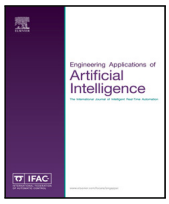
Engineering Applications of Artificial Intelligence 179 (2026) 115014

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Research paper

Short-term electricity load forecasting: Application-driven evaluation of machine learning models across spatial and temporal scales

Nikolaus Houben [a,*], Miguel Heleno [b], Han Li [b], Tianzhen Hong [b], Hans Auer [a], Amela Ajanovic [a], Reinhard Haas [a]

[a] Vienna University of Technology, Gußhausstraße 25 – 29/E37003, 1040 Vienna, Austria
[b] Lawrence Berkeley National Laboratory, One Cyclotron Road, Berkeley, CA 94720, USA

ARTICLE INFO

Keywords:
Short-term load forecasting
Machine learning
Multi-step ahead forecasting
Neural networks
Tree-based methods

ABSTRACT

As we transition towards a decarbonized economy, the integration of variable renewable energy resources and new demands (e.g., electric vehicles, heat pumps) into the electricity grid places unprecedented pressure on grid operators to effectively anticipate and manage peak load. In this context, machine learning algorithms are proving to be indispensable for accurate short-term load forecasting, a crucial task to address these challenges. This study benchmarks 6 machine learning algorithms, including three neural networks and three tree-based algorithms, across various levels of spatial aggregation and time horizons (1, 4, 8, 24, and 48 h). The central contribution of this work is the comparison and analysis of load forecasting models not only based on statistical metrics, but also based on a novel error metric, which evaluates the cost implications of forecast errors for power system stakeholders. Results show that tree-based models outperform neural networks, based on statistical metrics, and yield less skewed error distributions for most spatial scales. However, through the lens of the novel error metric, neural networks are the more competitive choice, especially for forecast horizons that exceed 8 h. The study concludes with actionable recommendations to grid operators and highlights the need for the development of error metrics that link forecasting accuracy to operational costs. To promote transparency and open science, the datasets and Python code are open-sourced via a supplementary repository.

1. Introduction

The transition to a decarbonized economy is underpinned by several pivotal components: the electrification of buildings, industry and transport, drastic energy efficiency upgrades, and the integration of variable renewable energy sources. As this transition unfolds, the integration of millions of assets into the grid will not only increase overall electricity consumption (IEA, 2023) but also introduce more irregular temporal patterns of electricity usage (Boßmann and Staffell, 2015). A primary challenge arising from this complexity is the anticipation and reduction of daily peak electricity consumption, or *peak load*. Managing peak load is a focal concern for grid operators, given its direct impact on the reliability of grid operations and the scale of associated investment costs (Pérez-Arriaga and Knittle, 2016). To tackle this, there is a growing reliance on using flexible assets – such as demand response mechanisms, electric batteries, and pumped-hydro storage – which play a pivotal role in energy balance by storing excess energy and supplying it during demand peaks.

Central to optimizing the use of these flexibilities is accurate load forecasts. Over the last few years, machine learning has become the most promising approach in this endeavor. Yet, the application of machine learning to time series forecasting presents a conundrum: While certain neural networks have ascended to dominance in fields such as image recognition (Krizhevsky et al., 2012) and language modeling (Bahdanau et al., 2014), time series forecasting remains a more contested domain (Kuster et al., 2017; Deb et al., 2017). The optimal algorithm often remains elusive, with tree-based algorithms persistently rivaling, if not surpassing, deep-learning techniques. This observation is validated by forecasting competitions (Hong et al., 2014; Miller et al., 2022; Januschowski et al., 2022) and aligns with the discourse by Shwartz-Ziv and Armon (2022) regarding the appropriateness of deep learning for tabular datasets. However, the rapid advances of deep-learning – especially with the advent of the transformer architecture (Vaswani et al., 2017), the foundation of modern language models – cannot be sidelined. Recent work has exhibited how transformer-based methodologies offer compelling results, both on benchmark datasets (Wen et al., 2022) and in real-world scenarios (Kunz et al., 2023; Gokhale et al., 2023).

