

A Benchmark for Electrical Load Forecasting Across Grid Levels: Time-Series Transformers Outperform Established Methods

MATTHIAS HERTEL*, SEBASTIAN PÜTZ[†], JONATHAN KOLAR, BENJAMIN SCHÄFER[†], RALF MIKUT, and VEIT HAGENMEYER, Karlsruhe Institute of Technology, Germany; [†]Helmholtz AI, Germany

Accurate load forecasting at multiple grid levels is essential for future smart grids, ranging from aggregated control area forecasts for balancing supply and demand to forecasts of individual end-consumer loads for demand-side management and energy management systems. We present a comprehensive benchmark for load forecasting across grid levels, comprising three datasets that represent a transmission system operator control area, low-voltage grid feeders, and individual end consumers. We evaluate ten methods for short-term load forecasting and find that Transformer-based approaches consistently outperform established methods, reducing forecast error by 6.6–10.7 %. To analyze the impact of architectural design, we introduce YAformer, a flexible Transformer architecture that integrates modifications from prior work and is optimized via hyperparameter optimization. However, the standard Transformer achieves superior performance, suggesting that these architectural modifications are not required for accurate load forecasting. We further evaluate the Transformer-based time-series foundation model Chronos-2, which demonstrates competitive zero-shot performance on two datasets but fails to accurately capture special events in the TSO data. Detailed analyses reveal model-specific strengths and weaknesses, and ablation studies highlight the importance of long input contexts, covariates and continuous retraining – aspects that are often overlooked in the time-series forecasting literature.

CCS Concepts: • **Applied computing** → **Forecasting**; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: Load Forecasting, Benchmark, High-Voltage, Low-Voltage, Transformer, Time-Series Foundation Model

Availability of Data and Material:

All three datasets come from public sources. Our code and our preprocessed versions of the datasets are made available at <https://github.com/KIT-IAI/load-forecasting-benchmark>.

ARIMA	Autoregressive Integrated Moving Average
CNN	convolutional neural network
DM test	Diebold–Mariano test
EHV	extra-high voltage
GEFCom	Global Energy Forecasting Competition
HPC	high-performance computing
HPO	hyperparameter optimization
HV	high voltage
kNN	k-Nearest Neighbor
LightGBM	Light Gradient Boosting Machine
LSTM	long short-term memory
LV	low voltage
MAE	mean absolute error
MLP	multilayer perceptron
nMAE	normalized MAE

*Matthias Hertel and Sebastian Pütz contributed equally.

Authors’ Contact Information: Matthias Hertel, matthias.hertel@kit.edu; Sebastian Pütz, sebastian.puetz@kit.edu; Jonathan Kolar, jonathan.kolar@student.kit.edu; Benjamin Schäfer, benjamin.schaefer@kit.edu; Ralf Mikut, ralf.mikut@kit.edu; Veit Hagenmeyer, veit.hagenmeyer@kit.edu, Karlsruhe Institute of Technology, Germany; [†]Helmholtz AI, Germany.

nRMSE	normalized RMSE
ReLU	rectified linear unit
RMSE	root mean square error
TFT	Temporal Fusion Transformer
TSMF	Time-Series Foundation Model
TSO	transmission system operator
YAformer	yet another Transformer

1 Introduction

The transition toward a climate-neutral energy system affects electricity demand and supply. Electrification of the heat, transport, and industrial sectors is increasing overall demand, while the growing integration of decentralized renewable generation at medium- and low-voltage levels introduces greater variability. As a result, maintaining the real-time balance between supply and demand becomes more challenging, and existing grid infrastructure reaches its limits during periods of peak load or high generation [54]. In this context, smart grid technologies aim to improve the utilization of existing assets [40], but their effectiveness relies on accurate electrical load forecasting across multiple levels of the energy system. At the extra-high voltage (EHV) and the high voltage (HV) level, forecasts support dispatch and outage planning as well as balancing energy procurement; at the low voltage (LV) level, they enable congestion management, demand response, and storage scheduling; and at the client level, they are essential for energy management and peak shaving. These applications are characterized by differing data properties, ranging from relatively stable and aggregated time series at the EHV and the HV level to highly heterogeneous and volatile patterns at the low-voltage and client levels, often with abrupt changes due to newly installed assets. At the same time, the ongoing rollout of measurement infrastructure, such as smart meters, is increasing data availability, creating new opportunities for data-driven forecasting approaches. The integration of covariates, such as weather data and calendar information, is crucial to achieve accurate load forecasts [19].

Recent publications on time-series forecasting make use of the Transformer model [46], including Transformer variants particularly designed for time-series forecasting, such as LogSparse Transformer [33], Temporal Fusion Transformer (TFT) [35], Informer [52], FEDformer [53], Autoformer [48] and PatchTST [38]. Since many different time-series Transformer models exist, the present paper analyzes which of the architectural modifications proposed in the literature are beneficial for electrical load forecasting, in order to derive recommendations for practical use cases.

Time-Series Foundation Models (TSFMs) [34] are a new paradigm in time-series forecasting. TSFMs are pretrained on a variety of time-series datasets and applicable to new time series in a zero-shot setting without the need for additional training. Most TSFMs

are built on Transformers, underlining the need for analyzing the architectural design of time-series Transformers.

When new forecasting methods are proposed, such as the time-series Transformers and TSFMs, it is important to benchmark them against established methods to evaluate whether they improve the model accuracy over the state of the art [23]. Because data characteristics vary across load aggregation levels, the present paper benchmarks Transformers and TSFMs against established methods on three different datasets, representing the transmission system operator (TSO) level, the LV feeder level, and the client level. In this context, our contributions are as follows:

- (1) We select and process three highly relevant and representative datasets for benchmarking of newly developed load forecasting methods across grid levels, representing a TSO control area, LV grid feeders and individual end consumers.
- (2) We study the effect of Transformer architectures on the forecast accuracy by introducing YAformer, a flexible Transformer architecture that integrates multiple modifications from architectures proposed in the literature and is optimized with a hyperparameter optimization (HPO).
- (3) We compare standard Transformers, YAformer and the TSFM Chronos-2 [2] with seven established forecasting methods and one baseline and highlight their advantages and disadvantages in terms of forecast accuracy and runtime.

The remainder of the paper is structured as follows: Section 2 presents the related work on time-series Transformers, TSFMs and benchmarking. Section 3 introduces YAformer and the comparison methods. Section 4 presents the datasets and experimental setup. Section 5 describes the results, which are discussed in Section 6. Section 7 concludes the paper and highlights potential future work.

2 Related Work

Electrical load forecasting has a long history, with first approaches dating back to the 1940s [25]. Since then, it became an active field of research, with recently more than 1,000 publications published annually¹. Despite the large number of publications in the field, literature reviews on load forecasting largely agree that there is a small set of established methods that are frequently used in research and in practice [18, 26, 43]. This includes statistical methods like Autoregressive Integrated Moving Average (ARIMA), machine learning methods like k-Nearest Neighbor (kNN) and gradient boosting machines, and deep learning methods like multilayer perceptrons (MLPs), convolutional neural networks (CNNs) and long short-term memory (LSTM) networks.

The Transformer is a deep learning model developed for machine translation [46] that led to improvements in many domains, including language modeling [39], computer vision [11] and tabular data [24]. The Transformer was also adopted for time-series forecasting [49] and electrical load forecasting in particular [51]. Many Transformer architectures for time-series forecasting exist, including LogSparse Transformer [33], TFT [35], Informer [52], FEDformer [53], Autoformer [48] and PatchTST [38]. The architectural modifications of these models include LSTM layers [35],

LogSparse attention [33], convolutional attention [33], ProbSparse attention [52], attention based on autocorrelation [48], max pooling [52] (called "self-attention distilling" in [52]), time series decomposition [53] and patched inputs [2, 38]. However, the evaluations in these publications do not generalize to electrical load forecasting, as they are often performed on datasets from different domains, focusing on long forecast horizons, and neglecting exogenous features which are important for electrical load forecasting [18]. While works on electrical load forecasting with Transformers show their potential [15, 21, 51], an evaluation of multiple Transformer architectures on electrical load datasets representing different aggregation levels is currently missing.

Recently, TSFMs were introduced, which are generalized time-series forecasting models trained on large amounts of time-series data from different domains, often including synthetic data, thereby being applicable to new time series in a zero-shot setting (i.e., without being trained on data from the particular time series). The TSFMs Moirai [47], Moirai-2 [36], TimeGPT [14], Lag-Llama [41], Chronos [3] and Chronos-2 [2] are based on Transformers. While time-series Transformers are predominantly used for TSFMs, other architectures also exist, including TabPFN-TS [27], which is based on a tabular Transformer model, and TiRex [4], which is based on an extended LSTM. We select Chronos-2 for the experiments, because it handles covariates well and outperforms other TSFMs on several electrical load forecasting datasets [7, 22, 32].

With the development of TSFMs, large benchmark datasets for the evaluation of forecasting models were introduced, but the ability to generalize the results to electrical load forecasting is limited. The Monash Time Series Forecasting Archive [16] contains 25 time series datasets, including two electrical load datasets, but it does not incorporate covariates. BuildingsBench [12] contains synthetic and real load time series from buildings, including calendar features but no weather data. Fev-bench [44] comprises 100 forecasting tasks, including electrical load data from ENTSO-E and from the Global Energy Forecasting Competitions (GEFCOMs) [26]. Shchur et al. [44] compare multiple TSFMs on fev-bench and find that Chronos-2 [2] performs best, but no comparisons are made to models trained from scratch. GIFT-Eval [1] contains only one electrical load dataset, which does not incorporate covariates. Therefore, the present paper evaluates Chronos-2 and models trained from scratch on a variety of electrical load datasets representing different aggregation levels.

