the under-researched area of CASH for probabilistic forecasting, the applied uncertainty quantification methods are still subject to the aforementioned limitations, i.e., they require assuming the underlying PDF and do not take into account the conditioning of the PDF by exogenous features.

2.3. Computing effort for methodological advances

Regarding the quantification of efforts required for performance improvements, the above-reviewed related works exhibit significant shortcomings. While some of the studies specify the hardware used and required training duration [23,25,26,28,36,39,44,49,52,60,64], estimating the corresponding energy consumption from this information involves considerable effort, with inherent uncertainty. Only in [44] the electricity consumed during training is measured; however, the values are normalized in comparison to their initial value and the absolute values are not provided. Yet, the electricity consumption is an essential metric as it is independent of data center-specific or geopolitical factors at the time of measurement [27]. Once electricity consumption is quantified, other metrics, such as the carbon footprint or the monetary costs, can be derived using assumed or local conditions.

[2] For more details on such a bijective mapping, refer to Section 3.2.

[3] For more details on conformal prediction, refer to [71].

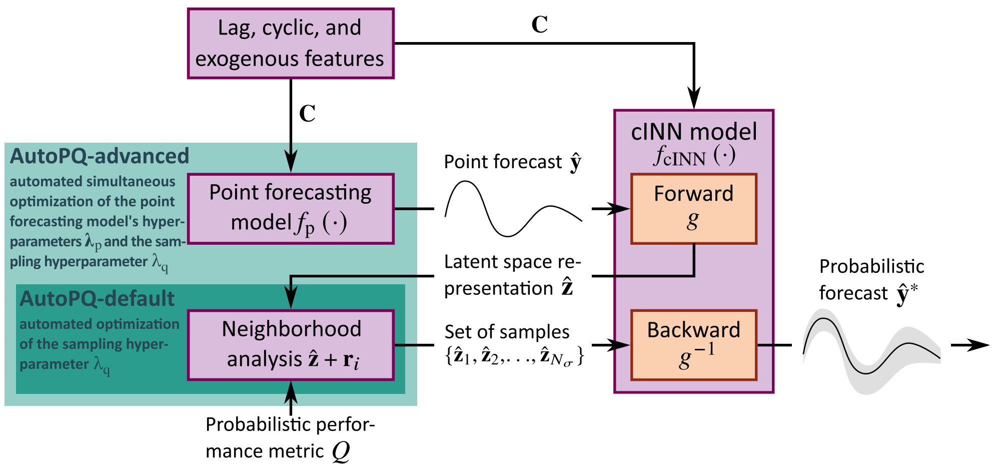

Lag, cyclic, and exogenous features
C
C
AutoPQ-advanced
automated simultaneous optimization of the point forecasting model's hyper-parameters λ_p and the sampling hyperparameter λ_q
Point forecasting model $f_p(\cdot)$
AutoPQ-default
automated optimization of the sampling hyper-parameter λ_q
Neighborhood analysis $\hat{\mathbf{z}} + \mathbf{r}_i$
Probabilistic performance metric Q
Point forecast $\hat{\mathbf{y}}$
Latent space representation $\hat{\mathbf{z}}$
Set of samples $\{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_{N_\sigma}\}$
cINN model $f_{\text{cINN}}(\cdot)$
Forward g
Backward g^{-1}
Probabilistic forecast $\hat{\mathbf{y}}^*$

Fig. 1. Overview of `AutoPQ`: Lag features, seasonal features, and exogenous features are selected and used as inputs by a point forecasting model to generate a point forecast in an unknown conditional distribution. This point forecast and the features are combined in a cINN, resulting in a representation of the forecast in a known and tractable conditional 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 conditional distribution to generate the probabilistic forecast. Automation methods are highlighted in green.

3. `AutoPQ`

To address the three key challenges in the automated design of probabilistic time series forecasting models, we present `AutoPQ`, an innovative method for electricity consumption-aware quantile forecasting based on point forecasts. The idea behind `AutoPQ` is to use a cINN [30] to generate a probabilistic forecast from any arbitrary point forecast [21] while automatically making corresponding design decisions to enhance probabilistic performance. In the following section, we describe (i) the generation of such a probabilistic forecast, (ii) details of the cINN, (iii) the corresponding design decisions, and (iv) the automation of these decisions, highlighted in green in Fig. 1. To accommodate various computing systems and performance requirements, we propose two variants of `AutoPQ`: `AutoPQ-default`, designed for general-purpose computing systems and capable of providing high-quality probabilistic forecasts [21], and `AutoPQ-advanced`, which requires HPC systems[4] to further increase forecasting quality for downstream applications with high decision costs.

3.1. *Creation of the probabilistic forecast*

Generating a probabilistic forecast with a cINN using a point forecast involves three steps [21].[5]

Creation of a point forecast in an unknown conditional distribution. A point forecast is generated using an arbitrary point forecasting model $f_\text{p}(\cdot)$, which estimates future values $\hat{\mathbf{y}}$ of the target time series $\mathbf{y}$ using current and past values. More precisely, the model $f_\text{p}(\cdot)$ makes a forecast for a specified forecast horizon H at origin k

$$\hat{\mathbf{y}}[k+1,\ldots,k+H] = f_\text{p}\Big(\mathbf{y}\left[k-H_1,\ldots,k\right],$$

$$\mathbf{X}\left[k-H_1,\ldots,k\right], \hat{\mathbf{X}}[k+1,\ldots,k+H], \mathbf{p}_\text{p}\Big), \tag{1}$$

$$k, H, H_1 \in \mathbb{N}_1, k > H_1,$$

taking into account historical values of $\mathbf{y}$ up to the past horizon H_1, values of exogenous forecasts $\hat{\mathbf{X}}$, and exogenous time series $\mathbf{X}$, where $\mathbf{p}_\text{p}$ are the trainable parameters of the model. Since the target time series $\mathbf{y}$ contains realizations of $\hat{\mathbf{y}}$, a point forecast with the horizon H can be interpreted as a sample of the random variable $\mathrm{Y} \sim f_\mathrm{Y}(\mathbf{y})$ in the realization space $\mathbb{Y}$ with an unknown H-dimensional conditional Probability Density Function (cPDF) $f_\mathrm{Y}(\mathbf{y})$.

Representation of the point forecast in a known and tractable conditional distribution. The point forecast $\hat{\mathbf{y}}[k+1,\ldots,k+H]$ as a sample from the unknown cPDF is represented in a space with a known and tractable H-dimensional cPDF $f_\mathrm{Z}(\mathbf{z})$ using a cINN. The cINN learns a conditional[6] and bijective function $g : \mathbb{Y} \to \mathbb{Z}$ [77] from the realization space $\mathbb{Y}$ with unknown cPDF into the latent space $\mathbb{Z}$ with the H-dimensional Gaussian cPDF.[7] This bijective function can be used to quantify the point forecast's uncertainty by mapping it into the latent space and analyzing its neighborhood as described in the following. Notably, the training of the cINN is independent of the point forecasting model, which allows using one cINN to generate probabilistic forecasts from multiple arbitrary point forecasts.

Neighborhood analysis of the point forecast's latent space representation. To obtain the point forecast's latent space representation, the point forecast $\hat{\mathbf{y}}[k+1,\ldots,k+H]$ itself together with additional temporal and exogenous information

$$\mathbf{C} = \left\{\mathbf{y}\left[k-H_1,\ldots,k\right], \mathbf{X}\left[k-H_1,\ldots,k\right], \hat{\mathbf{X}}[k+1,\ldots,k+H]\right\} \tag{2}$$

are passed forward through the trained cINN to obtain the latent space representation $\hat{\mathbf{z}}[k+1,\ldots,k+H]$. To quantify the point forecast's uncertainty, the neighborhood of this representation is analyzed. More specifically, we sample in the Gaussian-distributed latent space around the point forecast's latent space representation by adding a random noise vector $\mathbf{r}_i$ from a standard normal distribution with mean 0 and variance σ to the realization $\hat{\mathbf{z}}$

$$\hat{\mathbf{z}}_i = \hat{\mathbf{z}} + \mathbf{r}_i, \quad i \in \{1,\ldots,N_\sigma\}, \quad \mathbf{r}_i \sim \mathcal{N}(0,\sigma) \tag{3}$$

to generate a set of samples $\{\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, \ldots, \hat{\mathbf{z}}_{N_\sigma}\}$. These samples are similar but not identical to the original point forecast and represent the uncertainty in the neighborhood of the point forecast's latent space

[4] In this paper, we refer to servers for parallel computing with multiple nodes and Graphics Processing Unit (GPU) acceleration as HPC systems.

[5] For a detailed description, we refer to [21].

[6] The cINN can include additional exogenous information into the mapping using conditional-affine `Glow` coupling layers [76] as detailed below.

[7] The latent space cPDF has a separate dimension for each value within the forecast horizon H, as each value can be subject to different uncertainties.

representation. To this end, the samples can be passed backward through the cINN to calculate the quantiles of the probabilistic forecast $\hat{\mathbf{y}}^*[k+1,\dots,k+H]$.[8] Details regarding the sample generation are visualized in Fig. 12 in the Appendix A.

3.2. Details of the conditional invertible neural network

To implement the bijective function $g : \mathbb{Y} \rightarrow \mathbb{Z}$ from the realization space $\mathbb{Y}$ to the latent space $\mathbb{Z}$, we employ the cINN [30,77].

Architecture. The cINN is composed of multiple specially designed conditional-affine `Glow` coupling layers [76] with the conditioning inputs of those layers provided by a fully connected neural network [30,77]. This fully connected neural network leverages conditional information to account for the temporal and exogenous dependencies of the time series (see paragraph below on lag, cyclic, and exogenous features). Architecture details of the employed cINN can be found in Table 6 in the Appendix A.

Bijective mapping. The cINN's bijective mapping from the unknown cPDF $f_Y(\mathbf{y})$ to the known and tractable cPDF $f_Z(\mathbf{z})$ is described by the change of variable formula [77]

$$f_Z(\mathbf{z}) = f_Y\left(g^{-1}\left(\mathbf{z}[k+1,\dots,k+H], \mathbf{C}, \mathbf{p}_{\text{cINN}}\right)\right) |\det(J)| \tag{4}$$

with additional temporal and exogenous information $\mathbf{C}$ [30], where $J = \partial g^{-1}/\partial\mathbf{z}[k+1,\dots,k+H]$ is the Jacobian matrix, and $\mathbf{p}_{\text{cINN}}$ are the trainable parameters of the cINN.

Training. Training the cINN targets to learn the function g, ensuring that the latent space realizations $\mathbf{z} = g(\mathbf{y}, \mathbf{X}, \hat{\mathbf{X}}, \mathbf{p}_{\text{cINN}})$ follow a known and tractable H-dimensional Gaussian cPDF.[9] Therefore, the cINN is trained with the change of variables formula (4) in the maximum likelihood loss

$$\mathcal{L} = \mathbb{E}\left[\frac{1}{2}\left\|g\left(\mathbf{y}[k+1,\dots,k+H], \mathbf{C}, \mathbf{p}_{\text{cINN}}\right)\right\|_2^2 - \log|\det(J)|\right] + \lambda_{\mathcal{L}}\left\|\mathbf{p}_{\text{cINN}}\right\|_2^2, \tag{5}$$

using L2 regularization $\lambda_{\mathcal{L}}\|\mathbf{p}_{\text{cINN}}\|_2^2$ [30], where $J = \partial g/\partial\mathbf{y}[k+1,\dots,k+H]$ is the Jacobian matrix, and $\mathbf{p}_{\text{cINN}}$ are the trainable parameters of the cINN.

3.3. Task-dependent design decisions

Generating a quantile forecast requires specific design decisions tailored to the forecasting task. These include feature engineering, selecting the point forecasting method and its hyperparameters λ_{p}, and choosing the sampling variance $\lambda_{\text{q}} = \sigma$ for generating a quantile forecast with the cINN based on the point forecast's latent space representation.

Lag, cyclic, and exogenous features. Additional explanatory variables can enrich the input space of both the point forecasting model and the cINN. For their selection, prior knowledge and correlation analyses are used.

Lag features provide past values for the model as historical context. In the evaluation, point forecasting methods based on ML regression estimators require the explicit definition of lag features

$$x_{\text{lag},H_1}[k] = x\left[k - H_1\right], \tag{6}$$

unlike SM- and DL-based forecasting methods that consider past values implicitly. To guarantee a fair evaluation of the forecasting methods used, all lag features that lie within the past horizon of the SM and DL methods are provided for the ML methods.

[8] This is valid due to the equivalence of uncertainty in both spaces [21].

[9] The latent space cPDF has a separate dimension for each value within the forecast horizon H.

Seasonal features represent cyclic relationships, such as the hour of the day or the month,

$$x_{\text{s12}}[k] = \sin\left(\frac{2\pi \cdot \text{month}[k]}{12}\right), x_{\text{c12}}[k] = \cos\left(\frac{2\pi \cdot \text{month}[k]}{12}\right), \tag{7}$$

$$x_{\text{s24}}[k] = \sin\left(\frac{2\pi \cdot \text{hour}[k]}{24}\right), \quad x_{\text{c24}}[k] = \cos\left(\frac{2\pi \cdot \text{hour}[k]}{24}\right), \tag{8}$$

as well as work and non-work days:

$$x_{\text{wd}}[k] = \begin{cases} 1, & \text{if } \texttt{work day}(k) \text{ is True,} \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Since energy time series for smart grid applications contain predominantly daily, weekly, and yearly patterns [78–80], we draw on this prior knowledge and consider the above seasonal features in the evaluation.

Exogenous features are important if the target variable is influenced by external factors, such as weather conditions including temperature, humidity, wind speed, and solar irradiance. We thus select exogenous features depending on existing additional variables in the datasets and their correlation with the target variable. This selection is based on previous work [21] (`AutoPQ-default`) to ensure comparability in the benchmarking.[10]

Point forecasting methods and hyperparameters. As the point forecasting method in the above-described cINN approach significantly affects the probabilistic forecasting performance [21], we consider established methods from different forecasting method families in order to obtain a representative selection of methods that have proven effective and differ in their approach or architecture.

Statistical Modeling (SM): sARIMAX, ETS, TBATS,
Machine Learning (ML): MLP, SVR, XGB,
Deep Learning (DL): DeepAR, N-HiTS, TFT.

Since HPO can enhance performance beyond a method's default configuration, we apply HPO on the configuration spaces detailed in Tables 10–12 in the Appendix A.

Sampling hyperparameter of the latent space. The hyperparameter $\lambda_{\text{q}} = \sigma$ for generating N_σ samples[11] from the point forecast's latent space representation controls the magnitude of quantified uncertainty in the forecast: a larger σ includes more uncertainty and improves coverage rates, while a smaller σ produces sharper probabilistic forecasts. Both coverage rate and sharpness can be evaluated using probabilistic performance metrics, enabling HPO to tailor the sampling hyperparameter $\lambda_{\text{q}} = \sigma$ to the forecasting task [81]. The considered value range is given in Table 7 in the Appendix A.

3.4. Optimal design of the probabilistic forecast

`AutoPQ` automates the design decisions outlined above by optimally configuring the forecasting model. We propose `AutoPQ-default` for general-purpose computing systems to provide high-quality probabilistic forecasts, and `AutoPQ-advanced` for HPC systems to enhance the performance further.

3.5. `AutoPQ-default` for general-purpose computing systems

Given the high computational effort of HPO for point forecasting methods illustrated in Fig. 2, `AutoPQ-default` only focuses on optimizing the sampling hyperparameter to provide high-quality point forecast-based quantile forecasts using general-purpose computing systems for the model design.

[10] The selected exogenous features for the benchmarking are provided in Table 13 in the Appendix A.

[11] In the evaluation, we choose $N_\sigma = 100$ based on [21], as it is a one-sided insensitive hyperparameter: smaller values worsen the quality and larger values do not contribute to significant quality improvements.

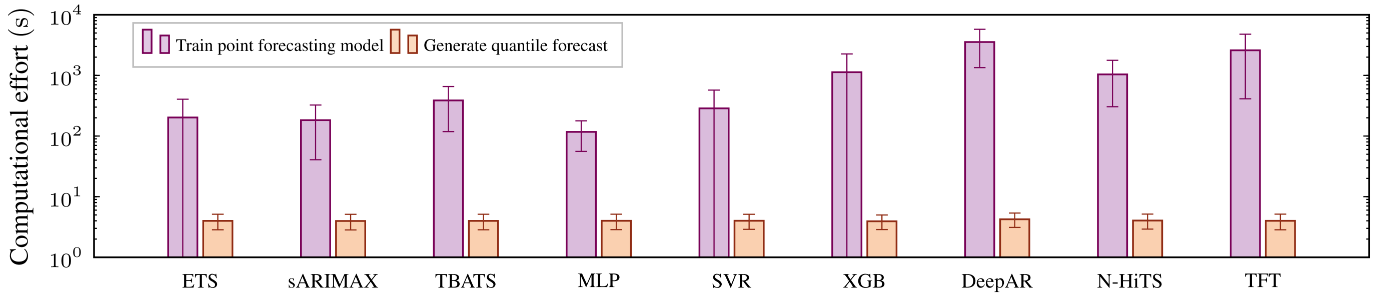


Fig. 2. Comparison of the average computational effort in terms of runtime across the six datasets used in the evaluation: effort for training a point forecasting model with configuration λ_p and effort for generating a quantile forecast based on the point forecast using the cINN with λ_q. Due to the 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.