* Corresponding author.
E-mail address: houben@eeg.tuwien.ac.at (N. Houben).

https://doi.org/10.1016/j.engappai.2026.115014
Received 9 April 2024; Received in revised form 22 March 2026; Accepted 29 April 2026
Available online 1 June 2026

Nomenclature

Abbreviations

BESS	Battery Energy Storage System
BOHP	Baysian Optimization and Hyperband
DC	Demand Charge
GBM	Gradient Boosting Machine
GPU	Graphical Processing Unit
GRU	Gated Recurrent Unit
MAE	Mean-absolute Error
MAPE	Mean-absolute-percentage Error
MIMO	Multi-Input Multi-Output
MPC	Model Predictive Control
N-BEATS	Neural Basis Expansion (Time Series)
NLE	Net Load Error
NN	Neural Network
R^2	R-squared Score
RF	Random Forest
RMSE	Root-mean-squared Error
SOC	State-of-Charge
TFT	Temporal Fusion Transformer
W	Watt
WandB	Weights & Biases
XGBoost	Extreme Gradient Boosting

Mathematical Symbols

ϵ	Generic error score
$\hat{y}$	Forecast of time series
y	Ground truth of time series

Superscripts

stored	State of charge of storage technology
opr	Operational
opt	Optimized

Subscripts

H	Optimization & forecast horizon
j	Time index of optimization/forecast
T	Final timestep of the dataset
t	Time index of the dataset

Due to the ambiguity around the most performant algorithm category, a plethora of research has been conducted on the comparison of models for load forecasting at specific spatial scales (Yildiz et al., 2017; Lusis et al., 2017; Hong, 2010; Elattar et al., 2020; Wang et al., 2021). These studies exclusively evaluate performance through statistical error metrics, which fail to capture the economic and operational impact of forecasts used as decision making inputs.

This context gives rise to the two central questions of this research:

1. To what extent is there a single machine learning algorithm or model category that consistently outperforms others across diverse spatial and temporal scales for short-term electricity load forecasting?
2. How can short-term load forecasting models be evaluated in a way that reflects their practical, economic value for operational tasks like peak load management?

To address these questions, this paper provides three key contributions to the state-of-the-art. First, we conduct a comprehensive benchmarking of six prominent machine learning algorithms across twelve independent electricity load time series, spanning four levels of spatial aggregation. This provides a broad understanding of algorithm performance in different contexts. Second, we perform a detailed error analysis, moving beyond standard metrics to offer a more contextualized understanding of model performance across different seasons and forecast horizons. Third, and most significantly, we introduce a novel, application-driven evaluation metric, the *Net Load Error (NLE)*. The NLE is designed to bridge the gap between statistical accuracy and operational costs by quantifying the economic consequences of forecast errors within a peak-shaving application. By evaluating models through the lens of both traditional metrics and the NLE, this study provides actionable recommendations for grid operators and highlights the need for evaluation methods that link forecasting accuracy to real-world outcomes.

1.1. Review of the state-of-the-art

A foundational understanding of electricity load forecasting requires recognizing its classifications. The field is broadly categorized by forecast horizon, from very short-term (sub-hourly) to long-term (multi-year), and by the nature of the output, which can be deterministic (a single point forecast) or probabilistic (a prediction interval) (Petropoulos et al., 2022). This study is positioned within the short-term, deterministic forecasting domain. General reviews of the field highlight that selecting appropriate models and evaluation metrics remain pivotal challenges for researchers and practitioners alike (Kazmi et al., 2023; Hong et al., 2020; Haben et al., 2021). Systematic reviews and meta-analyses have quantitatively confirmed that a multitude of factors – including the specific algorithms used, the forecast granularity, and the level of spatial aggregation – have a significant impact on forecast performance (Hopf et al., 2023; Nti et al., 2020; Li et al., 2025).

This literature review is built upon a comprehensive survey of the field. After filtering over 800 papers on load forecasting, a total of approximately 120 papers were thoroughly reviewed in this work. Of these, 18 were selected to build the literature review, which includes papers that adhere to one or more of the following criteria:

- Application and comparison of multiple machine learning algorithms for short-term load forecasting,
- Detailed discussions on performance, including error metric comparisons between algorithms for short-term load forecasting at a specific or multiple spatial scales,
- Examination of the real-world, economic, or application-driven impacts of short-term load forecasts.

1.1.1. Comparing forecasting algorithms for specific spatial scales

The performance of electricity load forecasting algorithms is often investigated at a specific level of spatial aggregation. A meta-regression analysis of 59 studies confirmed that the grid level (e.g., individual vs. system) is one of the most significant factors influencing forecast error (Hopf et al., 2023). At the building level, machine learning models have been shown to consistently outperform traditional linear regression models (Yildiz et al., 2017). While tree-based and neural network models often yield comparable statistical performance, some studies indicate that tree-based approaches can produce less biased error distributions over the test set (Lusis et al., 2017). A recurring finding at this scale is that accurately forecasting the timing and magnitude of peak load is a distinctly more difficult task than forecasting the overall load trajectory (Yildiz et al., 2017). Furthermore, the inherent characteristics of the data, such as the load's dispersion level, can significantly influence the suitability of a given forecasting model (Hu et al., 2023).