Complementary work evaluates the usage of TSFMs for peak load forecasting in the LV grid [29]. Their work focuses on probabilistic forecasting and specialized metrics for peak forecasting, whereas our work focuses on deterministic forecasting, experiments across grid levels, and evaluations of overall forecast accuracy.

3 Methodology

This section introduces a flexible Transformer architecture, the comparison methods, and the hyperparameter optimization.²

3.1 YAformer: a flexible Transformer architecture

We introduce yet another Transformer (YAformer), a flexible architecture that integrates architectural modifications to the standard

¹A SCOPUS search for publications with "load forecast" in the title returned 1,398 documents published in the year 2025 alone.

²Code is available on GitHub: <https://github.com/KIT-IAI/load-forecasting-benchmark>

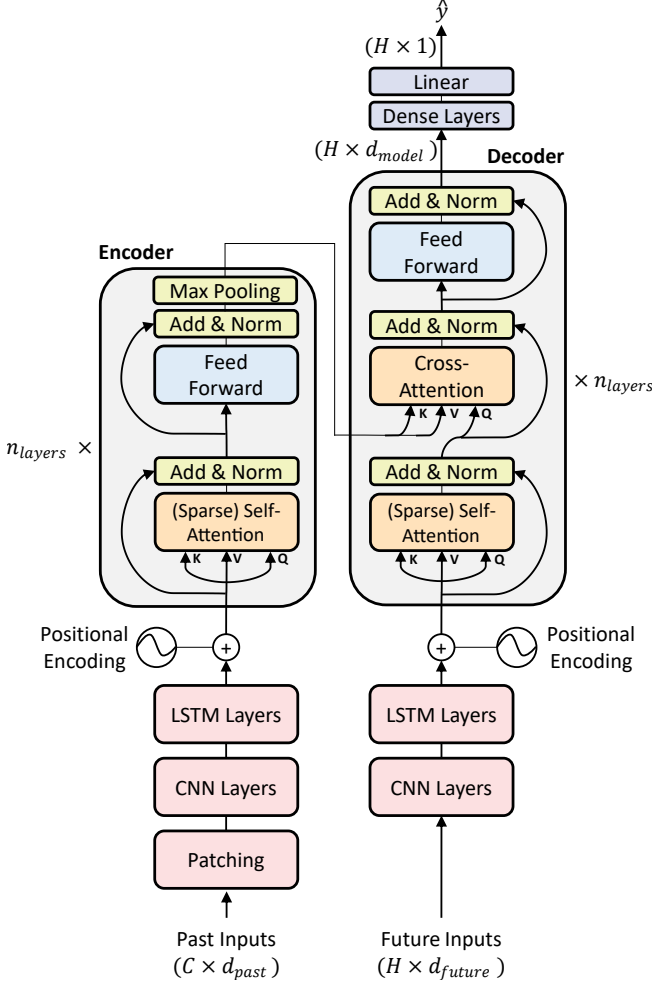


Fig. 1. Overview of the YAformer architecture based on an encoder-decoder Transformer. The inputs go through patching, CNN and LSTM layers. Sparse self-attention and max pooling can be optionally added in the encoder and decoder layers. The last linear layer produces a point forecast for the next H time steps. Figure based on [46].

Transformer proposed in prior work. Its design is controlled by a set of hyperparameters, allowing to represent a variety of time-series Transformer variants. This allows us to systematically compare an optimized YAformer model with the standard Transformer, thereby quantifying the impact of architectural modifications on forecasting accuracy. An overview of the model architecture is given in Figure 1. The model gets the past load, past covariates and future covariates as input. The basis of YAformer is the standard encoder-decoder Transformer [46] adapted for time series. For context length C , the encoder gets C vectors as input, each representing the load and covariates (weather data and calendar information) of one past time step. These vectors have one entry for the load and one for each covariate, so their size d_{past} equals the number of covariates plus one. The vectors go through a linear layer to embed them in the hidden dimension d_{model} . A stack of n_{layers} Transformer encoder

layers creates context-aware embeddings, using self-attention with keys K , values V and queries Q to retrieve information from the embedded input series. The encoder output has dimension $C \times d_{\text{model}}$. The decoder gets H vectors as input for a forecast horizon H , each representing the covariates of one future time step, so their size d_{future} equals the number of covariates. Like in the encoder, these vectors are embedded to dimension d_{model} via a linear layer. A stack of n_{layers} Transformer decoder layers create context-aware embeddings, using self-attention to retrieve information from the output of the previous layer and cross-attention to retrieve information about the past time steps from the encoder output. The decoder output has dimension $H \times d_{\text{model}}$ and goes through a stack of dense layers with rectified linear unit (ReLU) activation and a final linear output layer with one unit to create a point forecast of dimension $H \times 1$, containing the forecast values for the next H time steps.

The following architectural modifications can be added via hyperparameters:

- **Patching:** Neighboring past input vectors are concatenated before going through the linear embedding layer. The patch size p determines the number of vectors that are concatenated. Since p vectors get concatenated, the encoder input is transformed from dimension $C \times d_{\text{past}}$ to $\frac{C}{p} \times (d_{\text{past}} \cdot p)$.
- **CNN layers:** To encode local information, the linear embedding layer is replaced by a stack of convolutional layers with d_{model} filters. The number of convolutional layers is controlled by a hyperparameter n_{conv} and the kernel width by a hyperparameter w_{conv} .
- **LSTM layers:** Before the Transformer layers in the encoder and decoder, a stack of LSTM layers processes the embedded vectors sequentially.
- **Max pooling:** After each Transformer encoder layer, max pooling is used to reduce the length of the series. The pooling width w_{max} is controlled by a hyperparameter.
- **Sparse attention:** Sparse attention restricts the self-attention mechanisms in the encoder and decoder to use information only from the vectors of previous vectors with a logarithmic distance to the embedded position (the vector itself and 1, 2, 4, 8, ... steps before). A hyperparameter is used to switch between full attention and LogSparse attention [33].

3.2 Comparison methods

The following methods are selected for comparison, comprising a number of established methods [18, 26, 43] as well as recently developed Transformer models [35, 46] and the TSFM Chronos-2 [2].

Persistence baseline. The baseline predicts the load value from the same hour on the last observed day that has the same type as the predicted day, distinguishing between three types: workday, Saturday, and Sunday/holiday.

Linear model. Like the models from Zeng et al. [50], the linear model gets a feature vector as input and predicts the next H values. The model consists of a single linear layer with $n_{\text{features}} \times H$ weights. It is a global model, meaning that the weights are shared for all time series. The feature vector is the concatenation of C past vectors,

containing the load and covariates for the previous time steps, and H future vectors, only containing the covariates for the future time steps.

Light Gradient Boosting Machine (LightGBM). LightGBM [30] is a machine learning method that builds an ensemble of decision trees using gradient boosting. It is designed to train efficiently and handle large datasets with high performance. We train H individual models that predict the H future values based on past observations and future covariates.

Multilayer perceptron (MLP). The MLP gets a flat vector as input, which is the concatenation of the C past and H future vectors. Multiple fully-connected layers with ReLU activation are stacked on top of each other. The final layer is a linear output layer with H units returning the forecast values.

Convolutional neural network (CNN). The CNN gets a sequence of $C + H$ vectors as input. The load value in the future vectors is set to zero as it is not known at prediction time. Multiple convolutional layers with a stride of one are stacked on top of each other. The number of filters and the filter width are controlled via hyperparameters. The output of the last convolutional layer is flattened and fed into a stack of fully-connected layers with ReLU activation, where the last layer has H units and linear activation to create the forecast values.

Long short-term memory (LSTM). A hyperparameter for the LSTM allows to switch between an encoder-decoder model and an encoder-only model. The inputs to the encoder and decoder are the same as with YAformer. The vectors get embedded to dimension d_{model} by linear layers. The encoder and decoder are bidirectional LSTMs. The last hidden vector of the encoder is fed to the decoder as its first hidden vector. The decoder output for each future time step goes through a stack of linear layers and a final output layer with one unit to create the H forecast values. In the encoder-only variant, all input vectors are fed to the encoder LSTM stack, whose output is flattened and goes through a stack of linear layers and a final output layer with H units to create the H forecast values.

N-HITS. N-HITS [6] is a stack of MLPs, each of which returns a time series with a different time resolution. The forecast is finally computed as the sum of the returned (interpolated) time series.

Transformer. We use a standard encoder-decoder Transformer [46]. It gets the same inputs as YAformer, but does not make use of the architectural modifications introduced for YAformer.

Temporal Fusion Transformer (TFT). TFT [35] is an encoder-decoder model, receiving the same inputs as YAformer. The first layer of the encoder and decoder is a variable selection layer. It is followed by unidirectional LSTM layers and a single Transformer layer. Causal masking is used in the attention mechanism to prevent the Transformer layer from using information of time steps after the prediction time step.

Chronos-2. Chronos-2 [2] is a TSFM based on a Transformer with patched inputs. It is pretrained on a large dataset containing real time series data from various domains, as well as synthetic data

with covariate dependencies. It is the best open model on the GIFT-Eval benchmark, particularly outperforming the TSFMs Moirai [47], TimesFM [9] and TabPFN-TS [27].