Hyperparameter optimization. Instead of manually exploring a given configuration space $\boldsymbol{\Lambda}_\mathrm{q}$ to find the (estimated) optimal hyperparameter configuration $\lambda^{\hat{\star}}_\mathrm{q}$, we define the HPO problem

$$\lambda^{\hat{\star}}_\mathrm{q} = \min_{\lambda\in\boldsymbol{\Lambda}_\mathrm{q}} Q\left(\hat{\mathbf{y}}(\lambda_\mathrm{q}), \mathbf{y}\right), \tag{10}$$

with an arbitrary probabilistic performance metric as objective function Q and the hyperparameter λ_q for generating samples from the point forecast's latent space representation.

Optimization algorithm. Given that the objective function Q Eq. (10) is potentially noisy and non-differentiable and we do not have a closed-form expression of the configuration space $\boldsymbol{\Lambda}_\mathrm{q}$, we use the black-box optimizer `Hyperopt` [82]. `Hyperopt` acts as the HPO trial generator suggesting new candidate configurations to be assessed. The suggestion is based on BO using the Tree Parzen Estimator (TPE) as surrogate model, which balances exploration and exploitation of promising regions in the configuration space. The choice of `Hyperopt` is justified as follows: first, the TPE surrogate model allows the integration of prior knowledge, which is employed for `AutoPQ-advanced`. Second, the sequential algorithm allows for immediate termination as soon as a performance plateau is reached (early stopping). Third, `Hyperopt` is significantly more efficient than undirected sequential methods such as random search, as shown in ablation study 2.

Early stopping. The optimization can be stopped once the objective function's value Q has converged or reached a satisfactory value. To terminate the BO when additional trials are unlikely to improve Q, we use early stopping upon Q reaching a plateau, defined as the standard deviation of the five best trials falling below 0.0005, with a patience of five trials.[12]

Forecasting method selection. Probabilistic forecasting performance depends on not only the hyperparameter λ_q but also the underlying point forecasting method. In order to address this, `AutoPQ-default` selects the best-performing point forecasting method while optimizing the corresponding λ_q. First, all candidate point forecasting models are trained with default configurations $\boldsymbol{\lambda}_\mathrm{p,default}$. A quantile forecast is then generated for each model, with the associated sampling hyperparameter λ_q optimized with the previously described HPO. Finally, the combination of point forecasting model and optimized sampling hyperparameter achieving the best score on the validation dataset is selected.

3.5.1. *`AutoPQ-advanced` for HPC systems*

In smart grid applications with high decision costs, improving probabilistic forecasting performance is vital. `AutoPQ-advanced` addresses this requirement using HPC systems for the model design as detailed in the following.

Hyperparameter optimization. Similarly to `AutoPQ-default` Eq. (10), the HPO problem

$$\boldsymbol{\lambda}^{\hat{\star}} = \min_{\boldsymbol{\lambda}\in\boldsymbol{\Lambda}} Q\left(\hat{\mathbf{y}}(\boldsymbol{\lambda}_\mathrm{p}, \lambda_\mathrm{q}), \mathbf{y}\right) \tag{11}$$

is defined, with an arbitrary probabilistic performance metric as the objective function Q. In contrast to `AutoPQ-default`, `AutoPQ-advanced` considers the point forecasting method's hyperparameter configuration $\boldsymbol{\lambda}_\mathrm{p}$ together with the sampling hyperparameter λ_q for generating samples from the point forecast's latent space representation.

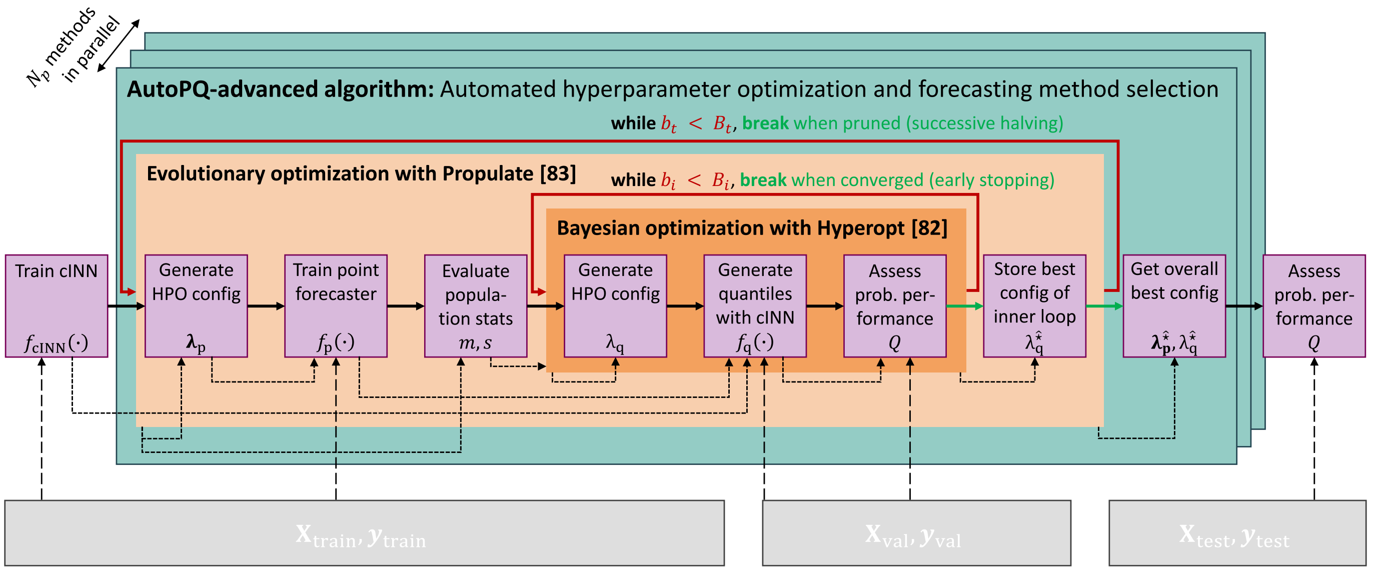
In order to solve (11) efficiently, the computing times between training the point forecasting model and generating a quantile forecast based on this point forecast must be properly balanced. Fig. 2 compares these efforts across the six evaluation datasets. Since generating a quantile forecast can be repeatedly refined with different sampling hyperparameters, we use two nested loops to solve (11) as outlined in Fig. 3. The following paragraphs detail this joint optimization, the application of Prior Knowledge (PK) during optimization, and the implementation details of the chosen algorithms.

Joint optimization. Algorithm 1 illustrates the nested joint HPO applied in `AutoPQ` with the total time budget B_t. The outer loop aims to identify the optimal hyperparameter configuration for the point forecasting method $\boldsymbol{\lambda}^{\hat{\star}}_\mathrm{p}$. First, the outer loop trial generator suggests a candidate $\boldsymbol{\lambda}_\mathrm{p}$. Importantly, after the random trial generator initialization, this suggestion is based on the probabilistic forecasting performance Q of previous trials and not on their point forecasting performance. Then, the point forecasting model $f_\mathrm{p}(\cdot)$ is trained. Afterward, the inner loop optimizes the corresponding sampling hyperparameter $\lambda^{\hat{\star}}_\mathrm{q}$ by solving (10). Specifically, the inner loop trial generator proposes a candidate λ_q, and the quantile forecast $f_\mathrm{q}(\cdot)$ is generated by passing the point forecast of $f_\mathrm{p}(\cdot)$ forward through the cINN, generating samples from the point forecast's latent space representation, and subsequently passing them backward through the cINN. Then, the probabilistic forecasting performance Q is assessed, and the inner loop generator is updated with Q and λ_q. The inner loop continues until an early stopping condition is met or the iteration budget B_i is exhausted, yielding the optimal sampling hyperparameter $\lambda^{\hat{\star}}_\mathrm{q}$ associated with $\boldsymbol{\lambda}_\mathrm{p}$. Note that the duration of computing the iterations B_i is billed to the total time budget B_t. Before starting a new outer loop iteration, the outer trial generator is updated with the probabilistic forecasting performance Q, $\boldsymbol{\lambda}_\mathrm{p}$, and $\lambda^{\hat{\star}}_\mathrm{q}$ to suggest a new candidate $\boldsymbol{\lambda}_\mathrm{p}$ for the point forecasting model. It continues until the total time budget B_t is exhausted, ultimately yielding the optimal configuration, i.e., $\boldsymbol{\lambda}^{\hat{\star}}_\mathrm{p}$ and the associated $\lambda^{\hat{\star}}_\mathrm{q}$.[13]

Prior knowledge. In initializing the inner trial generator, we leverage PK by assuming that the optimal sampling hyperparameter $\lambda^{\hat{\star}}_\mathrm{q}$ for a given point forecasting method configuration $\boldsymbol{\lambda}_\mathrm{p}$ is related to previously

[12] Note that the assessment is performed on normalized data, i.e., the values of Q across datasets are in similar ranges.

[13] The time budget refers to creating a model for one dataset.



N_p methods in parallel
AutoPQ-advanced algorithm: Automated hyperparameter optimization and forecasting method selection
while $b_t < B_t$, break when pruned (successive halving)
Evolutionary optimization with Propulate [83]
while $b_i < B_i$, break when converged (early stopping)
Bayesian optimization with Hyperopt [82]
Train cINN $f_{cINN}(\cdot)$
Generate HPO config λ_p
Train point forecaster $f_p(\cdot)$
Evaluate population stats m, s
Generate HPO config λ_q
Generate quantiles with cINN $f_q(\cdot)$
Assess prob. performance Q
Store best config of inner loop $\lambda_q^{\hat{\star}}$
Get overall best config $\lambda_p^{\hat{\star}}, \lambda_q^{\hat{\star}}$
Assess prob. performance Q
X_{train}, y_{train}
X_{val}, y_{val}
X_{test}, y_{test}

Fig. 3. The automated Hyperparameter Optimization (HPO) and forecasting method selection of `AutoPQ-advanced`: the hyperparameters of the point forecasting method $\boldsymbol{\lambda}_p$ and the sampling hyperparameter of the cINN λ_q are optimized jointly, see paragraph “Joint optimization”. For initializing the inner loop with the Bayesian optimization, the algorithm uses statistical parameters derived from the population of the evolutionary optimization, see paragraph “Prior knowledge”. The inner loop is stopped, when the performance reaches a plateau across trials, see paragraph “Early stopping”. During the parallel HPO all the N_p point forecasting methods, underperforming methods are pruned and resources are re-allocated to more promising ones, see paragraph “Forecasting method selection”. As long as a method is active, it uses its allocated resources to compute the inner and outer loops sequentially, i.e., the duration of the iterations B_i is billed to the total time budget B_t.

Algorithm 1 The `AutoPQ` HPO algorithm optimizes the hyperparameters of the point forecasting method and the sampling hyperparameter of the cINN jointly, taking into account the significantly lower computational effort of the latter.

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

```
    # Initialize Propulate [83] (EA)
 1: trial_generator_p($\boldsymbol{\Lambda}_p$)
 2: $b_t \leftarrow 0$ s
 3: while $b_t < B_t$ do
 4:     $t \leftarrow$ get_current_time()
 5:     $\boldsymbol{\lambda}_p \leftarrow$ trial_generator_q.get_config()
        # High computational effort
 6:     $f_p(\cdot) \leftarrow$ train_p_model($\boldsymbol{\lambda}_p, \mathbf{X}_{train}, \mathbf{y}_{train}$)
        # Distribution of $\lambda_q^{\hat{\star}}$ in active population
 7:     $m, s \leftarrow$ trial_generator_p.get_stats()
        # Initialize Hyperopt [82] (BO) with PK
 8:     trial_generator_q($\Lambda_q, m, s$)
 9:     for $b_i \leftarrow 0$ to $B_i$ do
10:         $\lambda_q \leftarrow$ trial_generator_q.get_config()
            # Low computational effort
11:         $f_q(\cdot) \leftarrow$ generate_q_model($f_p(\cdot), f_{cINN}(\cdot), \lambda_q$)
            # Low computational effort
12:         $Q \leftarrow$ assess_performance($f_q(\mathbf{X}_{val}, \mathbf{y}_{val})$)
13:         trial_generator_q.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:     $\lambda_q^{\hat{\star}} \leftarrow$ trial_generator_q.get_best_config()
        # Store $\lambda_q^{\hat{\star}}$ associated with $\boldsymbol{\lambda}_p$
20:     trial_generator_p.update($Q, \boldsymbol{\lambda}_p, \lambda_q^{\hat{\star}}$)
21:     $b_t \leftarrow b_t +$ (get_current_time() $- t$)
22: end while
23: $\boldsymbol{\lambda}_p^{\hat{\star}}, \lambda_q^{\hat{\star}} \leftarrow$ trial_generator_p.get_best_config()
```

Output: $\boldsymbol{\lambda}_p^{\hat{\star}}, \lambda_q^{\hat{\star}}$

$\boldsymbol{\Lambda}_p, \Lambda_q$: configuration space (point forecast, quantile forecast); B_i, B_t: budget (iteration, time); $\mathbf{X}_{train}, \mathbf{X}_{val}$: model inputs (training, validation); $\mathbf{y}_{train}, \mathbf{y}_{val}$: model outputs (training, validation); f_{cINN}, f_p, f_q: trained model (cINN, point forecast, quantile forecast); $\boldsymbol{\lambda}_p^{\hat{\star}}, \lambda_q^{\hat{\star}}$: optimal configuration (point forecast, quantile forecast)

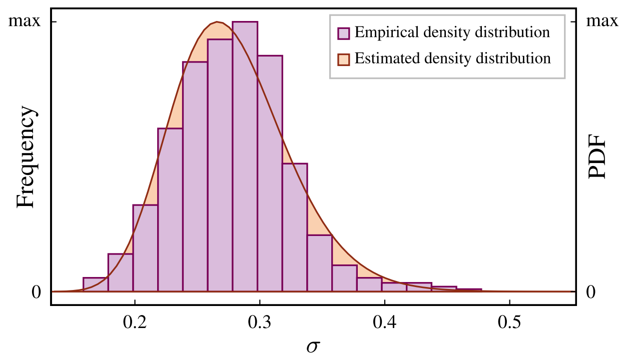
assessed configurations. Consequently, the inner trial generator begins with statistical parameters derived from these past configurations.[14]

Optimization algorithms. Since we do not have a closed-form expression of the configuration space $\boldsymbol{\Lambda}$ in Eq. (11) and the objective function Q is noisy, non-differentiable, and potentially non-convex, solving (11) requires black-box optimization methods. These methods serve as trial generators, suggesting new candidate configurations to be evaluated in both the inner and outer loop of Algorithm 1.

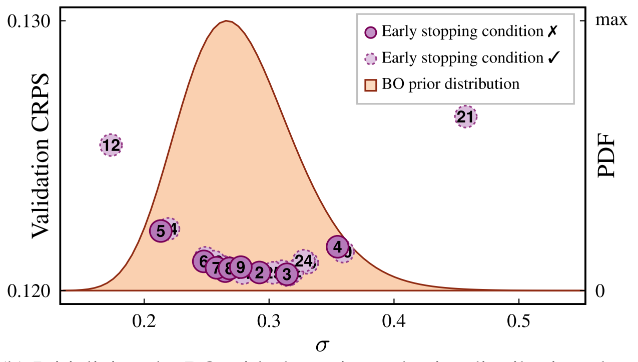
Training the point forecasting model in the **outer loop** is computationally expensive and benefits from GPU acceleration. Using multiple GPU enables evaluating multiple hyperparameter configurations $\boldsymbol{\lambda}_p$ (i.e., multiple trials of Q) in parallel. To handle varying training durations and avoid GPU idle time, asynchronous optimization is applied. For this purpose, we employ the Python package `Propulate` [83] as the outer trial generator. This evolutionary optimizer maintains a continuous population for asynchronous configuration assessments,[15] which also enables incorporating PK into the inner loop.