Moving to the neighborhood or low-voltage feeder level, a clear gap in the literature has been identified, suggesting that this scale has received less attention than others (Haben et al., 2019; Pinheiro

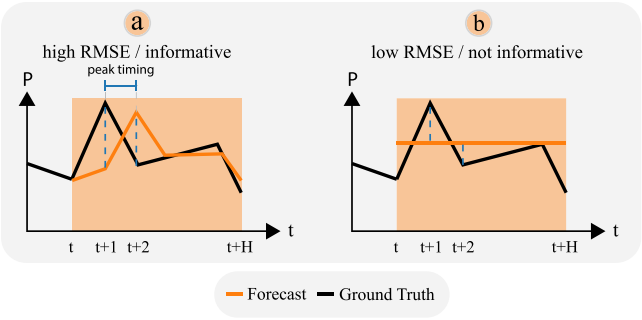


Fig. 1. Double Penalty Effect of Euclidean Error Metrics [single column, print full colors].

et al., 2023). Work in this area has demonstrated substantial performance gains, with machine learning models outperforming previous benchmarks by around 40% based on statistical error metrics (Pinheiro et al., 2023; Hong, 2010). Such studies also emphasize the need for a multi-criteria approach to model selection, where statistical accuracy is considered alongside practical factors like applicability, reproducibility, and interpretability (Pinheiro et al., 2023).

At higher levels of aggregation, such as large cities and counties, research has involved both the development of bespoke forecasting algorithms and the benchmarking of established methods (Elattar et al., 2020). Comparative studies at this scale have evaluated a wide range of models, including various neural networks, support vector regression, and tree-based ensembles (Elattar et al., 2020; Wang et al., 2021). These benchmarks have often highlighted the strong performance of gradient boosting models, with LightGBM yielding the lowest error scores in one such study, and have underscored the critical importance of including exogenous variables like weather to achieve high accuracy (Wang et al., 2021).

1.1.2. Models in load forecasting

A wide range of algorithms and model architectures have been developed and applied to tackle the task of short-term load forecasting. Out of all, two distinct model categories stand out: tree-based methods and neural networks. This is evident in the scientific literature (Yildiz et al., 2017; Lusis et al., 2017; Elattar et al., 2020; Wang et al., 2023; Li et al., 2025) and in forecasting competitions like the M-competition series (Hong et al., 2014; Ben Taieb and Hyndman, 2014; Hong et al., 2019). The reviewed studies provide a crucial lens on statistical performance and computational trade-offs, often favoring tree-based methods over neural networks. However, the same studies rely exclusively on conventional (statistical) error metrics and focus on select spatial scales to draw these conclusions. Therefore, a significant gap remains in comparing forecasting models on multiple spatial scales by evaluating them based on their economic and operational value.

1.1.3. Economic evaluation of forecasts

As the grid digitizes and data-driven load forecasting methods proliferate, understanding the real-world implications of forecast errors – *i.e.*, their economic impact on energy systems – is of increasing relevance. Yet, there remains a scarcity of research that explicitly links a forecast's accuracy to the operational or economic outcomes of its application (Haben et al., 2021). The need for such application-driven metrics is motivated by the shortcomings of conventional error scores. For instance, Euclidean-based metrics like the Root Mean Squared Error (RMSE) suffer from the *double penalty effect*, where a forecast that correctly predicts the shape and magnitude of a peak but mistimes it slightly is penalized more heavily than an uninformative, flat forecast (Keil and Craig, 2009), as illustrated in Fig. 1.

Motivated by this, a growing stream of research evaluates forecasts based on their applied value. The economic impact of forecast uncertainty has been a long-standing topic, with early work using Monte Carlo simulations to assess the costs of prediction errors (Ranaweera et al., 1997). More recently, Model Predictive Control (MPC) has emerged as a common framework for these evaluations. Within an MPC setup, forecast value can be assessed by its ability to optimize an objective, such as peak load reduction (Voss, 2020), or the monetary value of operating a complex energy system (Putz et al., 2023; Houben et al., 2023). These studies have compared the economic outcomes derived from different forecasting algorithms, such as XGBoost and Prophet, and analyzed how forecast errors impact specific objectives, finding that applications like demand charge minimization are particularly sensitive to forecast quality (Houben et al., 2023). This approach has also been used to evaluate advanced techniques, such as transfer learning with Temporal Fusion Transformers, by measuring both statistical error and resulting operational costs (Gokhale et al., 2023).

1.2. Structure of the paper

This paper is structured as follows: Section 2.3 discusses the machine learning modeling framework and evaluation metrics. Section 2.2 presents the load time series data of the various spatial scales. In Section 3, the most important results are shown, highlighting the differences of algorithms with regards to performance metrics based on time of the year and use case. Next, Section 4 synthesizes and critically discusses the main results, while also highlighting the major limitations of this work. Finally, Section 5 provides a conclusion and suggests directions for further research.