3.3 Hyperparameter optimization (HPO)

The architecture modifications of YAformer can be controlled via model hyperparameters. Note that the standard Transformer is a special case of YAformer with patch size 1, zero CNN and LSTM layers, pooling width 1 and full attention. The hyperparameters of all models (YAformer and the comparison models) are optimized using Bayesian Optimization in weights and biases [5] with 100 hyperparameter configurations. All deep learning models are trained with AdamW [37] and the mean squared error loss function, one validation per epoch on TransnetBW and ten validations per epoch on the other datasets, using early stopping with a patience of ten validations without improvement. The best hyperparameters found with the HPO are given in Appendix A.

4 Experimental Setup

This section describes the three datasets, the task definition and metrics, and the rolling evaluation scheme used in the experiments.

4.1 Datasets

The proposed benchmark contains three datasets from different load aggregation levels, thereby representing use cases in the EHv and HV grid, the low-voltage grid and for end consumers. These datasets are described next. An overview of the datasets is given in Table 1.³

TransnetBW. TransnetBW is the TSO of the German state Baden-Württemberg. We download the load values for the TSO control area for the years 2015–2025 from the ENTSO-E transparency platform [13]. We transform the data into hourly resolution by taking the mean of four quarter-hourly power values. The dataset is enriched with weather data for Baden-Württemberg from the ERA5 reanalysis model, downloaded from Copernicus [8]. We use the air temperature, irradiance, precipitation, wind speed at 10 meters height and a binary holiday feature as covariates. The year 2023 is used as validation set for the HPO and the years 2024 and 2025 are used as test set.

FeederBW. The FeederBW dataset [45] comprises the electrical load from 200 low-voltage feeders in Baden-Württemberg in minutely resolution, together with short-term weather forecasts associated with the post code of each feeder. The length of the time series is two years, ranging from 1.4.2023 to 31.3.2025. We transform the load data into hourly resolution and take March 2023 as validation set and the last twelve months (1.4.2024–31.3.2025) as test set. We use irradiance, air temperature, precipitation, and a binary holiday feature as covariates.

³All three datasets come from public sources and our preprocessed versions are made available to foster future comparisons to our results: <https://github.com/KIT-IAI/load-forecasting-benchmark>

Table 1. Dataset characteristics.

Dataset	Aggregation	Time series	Period
TransnetBW	TSO control area	1	1.1.2015–31.12.2025
FeederBW	LV feeders	200	1.4.2023–31.3.2025
Electricity-287	Clients	287	1.1.2011–31.12.2014

Electricity-287. To represent the client level, we use public data from the UCI Electrical Load Diagrams⁴, first mentioned by Rodrigues and Trindade [42]. This dataset contains 370 load time series from individual clients of an energy trading company from Portugal. The data is in 15-minutes resolution, ranging from 1.1.2011–31.12.2014. A subset of the dataset is contained in the GIFT-Eval benchmark, there called the ‘Electricity’ dataset. We take a cleaned version of the original dataset, where time series with lots of missing data or imputation artifacts are removed [20], and reduce it further to time series that have no missing data in the validation or test period. The resulting dataset has 287 time series, which we transform into hourly resolution. The original dataset has no weather information, so we combine it with weather data for Portugal from the ERA5 reanalysis model [8]. To the best of our knowledge, we are the first to combine the Electricity dataset with weather data and study the effect of weather covariates on the forecast accuracy. The same test split as in GIFT-Eval [1] and Chronos-2 [2] is used to allow for a fair comparison, meaning that the last 10 % (8.8.2014 – 31.12.2014) are used as test data. We use the same portion of data (15.3.2014 – 7.8.2014) as validation set.

4.2 Task definition and metrics

The task addressed in this work is to forecast the hourly loads for the next four days (96 hours) based on past observations and past and future covariates. We see this as representative for many applications, which often require planning more than one day ahead. Forecasts are generated with a stride of one hour, i.e., at every hour, the next 96 values are predicted. For the datasets with multiple time series (FeederBW and Electricity-287), individual forecasts for every time series are made. The deep learning models are trained with a global training strategy, which was found to be superior to local and multivariate models [17, 21, 38], whereas LightGBM is trained as local models.

In addition to the weather and holiday covariates, the models trained from scratch get covariates as input that are derived from the timestamp, namely the hour of day, day of week and month (all cyclically encoded as two features).

We use the mean absolute error (MAE) to evaluate the forecasts for the TransnetBW and FeederBW datasets. To account for the different magnitudes of the time series in the Electricity-287 dataset, the normalized MAE (nMAE) is used instead of the MAE. The nMAE is the MAE divided by the mean of the time series’ absolute ground truth values in the test period. The metrics are computed per time series on the entire test set and the mean of all time series is reported as the final metric value. Results for additional metrics are reported in Appendix B.

⁴<https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014>

4.3 Retraining

For the datasets with at least one year test period (TransnetBW and FeederBW), new models are trained every month, using the last month as validation data and everything before as training data. This allows the models to adapt to potential concept drifts. For the Electricity-287 dataset, a static split into training (1.1.2011–14.3.2014), validation (15.3.2014–7.8.2014) and test data (8.8.2014–31.12.2014) is used.

4.4 Statistical Comparison

For statistical evaluation of the observed performance differences, we apply Diebold–Mariano tests (DM tests) [10] on the per-series loss differentials between competing forecasting models. For each dataset and model, the absolute error per sample is averaged over the forecasting horizon, yielding one aggregated loss value per series and forecast origin. For each pair of models, we compute the corresponding loss differentials. For each series, we separately perform two one-sided DM tests, resulting in two p -values corresponding to the hypotheses that either model 1 or model 2 achieves significantly lower predictive loss. A model is counted as significantly better for a given series if the corresponding one-sided DM test rejects the null hypothesis at the 5% significance level. We implement the DM test in Python following the implementation of the R forecast package [28].

5 Results

This section describes the forecasting results on the three datasets, as well as runtimes and ablation studies.

5.1 Model comparison

We present the model comparison on the three datasets in Figure 2. All models outperform the baseline on all three datasets. The improvement of the best model compared to the baseline is 59.6 % on TransnetBW, 42.3 % on FeederBW and 23.3 % on Electricity-287, indicating that it is easier to outperform a simple approach on highly aggregated data than on the more volatile LV feeder and client-level data.

The Transformer achieves the lowest forecast error on the TransnetBW and FeederBW datasets, whereas it is outperformed by Chronos-2 on the Electricity-287 dataset. Note that the training set of the Electricity-287 dataset is contained in the pretraining data of Chronos-2, so the evaluation on this dataset is not zero-shot.

On all three datasets, the best models are based on time-series Transformer architectures; namely, the two best models on TransnetBW (Transformer and YAformer) and the four best models on FeederBW and Electricity-287 (Transformer, Chronos-2, YAformer and TFT). The best Transformer model achieves 8.3 %, 6.6 %, and

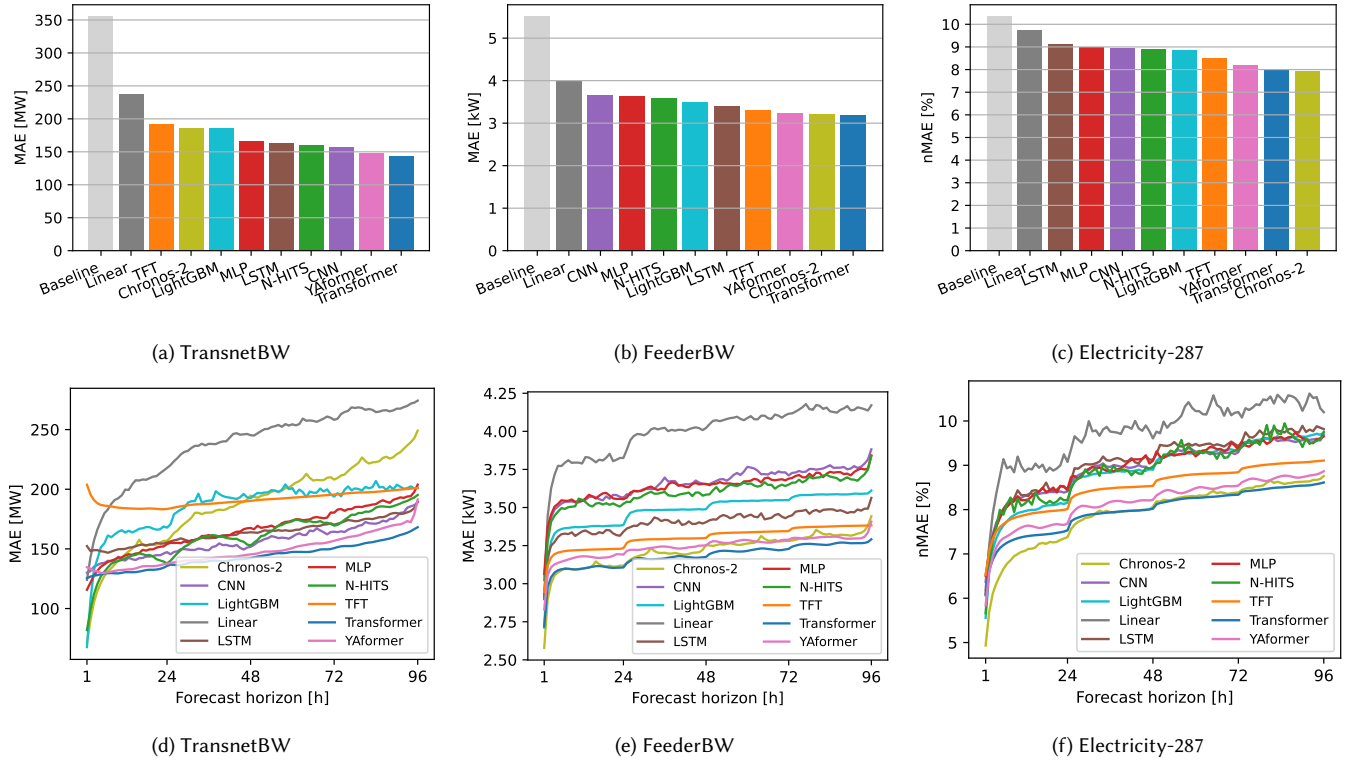


Fig. 2. Benchmarking results on the three datasets.