Generating a probabilistic forecast based on a point forecast in the **inner loop** is computationally cheap. Additionally, leveraging PK enables rapid convergence, often resulting in early stopping. Hence, parallelizing the inner loop would add unnecessary overhead without reducing computation time. Instead, the inner loop runs sequentially using the Python package `Hyperopt` [82] as trial generator.[16] `Hyperopt` employs BO with the TPE as the surrogate model to explore and exploit the configuration space by estimating the distribution of well-performing hyperparameter configurations in relation to underperforming ones [84]. To initialize `Hyperopt`, we use PK obtained from the outer loop trial generator `Propulate`. Specifically, the mean m and standard deviation s of the optimal sampling hyperparameters' natural logarithms are computed from

[14] These values are stored during outer trial generator updates.
[15] The EA's setup is based on [83] and detailed in Table 8 in the Appendix A.
[16] The choice of `Hyperopt` is justified above in the section of `AutoPQ-default`.



(a) Estimated 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. 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 provides a good estimate for the first trials, before the TPE surrogate model fitted on these trials gains influence. To illustrate the progression after the early stopping condition is met at the ninth iteration, the BO was continued until the 25th iteration (dashed dots), with each dot representing an iteration. After reaching this condition, sampling also continues at the edges of the distribution (iterations 12 and 21).

Fig. 4. Estimated BO prior distribution and its utilization in the HPO of the sampling hyperparameter.

already assessed candidates in Propulate's population (see Fig. 4(a)). These values are then used to define Hyperopt's prior distribution as $\log - \mathcal{N}(m, s)$. Assuming a log-normal distribution excludes negative values and is valid, as demonstrated in Fig. 4(a) with the MLP on the Load-BW dataset. A random set of λ_q values is initially drawn from this prior distribution to explore regions with high prior probabilities before fitting the TPE surrogate model. Afterward, the fitted TPE is used to schedule new candidates of $\{\lambda_{q,0}, \lambda_{q,1}, \dots\}$ through an acquisition function that balances exploration and exploitation. Each assessed $\lambda_{q,i}$ (i.e., each trial of Q) updates the TPE surrogate model with its performance Q, reducing the prior distribution's influence over iterations. Thus, values in regions with lower prior probabilities are also considered, unless early stopping occurs beforehand, as demonstrated in Fig. 4(b) with the MLP on the Load-BW dataset.

Early stopping. As with `AutoPQ-default`, optimization is stopped when the objective function's value Q has converged or reached a satisfactory value. The inner loop is terminated if Q reaches a plateau across trials, i.e., the standard deviation of the five best-performing trials is less than 0.0005 with a patience of five trials.

Forecasting method selection. `AutoPQ-advanced` aims to select the best point forecasting method while jointly optimizing its hyperparameters

Algorithm 2 The successive halving pruning strategy re-allocates computing resources from unpromising areas in the CASH configuration space to more promising ones.

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

```
      # Initialize result store
 1:  result_store($\boldsymbol{\Lambda}$)
 2:  while len($\boldsymbol{\Lambda}$) > 1 do
 3:    for $\boldsymbol{\Lambda}_i$ in $\boldsymbol{\Lambda}$ do
          # Parallel evaluation, split computation budget
 4:       $B_{t,i} \leftarrow B_t$/len($\boldsymbol{\Lambda}$)
          # Initialize HPO algorithm with stored results
 5:       hpo_algo(result_store)
          # Run until budget is exhausted
 6:       hpo_algo.run($\boldsymbol{\Lambda}_i, \mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}, \mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}, B_{t,i}$)
 7:       $Q_i^\star, \boldsymbol{\lambda}_i^\star \leftarrow$ hpo_algo.get_best_config($\boldsymbol{\Lambda}_i$)
          # Update result store
 8:       result_store.update($Q_i^\star, \boldsymbol{\lambda}_i^\star$)
 9:    end for
       # Shrink configuration space
10:    $\boldsymbol{\Lambda} \leftarrow$ result_store.get_best_half()
11:  end while
```

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

$\boldsymbol{\Lambda}$: configuration space; B_t: time budget; $\mathbf{X}_{\text{train}}, \mathbf{X}_{\text{val}}$: model inputs (training, validation); $\mathbf{y}_{\text{train}}, \mathbf{y}_{\text{val}}$: model outputs (training, validation); $\boldsymbol{\lambda}^\star$: optimal configuration

λ_p and the corresponding sampling hyperparameter λ_q. To efficiently combine method selection and joint HPO, we use successive halving pruning. This method prunes underperforming configurations and re-allocates resources to more promising ones. Starting from the considered candidate forecasting methods, successive halving (Algorithm 2) iteratively eliminates the worst-performing methods and re-allocates resources to the better-performing ones, until one top candidate remains when the total time budget B_t is exhausted. Crucially, HPO for the remaining candidates continues from checkpoints of the previous pruning round, allowing optimization runs to be split over multiple HPC jobs.

4. Evaluation

To evaluate `AutoPQ`[17] comprehensively, we first assess its probabilistic forecasting performance with default and advanced configurations to analyze the impacts of HPO on improving the probabilistic performance. Second, we conduct an ablation study to evaluate the effectiveness of `AutoPQ-advanced`, which jointly optimizes the hyperparameters of the point forecaster and the cINN's sampling hyperparameters. Third, we compare the computational efforts of `AutoPQ-default` and `AutoPQ-advanced`.

4.1. Benchmarking

The subsequent section outlines the experimental setup for assessing probabilistic forecasting performance, followed by a presentation of the results and key insights gained from the analysis.

4.1.1. Experimental setup

In the following, we detail the data utilized, the evaluation strategy employed, and the benchmarks considered.

Data. `AutoPQ` is evaluated on six distinct datasets: Load-BW, Load-GCP, Mobility, Price, PhotoVoltaic (PV), and Wind Power (WP). Each dataset is briefly described below, followed by details on pre-processing and feature selection. Additional information is provided in Table 13 in the Appendix A.

The **Load-BW** dataset [85] includes the gross electrical consumption of Baden-Württemberg, Germany, obtained from the Open Power System Data (OPSD) platform. The regional electrical consumption

[17] A Python implementation of AutoPQ can be found on GitHub: https://github.com/SMEISEN/AutoPQ.

reflects the aggregation of a large number of Grid Connection Points (GCPs).

The **Load-Grid Connection Point (GCP)** dataset [86] contains the electrical consumption time series of 370 GCPs within a distribution grid in Portugal. The time series `MT_158` is used to generate local forecasts, representing the local load of a single GCPs.

The **Mobility** dataset [87] comprises hourly records of bike rentals in Washington D.C., USA, which serve as an indicator of individual mobility within the smart grid context. Relevant weather and seasonal information such as air temperature, wind speed, and whether a day is a workday, weekend or holiday, are included.

The **Price** dataset features zonal electricity prices from an unspecified location, provided in the Global Energy Forecasting Competition (GEFCom) 2014 [88] price forecasting track. This challenge involved 14 tasks requiring day-ahead forecasts based on exogenous time series data, including total electrical consumption and zonal price. All 14 tasks are combined into one dataset for the evaluation.

The **PV** dataset includes the power generation of an Australian PhotoVoltaic (PV) plant, sourced from the GEFCom 2014 [88] PV power forecasting track. This track comprised 16 day-ahead forecasting tasks, supplemented by Numerical Weather Predictions (NWPs) for the site from European Centre for Medium-Range Weather Forecasts (ECMWF), including Global Horizontal Irradiance (GHI) and cloud cover. For the evaluation, all 16 tasks are combined into one dataset.

The **WP** dataset contains the power generation of an Australian wind farm, taken from the GEFCom 2014 [88] Wind Power (WP) forecasting track. This track consisted of 16 day-ahead forecasting tasks, also with NWPs for the site from ECMWF available, including wind speed and direction at heights of 10 m and 100 m. All 16 tasks are combined into one dataset for the evaluation.

Originally, all datasets considered are hourly resolved, except for the Load-GCPs dataset, which was resampled to hourly resolution.

Evaluation strategy. We evaluate the day-ahead forecasting performance of `AutoPQ` on the six datasets described above, comparing both `AutoPQ-default` and `AutoPQ-advanced` configurations. To ensure comparability across the datasets, we normalize each dataset before creating separate training, validation, and test sub-sets.[18] Detailed sample indices for these splits and selected exogenous features are provided in Table 13 in the Appendix A.

The cINN with the architecture detailed in Table 6 is trained on the training data set using the `Adam` optimizer [89] with a maximum of 100 epochs and L2 regularization [77,90]. The training of the cINN is performed prior to AutoPQ's optimization and completely independently of the point forecast, see Fig. 3. Consequently, the cINN must not be retrained if the point forecast or its hyperparameters are altered.

We consider nine different point forecasting methods, choosing three from each method family: sARIMAX, ETS, and TBATS from SM; MLP, SVR, and XGB from ML; and DeepAR, N-HiTS, and TFT from DL. For `AutoPQ-default`, we train all nine methods with their default configurations $\lambda_{\text{p,default}}$, generate probabilistic forecasts from the point forecasts, and optimize the sampling hyperparameter λ_{q} for each model, where the best-performing one is selected afterward.[19] These results are compared to our evaluation of `AutoPQ-advanced`, which jointly optimizes both configurations, i.e., λ_{p} and λ_{q}, and applies CASH with the successive halving pruning strategy for the specified total time budget of $B_{\text{t}} = 8$ h, and a maximal iteration budget of $B_{\text{i}} = 100$.[20]

[18] For the forecasting methods MLP, DeepAR, N-HiTS, and TFT, 20 % of the training dataset are hold-out for early stopping, i.e., the training process is terminated when the loss on the hold-out data increases.

[19] The results of `AutoPQ-default` for the datasets Load-GCPs, Mobility, Price, and PV are taken from [81], while the results for the additional datasets Load-BW and WP are not yet published.

[20] Note that the early stopping condition terminates the inner loop far before the iteration budget is exhausted (on average after nine iterations, as shown later in Fig. 7).

Two crucial properties in evaluating probabilistic forecasts are sharpness and calibration. According to Gneiting et al. [91], probabilistic forecasts should aim to maximize sharpness without negatively affecting the calibration. To balance these properties, the Continuous Ranked Probability Score (CRPS)

$$\mathrm{CRPS} = \frac{1}{K}\sum_{k=1}^{K}\int_{\mathbb{R}}\left(\hat{F}_{\mathrm{Y}}[k](x) - \mathbb{1}\{y[k] \le x\}\right)^2 \mathrm{d}x, \tag{12}$$

is used as metric for both tuning and assessment, averaged over all time points K. In Eq. (12), $\hat{F}[k]$ is the estimated CDF of the forecast at time point k and the realized value $y[k]$ is translated into a degenerate distribution with the indicator function $\mathbb{1}\{y[k] \le x\}$, which is one if $y[k]$ is less than the integration variable x, and zero otherwise.

The evaluation is performed five times on each dataset, with the arithmetic mean and the standard deviation reported to account for stochastic effects in training and optimization. We ensure consistent results using the same hardware for all runs.[21] Additionally, we report computing time and electricity consumption to quantify the computational effort and enable comparisons related to sustainability, as detailed in [27].

Benchmarks. We compare the probabilistic forecasting performance of `AutoPQ` against multiple benchmarks, categorized into two groups: direct probabilistic forecasts and probabilistic forecasts derived from existing point forecasts.[22]

The first category includes the direct probabilistic forecasts DeepAR, QRNNs, and the NNQF, which are detailed in Section 2. We implement DeepAR [28] using the Python package `PyTorch Forecasting` [92], the QRNNs with the Python package `Keras` [93] using the built-in PL function, and the NNQF [39] using the MLP of the `scikit-learn` [94] Python package.

The second category includes the point forecast-based probabilistic forecasts Gaussian PIs, Empirical PIs, and Conformal PIs. While all of these methods 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 $y[k]$ from a validation dataset, they differ in their approach to calculate PIs. Gaussian PIs [16] assume Gaussian-distributed residuals and estimate the standard deviation σ to calculate PIs centered around the point forecast, adjusting σ by a factor corresponding to the desired confidence level. In contrast, Empirical PIs [18] do not assume a specific parametric PDF but use the empirical PDF of residuals (i.e., sorting $r[k]$ from smallest to largest) to calculate PIs based on the desired confidence level. Conformal PIs for multi-step-ahead forecasts [19] calibrate the PIs for each forecast horizon without assuming a specific parametric PDF. A critical non-conformity score is calculated for each $r[k]$, and the temporal dependencies between these scores across the forecast horizon are obtained using Bonferroni correction. These corrected scores are used to calculate PIs centered around the point forecast.

For each point forecast-based probabilistic benchmark, we compare against the best-performing base forecast. That is, we select the point forecasting method achieving the lowest CRPS after calculating PIs using the Gaussian, the Empirical, and the Conformal approach, respectively.

4.1.2. Results

In the following, we summarize and visualize our benchmarking results, followed by an interpretation and discussion in Section 5.

[21] Each run is performed with four parallel trials using two Intel Xeon Platinum 8368 Central Processing Units (CPUs) with 76 cores, four NVIDIA A100-40 GPU with 40 GB memory (GPU are utilized only for DL-based forecasting methods), and 256 GB Random-Access Memory (RAM), all provided by the HPC system HoreKa.

[22] The benchmark results for the datasets Load-GCPs, Mobility, Price, and PV are taken from [81], while the results for the additional datasets Load-BW and WP are not yet published.

Table 2
The CRPS evaluated on the hold-out test sets with methods categorized into three groups: (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 datasets Load-GCP, Mobility, Price, and PV originate from [21]. The lowest values per data set are highlighted in bold.

	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 ± 0.000$^{(1)}$	0.143 ± 0.000$^{(1)}$	0.143 ± 0.000$^{(1)}$	0.147 ± 0.001$^{(1)}$	**0.138** ± 0.001$^{(1)}$
Load-GCP	0.312 ± 0.009	0.287 ± 0.002	0.263 ± 0.001	0.299 ± 0.000$^{(1)}$	0.234 ± 0.000$^{(1)}$	0.234 ± 0.000$^{(1)}$	0.234 ± 0.002$^{(1)}$	**0.218** ± 0.001$^{(1)}$
Mobility	0.299 ± 0.007	0.443 ± 0.006	0.542 ± 0.004	0.360 ± 0.021$^{(2)}$	0.268 ± 0.015$^{(2)}$	0.268 ± 0.015$^{(2)}$	0.263 ± 0.004$^{(2)}$	**0.258** ± 0.005$^{(2)}$
Price	0.158 ± 0.005	0.157 ± 0.002	0.183 ± 0.003	0.279 ± 0.008$^{(2)}$	0.161 ± 0.005$^{(2)}$	0.161 ± 0.005$^{(2)}$	0.140 ± 0.006$^{(2)}$	**0.131** ± 0.001$^{(3)}$
PV	0.151 ± 0.013	**0.101** ± 0.001	0.119 ± 0.000	0.208 ± 0.000$^{(1)}$	0.125 ± 0.000$^{(1)}$	0.125 ± 0.000$^{(1)}$	0.106 ± 0.001$^{(1)}$	**0.101** ± 0.001$^{(1)}$
WP	0.618 ± 0.028	0.374 ± 0.002	0.394 ± 0.004	0.419 ± 0.000$^{(1)}$	0.373 ± 0.000$^{(1)}$	0.373 ± 0.000$^{(1)}$	0.377 ± 0.001$^{(1)}$	**0.362** ± 0.001$^{(1)}$
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 3
The percentage improvement of `AutoPQ-advanced` over the comparison methods in terms of the CRPS evaluated on the hold-out test data. `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.

	DeepAR	QRNNs	NNQF	Gaussian PIs	Empirical PIs	Conformal PIs	AutoPQ default
Load-BW	28.1 %*	4.8 %*	33.7 %*	11.5 %$^{*}_{(1)}$	3.5 %$^{*}_{(1)}$	3.5 %$^{*}_{(1)}$	6.1 %$^{*}_{(1)}$
Load-GCP	30.1 %*	24.0 %*	17.1 %*	27.1 %$^{*}_{(1)}$	6.8 %$^{*}_{(1)}$	6.8 %$^{*}_{(1)}$	6.8 %$^{*}_{(1)}$
Mobility	13.7 %*	41.8 %*	52.4 %*	28.3 %$^{*}_{(2)}$	3.7 %$_{(2)}$	3.7 %$_{(2)}$	1.9 %$_{(2)}$
Price	17.1 %*	16.6 %*	28.4 %*	53.0 %$^{*}_{(2)}$	18.6 %$^{*}_{(2)}$	18.6 %$^{*}_{(2)}$	6.4 %$^{*}_{(2)}$
PV	33.1 %*	0.0 %	15.1 %*	51.4 %$^{*}_{(1)}$	19.2 %$^{*}_{(1)}$	19.2 %$^{*}_{(1)}$	4.7 %$^{*}_{(1)}$
WP	41.4 %*	3.2 %*	8.1 %*	13.6 %$^{*}_{(1)}$	2.9 %$^{*}_{(1)}$	2.9 %$^{*}_{(1)}$	4.0 %$^{*}_{(1)}$
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 key observations can be made from the benchmarking results in Table 2: first, and most importantly, `AutoPQ-advanced` shows a significant performance advantage over `AutoPQ-default` and other benchmarks. Second, the performance variability across datasets is greater for the direct probabilistic forecasting methods DeepAR, QRNNs, and NNQF compared to the point forecast-based probabilistic forecasting methods Gaussian, Empirical, and Conformal PIs as well as both `AutoPQ-default` and `AutoPQ-advanced`. Third, the average performance across datasets is lower for DeepAR, QRNNs, NNQF, and Gaussian PIs compared to Empirical PIs, Conformal PIs, `AutoPQ-default`, and `AutoPQ-advanced`.