2. Methods: Economic evaluation of data-driven load forecasts

The methodology of this work can be separated into three parts: Firstly, the description of a novel error score, the Net Load Error, which is used to economically evaluate the short-term forecasts of the studied algorithms is given. Secondly, the selection and processing of the data. Thirdly, the machine learning modeling framework, which includes the selection of forecasting algorithms and features, the parameter and hyperparameter optimization process, and model evaluation.

2.1. Net load error

Motivated by the lack of application-driven error metrics (as surveyed in Section 1.1.3), this work introduces a novel error score, the *Net Load Error* (NLE), which is designed to capture not only the accuracy of peak load forecasting, but also to translate the consequences of any forecasting errors into tangible operational costs for power systems. This is particularly crucial both from the perspective of the system operators, and end-consumers. On the one hand, system operators may employ peak load forecasts to efficiently dispatch resources, such as generation units or reserves, and in extreme cases, implement load shedding measures. For consumers, on the other hand, performant peak load forecasts are vital for managing loads or building-level Distributed Energy Resources (DERs) to evade demand charges or contracted power, which constitute a significant portion of avoidable electricity costs.

The NLE quantifies the costs that can be attributed to the errors of a load forecast by controlling a stylized[1] energy system, which includes a Battery Energy Storage System (BESS), an electricity load forecast, its accompanying ground truth time series, and a daily demand charge. For the purpose of developing a specific test system to represent the

[1] As opposed to realistic operating reserve procurement by the grid operator, this energy system is a highly simplified setup. Major assumptions are provided in Appendix B and limitations are discussed in Section 4.3.

diverse resource allocation scenarios of consumers or system operators, we utilize BESS operation as a representation. BESS, with its inherent temporal power/energy relationships, effectively abstracts various forms of sequential dispatch decision and serves as a common stylized system to translate forecast errors into actual costs.

The operational model within this framework uses an economic Model Predictive Control (MPC) setup (see Drgoňa et al., 2020; Rawlings, 2000), running two scenarios: one with perfect forecasts, i.e., the ground truth, and the other with the forecasts to be evaluated. This process results in two distinct operational *net loads*. The observed operational net loads of each scenario are evaluated ex-post through the daily demand charge, leading to the calculation of *operational costs*. The difference in these operational costs, stemming from net loads informed by the ground truth versus those informed by the forecast, is defined as the Net Load Error.

The following sub-sections provide a detailed description of the NLE framework, with mathematical details and major assumptions outlined in Appendix B.

2.1.1. System description

Consider the system setup depicted in Fig. 2a, in which $\forall t \in T$ a forecast load $\hat{y}_{j=0:H;t}$ with horizon H, its corresponding ground truth $y_{j=0:H;t}$, a demand charge P_{DC}, and a battery electric storage system (BESS) with specification parameters θ are available. The daily demand charge refers to a pricing scheme that penalizes the highest measured value of the day, as given by Eq. (1) and henceforth referred to as *operational costs*. These costs are calculated ex-post, *i.e.* looking back onto an observed load time series.

$$C^{opr} = \sum_{d} max\ l_{t;d} * P_{DC} \tag{1}$$

Note that $l_{t;d}$ is the *operational net load*, which is the original load y_t modified by the BESS (dis-)charging action u_t, as shown in Eq. (2). It is indexed by t not indexed by j, since it is the realized load after forecasting and optimizing for $j \in H$.

$$l_t^{opr} = y_t + u_t \tag{2}$$

Following the classical economic MPC formulation, shown in Fig. 2(a), at each timestep an optimizer solves the optimization problem[2] shown in Eq. (3), by finding an optimal trajectory of charging actions $\boldsymbol{u}$.

$$\min_{u} C^{opt}(l^{opt}, P_{DC}) + V(SOC_{j=H}) \quad \text{s.t.} \quad g(l^{opt}, P_{DC}, \theta) \geq 0,$$
$$h(l^{opt}, P_{DC}, \theta) = 0. \tag{3}$$

where,

C^{opt}	= demand charge proxy cost for horizon H
u	= charging actions $u_{j=0:H}$
l^{opt}	= optimized net load $l^{opt}_{j=0:H}$
V	= terminal costs, avoiding complete discharge at the final optimization timestep
SOC	= state-of-charge of the BESS
P_{DC}	= daily demand charge
θ	= BESS parameters

Considering Fig. 2(b), analogous to Eq. (2) but within the optimization (indexed by j), the charging actions modify the load (the forecast) so that it has a minimum peak value, given by Eq. (4).