10.7 % lower forecast errors than the best non-Transformer model (CNN, LSTM and N-HITS) on TransnetBW, FeederBW and Electricity-287, respectively. The Transformer outperforms YAformer on all three datasets, indicating that the architectural modifications do not improve the forecast error compared to that of a standard Transformer.

The forecast performance of Chronos-2 varies greatly across the three datasets. Chronos-2 is the best model on the Electricity-287 dataset, and second-best on the FeederBW dataset, but has a much larger MAE than the best models on the TransnetBW dataset. A closer look at the model performance over the test period in Figure 3a reveals that Chronos-2 has large errors around holidays (e.g., 1.1.2024, 28.3.2024, 1.5.2024, 3.10.2024, the Christmas period, 17.4.2025, 3.10.2025). Figure 4 shows four forecasts where Chronos-2 exhibits large errors. The beginning of January is characterized by low loads due to winter holidays, and Chronos-2 fails to adjust the load to normal levels after the holiday period. Chronos-2 predicts a higher load on Good Friday (29.3.2024), potentially because the previous holiday (6.1.2024) did not affect the load because it was on a weekend. Ascension Day (9.5.2024) is predicted correctly, but Chronos-2 does not predict the long weekend. Before Christmas, many people take the full week off, and 24.12.2024 is not an official holiday, so Chronos-2 predicts a load that is too high. We evaluated Chronos-2 on TransnetBW with additional binary features for long weekends and school holidays, which reduces its MAE to 180 MW,

but it is still outperformed by the deep learning models trained from scratch.

On the FeederBW dataset (Figure 3b), the forecast errors are higher in winter, when the load is generally higher, with spikes in April and around Christmas. In this case, the Transformer outperforms Chronos-2, especially from September to November. On the Electricity-287 dataset (Figure 3c), Chronos-2 outperforms the Transformer consistently, except for the Christmas period, where both models exhibit large errors.

The forecast error for forecast horizons ranging from one hour to four days are shown in Figure 2d-f. LightGBM is the best model for one-step forecasts on the TransnetBW dataset. Chronos-2 is good for short forecast horizons, outperforming all other models for two to six hours horizon on TransnetBW, for the first six hours on FeederBW, and for the first 31 hours on Electricity-287. The error of the Transformer increases less with longer horizons, making it the best model for long horizons.

The results of the DM tests to compare the models per time series are given in Figure 5. For each pair of models, the number in the figure is the fraction of time series where one model is significantly better than the other model with $p < 0.05$. The TransnetBW dataset contains only one time series, so the fraction of time series can only be 100% or 0%. Transformer and YAformer are not significantly different, and they outperform all other models significantly. On the FeederBW dataset, the result is mixed, with the Transformer outperforming Chronos-2 on 46% of the time series, but vice versa on

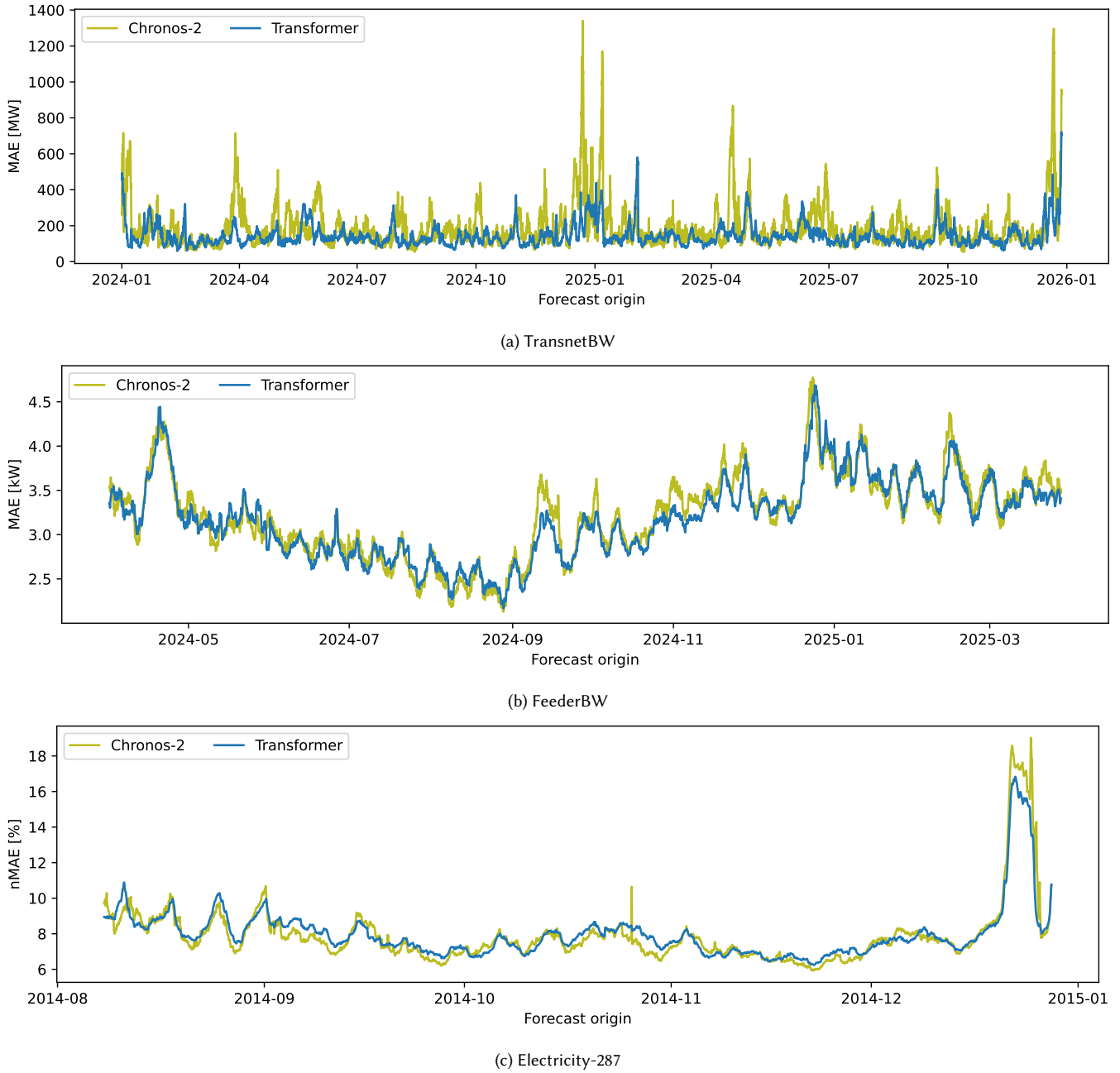


Fig. 3. Comparison of the forecast error of the Transformer and Chronos-2 over time.

20% of the time series (and no significant difference on the remaining time series). On the Electricity-287 dataset, Chronos-2 has the best nMAE, but it outperforms the Transformer only on 14% of the time series, whereas the result is vice versa on 39% of the time series (and no significant difference on the remaining time series). The reason is that Chronos-2 is particularly good on difficult time series with a high nMAE, but it is outperformed on many time series with lower nMAE.

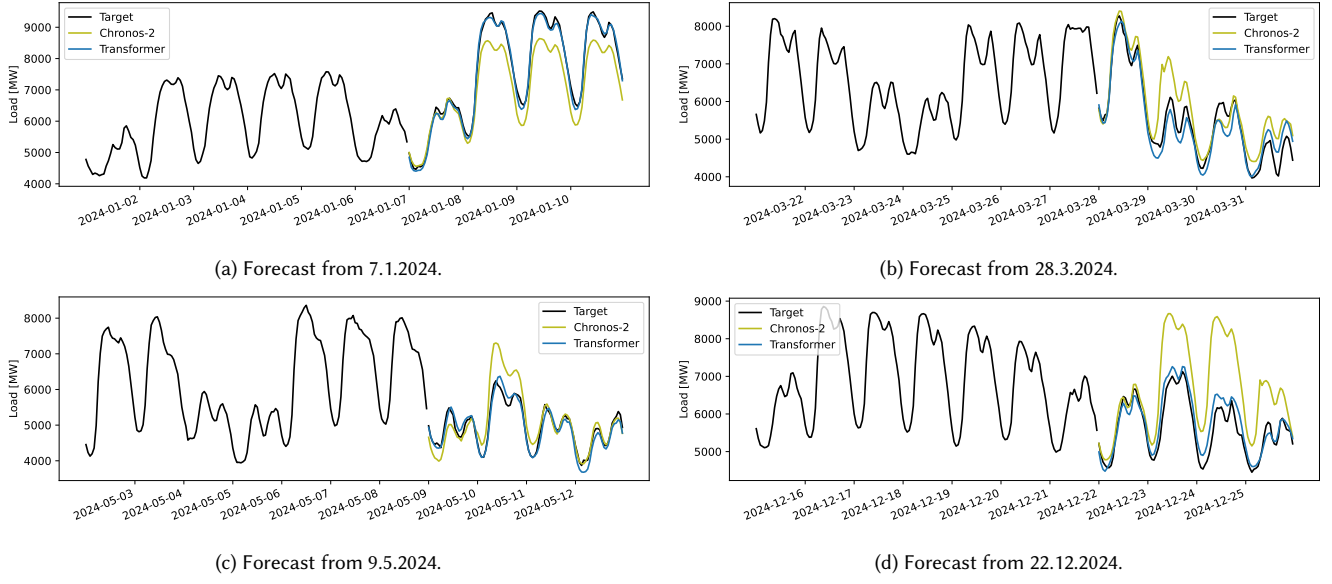


Fig. 4. Exemplary forecasts of the Transformer and Chronos-2 on the TransnetBW data, showing cases where Chronos-2 exhibits large errors.