Table 3 shows the percentage improvement of `AutoPQ-advanced` relative to the six benchmarks and `AutoPQ-default` per dataset. Improvements are highlighted with an asterisk if they are significant, i.e., the p-value of the one-tailed t-test is less than 0.05. The following insights can be drawn from the table: First, we observe significant improvements in 38 out of 42 tests (across seven methods and six datasets). Second, `AutoPQ-advanced` shows significant improvement over each direct probabilistic benchmark, averaging at least 15.1 %, and each point forecast-based probabilistic benchmark, averaging at least 9.1 %. Third, `AutoPQ-advanced` outperforms `AutoPQ-default` by 5.0 % on average.

4.1.3. Insights

Fig. 5 provides a visual comparison of PIs generated by `AutoPQ-advanced` using three exemplarily selected point forecasting methods (ETS, XGB, and TFT) compared to the probabilistic benchmarks for the Mobility dataset. The `AutoPQ-advanced` PIs are generated for the best-performing configuration identified through CASH with successive halving, with TFT being the best method for this dataset (as detailed later in Fig. 8). Fig. 5(a)–(c) shows that the PIs generated from the three point forecasting methods vary in sharpness, as reflected in their widths. Notably, clear differences are evident between the considered methods TFT, XGB, and ETS, ranking first, fourth, and ninth in the CASH, respectively.

Another observation concerns the PIs when the realized values are near zero or zero. Since the forecasted value (mobility indicator) is restricted to non-negative values, negative PIs are unreasonable. `AutoPQ`'s PIs account for this restriction, unlike the probabilistic benchmarks' PIs, cf. Fig. 5(a)–(c) and (d)–(i). However, the post-processing required for `AutoPQ`'s quantiles to comply with this non-zero constraint on the CRPS is minimal, as detailed in Table 9 in the Appendix A.

4.2. Ablation study

This section outlines the experimental setup of the ablation study, which evaluates the effectiveness of the automation methods applied in `AutoPQ-advanced`, followed by the presentation of results and insights.

4.2.1. Experimental setup

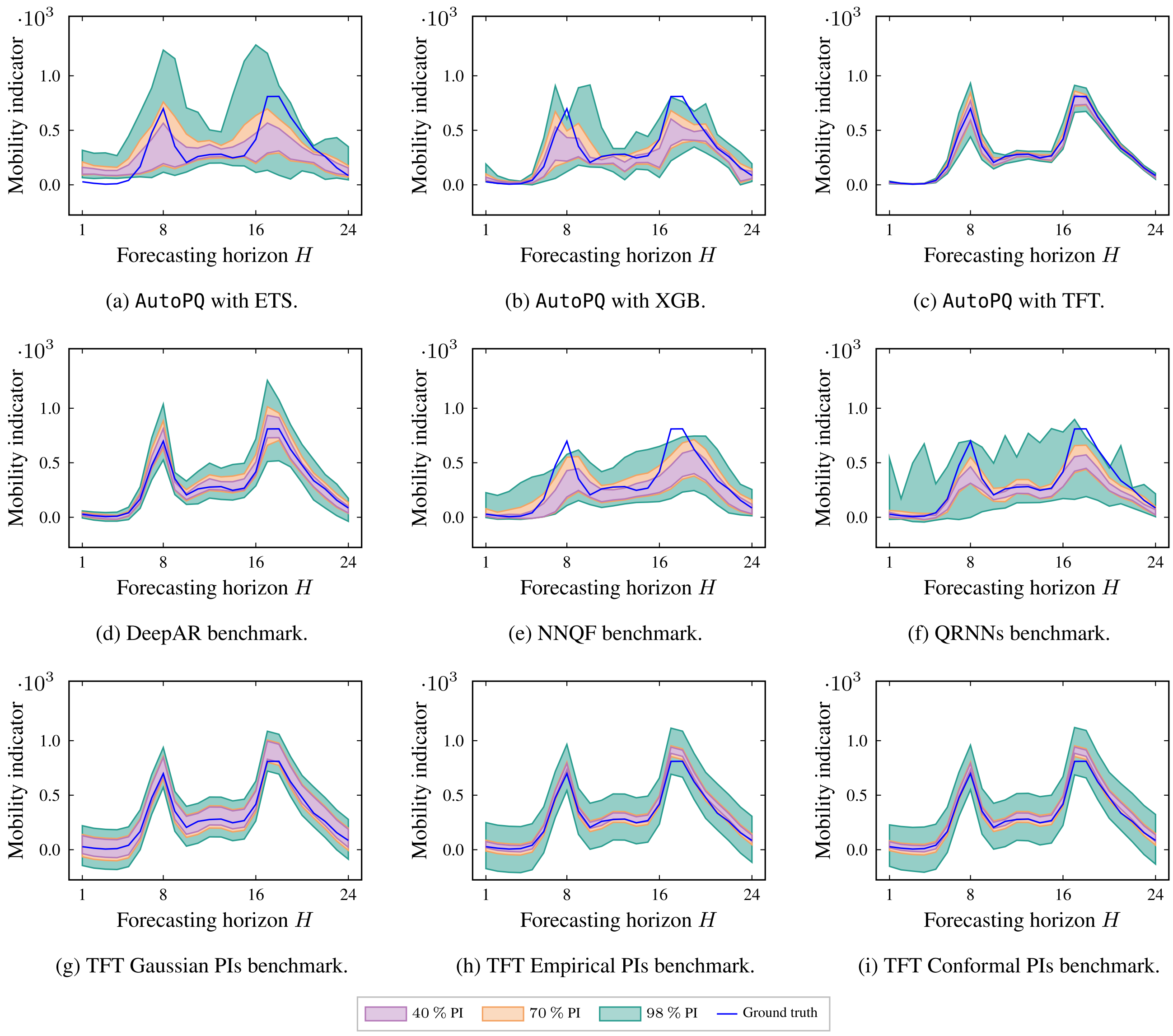
In the following, we outline the data used, the evaluation strategy employed, and the ablations considered in our experimental setup.

Data. As for benchmarking, the ablation study of `AutoPQ-advanced` is performed on six different datasets: Load-BW, Load-GCPs, Mobility, Price, PV, and WP (see Section 4.1 for an overview).

Evaluation strategy. The ablation study is based on a similar evaluation strategy as the previous benchmarking. Specifically, the six datasets outlined in Section 4.1 are normalized and divided into training, validation, and test sets, as detailed in Table 13 in the Appendix A.

To reduce the computational effort, we only consider the MLP and N-HiTS point forecasting methods without using successive halving. This selection is reasoned by (i) the computational effort, (ii) the probabilistic performance, and (iii) the sensitivity to HPO. As illustrated in Fig. 2, MLP and N-HiTS require low and moderate to high computational effort, respectively. In addition, both methods achieve competitive probabilistic performance and respond well to HPO, as shown later in Table 4.

In line with the benchmarking, the CRPS metric (12) is used for both tuning and assessment, with evaluations conducted five times on each



(a) AutoPQ with ETS. (b) AutoPQ with XGB. (c) AutoPQ with TFT.

(d) DeepAR benchmark. (e) NNQF benchmark. (f) QRNNs benchmark.

(g) TFT Gaussian PIs benchmark. (h) TFT Empirical PIs benchmark. (i) TFT Conformal PIs benchmark.

Fig. 5. Exemplary 40 %, 70 %, and 98 % PIs for the Mobility dataset. The probabilistic forecasts of `AutoPQ-advanced` (a–c) are generated based on the point forecasting methods ETS (SM), XGB (ML), and TFT (DL), respectively, showing different performance. Note that `AutoPQ-advanced` automatically optimizes the hyperparameters and selects the best-performing method for each dataset, i.e., TFT for Mobility. The probabilistic benchmarks (d–i) can be categorized into direct probabilistic methods (DeepAR, NNQF, QRNNs) and point forecast-based probabilistic methods (Gaussian PIs, Empirical PIs, Conformal PIs based on a TFT forecaster).

Table 4
The percentage improvement of each optimized configuration over the corresponding default configuration in the successive halving ablation study without pruning in terms of the CRPS evaluated on the validation data.

	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.

dataset. To ensure consistent runtime measurements, we utilize the same hardware as used for the benchmarking in all runs (see Section 4.1).

Ablation 1. The effectiveness of the inner loop in Algorithm 1, responsible for finding the optimal sampling hyperparameter $\lambda^{\hat{\star}}_{q}$, is assessed. For comparison, a conventional HPO is used that does not take into account the different computational efforts required for training the point forecasting model and generating probabilistic forecasts. This is achieved by omitting the inner loop of Algorithm 1, resulting in

Algorithm 3 HPO algorithm for optimizing the point forecasting method's hyperparameters and the cINN's sampling hyperparameter simultaneously without considering the significantly lower computational effort of the latter.

```
Input: $\Lambda_p, \Lambda_q, B_i, B_t, X_{train}, X_{val}, y_{train}, y_{val}, f_{cINN}(\cdot)$

    # Initialize Propulate [83] (EA)
 1: trial_generator($\Lambda_p, \Lambda_q$)
 2: $b_t \leftarrow 0$ s
 3: while $b_t < B_t$ do
 4:     $t \leftarrow$ get_current_time()
 5:     $\lambda_p, \lambda_q \leftarrow$ trial_generator.get_config($\Lambda_p, \Lambda_q$)
        # High computational effort
 6:     $f_p(\cdot) \leftarrow$ train_p_model($\lambda_p, X_{train}, y_{train}$)
        # Low computational effort
 7:     $f_q(\cdot) \leftarrow$ generate_q_model($f_p(\cdot), f_{cINN}(\cdot), \lambda_q$)
        # Low computational effort
 8:     $Q \leftarrow$ assess_performance($f_q(X_{val}, y_{val})$)
 9:     trial_generator.update($\lambda_p, \lambda_q, Q$)
10:     $b_t \leftarrow b_t +$ (get_current_time() $- t$)
11: end while
12: $\lambda_p^\star, \lambda_q^\star \leftarrow$ trial_generator.get_best_config($\Lambda_p, \Lambda_q$)

Output: $\lambda_p^\star, \lambda_q^\star$
```

Λ_p, Λ_q: configuration space (point forecast, quantile forecast); B_i, B_t: budget (iteration, time); X_{train}, X_{val}: model inputs (training, validation); y_{train}, y_{val}: model outputs (training, validation); f_{cINN}, f_p, f_q: trained model (cINN, point forecast, quantile forecast); $\lambda_p^\star, \lambda_q^\star$: optimal configuration (point forecast, quantile forecast)

Algorithm 3. In this setup, the trial generator suggests configurations for both the point forecaster λ_p and the sampling hyperparameter λ_q as combined configuration space $\lambda = \lambda_p \times \lambda_q$. Thus, only one sampling hyperparameter is assessed for each trained point forecasting model. To ensure comparability, both Algorithms 1 and 3 are run without successive halving, i.e., using the full time budget $B_t = 8$ h and considering the same configuration spaces.

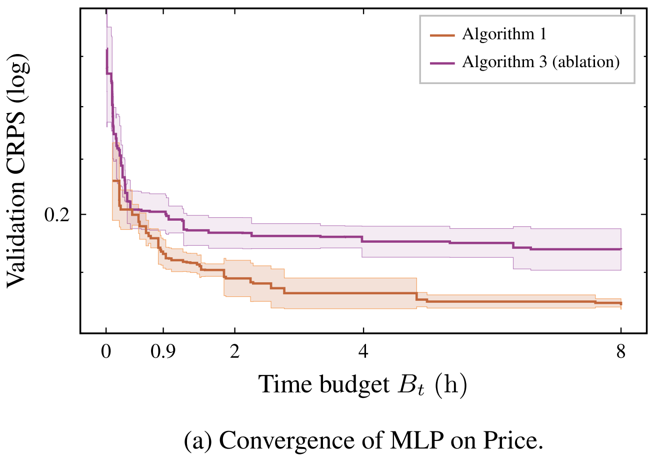
Ablation 2. Different trial generators for the inner loop of Algorithm 1 are compared. Specifically, we compare PK-based BO against random search and BO, neither of which uses PK. The comparison is based on the number of iterations required to fulfill the early stopping condition outlined in Section 3.5.1. To avoid the computationally expensive task of repeating the entire CASH process for all six datasets five times across all nine forecasting methods, we limit the comparison to repeating only the inner loop of Algorithm 1. Thus, the inner loop is executed five times for each dataset using the best configuration for each forecasting method and evaluated with the two trial generators being compared.

Ablation 3. The decision quality of the successive halving is evaluated, specifically the impact of pruning underperforming forecasting methods in the CASH. To achieve this, we compare the evolution of the forecasting methods' validation performances with successive halving to their performance evolution without pruning. Each pruned forecasting method from the benchmarking (see Section 4.1) is continued until the full time budget $B_t = 8$ h is exhausted. This allows us to analyze whether successive halving retains forecasting methods that ultimately perform well while pruning those that do not improve with additional computational effort. Additionally, we evaluate how the nine forecasting methods respond to HPO, i.e., the improvement over the point forecasting methods' default hyperparameter configurations.

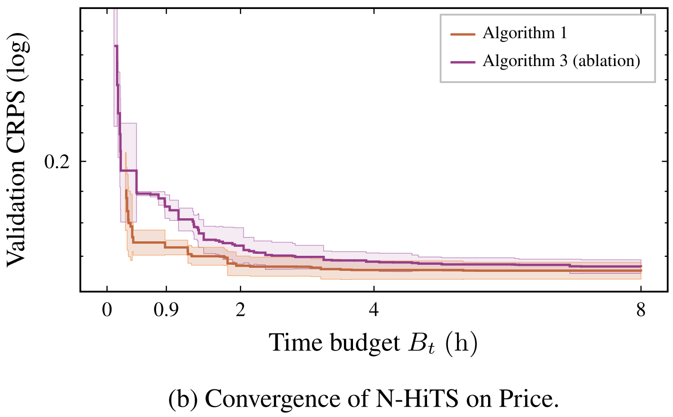
4.2.2. Results

The following section summarizes the results of the ablation study and provides supporting visualizations, which are further interpreted and discussed in Section 5.

Ablation 1. The results comparing the effectiveness of the originally proposed AutoPQ HPO Algorithm 1 to its ablated version Algorithm 3 are presented in Fig. 6 for the MLP and N-HiTS. These figures exemplarily



(a) Convergence of MLP on Price.



(b) Convergence of N-HiTS on Price.

Fig. 6. Comparison of the convergence of Algorithms 1 and 3 (ablation) for the HPO of the MLP and N-HiTS on the two exemplary datasets. The thick solid line represents the mean value, and the opaque area is the standard deviation over five runs.

illustrate the CRPS evolution on the validation dataset over the HPO budget. The time budget marks for the pruning rounds are also shown to contextualize the validation performance within successive halving. We can draw two key observations from these figures: First, the mean value of Algorithm 1 is consistently lower than that of Algorithm 3 for most of the HPO time budget. Second, the mean value of Algorithm 1 is notably lower than that of Algorithm 3 at the beginning of the HPO. This gap narrows toward the end of the HPO time budget, with both values appearing to converge to the same value.[23]

Ablation 2. Fig. 7 compares the results of different trial generators for the inner loop in Algorithm 1. The figure shows the number of iterations required until the early stopping condition is met, aggregated over all nine forecasting methods, the six datasets, and five runs.[24] Three key observations emerge: First, BO using the TPE surrogate model (Hyperopt [82]) significantly reduces the number of iterations compared to random search. Second, integrating PK into BO-TPE further decreases the number of required iterations. Notably, even the upper outliers fall within the range of the lower whisker of BO-TPE without PK. Third, all three trial generators achieve similar validation CRPS, as detailed

[23] Both observations also hold for the other four datasets and both of the considered forecasting methods.
[24] The computational effort scales approximately linearly with the number of iterations since the initialization effort is negligible.