$$l_j^{opt} = \hat{y}_j + u_j \tag{4}$$

In standard MPC fashion, having optimized this system at timestep t w.r.t. to the available load forecast $\hat{y}$, the first optimal charging setpoint $u_{j=0}$ is applied to the real system at timestep $t+1$, resulting in an *operational net load* l_{opr} at the next timestep $t+1$ given by Eq. (5):

$$l_{t+1}^{opr} = y_{t+1} + u_{j=1} \tag{5}$$

This is followed by carrying over the optimized SOC of the BESS of the previous timestep t, to the next optimization run at $t+1$, given by Eq. (6):

$$SOC_{j=0,t+1} = SOC_{j=1,t} \tag{6}$$

Crucially, note that the operational net load l_{t+1}^{opr} and the optimized net load $l_{j=1}^{opt}$ are different if the errors of forecast $\hat{y}$ are non-zero for any $j \in H$. This is shown in Fig. 2c, where the underestimation by the load forecast at timestep t, results in an operational net load that exceeds the optimized net load at timestep $t+1$. Inadvertently, a new peak is created, which will lead to higher demand charge costs in the ex-post evaluation.

Shown in Fig. 2a, in the economic MPC formulation the process of updating the ground truth load y_{t+1} with charging actions $u_{j=1}$ is repeated for the entire test period ($\forall t \in T$). Once finished, an ex-post evaluation based on the daily demand charge in Eq. (1) is performed.

In summary, the key idea of the NLE framework is that if ground truth values were available as perfect forecasts, the operational and optimized net load do not deviate, providing a baseline for optimal peak shaving. In other words, in such a case the charging actions from each optimization remain optimal when applied to the real system. Thus the difference in operational costs between an MPC run with the ground truth as perfect forecasts and another using the actual forecasts in question, yields a performance score, which assesses the ability of that forecast to effectively predict peak load (see Fig. 2d).

2.2. Dataset selection and processing

This paper draws on three open-source datasets, each including multiple load time series covering a broad range of geographies and consumer types. Specifically, the levels of spatial aggregation (referred to as *spatial scales*) between the datasets vary widely; with dataset collected at what will henceforth be referred to as county, town, neighborhood, and building levels. For instance, the county-level load time series, taken from the *EIA Cleaned Hourly Electricity Demand* dataset presented in Ruggles et al. (2020), encompasses millions of end-users composed of households, industries, transport, and public lighting, whereas the neighborhood-level dataset from USA (Miller et al., 2020) consists of appliance loads of office buildings. The diversity of datasets represents the electricity demand across all scales and consumer types, forming a robust and transparent foundation for the subsequent evaluations (see Table 1). To ensure 1 year of training and 2 months for development and testing datasets, a requirement of the data collection process was that all data included at least 2 years of timeseries (see Fig. 3).

From each dataset three independent timeseries, corresponding to three electricity consumer groups, were selected. The three time series per spatial scale were selected to represent a diverse range of load patterns. For instance, in the county scale one of the time series is the aggregate load of the Los Angeles greater area, the other New York. These two locations differ strongly in degree of electrification of transport, heating and cooling, which is clearly visible in the data and discussed in Section 4.1. Similar data inspections were carried out for the other spatial scales, ensuring significant differences of basic statistics (mean and standard deviation) and correlations between exogenous input features.

The timeseries range of values, temporal resolution, country of origin, column labels within the original datasets, and reference are provided in Table 1. For the purposes of this work, all data was resampled to 60 min resolution. Furthermore, even though data had been cleaned by their providers, additional cleaning, which was necessary for a systematic comparison was performed and is described in

[2] The complete optimization problem formulation is given in Appendix B.