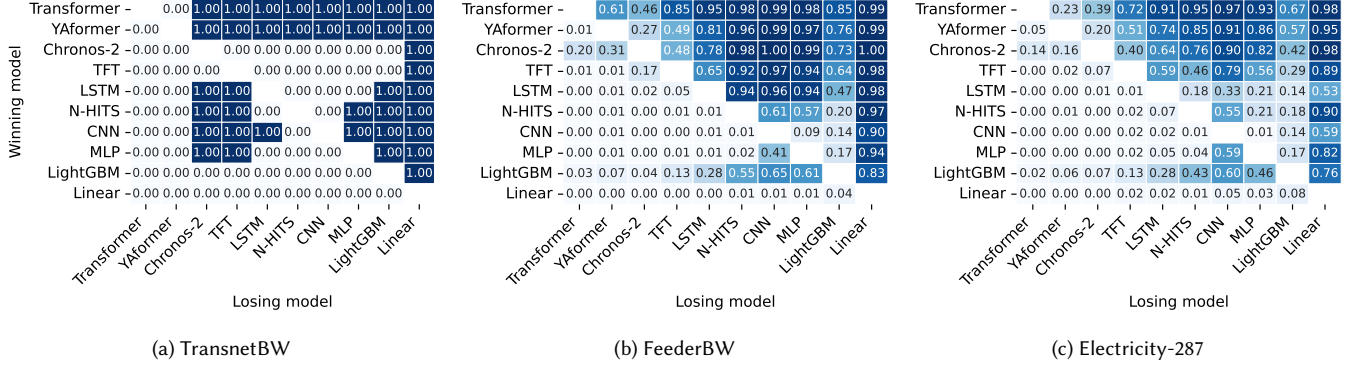


Fig. 5. Statistical test results on the three datasets. A Diebold-Mariano test is performed per time series for each pair of models. The fraction of time series on which the model in the column is better than the model in the row with $p < 0.05$ is given.

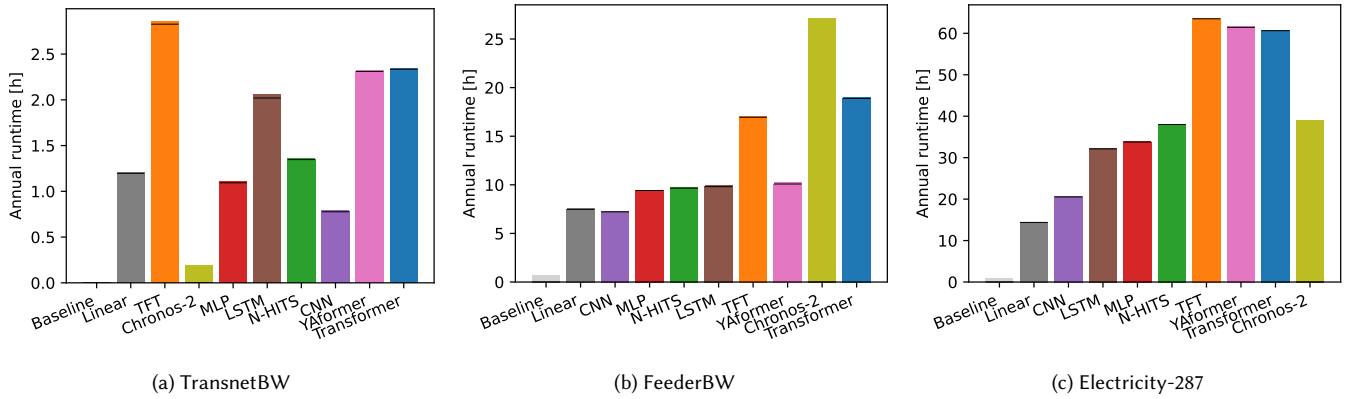


Fig. 6. Annual total runtimes for twelve monthly model trainings and 8760 inferences on the three datasets. The training time is shown as dashed bar.

Table 2. Runtimes for one model training, one inference (one forecast for all time series), and annual total runtime ($12 \cdot \text{training} + 8760 \cdot \text{inference}$).

Model	TransnetBW			FeederBW			Electricity-287		
	Training [h:mm:ss]	Inference [ms]	Annual [h:mm:ss]	Training [h:mm:ss]	Inference [ms]	Annual [hh:mm:ss]	Training [hh:mm:ss]	Inference [ms]	Annual [hh:mm:ss]
Baseline	-	1.4	0:00:12	-	266.25	0:38:52	-	373.4	0:54:30
Chronos-2	-	78.3	0:11:26	-	11,160.7	27:09:27	-	16,015.6	38:58:16
CNN	0:03:53	5.4	0:47:29	0:36:06	28.1	7:17:19	1:42:37	35.8	20:36:37
LightGBM	0:18:42	1.2	3:44:36	10:04:12	269.7	121:29:46	50:20:28	239.3	604:40:32
Linear	0:05:59	5.2	1:12:39	0:37:22	38.5	7:17:19	1:11:35	38.9	14:28:16
LSTM	0:10:07	15.3	2:03:37	0:49:03	74.6	9:59:32	2:40:34	55.2	32:14:51
MLP	0:05:29	5.2	1:06:29	0:47:05	38.5	9:30:42	2:48:51	29.2	33:50:27
N-HITS	0:06:45	5.9	1:21:49	0:48:18	39.2	9:45:20	3:09:58	41.5	38:05:39
TFT	0:14:08	10.8	2:51:12	1:24:45	46.1	17:03:44	5:17:30	75.2	63:40:58
Transformer	0:11:40	6.4	2:21:00	1:34:26	72.5	19:03:48	5:03:10	60.6	60:46:50
YAformer	0:11:33	6.3	2:19:32	0:50:25	82.6	10:17:04	5:07:09	95.2	61:39:42

5.2 Runtime

Training and inference runtimes for all models are given in Table 2 (evaluated on a single high-performance computing (HPC) node with an NVIDIA A100 GPU with 40 GB VRAM, 19 CPU cores of an Intel Xeon Platinum 8368 and 128 GB RAM). The training time is the average time needed to train a global model. The inference time is the average time needed to create a forecast for all time series in a dataset (one for TransnetBW, 200 for FeederBW, 287 for Electricity-287). The deep learning models trained by us are evaluated with a batch size equal to the number of time series, i.e. all time series are put into a single batch, whereas Chronos-2 is evaluated with a maximum batch size of 32, which resulted in the fastest throughput. We calculate the annual runtime in a scenario where the models are retrained every month and a forecast is made at every hour for the next 96 hours. This means that twelve models are trained and 8760 forecasts are made per year, so the annual runtime consists of twelve trainings and 8760 inference instances. This annual runtime is visualized in Figure 6, with the training part dashed. For the models trained by us, the annual runtime is dominated by the training time, which comprises almost all of the annual runtime. The inference time is fast, ranging between 5 ms and 95 ms per inference on the three datasets, so that it is barely visible in Figure 2. Chronos-2 is used as a pretrained model, so it has zero training time in our scenario. This makes it the fastest model overall on the TransnetBW dataset, where only one time series needs to be forecast. On the other datasets, which have 200 and 287 time series, the slower inference time of Chronos-2 outweighs the fact that it has zero training time.

5.3 Ablations

Figure 7 shows results of ablated models, where covariates are removed, static training instead of monthly retraining is performed, and the context length is altered.

Covariates. Covariates are important to achieve the lowest forecast errors. This holds for all three datasets, as the Transformer with all features is always better than the univariate Transformer.

The same holds for Chronos-2, which also benefits from covariates. The improvement by the integration of covariates is large on the TransnetBW and FeederBW datasets, which are largely influenced by behind-the-meter photovoltaic power generation. The effect is smaller, but also important, on the Electricity-287 dataset, where some of the load is also driven by weather effects. Apart from weather covariates, calendar features are important on the TransnetBW and Electricity-287 datasets, as the industrial activity is reduced during holidays, leading to a lower electrical load.

Continuous retraining. On TransnetBW and FeederBW, a rolling evaluation is performed on a monthly basis, meaning that new models are trained every month. The previous month is used as validation data, and all data before that as training data, so that models can adapt to concept drifts due to newly installed appliances or changed consumption patterns. We compare the retrained Transformer to a static model that is only trained once per year. The retrained Transformer outperforms the static counterpart on both datasets, signifying that retraining helps the model to adapt.

Context length. Figures 7d-f show the forecast error of Chronos-2 with one week, two weeks, one month context and the full context length of 8192 time steps. The forecast accuracy of Chronos-2 improves with longer contexts on all three datasets. With only one week context, Chronos-2 barely outperforms the baseline, but it quickly improves with longer contexts. Using the maximum context of 8192 time steps gives only a small improvement over using one month context.