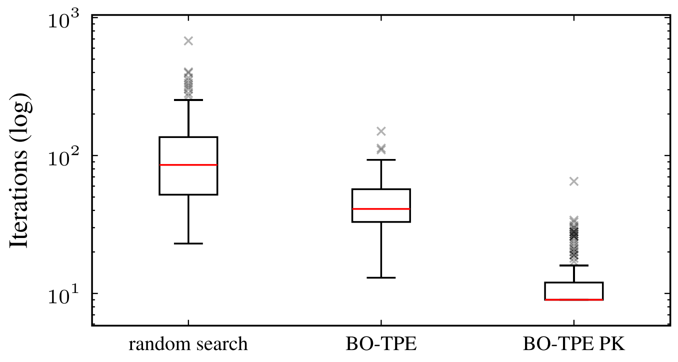


Fig. 7. Number of iterations required for the inner loop of Algorithm 1 until the early stopping criterion is fulfilled, compared for three trial generators: random search, BO-TPE, and BO-TPE-PK. The box plot considers all nine forecasting methods, the six datasets, and five runs, i.e., 270 data tuples per trial generator.

in Table 14 in the Appendix A. Specifically, BO-TPE-PK delivers similar probabilistic performance while requiring significantly fewer iterations compared to the other two generators.

Ablation 3. Fig. 8 shows the results of the successive halving ablation study.[25] Three key findings are notable: first, the best-performing forecasting method remains consistent across all datasets, meaning it is not erroneously deactivated during pruning. Second, this best-performing method is already identified by `AutoPQ-default`. Rank changes occur among the following methods during successive halving. Third, while SM methods show significant improvements in the first pruning round (`AutoPQ-advanced`) compared to their default configuration (`AutoPQ-default`), they still rank last even with additional time budget. In contrast, the middle and upper ranks are competitive among methods from the ML and DL families.

4.2.3. *Insights*

Fig. 9 illustrates the impact of leveraging PK for BO-TPE (`Hyperopt` [82]) on each dataset. The plots depict the CRPS on the validation dataset together with the prior distribution over the sampling hyperparameter σ for both BO-TPE with and without PK. For BO-TPE, a continuous uniform PDF $\mathcal{U}(0,3)$ is assumed, while BO-TPE-PK uses a log-normal PDF $\mathcal{N}(\mu,\sigma)$ with the parameters derived from the EA population (`Propulate` [83]). The PDF in Fig. 9 are normalized to their maximum value to ensure visual comparability. These plots, while only exemplarily showing the sampling hyperparameter optimization for the XGB point forecast, are representative of other point forecasting methods concerning the following observations: First, the optimal sampling hyperparameter $\lambda_{\mathrm{q}}^{\hat{\star}} = \sigma^{\hat{\star}}$ is highly dependent on the dataset, showing differences in both location and sensitivity. For instance, the validation CRPS valley in Fig. 9(b) is flatter than the one in Fig. 9(a). Correspondingly, the prior distribution of BO-TPE-PK shows varying spread. Second, Fig. 9 confirms our earlier findings that BO-TPE-PK requires significantly fewer iterations to meet the early stopping condition. This condition is often satisfied before the TPE surrogate model starts trading off exploitation and exploration.

Examining the improvement in the validation CRPS without pruning as shown in Fig. 8, we can calculate the percentage improvement. Results are listed in Table 4, where significant improvements with a p-value of the one-tailed t-test smaller than 0.05 are highlighted with an asterisk. Three observations stand out: First, the SM methods show inconsistent improvements; some exhibit large and significant enhancements, while others demonstrate little to no improvement. Second, the ML methods achieve significant improvements, with the MLP showing

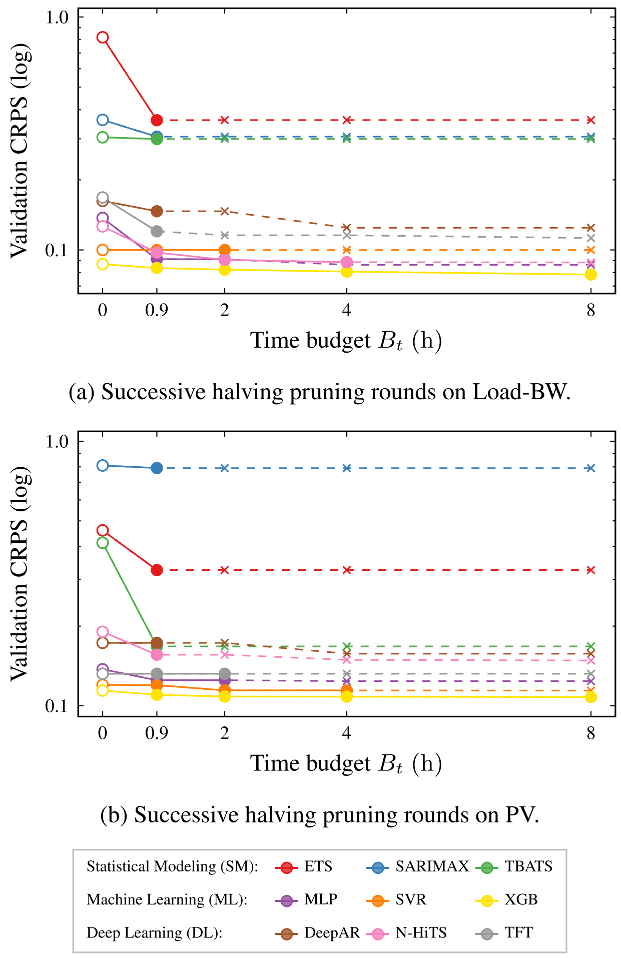


Fig. 8. Evolution of the considered forecasting methods' validation performances in the successive halving-based CASH of `AutoPQ-advanced`. The performance at 0 h reflects `AutoPQ-default`, shown as a reference and thus not included in the time budget B_t. Solid lines with dot markers show the performance of active methods, while dotted lines with cross markers show the performance of inactive methods as it would have evolved without pruning. For simplicity, the progress in the successive halving pruning rounds is assumed to be linear.

the best response to HPO, improving on average by 12.3 %. Third, the DL methods achieve significant improvements only for DeepAR and N-HiTS, with average enhancements by 14.7 % and 15.8 %, respectively. In contrast, the TFT exhibits a weak response to HPO, showing no significant improvement in five of the six datasets.

To visualize the impact of the point forecast's quality on `AutoPQ`'s probabilistic forecasting quality, we compare the validation metrics of all evaluated configurations in the HPO exemplarily for MLP and N-HiTS.[26] Specifically, we plot the Mean Squared Error (MSE) (point forecast) over the CRPS (probabilistic forecast) in Fig. 10.

In this comparison, we make two findings: First, a low MSE generally results in a low CRPS. Second, the lowest MSE, however, does not necessarily yield the lowest CRPS. A considerable dependency on the data set is recognizable. While the relationship between CRPS and MSE in the Load-BW data set is approximately linear, the relationship in the PV data set is weaker. Furthermore, a dependency on the point forecasting

[25] Fig. 8 shows the representative results of run no. 1; the results of all runs are given in Table 15 in the Appendix A.

[26] The selection is based on their low and moderate computational effort (i.e. many data points for the analysis), and their good performance in the benchmarking and high sensitivity to HPO.

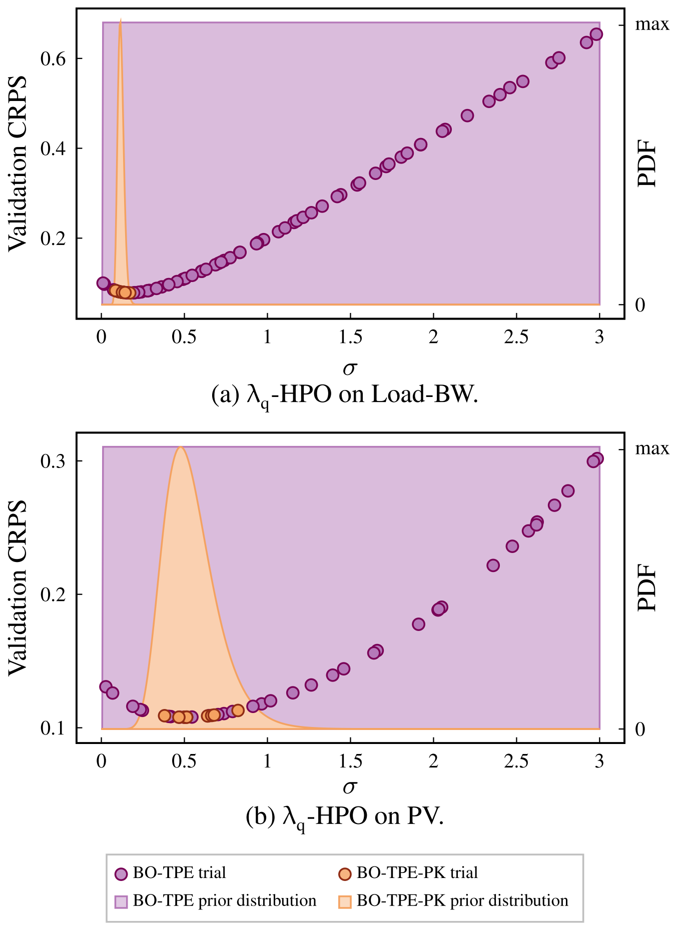


Fig. 9. The HPO of the sampling hyperparameter conducted in the inner loop of Algorithm 1 using BO-TPE and BO-TPE-PK. The validation performance (CRPS) over the trial configurations (λ_q) is represented as dots, and the TPE prior distribution is shown as a filled area normalized by the maximum of the respective PDF. The plots illustrate the HPO for a single XGB configuration but are representative of the other point forecasting methods.

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.

4.3. Electricity consumption-awareness

Our benchmarking results demonstrate that `AutoPQ-advanced` can significantly enhance performance compared to `AutoPQ-default`. Since this improvement incurs a higher computational effort, the following evaluation aims to increase the awareness for energy-intensive performance improvements.

4.3.1. Experimental setup

Our evaluation is based on the average computational effort across the six datasets used in the benchmarking (Section 4.1). We measure the computational effort in terms of the electricity consumption because it is independent of data-center-specific or geopolitical factors at the time of collection [27].[27] Since the research-exclusive HPC system HoreKa[28] was used for benchmarking, we additionally estimate the monetary costs that would be incurred by the Amazon AWS cloud computing service with equivalent hardware.

[27] Other metrics based on electricity consumption like the carbon footprint can be calculated using assumed or local conditions.

[28] https://www.nhr.kit.edu/userdocs/horeka/.

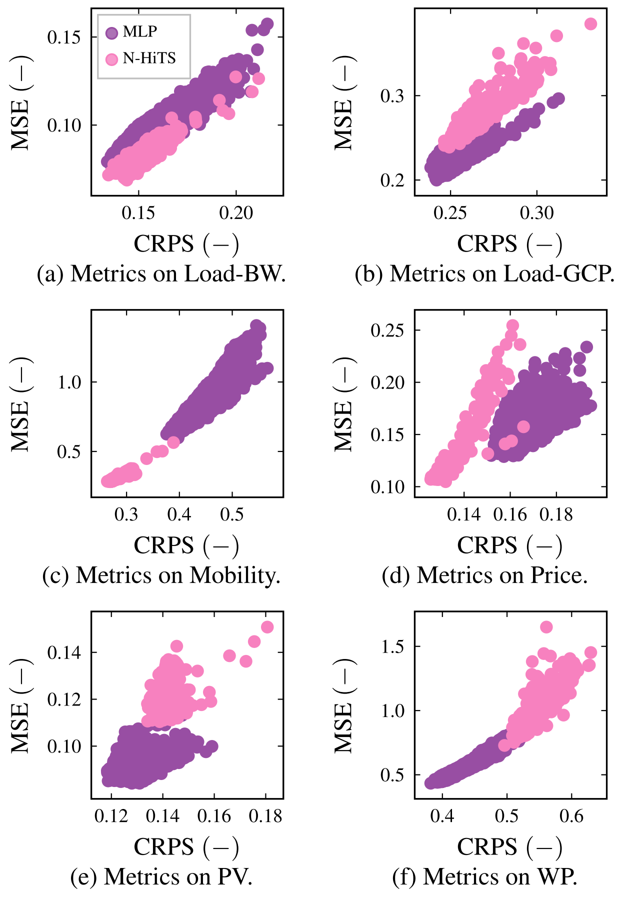


Fig. 10. Comparison of the MSE (point forecast) with the CRPS (probabilistic forecast) on the six data sets in the HPO of AutoPQ-advanced. Each dot represents the performance of a trial configuration in the HPO and legend of (a) applies to all sub-plots.

4.3.2. Results

The following section summarizes the electricity consumption required for performance improvements and corresponding monetary costs, which are further interpreted and discussed in Section 5.

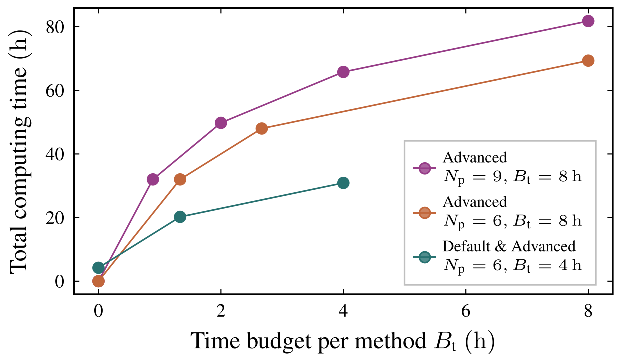
Electricity consumption.

Fig. 11 shows the total computing time and the corresponding electricity consumption of `AutoPQ-default` and `AutoPQ-advanced` required for a single run, averaged across datasets and runs.

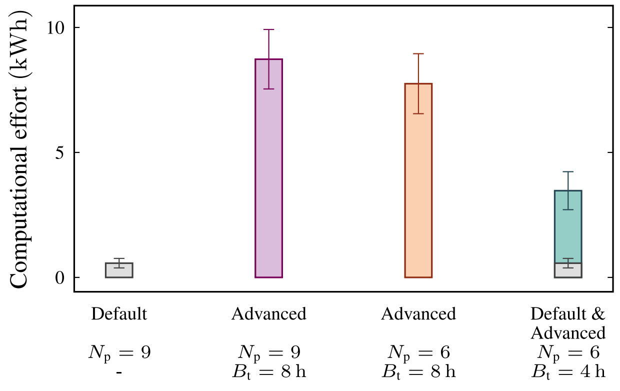
While a single `AutoPQ-default` run uses 4.22 h of compute time and consumes 0.57 kWh of electrical energy, `AutoPQ-advanced` requires 81.78 h and consumes 8.73 kWh for a run, with $N_p = 9$ considered point forecasting methods and a time budget $B_t = 8$ h. This increased resource usage is due to the parallelized CASH, which evaluates four trial configurations simultaneously for each point forecasting method. As a result, the total computing time accumulates over all pruning rounds of successive halving. Specifically, the total computing time is calculated as the product of the number of active forecasting methods per round, the time budget per round, and the number of parallel trials conducted.

The benchmarking shows that the three SM point forecasting methods perform significantly worse than the ML- and DL-based methods. Omitting SM methods due to their weak performance, i.e., $N_p = 6$, would reduce the computational effort to 69.33 h and 7.75 kWh.

The ablation study reveals that the performance of `AutoPQ-default`, which uses default hyperparameters for the point forecasting methods $\lambda_{p,\text{default}}$ with an optimized sampling hyperparameter $\lambda_q^{\hat{\star}}$, reflects the potential performance improvements achievable through HPO to identify $\lambda_p^{\hat{\star}}$ with `AutoPQ-advanced`. Thus, `AutoPQ-default` can be viewed



(a) Evolution of the total computing time.



(b) Resulting total electricity consumption.

Fig. 11. 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 system.

as the initial pruning round in the successive halving-based CASH of `AutoPQ-advanced`. Since the performance gains in the final pruning round are minimal, the overall time budget could be further reduced to $B_t = 4$ h. Both adjustments would decrease the computational effort to 30.89 h and 3.47 kWh.

Monetary costs. We calculate the monetary costs of using Amazon AWS for both `AutoPQ-default` and `AutoPQ-advanced`, based on equivalent hardware on the Amazon EC2 G4 instances [95]. For `AutoPQ-advanced`, the four-GPU instance `g4ad.16xlarge` (3.468 \$/h on demand) closely matches the hardware utilized on the HoreKa HPC system. Since four trials are evaluated in parallel, the billed computing time shown in Fig. 11 is reduced by a factor of four. As the HPO of the sampling hyperparameter is performed sequentially for `AutoPQ-default`, the single-GPU instance `g4ad.4xlarge` (0.867 \$/h on demand) is comparable to the hardware utilized on HoreKa.

The costs for employing `AutoPQ-default` and `AutoPQ-advanced` on the above Amazon EC2 G4 instances are summarized in Table 5. Using `AutoPQ-default` would cost 3.66 \$, while using `AutoPQ-advanced` would cost 26.78 \$ with the electricity consumption-aware configuration ($N_p = 6$, $B_t = 4$ h). For simplicity, we assume that all forecasting methods utilize GPU instances, even though only the three DL methods benefit from them. To optimize costs and avoid GPU idle time, CPU-only instances should be employed for SM and ML point forecasting methods, as used for the benchmarking on HoreKa.

5. Discussion

The subsequent section discusses the results of the benchmarking, the ablation studies, as well as limitations and benefits.