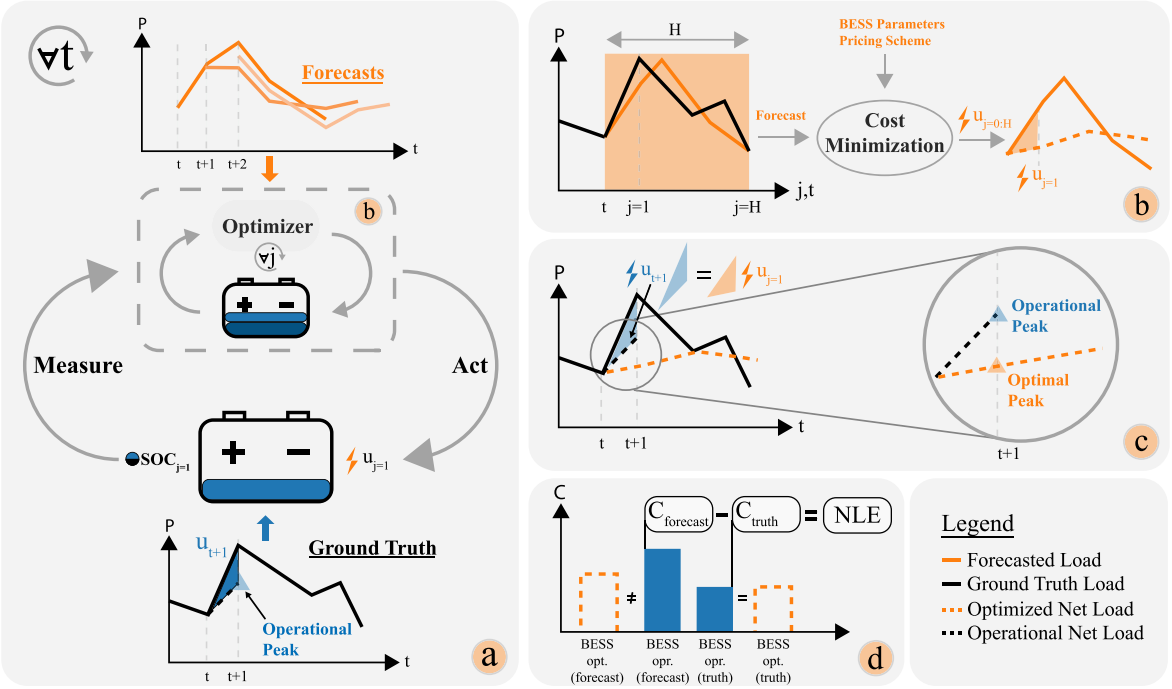


Fig. 2. NLE: System considered to evaluate operational costs of a BESS modifying a load curve[double column, print full colors].

Table 1
Overview of the datasets.

Spatial scale	Power range	Resolution	Country	Meter label	Source
County	1–50 GW	60 min	USA	LDWP, BANC, NYIS	Ruggles et al. (2020)
Town	5–50 MW	15 min	Portugal	MT_196, MT_279, MT_208	Trindade (2015)
Neighborhood	5–2000 kW	60 min	USA	Bull, Hog, Bobcat	Miller et al. (2020)
Building	0–2 kW	60 min	USA	Be_Sandy, Be_Millie, Co_Joel	Miller et al. (2020)

Section 2.2.1. In conjunction to the load data described above, hourly temperature data was collected for each location as an exogenous covariate due to its considerable effect on electricity consumption (Pardo et al., 2002; Behm et al., 2020). This data was sourced from the *Open-Meteo API* (Open-Meteo, 0000).

2.2.1. Data exploration & cleaning

Time series datasets were thoroughly examined with regards to outliers and missing values (NaN). Outliers were detected with the empirical rule, using the 99th percentile as a threshold, and set as missing. For load time series that included significant power generation, *i.e.* included negative values, the symmetric 1st percentile is used as a lower bound. As the examined machine learning algorithms do not deal with missing values internally, these were dealt with in a three-step process. For a given time series these process is as follows:

1. The missing values per day of the time series were counted. If for any given day its count exceeded 10% of the total values of that day, the day was removed.
2. If the number of consecutive missing values equated to less than an hour, the missing value(s) were interpolated by taking the mean of the neighboring values. If they exceeded an hour, they were kept missing (NaN).
3. The time series was split into multiple sub-series at the missing values as breakpoints. Specifically, each sub-series starts with the first valid value and ends with the last valid value. If the length of the sub-series was shorter than the longest look-back window used by the forecasting models (l_{max} = 72 h) plus the longest forecast horizon (H_{max} = 48 h), they were removed.

2.3. Machine learning modeling framework

To systematically tune, train, evaluate, and compare multiple algorithms across diverse datasets, a forecasting framework was developed. This framework[3] is anchored by the *Darts* framework (Herzen et al., 2022) and the *Weights & Biases* experiment tracking platform. It encompasses all stages integral to a machine learning time series forecasting project: data exploration and cleaning, feature engineering, data splitting, hyperparameter optimization, evaluation, and algorithm comparison. Subsequent sub-sections delve into each of these stages, detailing the applied methodologies.

2.3.1. Forecasting algorithms

A total number of 6 algorithms are studied in this work. They were selected to be representative of the two major algorithm types used in time series forecasting, namely tree-based and neural network-based algorithms (see Section 1.1.2). These two algorithm types are also the most prevalent in the load forecasting literature, with tree-based methods often outperforming neural networks in terms of statistical error metrics.