6 Discussion

Transformer-based models are effective for load forecasting on all tested grid levels. On the three datasets, the improvement over the best non-Transformer model is considerable, with 6.6–10.7 % lower errors for the best Transformer compared to the best other method. This includes datasets with different characteristics, such as highly aggregated TSO load and more diverse and volatile LV feeder loads

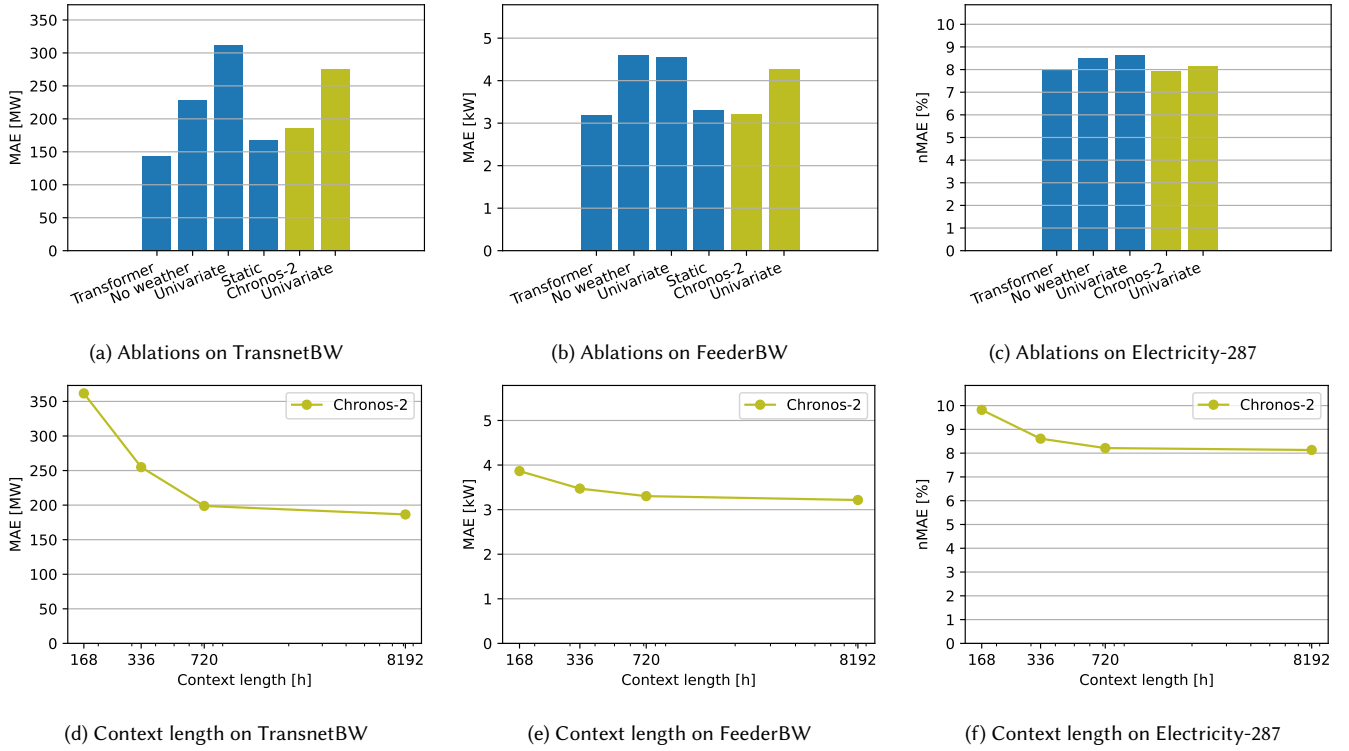


Fig. 7. Ablation results on test data for models without weather data (*no weather*), without covariates (*univariate*), and without retraining (*static*), as well as reduced context length for Chronos-2.

and client loads, where time series from multiple hundred measurement points are available. After hyperparameter optimization, Transformer models are effective for all these data types.

The architectural modifications to the Transformer – including patching, CNN layers, LSTM layers, max pooling and sparse attention – are not successful in reducing the forecast error. Instead, the standard Transformer outperforms the flexible YAfformer after hyperparameter optimization by a small margin. This finding is further strengthened by the fact that the standard Transformer also outperforms the established Transformer-based model TFT in our experiments. Although the standard Transformer is a subclass of YAfformer, the larger hyperparameter search space of YAfformer hinders the HPO in finding the optimal configuration. In fact, there is no guarantee that the Transformer configuration (or the configuration of any other model) after HPO is the global optimum, but we had to restrict the HPO to 100 evaluations per model for practical reasons.

Chronos-2 delivers accurate zero-shot forecasts for diverse time series, but it has particular weaknesses with rare events and long horizons, which lead to larger overall errors on the TSO data. While Chronos-2 is very accurate for short forecast horizons between a few hours up to one day, it is outperformed by models trained particularly for the datasets at hand for longer horizons. On the TransnetBW data, it exhibits large errors around holidays, where it needs to incorporate not only the lower industrial activity on the

holiday, but also indirect effects such as long weekends and school holidays. Future work can try to improve the results from Chronos-2 by designing additional features or by fine-tuning the model.

The advantage of Chronos-2 is that no dataset-specific training is required. Training a model from scratch takes between half an hour and five hours, depending on the dataset and model. This computational cost is saved with Chronos-2. On the other hand, Chronos-2 is a very large model, resulting in slower inference times. Inference times for our models are below 100 ms for creating one forecast for all time series in a dataset, whereas Chronos-2 needs up to 21 seconds. Therefore, Chronos-2 is useful when inference time plays a minor role, e.g. because few time series need to be forecasted or forecasts are only made once per day. For continuous forecasts during the day and for scenarios with many time series, dataset-specific models can be advantageous.

Long contexts, covariates and continuous retraining are important to achieve the lowest electrical load forecast errors. This means that univariate TSFMs will likely not succeed in achieving the lowest possible forecast errors. Comparisons on datasets without covariates, like the original UCI Electrical Load Diagrams, can indicate which models capture autocorrelated energy demand patterns, but are to be taken with caution because they do not show the full picture.

Deterministic forecasts do not cover the uncertainty of a forecast. This is especially relevant for volatile time series like FeederBW

and Electricity-287. On both datasets, it is difficult to make good point forecasts due to the volatile and unpredictable pattern of the load, so that even the best models sometimes exhibit large errors. In such cases, it can be beneficial to create probabilistic forecasts that cover the uncertainty in form of predicted distributions. Future work can extend our benchmarking to probabilistic forecasting models. Chronos-2 offers the capability to predict quantiles. The deep learning models from our work can be extended to probabilistic models by replacing the linear output layer with a probabilistic output layer and training the models with the pinball loss function.

We agree with scholars [18, 23] that benchmarking of electrical load forecasting methods on different public datasets, with comparable test splits and metrics, is necessary to establish the state of the art in electrical load forecasting. Therefore, we provide a comparison of a large set of methods on three datasets. We share our preprocessed datasets publicly, together with the predictions of all evaluated models, to allow for future comparisons to our results⁵.

The achieved forecast errors might be unrealistically low for real applications. For example, real-time measurements of past load values might become available only after a few hours or days, but are assumed to be available immediately in our experiments. In addition, we have used weather data that is not available at the forecast time in practice, such as reanalysis data or short-term weather forecasts. However, our primary goal is to compare models under fair conditions, which we achieve by using the same data for all.

7 Conclusion & Outlook

The present paper presents a benchmark for electrical load forecasting across grid levels, including three datasets representing the TSO level, the LV grid feeder level and the end consumer level. In a comparison of time-series Transformers and the TSFM Chronos-2 with established machine learning and deep learning methods for electrical load forecasting on the three datasets, Transformer-based models outperform the established methods across all grid levels, reducing the forecast error by 6.6–10.7 %. The standard time-series Transformer outperforms YAformer, a flexible Transformer with patching, CNN layers, LSTM layers and sparse attention. This indicates that the standard architecture is already a strong model, and architectural modifications are not needed and can even lead to worse accuracy as it makes hyperparameter optimization (HPO) more difficult due to an extended hyperparameter search space. The TSFM Chronos-2 gives competitive forecasts on the feeder and client levels, being particularly accurate for short forecast horizons, but it is outperformed by specialized models on the TSO level and it is comparably slow in inference.

All our datasets and results are made publicly available to enable benchmarking of new approaches against our results, as well as to allow for an extension to probabilistic models in the future. We plan to integrate the best approaches into the Energy Arena⁶ [31] for continuous performance comparisons under realistic operational conditions. Ensembling of forecast models could improve the results beyond single-model forecasts.

⁵The datasets and results are available on GitHub: <https://github.com/KIT-IAI/load-forecasting-benchmark>

⁶<https://energy-arena.org>

Acknowledgments

The authors thank Alexander Kreusel for assistance with Chronos-2.

The authors gratefully acknowledge funding by the Helmholtz Association under the program “Energy System Design” and the Helmholtz Association’s Initiative and Networking Fund through Helmholtz AI and the HAICORE@KIT partition. The authors gratefully acknowledge the computing time provided on HoreKa by the NHR@KIT center. This center is jointly supported by the Federal Ministry of Education and Research and the Ministry of Science, Research and the Arts of Baden-Württemberg, as part of the National High-Performance Computing (NHR) joint funding program. HoreKa is partly funded by the German Research Foundation (DFG).

ChatGPT was used to assist in parts of the writing and coding. All outputs were edited carefully and the authors take the full responsibility for the published article.