5.1. Benchmarking

We discuss two aspects of the benchmarking results for probabilistic day-ahead forecasting performance across the six datasets: First, we observe a greater performance variability among the direct forecasting methods, i.e., DeepAR, QRNNs, and NNQF, compared to the point forecast-based probabilistic forecasting methods, i.e., Gaussian, empirical, and conformal PIs, as well as `AutoPQ-default` and `AutoPQ-advanced`. The reduced variability in the latter group can be attributed to the probabilistic forecast being based on multiple point forecasting methods, from which the best-performing method is selected. This performance-based selection ensures that only the top method prevails, thus mitigating the risk of poor performance associated with any method underperforming on the dataset.

Second, `AutoPQ-advanced` consistently outperforms `AutoPQ-default` and all benchmarking methods. Specifically, the average improvement of `AutoPQ-advanced` surpasses that of `AutoPQ-default` when compared to both direct probabilistic benchmarks and point forecast-based probabilistic benchmarks. This improvement is significant for five of the six datasets, suggesting that `AutoPQ-advanced` is particularly well-suited for performance-critical smart grid applications subject to high decision costs.

5.2. Ablation study

Our results from the ablation study reveal four key points: First, the performance advantage of Algorithm 1 over Algorithm 3 (ablation), evident at the beginning of the HPO, is decisive for the effectiveness of successive halving. Since Algorithm 3 (ablation) risks prematurely pruning forecasting methods that could potentially perform well, its performance deficit is particularly high during the initial and subsequent pruning rounds. This is because the EA requires several generations to identify a suitable sampling hyperparameter, with random initialization

Table 5
The monetary costs of using the Amazon EC2 G4 instances with hardware similar to the HoreKa HPC system.

`AutoPQ` configuration	Default	Advanced	Advanced	Default & advanced
Point forecasting methods	$N_p = 9$	$N_p = 9$	$N_p = 6$	$N_p = 6$
Successive halving budget	–	$B_t = 8$ h	$B_t = 4$ h	$B_t = 4$ h
Amazon EC2 G4 instance	`g4ad.4xlarge`	`g4ad.16xlarge`	`g4ad.16xlarge`	`g4ad.16xlarge`
On-demand price	0.87 \$/h	3.47 \$/h	3.47 \$/h	3.47 \$/h
Computing hours	4.22 h	20.45 h	17.33 h	7.72 h
Monetary costs	3.66 \$	70.9 \$	60.11 \$	26.78 \$

potentially yielding both suitable and unsuitable parameters. In contrast, Algorithm 1 effectively identifies the optimal sampling hyperparameter for each trained model, thereby reducing this variability (i.e., denoising via structured search). We observe this benefit for both N-HiTS (moderate to high computational effort) and the MLP (low computational effort), demonstrating the general effectiveness of Algorithm 1 across different forecasting methods.

Second, leveraging PK for the inner loop trial generator significantly reduces the number of iterations. Maintaining a low iteration count is crucial for achieving the performance advantage of Algorithm 1 over Algorithm 3 (ablation). This improvement stems from the fact that the inner loop's effectiveness relies on keeping the computational effort needed to identify the optimal sampling hyperparameter low compared to the time required to train the model.

Third, successive halving proves effective in both pruning underperforming forecasting methods and retaining promising ones. The first conclusion is underpinned by the ablation study that shows no premature pruning of promising configurations for the given data sets as no such cases occur across five runs (see Table 15 in the Appendix A). Additionally, Algorithm 1 denoises the objective function to reduce the risk of premature pruning as discussed above (see Fig. 6). The second conclusion is supported by our finding that the three SM methods consistently perform poorly across all datasets, lagging far behind ML and DL methods. Therefore, we recommend excluding these methods to reduce the computational effort. For ML and DL methods, the performance of their default hyperparameter configuration already provides a strong indicator of the method's potential for performance improvement in the HPO. This suggests that if one method significantly outperforms another using default configurations, this performance gap is unlikely to change substantially through HPO.

Fourth, the evaluation does not show a clear correlation between the point forecasting metric MSE and the probabilistic metrics CRPS. Therefore, we conclude that optimizing the desired probabilistic properties of `AutoPQ` requires selecting the most suitable point forecasting method while jointly optimizing its hyperparameters λ_p, and the sampling hyperparameter λ_q. In other words, a separate optimization that first optimizes the point forecasting method's MSE and then the CRPS, can leave untapped performance potential.

5.3. Electricity consumption-awareness

In the context of electricity consumption-awareness in view of sustainability, `AutoPQ` offers two configurations tailored to accommodate varying computing systems and performance needs: `AutoPQ-default` for general-purpose computing systems, delivering high-quality probabilistic forecasts, and `AutoPQ-advanced`, which necessitates HPC systems to enhance forecasting quality further. The latter uses a built-in successive halving strategy for optimizing resource use by pruning underperforming configurations and re-allocating computational resources to more promising ones. We recommend running `AutoPQ-default` first and using the model if the forecasting quality is satisfactory. Otherwise, the results can be used to initialize the successive halving-based CASH of `AutoPQ-advanced`, which can improve the CRPS by up to 6.8 %. In terms of the efforts required for performance improvements, we quantify the energy footprint and monetary costs of both `AutoPQ-default` and `AutoPQ-advanced`. Given that the monetary costs are relatively low compared to the personnel expenses associated with an iterative manual design process, we conclude that both `AutoPQ-default` and `AutoPQ-advanced` are cost-effective for most smart grid applications. For applications in which decision-making heavily relies on forecasting quality, we anticipate that `AutoPQ-advanced` will quickly amortize its higher initial costs, ultimately becoming more cost-efficient than `AutoPQ-default` in the long run.

5.4. Limitations

With regard to the limitations of `AutoPQ`, two key aspects are discussed: possibly untapped potential for further performance improvements and desirable probabilistic properties.

Possibly untapped performance improvement potential. Performance could be further improved through the configuration space design, the feature selection, and integrating PK. An improperly sized configuration space may either waste computational resources on insensitive hyperparameters or overlook sensitive hyperparameters, missing potential performance gains. While a comprehensive sensitivity analysis across various datasets is impractical due to high computational costs, `AutoPQ`'s configuration space design is based on best practices. Specifically, the hyperparameters and their value ranges are chosen according to recommendations provided in the respective forecasting methods' documentation. The feature selection for the evaluation is based on [21] (`AutoPQ-default`) to ensure comparability in benchmarking. However, automated selection is preferable for real-world applications. This limitation could be addressed by using automated feature selection methods, e.g., the Minimum Redundancy Maximum Relevance (MRMR) filter method [96].

Possible bottlenecks in the resource allocation. As discussed above for the ablation study, the effectiveness of `AutoPQ-advanced` depends on keeping the computational effort required to identify the optimal sampling hyperparameter low, relative to the time required to train the point forecasting model. That is, a bottleneck may occur if the training time is shorter than the time required for identification. However, such a situation does not arise with the data sets and point forecasting methods considered: The identification effort is independent of the point forecasting method and the data set, and the training takes significantly longer than the identification (see Fig. 2). Nevertheless, bottlenecks in cases of fast training times (e.g., when fine-tuning pre-trained point forecasting models) could be avoided by switching from Algorithms 1 to 3 once the training time is shorter than the identification time.

Desirable probabilistic properties. The benchmarking demonstrates that `AutoPQ` can be optimized to effectively minimize CRPS. Moreover, our previous work [81] highlights that optimizing the sampling hyperparameter for a customized metric allows the forecast to be tailored to different probabilistic properties. However, it remains unclear which probabilistic properties should be prioritized for specific real-world smart grid applications. This gap could be addressed by directly measuring the so-called forecast value of a configuration within the application, rather than relying on a probabilistic metric as a proxy. For instance, the forecast value could be assessed by using the resulting costs in a forecast-based electricity cost optimization problem, as shown in [97].[29] This approach enables the HPO to assess how well the resulting probabilistic properties align with the application's needs, rather than relying on a chosen probabilistic metric as a proxy.

5.5. Benefits

`AutoPQ`'s ability to automatically design high-quality probabilistic forecasts with tailored properties is essential for scaling smart grid applications. With the cost-effectiveness of training `AutoPQ-advanced`, we anticipate a swift return on investments for many smart grid applications, particularly where forecast quality is critical for decision-making. A key advantage of our approach is its versatility: the cINN can generate quantiles or PIs for any point forecasting method. This flexibility enables the integration of new forecasting methods into `AutoPQ`'s CASH

[29] In this study, several MLP-based point forecasting models are trained using different loss functions, and the model with the highest forecast value is selected.

process, while consistently removing underperforming point forecasting methods. Moreover, the joint optimization of the point forecasting methods' hyperparameters and the cINN's sampling hyperparameter is not limited to mathematically derivable validation metrics. This is crucial because it supports the use of any metrics for the assessment in the HPO process, including customized metrics or direct measurements of the so-called forecast value within smart grid applications, as previously suggested.

6. Conclusion and outlook

Designing probabilistic time series forecasting models for smart grid applications includes several challenges, i.e., (i) quantifying forecast uncertainty in an unbiased and accurate manner, (ii) automating the selection and enhancement of forecasting models to meet application needs, and (iii) quantifying the environmental impacts required for performance improvements in view of sustainability.

In order to address these challenges, we introduce `AutoPQ`, a novel method for generating probabilistic forecasts. `AutoPQ` uses a cINN to convert point forecasts into quantile forecasts, leveraging existing well-designed point forecasting methods. `AutoPQ` automates the selection and optimization of these methods, ensuring the best model is chosen and fine-tuned to improve probabilistic forecasting performance. To handle varying performance needs and available computing power, `AutoPQ` comes with two configurations: the default configuration suitable for providing high-quality probabilistic forecasts on general-purpose computing systems, and the advanced configuration for further enhancing forecast quality on HPC systems.

The evaluation demonstrates that `AutoPQ` is both cost-effective and scalable, with significant performance improvements in forecasting quality over existing direct probabilistic forecasting benchmarks, averaging at least 15.1 %, and point forecast-based probabilistic benchmarks, averaging at least 9.1 %. Additionally, the evaluation addresses environmental concerns by assessing `AutoPQ`'s strategies for saving computing resources. Specifically, the energy consumption can be reduced by 60 % by adjusting `AutoPQ-advanced`'s successive halving strategy based on the ablation study results (omit SM methods, reduce time budget, and use `AutoPQ-default` as initial pruning round). Furthermore, information on `AutoPQ`'s electricity consumption and monetary costs required for performance improvements is reported. Specifically, the average improvement in forecast quality of 5 % of `AutoPQ-advanced` over `AutoPQ-default` additionally consumes 2.9 kWh and additionally costs 23.12 $.

Future work will focus on refining the feature selection of `AutoPQ`, and incorporating the forecast value as the assessment metric to tailor the forecast directly to the application's needs, rather than using a probabilistic metric like the CRPS as a proxy.

CRediT authorship contribution statement

Stefan Meisenbacher: Writing – review & editing, Writing – original draft, Visualization, Methodology, Formal analysis, Conceptualization. **Kaleb Phipps:** Writing – review & editing, Methodology, Data curation, Conceptualization. **Oskar Taubert:** Writing – review & editing. **Marie Weiel:** Writing – review & editing. **Markus Götz:** Writing – review & editing, Supervision, Funding acquisition. **Ralf Mikut:** Writing – review & editing, Supervision, Funding acquisition. **Veit Hagenmeyer:** Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project is funded by the Helmholtz Association under the Program "Energy System Design", the Helmholtz Association's Initiative and Networking Fund through Helmholtz AI, and was performed on the HoreKa supercomputer funded by the Ministry of Science, Research and the Arts Baden-Württemberg and by the Federal Ministry of Education and Research.

Appendix A.

Table 6
The architecture of the cINN, which transforms a point forecasting model into a probabilistic one [21].

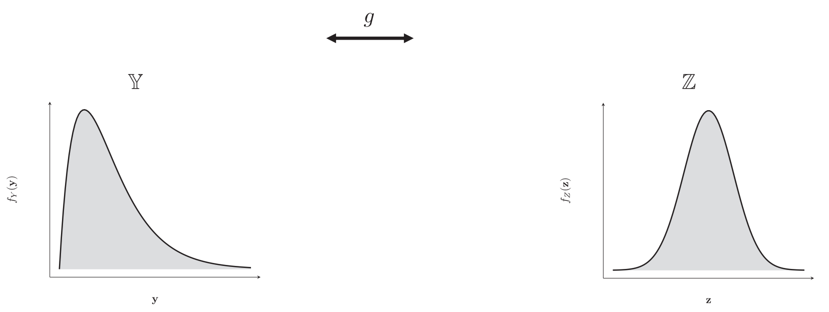
Parameter	Description
Layers per block	`Glow` coupling layer [76] and random permutation
Number of blocks	5
Sub-network in block	Fully connected
Input	[Output of previous coupling layer, output of conditioning network]
Hidden layer 1	Dense 32 neurons; activation: tanh
Hidden layer 2	Dense 24 neurons; activation: tanh
Conditioning network	Fully connected
Input	[Calendar information, historical information, exogenous features if available]
Hidden layer 1	Dense 8 neurons; activation: tanh
Hidden layer 2	Dense 4 neurons; activation: linear

Table 7
The configuration space Λ_q for the cINN, which transforms a point forecasting model into a probabilistic one, in the respective naming convention [81].

Hyperparameter	Value range	Default value
`sampling_std`	$[0.01, 3.0]$	0.1

Table 8
The configuration of the `Propulate` EA for HPO, in the respective naming convention [83].

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



$$\begin{tikzcd}
\hat{\mathbf{y}}[k+1,\ldots,k+H] \arrow[r, "\text{apply } g"] & \hat{\mathbf{z}}[k+1,\ldots,k+H] = g(\hat{\mathbf{y}}[k+1,\ldots,k+H],\mathbf{C}) \arrow[d, "\text{sample in latent space}"'] \\
 & \hat{\mathbf{z}}_i = \hat{\mathbf{z}} + \mathbf{r}^i, \quad i \in \{1,\ldots,N_\sigma\}, \quad \mathbf{r}^i \sim \mathcal{N}(0,\sigma) \arrow[d, "\text{repeat to include uncertainty}"'] \\
\hat{\mathbf{y}}^\sigma[k+1,\ldots,k+H] = \{\hat{\mathbf{y}}_1,\hat{\mathbf{y}}_2,\ldots,\hat{\mathbf{y}}_{N_\sigma}\} \arrow[d, "\text{calculate quantiles}"'] & \hat{\mathbf{z}}^\sigma[k+1,\ldots,k+H] = \{\hat{\mathbf{z}}_1,\hat{\mathbf{z}}_2,\ldots,\hat{\mathbf{z}}_{N_\sigma}\} \arrow[l, "\text{apply } g^{-1}"'] \\
\hat{\mathbf{y}}^*[k+1,\ldots,k+H] &
\end{tikzcd}$$

Fig. 12. The sampling process for including uncertainty: first, the point forecast is mapped from the unknown distribution into the known latent space distribution (latent space original representation). Since the latent space is Gaussian-distributed, we can sample in the neighborhood of the representation by adding Gaussian noise to the original representation. Given the set of forecasts representing the uncertainty in the latent space, we map them back to the realization space. Finally, the quantiles are calculated to generate a probabilistic forecast.

Table 9
The CRPS evaluated on the hold-out test sub-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 datasets (Load-BW, Load-GCP, and Price), either no negative values exist in the quantile forecasts, respectively, or negative values are valid.

	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 10
The configuration spaces $\mathbf{\Lambda}_\mathrm{p}$ for the SM-based point forecasting methods in sktime naming convention [98] with the seasonal period $s = 24$.

Forecasting method	Hyperparameter	Value range	Default value
`SARIMAX` [16]	`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` [16]	`error`	`{add, mul}`	`add`
	`trend`	`{add, mul, None}`	`None`
	`seasonal`	`{add, mul, None}`	`None`
	`damped_trend`	`{True, False}`	`False`
`TBATS` [59]	`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 11
The configuration spaces Λ_p for the ML-based point forecasting methods in the respective naming conventions.

Forecasting method	Hyperparameter	Value range	Default value
`MLPRegressor` [94]	`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` [99]	`C`	$[0.01, 100]$	1.0
	`epsilon`	$[0.001, 1]$	0.1
	`kernel`	{`rbf, laplacian, sigmoid`}	`rbf`
`XGBRegressor` [100]	`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 12
The configuration spaces Λ_p for the DL-based (point) forecasting methods in PyTorch Forecasting naming convention [92].

Forecasting method	Hyperparameter	Value range	Default value
`DeepAR` [28]	`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` [12]	`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` [67]	`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 13
Overview of the training, validation, and test sub-sets of the data with hourly resolution used for evaluation, as well as the selected features based on [21]. The names of the target variables and the features refer to the column names in the datasets.