In the literature Random Forest (RF), XGBoost, and LightGBM stand out as key representatives of tree-based methods. Random Forest constructs its trees independently, offering robustness through diversity through bootstrap aggregation (see Appendix A). In contrast, XGBoost and LightGBM build trees sequentially, addressing the errors of previous ones, with both using gradient tree-boosting techniques. By virtue

[3] Please find the framework in the GitHub repository https://github.com/nikohou/WattCast.

Table 2
Overview of the studied machine learning algorithms.

Algorithm	Type	Mechanism	Multi-Step	Reference
RF	Decision tree ensemble	Bagging	Direct	Breiman (2001)
XGBoost	Decision tree ensemble	Grad. Boosting & Depth First	Direct	Chen and Guestrin (2016)
LightGBM	Decision tree ensemble	Grad. Boosting & Breath First	Direct	Ke et al. (2017)
GRU	Deep neural network	Memory Gates	MIMO	Chung et al. (2014)
N-BEATS	Deep neural network	Basis Expansion	MIMO	Oreshkin et al. (2019)
TFT	Deep neural network	Attention	MIMO	Vaswani et al. (2017) and Lim et al. (2021)

of gradient-based methods, each of these models includes regularization in their cost functions, to strike the balance between over- and underfitting. While similar in approach XGBoost and LightGBM differ in their strategy in growing trees (depth vs. breath first). The inclusion of both provides nuanced insights as to which one is more suitable for specific scales, given the same training budget (see Section 2.3.4).

On the side of neural networks, representative model architectures found in the load forecasting literature include Gated Recurrent Units (GRU), Neural Basis Expansion for Time Series (N-BEATS) and Temporal Fusion Transformer (TFT). While these neural networks are implemented in a multi-input multi-output way to achieve multi-step outputs, their operational mechanisms differ significantly. Firstly, GRU are recurrent neural networks developed for sequence-to-sequence tasks, propagating a hidden state from cell to cell, each responsible of processing one timestep of a time series. Secondly, N-BEATS casts the timeseries forecasting problem into a non-linear multi-variate regression by stacking multiple feed-forward neural network blocks. Thirdly, TFT combines the recurrent neural networks with the attention mechanism. Given their strong performances in the time series forecasting literature and the aforementioned differences in mechanisms, these algorithms were selected as representatives for the neural network type.

An overview of algorithms is given in Table 2, while the mathematical description of each algorithm, the mechanism and specific implementation are provided in Appendix A. For a more detailed description of each algorithm the reader can refer to the references listed in Table 2. All forecasting models were implemented in Python using the Darts Library (Herzen et al., 2022), which provides wrappers for sklearn, dmlc xgboost, and pytorch models (Pedregosa et al., 2011; Chen et al., 2018; Paszke et al., 2019). Model training and inference was performed on a Nvidia RTX8000 GPU.

2.3.2. Feature selection & engineering

Feature engineering is an important step in supervised learning. In time series modeling features can be distinguished between endogenous and exogenous, where endogenous features are crafted from the time series itself, and exogenous features have to be derived from additionally collected data. This study included the following endogenous features: lagged measurements of up to 72 h, and trigonometric and one-hot encoding of the timestamp of each measurement. For reference, the equation of the hourly trigonometric encoding is shown in Eq. (7), where k is the hour of the current timestep, and C is the period of one cycle, *i.e.* 24 h.

$$m_{C,\sin} = \sin\left(\frac{2\pi k}{C}\right) \quad m_{C,\cos} = \cos\left(\frac{2\pi k}{C}\right) \tag{7}$$

In this work, outdoor air temperature is used as the basis for exogenous features. Specifically, the temperature time series itself and a binary heat-wave variable was implemented. The latter feature marked days when the outdoor temperature was greater or equal to the 95th percentile of past temperature data. This helps the model identify any patterns or changes related to very high temperatures. Though results by Wang et al. (2021) have shown that training machine learning models with weather forecasts, instead of measurements, yields improved performance, this work opts for using measurements, due to improved data accessibility.

A summary of available features is given in Table 3.

Furthermore, time series were normalized as shown in Eq. (41), with values scaled to the range of $[0, 1]$, and *Boxcox-transformed* (Sakia, 1992), to make the data distribution quasi-normal. As described in Section 2.3.4, the decision of including any feature in the final model resulted directly from the hyperparameter tuning approach (see Section 2.3.4). Consequently, some of the final models include features that others do not.

2.3.3. Data splits

Each time series dataset was split into a training set consisting of one year (Y1) of data and testing set of a full month in summer and in winter from another year (Y2) (see Fig. 3). The two months in the testing set were manually selected to contain days with extreme weather conditions, such as heat waves or cold snaps.