References

- [1] Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. 2024. GIFT-Eval: A Benchmark For General Time Series Forecasting Model Evaluation. doi:10.48550/arXiv.2410.10393
- [2] Abdul Fatir Ansari, Oleksandr Shchur, Jari Kükken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Gueron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. 2025. Chronos-2: From Univariate to Universal Forecasting. doi:10.48550/ARXIV.2510.15821
- [3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. 2024. Chronos: Learning the Language of Time Series. doi:10.48550/ARXIV.2403.07815
- [4] Andreas Auer, Patrick Podest, Daniel Klotz, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. 2025. TiRex: Zero-Shot Forecasting Across Long and Short Horizons with Enhanced In-Context Learning. doi:10.48550/arXiv.2505.23719
- [5] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. wandb.com
- [6] Cristian Chailu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. 2023. NHITS: Neural Hierarchical Interpolation for Time Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 6 (June 2023), 6989–6997. doi:10.1609/aaai.v37i6.25854
- [7] Wei Soon Cheong, Lian Lian Jiang, and Jamie Ng Suat Ling. 2026. Assessing Electricity Demand Forecasting with Exogenous Data in Time Series Foundation Models. doi:10.48550/ARXIV.2602.05390
- [8] Copernicus Climate Change Service. 2020. Climate and energy indicators for Europe from 1979 to present derived from reanalysis. doi:10.24381/CDS.4BD77450
- [9] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. 2024. A decoder-only foundation model for time-series forecasting. doi:10.48550/arXiv.2310.10688
- [10] Francis X. Diebold and Roberto S. Mariano. 1995. Comparing Predictive Accuracy. *Journal of Business & Economic Statistics* 13, 3 (1995), 253–263. jstor:1392185 doi:10.2307/1392185
- [11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiahua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=YicbFdNTTy>
- [12] Patrick Emami, Abhijeet Sahu, and Peter Graf. 2023. BuildingsBench: A Large-Scale Dataset of 900K Buildings and Benchmark for Short-Term Load Forecasting. In *37th Conference on Neural Information Processing Systems (NeurIPS 2023) Track on Datasets and Benchmarks*. doi:10.48550/arXiv.2307.00142
- [13] ENTSO-E. 2026. Transparency platform. <https://transparency.entsoe.eu>
- [14] Azul Garza, Cristian Chailu, and Max Mergenthaler-Canseco. 2023. TimeGPT-1. doi:10.48550/ARXIV.2310.03589
- [15] Elena Giacomazzi, Felix Haag, and Konstantin Hopf. 2023. 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*. ACM, Orlando FL USA, 353–360. doi:10.1145/3575813.3597345

- [16] Rakshitha Wathsadini Godaheva, Christoph Bergmeir, Geoffrey I. Webb, Rob Hyndman, and Pablo Montero-Manso. 2021. Monash Time Series Forecasting Archive. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. <https://openreview.net/forum?id=wEc1mgAjU->
- [17] Miha Grabner, Yi Wang, Qingsong Wen, Boštjan Blažič, and Vitomir Štruc. 2023. A Global Modeling Framework for Load Forecasting in Distribution Networks. *IEEE Transactions on Smart Grid* 14, 6 (Nov. 2023), 4927–4941. doi:10.1109/TSG.2023.3264525
- [18] Stephen Haben, Siddharth Arora, Georgios Giasemidis, Marcus Voss, and Danica Vukadinović Greetham. 2021. Review of low voltage load forecasting: Methods, applications, and recommendations. *Applied Energy* 304 (Dec. 2021), 117798. doi:10.1016/j.apenergy.2021.117798
- [19] Stephen Haben, Marcus Voss, and William Holderbaum. 2023. *Core Concepts and Methods in Load Forecasting: With Applications in Distribution Networks*. Springer International Publishing, Cham. doi:10.1007/978-3-031-27852-5
- [20] Matthias Hertel, Lara Ambrosius, Manuel Treutlein, Ralf Mikut, and Veit Hagenmeyer. 2025. A comparison of local, cluster-specific and global Transformer models for forecasting electrical loads of individual buildings and substations. In *2025 IEEE Kiel PowerTech*. IEEE, Kiel, Germany, 1–8. doi:10.1109/PowerTech59965.2025.11180482
- [21] Matthias Hertel, Maximilian Beichter, Benedikt Heidrich, Oliver Neumann, Benjamin Schäfer, Ralf Mikut, and Veit Hagenmeyer. 2023. Transformer training strategies for forecasting multiple load time series. *Energy Informatics* 6, 1 (Oct. 2023), 20. doi:10.1186/s42162-023-00278-z
- [22] Matthias Hertel, Alexandra Nikoltchovska, Sebastian Pütz, Benjamin Schäfer, Ralf Mikut, and Veit Hagenmeyer. 2026. Explainable Load Forecasting with Covariate-Informed Time Series Foundation Models. In *Proceedings of the 17th ACM International Conference on Future and Sustainable Energy Systems*. ACM, Banff, Alberta, Canada, 612–626. doi:10.1145/3744255.3811724
- [23] Hansika Hewamalage, Klaus Ackermann, and Christoph Bergmeir. 2023. Forecast evaluation for data scientists: common pitfalls and best practices. *Data Mining and Knowledge Discovery* 37, 2 (March 2023), 788–832. doi:10.1007/s10618-022-00894-5
- [24] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. 2023. TabPFN: A Transformer That Solves Small Tabular Classification Problems in a Second. doi:10.48550/arXiv.2207.01848
- [25] Tao Hong. 2014. Energy Forecasting: Past, Present, and Future. *Foresight: The International Journal of Applied Forecasting* 32 (2014), 43–48.
- [26] Tao Hong, Pierre Pinson, Yi Wang, Rafal Weron, Dazhi Yang, and Hamidreza Zareipour. 2020. Energy Forecasting: A Review and Outlook. *IEEE Open Access Journal of Power and Energy* 7 (2020), 376–388. doi:10.1109/OAJPE.2020.3029979
- [27] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. 2025. From Tables to Time: How TabPFN-v2 Outperforms Specialized Time Series Forecasting Models. doi:10.48550/arXiv.2501.02945
- [28] Rob J Hyndman and Yeasmin Khandakar. 2008. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software* 27, 3 (2008), 1–22. doi:10.18637/jss.v027.i03
- [29] Benedikt Kaas, Manuel Treutlein, Hannes Benedikt Gerber, Oliver Neumann, Cheewan Phathanakhuha, Oliver Resch, Ralf Mikut, and Veit Hagenmeyer. 2026. Probabilistic Low-Voltage Peak Load Forecasting with Time Series Foundation Models Evaluated on Application-Oriented Metrics. doi:10.48550/ARXIV.2607.01966
- [30] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. LightGBM: a highly efficient gradient boosting decision tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS’17)*. Curran Associates Inc., Red Hook, NY, USA, 3149–3157. <https://dl.acm.org/doi/10.5555/3294996.3295074>
- [31] Max Kleinebrahm, Jonathan Berrisch, Philipp Eiser, Wolf Fichtner, Veit Hagenmeyer, Matthias Hertel, Nils Koster, Sebastian Lerch, Ralf Mikut, Jan Priesmann, Melanie Schienle, Benjamin Schaefer, Jann Weinand, and Florian Ziel. 2026. Energy-Arena: A Dynamic Benchmark for Operational Energy Forecasting. doi:10.48550/arXiv.2604.24705
- [32] Alexander Kreusel, Matthias Hertel, Moritz Noskiewicz, Frederik Zahn, Heiko Maaß, Ralf Mikut, and Veit Hagenmeyer. 2026. Covariates Are the Key to Accurate Probabilistic Building Energy Forecasting with Time Series Foundation Models. In *Proceedings of the 2026 ACM Sustainability Week*. ACM, Banff, Alberta, Canada, 172–182. doi:10.1145/3765611.3815507
- [33] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyu Zhou, Wenhui Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc. doi:10.5555/3454287.3454758
- [34] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. 2024. Foundation Models for Time Series Analysis: A Tutorial and Survey. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6555–6565. doi:10.1145/3637528.3671451
- [35] Bryan Lim, Sercan O. Arik, Nicolas Loeff, and Tomas Pfister. 2021. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* 37, 4 (Oct. 2021), 1748–1764. doi:10.1016/j.ijforecast.2021.03.012
- [36] Chenghao Liu, Taha Aksu, Juncheng Liu, Xu Liu, Hanshu Yan, Quang Pham, Silvio Savarese, Doyen Sahoo, Caiming Xiong, and Junnan Li. 2026. Moirai 2.0: When Less Is More for Time Series Forecasting. doi:10.48550/arXiv.2511.11698
- [37] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization. doi:10.48550/arXiv.1711.05101
- [38] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=Jbdc0vTOcol>
- [39] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI* (2019). https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- [40] Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nicholas R. Jennings. 2012. Putting the ‘smarts’ into the smart grid: a grand challenge for artificial intelligence. *Commun. ACM* 55, 4 (April 2012), 86–97. doi:10.1145/2133806.2133825
- [41] Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. 2023. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. doi:10.48550/ARXIV.2310.08278
- [42] Fátima Rodrigues and Artur Trindade. 2018. Load forecasting through functional clustering and ensemble learning. *Knowledge and Information Systems* 57, 1 (Oct. 2018), 229–244. doi:10.1007/s10115-018-1169-y
- [43] Frederik vom Scheidt, Hana Medinová, Nicole Ludwig, Bent Richter, Philipp Staudt, and Christof Weinhardt. 2020. Data analytics in the electricity sector – A quantitative and qualitative literature review. *Energy and AI* 1 (Aug. 2020), 100009. doi:10.1016/j.egyai.2020.100009
- [44] Aleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Gueron, Michael Bohlke-Schneider, and Yuyang Wang. 2025. fev-bench: A Realistic Benchmark for Time Series Forecasting. doi:10.48550/ARXIV.2509.26468
- [45] Manuel Treutlein, Pascal Bothe, Marc Schmidt, Roman Hahn, Oliver Neumann, Ralf Mikut, and Veit Hagenmeyer. 2026. Real-world energy data of 200 feeders from low-voltage grids with metadata in Germany over two years. doi:10.48550/arXiv.2602.03521
- [46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, Vol. 30. Curran Associates, Inc., 5998–6008. <https://dl.acm.org/doi/10.5555/3295222.3295349>
- [47] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. 2024. Unified Training of Universal Time Series Forecasting Transformers. doi:10.48550/arXiv.2402.02592
- [48] Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting. In *Advances in Neural Information Processing Systems*, Vol. 34. Curran Associates, Inc., 22419–22430. <https://dl.acm.org/doi/10.5555/3540261.3541978>
- [49] Neo Wu, Bradley Green, Xue Ben, and Shawn O’Banion. 2020. Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case. doi:10.48550/arXiv.2001.08317
- [50] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are Transformers Effective for Time Series Forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 9 (June 2023), 11121–11128. doi:10.1609/aaai.v37i9.26317
- [51] Guangqi Zhang, Chuyuan Wei, Changfeng Jing, and Yanxue Wang. 2022. Short-Term Electrical Load Forecasting Based on Time Augmented Transformer. *International Journal of Computational Intelligence Systems* 15, 1 (Aug. 2022), 67. doi:10.1007/s44196-022-00128-y
- [52] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 12 (May 2021), 11106–11115. doi:10.1609/aaai.v35i12.17325
- [53] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 27268–27286. <https://proceedings.mlr.press/v162/zhoul22g.html>
- [54] Hüseyin K. Çakmak and Veit Hagenmeyer. 2022. Using Open Data for Modeling and Simulation of the All Electrical Society in eASiMOV. In *2022 Open Source Modelling and Simulation of Energy Systems (OSMES)*. 1–6. doi:10.1109/OSMES54027.2022.9769145