Data set	Target variable	Features	Training sub-set	Validation sub-set	Test sub-set
Load-BW	`load_power_statistics`	`Seasonal features` (7)–(9) `Lag features` (6)	[0, 4904] (204.3 d)	[4905, 7007] (87.6 d)	[7008, 8760] (73.0 d)
Load-GCP	`MT_158`	`Seasonal features` (7)–(9) `Lag features` (6)	[0, 14716] (613.2 d)	[14717, 21023] (262.8 d)	[21024, 26280] (219.0 d)

Mobility	`cnt`	`Seasonal features` (7)–(9)
		`Lag features` (6)
		`temp`
		`hum`
		`windspeed`
		`weathersit`
			[0, 9824]
			(409.3 d)
				[9825, 14034]
				(175.4 d)
					[14035, 17544]
					(146.2 d)

Price	`Zonal Price`	`Seasonal features` (7)–(9)
		`Lag features` (6)
		`Forecast Total Load`
		`Forecast Zonal Load`
			[0, 14541]
			(605.9 d)
				[14542, 20773]
				(259.6 d)
					[20774, 25968]
					(216.4 d)

PV	`POWER`	`Seasonal features` (7)–(9)
		`Lag features` (6)
		`SSRD`
		`TCC`
			[0, 11033]
			(459.7 d)
				[11034, 15762]
				(197.0 d)
					[15763, 19704]
					(164.2 d)

WP	`TARGETVAR`	`Seasonal features` (7)–(9)
		`Lag features` (6)
		`U100`
		`V100`
		`Speed100`
			[0, 9400]
			(391.7 d)
				[9401, 13430]
				(167.9 d)
					[13431, 16789]
					(139.9 d)

`cnt`: count; `temp`: air temperature; `hum`: humidity; `weathersit`: weather situation; `SSRD`: Surface Short-wave (solar) Radiation Downwards; `TCC`: Total Cloud Cover; `U100`, `V100`, `Speed100`: wind speeds at 100 m above ground, with the eastward component (U), the northward component (V), and the total wind speed (Speed).

Table 14
The validation CRPS compared for three trial generators in the inner loop of Algorithm 1 (random search, BO-TPE, BO-TPE-PK). Across the six datasets and the nine forecasting methods, only minor differences occur.

	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 15
The ranks of the point forecasting methods in the successive halving pruning strategy, evaluated over five runs. Black numbers are the ranks of active methods, while red numbers are the ranks of inactive methods as it would have evolved without pruning.

	ETS					sARIMAX					TBATS					MLP					SVR					XGB					DeepAR					N-HiTS					TFT					
Load-BW	9	9	9	9	9	8	8	8	8	8	7	7	7	7	7	4	3	3	3	3	2	2	2	2	2	1	1	1	1	1	5	6	6	6	6	3	4	4	5	5	6	5	5	4	4	$B_t = 0$ h
	9	9	9	9	9	8	8	7	8	8	7	7	8	7	7	2	2	2	2	2	4	4	4	4	4	1	1	1	1	1	6	6	6	6	6	3	3	3	3	3	5	5	5	5	5	$B_t = 0.9$ h
	9	9	9	9	9	8	8	7	8	8	7	7	8	7	7	3	3	2	2	2	4	4	4	4	4	1	1	1	1	1	6	6	6	6	6	2	2	3	3	3	5	5	5	5	5	$B_t = 2$ h
	9	9	9	9	9	8	8	7	8	8	7	7	8	7	7	2	3	2	3	2	4	4	4	4	4	1	1	1	1	1	6	6	6	6	6	3	2	3	2	3	5	5	5	5	5	$B_t = 4$ h
	9	9	9	9	9	8	8	7	8	8	7	7	8	7	7	2	3	2	2	2	4	4	4	4	4	1	1	1	1	1	6	6	6	6	6	3	2	3	3	3	5	5	5	5	5	$B_t = 8$ h
Load-GCP	9	9	9	9	9	7	7	7	7	7	8	8	8	8	8	4	3	3	3	4	3	2	2	2	3	1	1	1	1	1	6	6	6	6	6	5	4	5	5	5	2	5	4	4	2	$B_t = 0$ h
	9	9	9	9	9	8	8	8	7	7	7	7	7	8	8	4	3	3	3	4	2	2	2	2	2	1	1	1	1	1	6	6	6	6	6	5	4	5	5	5	3	5	4	4	3	$B_t = 0.9$ h
	9	9	9	9	9	8	8	8	7	7	7	7	7	8	8	4	3	4	3	4	3	4	3	2	3	1	1	1	1	1	6	6	6	6	6	5	5	5	5	5	2	2	2	4	2	$B_t = 2$ h
	9	9	9	9	9	8	8	8	7	7	7	7	7	8	8	4	3	4	3	4	3	4	3	2	2	1	1	1	1	1	6	6	6	6	6	5	5	5	4	5	2	2	2	5	3	$B_t = 4$ h
	9	9	9	9	9	8	8	8	7	7	7	7	7	8	8	4	3	4	3	4	3	4	3	2	3	1	1	1	1	1	6	6	6	6	6	5	5	5	4	5	2	2	2	5	2	$B_t = 8$ h
Mobility	9	9	9	9	9	8	8	8	8	8	7	7	7	7	7	4	4	4	4	4	6	6	6	6	6	5	5	5	5	5	2	2	2	2	2	3	3	3	3	3	1	1	1	1	1	$B_t = 0$ h
	9	9	9	7	9	8	8	8	8	8	7	7	7	9	7	4	4	4	4	4	5	5	5	5	5	6	6	6	6	6	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	$B_t = 0.9$ h
	9	9	9	7	9	8	8	8	8	8	7	7	7	9	7	4	4	4	4	4	6	5	5	5	5	5	6	6	6	6	2	3	3	3	3	3	2	2	2	2	1	1	1	1	1	$B_t = 2$ h
	9	9	9	7	9	8	8	8	8	8	7	7	7	9	7	4	4	4	4	4	6	5	6	5	6	5	6	5	6	5	2	3	3	3	3	3	2	2	2	2	1	1	1	1	1	$B_t = 4$ h
	9	9	9	7	9	8	8	8	8	8	7	7	7	9	7	4	4	4	4	4	6	6	6	6	6	5	5	5	5	5	2	3	3	3	3	3	2	2	2	2	1	1	1	1	1	$B_t = 8$ h
Price	8	8	8	8	9	9	9	9	9	8	7	7	7	7	7	3	2	4	2	3	4	4	5	5	5	5	6	6	6	6	6	3	3	4	2	1	1	1	1	1	2	5	2	3	4	$B_t = 0$ h
	9	9	9	9	9	7	8	7	7	7	8	7	8	8	8	2	2	3	2	3	4	4	5	5	5	5	6	6	6	6	6	3	4	4	2	1	1	1	1	1	3	5	2	3	4	$B_t = 0.9$ h
	9	9	9	9	9	7	8	7	7	7	8	7	8	8	8	2	2	3	2	3	5	5	5	5	5	6	6	6	6	6	3	3	2	3	4	1	1	1	1	1	4	4	4	4	2	$B_t = 2$ h
	9	9	9	9	9	7	8	7	7	7	8	7	8	8	8	2	2	2	2	2	5	5	5	5	5	6	6	6	6	6	3	3	3	3	4	1	1	1	1	1	4	4	4	4	3	$B_t = 4$ h
	9	9	9	9	9	7	8	7	7	7	8	7	8	8	8	2	2	2	2	2	6	5	5	6	5	5	6	6	5	6	3	4	3	4	4	1	1	1	1	1	4	3	4	3	3	$B_t = 8$ h
PV	8	8	8	8	8	9	9	9	9	9	7	7	7	7	7	4	3	3	4	3	2	2	2	2	2	1	1	1	1	1	5	5	6	6	5	6	6	5	5	6	3	4	4	3	4	$B_t = 0$ h
	8	8	8	8	8	9	9	9	9	9	6	6	7	6	7	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	7	7	6	7	6	5	5	5	5	5	4	4	4	4	4	$B_t = 0.9$ h
	8	8	8	8	8	9	9	9	9	9	6	7	7	6	7	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	7	5	6	7	6	5	6	5	5	5	4	4	4	4	4	$B_t = 2$ h
	8	8	8	8	8	9	9	9	9	9	7	7	7	6	7	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	6	6	6	7	6	5	5	5	5	5	4	4	4	4	4	$B_t = 4$ h
	8	8	8	8	8	9	9	9	9	9	7	7	7	7	7	3	3	4	3	3	2	2	2	2	2	1	1	1	1	1	6	6	6	6	6	5	5	5	5	5	4	4	3	4	4	$B_t = 8$ h
WP	9	9	9	9	9	7	7	5	7	6	8	8	7	8	7	3	3	3	3	3	2	2	2	2	2	1	1	1	1	1	6	6	8	6	8	5	4	4	5	4	4	5	6	4	5	$B_t = 0$ h
	9	9	9	9	9	7	7	5	7	6	8	8	7	8	7	2	2	2	2	3	3	3	3	3	2	1	1	1	1	1	6	6	8	6	8	5	4	4	4	4	4	5	6	5	5	$B_t = 0.9$ h
	9	9	9	9	9	7	7	7	7	7	8	8	8	8	8	3	2	3	2	3	2	3	2	3	2	1	1	1	1	1	6	5	5	5	6	5	4	4	4	4	4	6	6	6	5	$B_t = 2$ h
	9	9	9	9	9	7	7	7	7	7	8	8	8	8	8	3	2	3	2	3	2	3	2	3	2	1	1	1	1	1	6	5	5	5	6	4	4	4	4	4	5	6	6	6	5	$B_t = 4$ h
	9	9	9	9	9	7	7	7	7	7	8	8	8	8	8	3	2	3	2	2	2	3	2	3	3	1	1	1	1	1	5	6	6	6	6	4	4	4	4	4	6	5	5	5	5	$B_t = 8$ h
run no.	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	

Data availability

The data is openly available and referenced accordingly to enable reproduction.

References

[1] Khajeh H, Laaksonen H. Applications of probabilistic forecasting in smart grids: a review. Appl Sci 2022;12. doi: https://doi.org/10.3390/app12041823.
[2] Summers T, Warrington J, Morari M, Lygeros J. Stochastic optimal power flow based on conditional value at risk and distributional robustness. Int J Elec Power 2015;72:116–25. doi: https://doi.org/10.1016/j.ijepes.2015.02.024, The Special Issue for 18th Power Systems Computation Conference.
[3] Mieth R, Dvorkin Y. Data-driven distributionally robust optimal power flow for distribution systems. IEEE Control Syst Lett 2018;2:363–8.
[4] Appino RR, González Ordiano JA, Mikut R, Faulwasser T, Hagenmeyer V. On the use of probabilistic forecasts in scheduling of renewable energy sources coupled to storages. Appl Energy 2018;210:1207–18. doi: https://doi.org/10.1016/j.apenergy.2017.08.133.
[5] Henni S, Becker J, Staudt P, vom Scheidt F, Weinhardt C. Industrial peak shaving with battery storage using a probabilistic forecasting approach: economic evaluation of risk attitude. Appl Energy 2022;327:120088. doi: https://doi.org/10.1016/j.apenergy.2022.120088.
[6] Phipps K, Schwenk K, Briegel B, Mikut R, Hagenmeyer V. Customized uncertainty quantification of parking duration predictions for EV smart charging. IEEE Internet Things J 2023;10:20649–61. doi: https://doi.org/10.1109/JIOT.2023.3299201.
[7] Huber J, Dann D, Weinhardt C. Probabilistic forecasts of time and energy flexibility in battery electric vehicle charging. Appl Energy 2020;262:114525. doi: https://doi.org/10.1016/j.apenergy.2020.114525.
[8] Zhang Y, Cheng C, Cao R, Li G, Shen J, Wu X. Multivariate probabilistic forecasting and its performance's impacts on long-term dispatch of hydro-wind hybrid systems. Appl Energy 2021;283:116243. doi: https://doi.org/10.1016/j.apenergy.2020.116243.
[9] Han J, Yan L, Li Z. A task-based day-ahead load forecasting model for stochastic economic dispatch. IEEE Trans Power Syst 2021;36:5294–304. doi: https://doi.org/10.1109/TPWRS.2021.3072904.
[10] Oreshkin BN, Carpov D, Chapados N, Bengio Y. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In: International conference on learning representations; 2020.
[11] Nie Y, Nguyen NH, Sinthong P, Kalagnanam J. A time series is worth 64 words: long-term forecasting with transformers. In: International Conference on Learning Representations; 2023.
[12] Challú C, Olivares KG, Oreshkin BN, Garza Ramirez F, Mergenthaler Canseco M, Dubrawski A. N-HiTS: neural hierarchical interpolation for time series forecasting. Proc AAAI Conf Artif Intell 2023;37:6989–97. doi: https://doi.org/10.1609/aaai.v37i6.25854.
[13] Das A, Kong W, Leach A, Mathur SK, Sen R, Yu R. Long-term forecasting with TiDE: time-series dense encoder. Trans Mach Learn Res 2023; 1–21. doi: https://doi.org/10.48550/arXiv.2304.08424.
[14] Zeng A, Chen M, Zhang L, Xu Q. Are transformers effective for time series forecasting? Proc AAAI Conf Artif Intell 2023;37:11121–8. doi: https://doi.org/10.1609/aaai.v37i9.26317.
[15] Petropoulos F, Apiletti D, Assimakopoulos V, Babai MZ, Barrow DK, Ben Taieb S, et al. Forecasting: theory and practice. Int J Forecast 2022;38:705–871. doi: https://doi.org/10.1016/j.ijforecast.2021.11.001.
[16] Hyndman RJ, Athanasopoulos G. Forecasting: principles and practice. 3rd ed. Melbourne, Australia: OTexts; 2021.
[17] Camporeale E, Chu X, Agapitov OV, Bortnik J. On the generation of probabilistic forecasts from deterministic models. Space Weather 2019;17:455–75. doi: https://doi.org/10.1029/2018SW002026.
[18] Williams WH, Goodman ML. A simple method for the construction of empirical confidence limits for economic forecasts. J Am Stat Assoc 1971;66:752–4. doi: https://doi.org/10.1080/01621459.1971.10482340.
[19] Stankeviciute K, Alaa AM, van der Schaar M. Conformal time series forecasting. In: Ranzato M, Beygelzimer A, Dauphin Y, Liang P, Vaughan JW, editors. Advances in neural information processing systems vol. 34. Curran Associates, Inc.; 2021. p. 6216–28.
[20] Wang Y, Hug G, Liu Z, Zhang N. Modeling load forecast uncertainty using generative adversarial networks. Electr Power Syst Res 2020;189:106732. doi: https://doi.org/10.1016/j.epsr.2020.106732.
[21] Phipps K, Heidrich B, Turowski M, Wittig M, Mikut R, Hagenmeyer V. Generating probabilistic forecasts from arbitrary point forecasts using a conditional invertible neural network. Appl Intell 2024;54:6354–82. doi: https://doi.org/10.1007/s10489-024-05346-9.
[22] Hutter F, Kotthoff L, Vanschoren J. Automated machine learning: methods, systems, challenges. In: The Springer series on challenges in machine learning. Cham, Switzerland: Springer International Publishing; 2019. doi: https://doi.org/10.1007/978-3-030-05318-5.
[23] Rätz M, Javadi AP, Baranski M, Finkbeiner K, Müller D. Automated data-driven modeling of building energy systems via machine learning algorithms. Energy Build 2019;202:109384. doi: https://doi.org/10.1016/j.enbuild.2019.109384.
[24] Shah SY, Patel D, Vu L, Dang X-H, Chen B, Kirchner P, et al. AutoAI-TS: AutoAI for time series forecasting. In: Proceedings of the 2021 international conference on management of data, SIGMOD '21. New York, USA: Association for Computing Machinery; 2021. p. 2584–96. doi: https://doi.org/10.1145/3448016.3457557.
[25] Deng D, Karl F, Hutter F, Bischl B, Lindauer M. Efficient automated deep learning for time series forecasting. In: Amini M-R, Canu S, Fischer A, Guns T, Novak PK, Tsoumakas G, editors. Machine learning and knowledge discovery in databases. Cham, Switzerland: Springer Nature; 2023. p. 664–80.
[26] Shchur O, Turkmen AC, Erickson N, Shen H, Shirkov A, Hu T, et al. AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting. In: International conference on automated machine learning. PMLR; 2023. p. 9-1.
[27] Debus C, Piraud M, Streit A, Theis F, Götz M. Reporting electricity consumption is essential for sustainable AI. Nat Mach Intell 2023;5:1176–8. doi: https://doi.org/10.1038/s42256-023-00750-1.
[28] Salinas D, Flunkert V, Gasthaus J, Januschowski T. DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int J Forecast 2020;36:1181–91. doi: https://doi.org/10.1016/j.ijforecast.2019.07.001.
[29] Rangapuram SS, Werner LD, Benidis K, Mercado P, Gasthaus J, Januschowski T. End-to-end learning of coherent probabilistic forecasts for hierarchical time series vol. 139. In: Meila M, Zhang T, editors. Proceedings of the 38th international conference on machine learning of proceedings of machine learning research. PMLR; 2021. p. 8832–43.
[30] Heidrich B, Turowski M, Phipps K, Schmieder K, Süß W, Mikut R, et al. Controlling non-stationarity and periodicities in time series generation using conditional invertible neural networks. Appl Intell 2023;53:8826–43. doi: https://doi.org/10.1007/s10489-022-03742-7.
[31] Cannon AJ. Quantile regression neural networks: implementation in R and application to precipitation downscaling. Comput Geosci 2011;37:1277–84. doi: https://doi.org/10.1016/j.cageo.2010.07.005.
[32] Wang Y, Gan D, Sun M, Zhang N, Lu Z, Kang C. Probabilistic individual load forecasting using pinball loss guided LSTM. Appl Energy 2019;235:10–20. doi: https://doi.org/10.1016/j.apenergy.2018.10.078.
[33] Hu J, Luo Q, Tang J, Heng J, Deng Y. Conformalized temporal convolutional quantile regression networks for wind power interval forecasting. Energy 2022;248:123497. doi: https://doi.org/10.1016/j.energy.2022.123497.
[34] Masood Z, Gantassi R, Choi Y. Enhancing short-term electric load forecasting for households using quantile LSTM and clustering-based probabilistic approach. IEEE Access 2024;12:77257–68. doi: https://doi.org/10.1109/ACCESS.2024.3406439.
[35] Hu J, Tang J, Liu Z. A novel time series probabilistic prediction approach based on the monotone quantile regression neural network. Inf Sci (NY) 2024;654:119844. doi: https://doi.org/10.1016/j.ins.2023.119844.
[36] Zhang W, Quan H, Srinivasan D. Parallel and reliable probabilistic load forecasting via quantile regression forest and quantile determination. Energy 2018;160:810–9. doi: https://doi.org/10.1016/j.energy.2018.07.019.
[37] He Y, Liu R, Li H, Wang S, Lu X. Short-term power load probability density forecasting method using kernel-based support vector quantile regression and copula theory. Appl Energy 2017;185:254–66. doi: https://doi.org/10.1016/j.apenergy.2016.10.079.
[38] He Y, Yan Y, Xu Q. Wind and solar power probability density prediction via fuzzy information granulation and support vector quantile regression. Int J Elec Power 2019;113:515–27. doi: https://doi.org/10.1016/j.ijepes.2019.05.075.
[39] González Ordiano JA, Gröll L, Mikut R, Hagenmeyer V. Probabilistic energy forecasting using the nearest neighbors quantile filter and quantile regression. Int J Forecast 2020;36:310–23. doi: https://doi.org/10.1016/j.ijforecast.2019.06.003.
[40] Verbois H, Rusydi A, Thiery A. Probabilistic forecasting of day-ahead solar irradiance using quantile gradient boosting. Sol Energy 2018;173:313–27. doi: https://doi.org/10.1016/j.solener.2018.07.071.
[41] Guo H, Huang B, Wang J. Probabilistic load forecasting for integrated energy systems using attentive quantile regression temporal convolutional network. Adv Appl Energy 2024;14:100165. doi: https://doi.org/10.1016/j.adapen.2024.100165.
[42] Aprillia H, Yang H-T, Huang C-M. Statistical load forecasting using optimal quantile regression random forest and risk assessment index. IEEE Trans Smart Grid 2021;12:1467–80. doi: https://doi.org/10.1109/TSG.2020.3034194.
[43] Brusaferri A, Matteucci M, Spinelli S, Vitali A. Probabilistic electric load forecasting through Bayesian mixture density networks. Appl Energy 2022;309:118341. doi: https://doi.org/10.1016/j.apenergy.2021.118341.
[44] Sprangers O, Schelter S, de Rijke M. Parameter-efficient deep probabilistic forecasting. Int J Forecast 2023;39:332–45. doi: https://doi.org/10.1016/j.ijforecast.2021.11.011.
[45] Wen R, Kari T. Deep generative quantile-copula models for probabilistic forecasting. In: Proceedings of the 36th international conference on machine learning; 2019. Paper: Time Series Workshop at ICML 2019.
[46] Rasul K, Sheikh A-S, Schuster I, Bergmann UM, Vollgraf R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In: International conference on learning representations; 2021.
[47] Jamgochian A, Wu D, Menda K, Jung S, Kochenderfer MJ. Conditional approximate normalizing flows for joint multi-step probabilistic forecasting with application to electricity demand. 2022. arXiv preprint arXiv:2201.02753.
[48] Arpogaus M, Voss M, Sick B, Nigge-Uricher M, Dürr O. Short-term density forecasting of low-voltage load using Bernstein-polynomial normalizing flows. IEEE Trans Smart Grid 2023;14:4902–11. doi: https://doi.org/10.1109/TSG.2023.3254890.
[49] Fanfarillo A, Roozitalab B, Hu W, Cervone G. Probabilistic forecasting using deep generative models. GeoInformatica 2021;25:127–47. doi: https://doi.org/10.1007/s10707-020-00425-8.
[50] Zhang L, Zhang B. Scenario forecasting of residential load profiles. IEEE J Sel Areas Commun 2020;38:84–95. doi: https://doi.org/10.1109/JSAC.2019.2951973.