Table 3. Number of trainable parameters per model on the three datasets.

Model	TransnetBW	FeederBW	Electricity-287
CNN	8,263,904	7,710,304	3,692,768
Linear	940,128	456,288	940,128
LSTM	13,855,456	117,325,408	117,327,456
MLP	2,966,624	14,127,200	6,457,440
N-HITS	7,308,894	48,654,500	24,865,431
TFT	195,828	696,810	741,748
Transformer	1,078,657	3,304,961	279,361
YAformer	4,350,849	2,823,297	8,608,769

Table 4. Test results on the three datasets evaluated with the mean absolute error (MAE), the root mean square error (RMSE), the normalized MAE (nMAE) and the normalized RMSE (nRMSE). The best result is highlighted in bold.

Model	TransnetBW				FeederBW				Electricity-287			
	MAE [MW]	RMSE [MW]	nMAE [%]	nRMSE [%]	MAE [kW]	RMSE [kW]	nMAE [%]	nRMSE [%]	MAE [kW]	RMSE [kW]	nMAE [%]	nRMSE [%]
Baseline	355.3	487.1	5.49	7.53	5.51	9.24	33.5	51.8	56.1	566.5	10.35	16.79
Chronos-2	186.5	280.0	2.88	4.33	3.22	4.91	19.4	28.4	42.2	407.0	7.92	12.95
CNN	156.5	211.1	2.42	3.26	3.66	5.41	22.2	31.3	47.3	430.2	8.97	13.34
LightGBM	186.0	259.8	2.87	4.01	3.49	5.31	21.1	30.6	43.6	377.3	8.86	13.08
Linear	237.7	322.2	3.67	4.98	3.99	6.00	24.2	34.4	54.1	508.7	9.76	14.08
LSTM	163.8	217.5	2.53	3.36	3.41	5.05	20.6	29.2	48.2	453.1	9.10	13.77
MLP	166.0	222.4	2.57	3.44	3.64	5.40	22.0	31.1	47.9	438.3	8.98	13.25
NHITS	160.5	215.7	2.48	3.33	3.60	5.32	21.8	30.7	47.5	445.3	8.92	13.33
TFT	191.3	259.5	2.96	4.01	3.30	5.01	20.0	28.9	44.1	413.3	8.51	13.47
Transformer	143.6	191.7	2.22	2.96	3.18	4.82	19.2	27.8	40.6	378.9	7.99	12.56
YAformer	148.4	199.8	2.29	3.09	3.24	4.90	19.6	28.4	42.7	415.1	8.18	12.77

A Hyperparameters

The hyperparameters of the best configurations found with the HPO are given in Table 5 and the resulting model sizes in Table 3.

B Additional metrics

The mean absolute error (MAE), root mean square error (RMSE), normalized MAE (nMAE) and normalized RMSE (nRMSE) results on the test sets of the three datasets are given in Table 4.

Received 24 April 2026

Table 5. Model hyperparameter search range and the best configurations found with a Bayesian Optimization HPO for each dataset.

Hyperparameter	Search range	TransnetBW	FeederBW	Electricity-287
<u>CNN</u>				
Context length C	168, 336, 720	336	168	336
Patch size	1, 4, 8, 24	8	24	24
Kernel width	3, 5, 7	3	7	5
Convolutional layers	1, 2, 3	1	1	1
Max pooling width	1, 2, 4, 8	1	1	1
Dense layers	1, 2, 3	2	1	2
Dense units	128, 256, 512, 1024	1024	512	1024
Batch size	128, 256, 512	256	128	512
Learning rate	(0.0001, 0.001)	0.00079	0.00016	0.00014
Dropout	(0.0, 0.2)	0.00	0.09	0.12
<u>LightGBM</u>				
Context length C	168	168	168	168
Learning rate	(0.01, 0.3)	0.153	0.0492	0.0519
Minimum child samples	(1, 200)	32	58	7
Number of estimators	(1, 250)	248	232	211
<u>Linear</u>				
Context length C	168, 336, 720	720	336	720
Batch size	128, 256, 512	512	512	512
Learning rate	(0.0001, 0.001)	0.00012	0.00010	0.00017
Dropout	(0.0, 0.2)	0.15	0.08	0.05
<u>LSTM</u>				
Context length C	168, 336, 720	720	720	720
Architecture	encoder, encoder-decoder	encoder	encoder	encoder
Patch size	1, 4, 8, 24	4	4	4
LSTM layers	1, 2, 3	2	3	3
d_{model}	32, 64, ..., 1024	128	512	512
Dense units	128, 256, 512, 1024	512	1024	1024
Dense layers	1, 2, 3	1	1	1
Batch size	128, 256, 512	256	128	128
Learning rate	(0.0001, 0.001)	0.00080	0.00037	0.00012
Dropout	(0.0, 0.2)	0.19	0.04	0.08
<u>MLP</u>				
Context length C	168, 336, 720	336	336	336
Layers	1, 2, 3	2	2	2
Units	128, 256, 512, 1024, 2048	512	2048	1024
Batch size	128, 256, 512	512	256	512
Learning rate	(0.0001, 0.001)	0.00019	0.00018	0.00012
Dropout	(0.0, 0.2)	0.00	0.12	0.09
<u>N-HITS</u>				
Context length C	168, 336, 720	168	336	720
Blocks	(1,1,1), (2,2,2), (3,3,3)	(1,1,1)	(2,2,2)	(3,3,3)
Kernel size	(1,1,1), (2,2,2), (4,4,4), (2,2,1), (8,4,1), (16,8,1)	(2,2,1)	(8,4,1)	(16,8,1)
Downsample	(1,1,1), (24,12,1), (40,20,1), (60,8,1), (168,24,1), (180,60,1)	(40,20,1)	(180,60,1)	(60,8,1)
Dense units	32, 64, 128, ..., 1024	512	1024	256
Batch size	128, 256, 512	256	512	128
Learning rate	(0.0001, 0.001)	0.00025	0.00014	0.00010
<u>TFT</u>				
Context length C	168, 336, 720	720	336	720
d_{model}	32, 64, 128, 256, 512	32	64	64
n_{heads}	1, 2, 4, 8	1	4	4
Batch size	128, 256, 512	128	128	128
Learning rate	(0.0001, 0.001)	0.00049	0.00010	0.00035
Dropout	(0.0, 0.2)	0.04	0.03	0.02
<u>Transformer</u>				
Context length C	168, 336, 720	720	720	720
n_{layers}	1, 2, 3, 4	1	3	4
d_{model}	32, 64, 128, 256, 512	256	256	64
n_{heads}	1, 2, 4, 8	1	4	4
Dense layers	1, 2, 3	1	2	1
Dense units	64, 128, 256, 512	64	256	128
Batch size	128, 256, 512	128	512	512
Learning rate	(0.0001, 0.001)	0.00028	0.00017	0.00033
Dropout	(0.0, 0.2)	0.17	0.10	0.07
<u>YAformer</u>				
Context length C	168, 336, 720	336	720	720
Patch size	1, 4, 8, 24	1	1	1
w_{conv}	1, 3, 5	5	1	5
LSTM layers	0, 1, 2	0	1	0
n_{layers}	1, 2, 3, 4	1	1	2
d_{model}	32, 64, 128, 256, 512	512	256	512
n_{heads}	1, 2, 4, 8	2	1	8
Attention	full, sparse	full	sparse	full
Max pooling width	1, 2, 4	4	1	1
Dense layers	1, 2, 3	2	2	1
Dense units	64, 128, 256, 512	128	128	256
Batch size	128, 256, 512	256	128	512
Learning rate	(0.0001, 0.001)	0.00014	0.00038	0.00056
Dropout	(0.0, 0.2)	0.05	0.11	0.10