[51] Ge L, Liao W, Wang S, Bak-Jensen B, Pillai JR. Modeling daily load profiles of distribution network for scenario generation using flow-based generative network. IEEE Access 2020;8:77587–97. doi: https://doi.org/10.1109/ACCESS.2020.2989350.

[52] Dumas J, Wehenkel A, Lanaspeze D, Cornélusse B, Sutera A. A deep generative model for probabilistic energy forecasting in power systems: normalizing flows. Appl Energy 2022;305:117871. doi: https://doi.org/10.1016/j.apenergy.2021.117871.

[53] Cramer E, Witthaut D, Mitsos A, Dahmen M. Multivariate probabilistic forecasting of intraday electricity prices using normalizing flows. Appl Energy 2023;346:121370. doi: https://doi.org/10.1016/j.apenergy.2023.121370.

[54] Chernozhukov V, Wüthrich K, Zhu Y. Distributional conformal prediction. Proc Natl Acad Sci 2021;118:e2107794118. doi: https://doi.org/10.1073/pnas.2107794118.

[55] Zaffran M, Féron O, Goude Y, Josse J, Dieuleveut A. Adaptive conformal predictions for time series. In: International conference on machine learning. PMLR; 2022. p. 25834–66.

[56] Hu J, Hu W, Cao D, Sun X, Chen J, Huang Y, et al. Probabilistic net load forecasting based on transformer network and Gaussian process-enabled residual modeling learning method. Renew Energy 2024;225:120253. doi: https://doi.org/10.1016/j.renene.2024.120253.

[57] Hyndman RJ, Koehler AB, Snyder RD, Grose S. A state space framework for automatic forecasting using exponential smoothing methods. Int J Forecast 2002;18:439–54. doi: https://doi.org/10.1016/S0169-2070(01)00110-8.

[58] Hyndman RJ, Khandakar Y. Automatic time series forecasting: the forecast package for R. J Stat Softw 2008;27:1–22. doi: https://doi.org/10.18637/jss.v027.i03.

[59] De Livera AM, Hyndman RJ, Snyder RD. Forecasting time series with complex seasonal patterns using exponential smoothing. J Am Stat Assoc 2011;106:1513–27. doi: https://doi.org/10.1198/jasa.2011.tm09771.

[60] Maldonado S, González A, Crone S. Automatic time series analysis for electric load forecasting via support vector regression. Appl Soft Comput 2019;83:105616. doi: https://doi.org/10.1016/j.asoc.2019.105616.

[61] Valente JM, Maldonado S. SVR-FFS: a novel forward feature selection approach for high-frequency time series forecasting using support vector regression. Expert Syst Appl 2020;160:113729. doi: https://doi.org/10.1016/j.eswa.2020.113729.

[62] Fan S, Qin X, Jia Z, Qi X, Lin M. ELM-based improved layered ensemble architecture for time series forecasting. IEEE Access 2019;7:97827–37. doi: https://doi.org/10.1109/ACCESS.2019.2927047.

[63] Barros FS, Cerqueira V, Soares C. Empirical study on the impact of different sets of parameters of gradient boosting algorithms for time-series forecasting with LightGBM. In: Pham DN, Theeramunkong T, Governatori G, Liu F, editors. PRICAI 2021: trends in artificial intelligence. Cham, Switzerland: Springer International Publishing; 2021. p. 454–65.

[64] Wu X, Zhang D, Guo C, He C, Yang B, Jensen CS. AutoCTS: automated correlated time series forecasting. Proc VLDB Endow 2021;15:971–83. doi: https://doi.org/10.14778/3503585.3503604.

[65] Kong W, Dong ZY, Luo F, Meng K, Zhang W, Wang F, et al. Effect of automatic hyperparameter tuning for residential load forecasting via deep learning. In: 2017 Australasian universities power engineering conference (AUPEC); 2017. p. 1–6. doi: https://doi.org/10.1109/AUPEC.2017.8282478.

[66] Al Mamun A, Hoq M, Hossain E, Bayindir R. 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. p. 886–91. doi: https://doi.org/10.1109/ICRERA47325.2019.8996550

[67] Lim B, Arık SO, Loeff N, Pfister T. Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int J Forecast 2021;37:1748–64. doi: https://doi.org/10.1016/j.ijforecast.2021.03.012

[68] Xu C, Sun Y, Du A, Gao D-C. Quantile regression based probabilistic forecasting of renewable energy generation and building electrical load: a state of the art review. J Building Eng 2023;79:107772. doi: https://doi.org/10.1016/j.jobe.2023.107772

[69] Haupt SE, Garcia Casado M, Davidson M, Dobschinski J, Du P, Lange M, et al. The use of probabilistic forecasts: applying them in theory and practice. IEEE Power Energy Mag 2019;17:46–57. doi: https://doi.org/10.1109/MPE.2019.2932639

[70] Smith RC. Uncertainty quantification: theory, implementation, and applications. In: Society for industrial and applied mathematics. 1st ed. Philadelphia, USA; 2013. doi: https://doi.org/10.1137/1.9781611973228

[71] Fontana M, Zeni G, Vantini S. Conformal prediction: a unified review of theory and new challenges. Bernoulli 2023;29:1–23. doi: https://doi.org/10.3150/21-BEJ1447

[72] Alexandrov A, Benidis K, Bohlke-Schneider M, Flunkert V, Gasthaus J, Januschowski T, et al. GluonTS: probabilistic and neural time series modeling in python. J Mach Learn Res 2020;21:1–6.

[73] Olivares KG, Challú C, Garza F, Canseco MM, Dubrawski A. NeuralForecast: user friendly state-of-the-art neural forecasting models. Salt Lake City, Utah, US: PyCon; 2022.

[74] Meisenbacher S, Turowski M, Phipps K, Rätz M, Müller D, Hagenmeyer V, et al. Review of automated time series forecasting pipelines. WIREs Data Min Knowl Discov 2022;12:e1475. doi: https://doi.org/10.1002/widm.1475

[75] Bergstra J, Bengio Y. Random search for hyperparameter optimization. J Mach Learn Res 2012;13:281–305.

[76] Kingma DP, Dhariwal P. Glow: generative flow with invertible 1x1 convolutions. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in neural information processing systems, vol. 31. Curran Associates, Inc.; 2018.

[77] Ardizzone L, Lüth C, Kruse J, Rother C, Köthe U. Guided image generation with conditional invertible neural networks; 2019. arXiv preprint arXiv:1907.02392.

[78] Heidrich B, Turowski M, Ludwig N, Mikut R, Hagenmeyer V. Forecasting energy time series with profile neural networks. In: Proceedings of the 11th ACM international conference on future energy systems, e-Energy ’20. New York, USA: Association for Computing Machinery; 2020. p. 220–30. doi: https://doi.org/10.1145/3396851.3397683

[79] Giacomazzi E, Haag F, Hopf K. 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, e-Energy ’23. New York, USA: Association for Computing Machinery; 2023. p. 353–60. doi: https://doi.org/10.1145/3575813.3597345

[80] Richter L, Bauer F, Klaiber S, Bretschneider P. Day-ahead electricity load prediction based on calendar features and temporal convolutional networks. In: Valenzuela O, Rojas F, Herrera LJ, Pomares H, Rojas I, editors. Theory and applications of time series analysis and forecasting. Cham, Switzerland: Springer International Publishing; 2023. p. 243–53.

[81] Phipps K, Meisenbacher S, Heidrich B, Turowski M, Mikut R, Hagenmeyer V. Loss-customised probabilistic energy time series forecasts using automated hyperparameter optimisation. In: Proceedings of the 14th ACM international conference on future energy systems, e-Energy ’23 New York, USA: Association for Computing Machinery; 2023. p. 271–86. doi: https://doi.org/10.1145/3575813.3595204

[82] Bergstra J, Yamins D, Cox D. 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, ICML ’13, proceedings of machine learning research. PMLR; 2013. p. 115–23.

[83] Taubert O, Weiel M, Coquelin D, Farshian A, Debus C, Schug A, et al. Massively parallel genetic optimization through asynchronous propagation of populations. In: Bhatele A, Hammond J, Baboulin M, Kruse C, editors. High performance computing. Cham, Switzerland: Springer Nature; 2023. p. 106–24.

[84] Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, et al. Hyperparameter optimization: foundations, algorithms, best practices, and open challenges. WIREs Data Min Knowl Discov 2023;13:e1484. doi: https://doi.org/10.1002/widm.1484

[85] Wiese F, Schlecht I, Bunke W-D, Gerbaulet C, Hirth L, Jahn M, et al. Open power system data: frictionless data for electricity system modelling. Appl Energy 2019;236:401–9. https://open-power-system-data.org/. doi: https://doi.org/10.1016/j.apenergy.2018.11.097.

[86] Trindade A. Electricity load diagrams 2011-2014. UCI Machine Learning Repository; 2015. doi: https://doi.org/10.24432/C58C86

[87] Fanaee-T H. Bike sharing dataset. UCI Machine Learning Repository; 2013. doi: https://doi.org/10.24432/C5W894

[88] Hong T, Pinson P, Fan S, Zareipour H, Troccoli A, Hyndman RJ. Probabilistic energy forecasting: global energy forecasting competition 2014 and beyond. Int J Forecast 2016;32:896–913. doi: https://doi.org/10.1016/j.ijforecast.2016.02.001

[89] Kingma DP, Ba J. Adam: a method for stochastic optimization. In: Bengio Y, LeCun Y, editors. 3rd International conference on learning representations, ICLR 2015, conference track proceedings, May 7–9, 2015. San Diego, CA, USA; 2015.

[90] Heidrich B, Mannsperger L, Turowski M, Phipps K, Schäfer B, Mikut R, et al. Boost short-term load forecasts with synthetic data from transferred latent space information. Energy Inform 2022;5:20. doi: https://doi.org/10.1186/s42162-022-00214-7

[91] Gneiting T, Balabdaoui F, Raftery AE. Probabilistic forecasts, calibration and sharpness. J R Stat Soc 2007;69:243–68. doi: https://doi.org/10.1111/j.1467-9868.2007.00587.x

[92] Beitner J. PyTorch forecasting; 2020. https://towardsdatascience.com/introducing-pytorch-forecasting-64de99b9ef46.

[93] Chollet F, et al. Keras; 2015. https://keras.io.

[94] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in Python. J Mach Learn Res 2011;12:2825–30.

[95] Amazon Web Services, Inc.. Amazon EC2 G4 instances; 2024. https://aws.amazon.com/ec2/instance-types/g4/.

[96] Ding C, Peng H. Minimum redundancy feature selection from microarray gene expression data. In: Computational systems bioinformatics. CSB2003. Proceedings of the 2003 IEEE bioinformatics conference. CSB2003; 2003. p. 523–8. doi: https://doi.org/10.1109/CSB.2003.1227396

[97] Werling D, Beichter M, Heidrich B, Phipps K, Mikut R, Hagenmeyer V. 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, e-Energy ’23 companion. New York, USA: Association for Computing Machinery; 2023. p. 59–71. doi: https://doi.org/10.1145/3599733.3600251

[98] Löning M, Bagnall A, Ganesh S, Kazakov V, Lines J, Király FJ. Sktime: a unified interface for machine learning with time series. In: 2019 workshop on systems for ML at NeurIPS; 2019.

[99] Bao Y, Xiong T, Hu Z. Multi-step-ahead time series prediction using multiple-output support vector regression. Neurocomputing 2014;129:482–93. doi: https://doi.org/10.1016/j.neucom.2013.09.010

[100] Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, KDD ’16. New York, USA: Association for Computing Machinery; 2016. p. 785–94. doi: https://doi.org/10.1145/2939672.2939785