To save time on hyperparameter tuning (see Section 2.3.4) each dataset was shortened to only include the first week of each month of the training set during hyperparameter search phase. During these runs the one of the summer months was used as the validation set.

2.3.4. Hyperparameter optimization & model selection

The performance of the studied supervised learning models is sensitive to their hyperparameters. One of the key objectives of this work is the objective comparison of algorithms by providing an equal training budget in terms of number of hyperparameter samples. This means that each algorithm received a total of 25 training runs within the hyperparameter tuning framework, to find the optimal hyperparameter configuration for a given dataset. With twelve datasets, 6 algorithms, and 25 training runs, a total of 1800 model training runs were completed. The optimal hyperparameters were then used to train a final model with the entire dataset for the maximum horizon of 48 h.

There are different techniques to find a suitable combination of hyperparameters, including grid search, random search, scheduled Markov Chain Monte Carlo (see Lechner et al., 2022), and Bayesian Optimization (see Turner et al., 2021). In this study, the hyperparameters of the model are tuned with Bayesian Optimization and Hyperband, as implemented in WandB (Falkner et al., 2018; Biewald, 2020). This method combines Bayesian Optimization with those of bandit-based methods, and has been shown to yield superior performance compared to using just Bayesian optimization or Hyperband across a diverse array of problem types.

As an example, the hyperparameter search space of the three tree-based algorithms, as well as the optimal parameters for the LDWP dataset are provided in Table 4. The analogous table for all other datasets and algorithms (*i.e.* the neural networks) can be found in the *WandB project*.[4]

Even though Table 4 only presents an example of hyperparameters for tree-based models, a short explanation on the mechanisms and types of hyperparameters is in order. By tuning hyperparameters the algorithm is shaped in a way to learn the training data as well as possible, while not overfitting to it. As such, there are two types of hyperparameters that can be considered in the presented algorithms. The first type concerns the size of the model, effectively setting the total number of parameters (degrees of freedom). In tree-based methods this entails

[4] https://wandb.ai/wattcast/Wattcast_tuning.

Table 3
Summary of features used in the study.

Feature	Description	Feature type	Temporal
Lagged measurements	Derived from the time series	Endogenous	Past
Hour of the Day	Trig. encoding of timestamp, see Eq. (7)	Endogenous	Future
Day of the Week	One-hot encoding of timestamp	Endogenous	Future
Week of the Year	One-hot encoding of timestamp	Endogenous	Future
Temperature time series	External temperature measurements	Exogenous	Future
Binary heat-wave variable	95th percentile temperature	Exogenous	Future

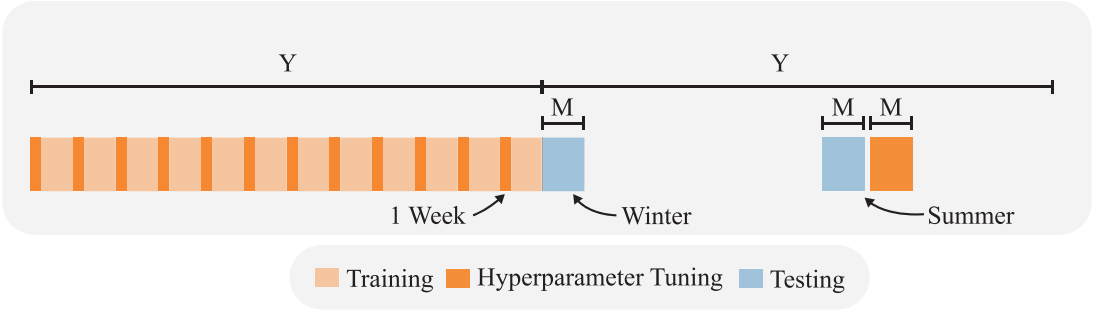


Fig. 3. Train and Test, Hyperparameter tuning data splits [single column, print full colors].

Table 4
Hyperparameter and Feature search space for tree-based algorithms and best configurations for Random Forest, XGBoost, and LGBM for "LDWP (Los Angeles)". The configuration names, as logged in WandB, are indicated below the algorithm abbreviation.

Hyperparameter	Values	RF *summer-sweep-20*	XGBoost *woven-sweep-14*	LGBM *proud-sweep-16*
n_estimators	50, 100, 500, 1000	1000	1000	1000
learning_rate	0.05, 0.1, 0.2, 0.3	–	0.05	0.05
max_depth	3, 6, 12	15	15	5
min_child_weight	1, 5, 10	–	10	5
min_samples_split	2, 5, 10 , 15	2	–	–
min_samples_leaf	2, 5, 10 , 15	2	–	–
objective	mse, huber	mse	huber	huber
reg_lambda	0.1, 0.3, 0.5, 0.7, 1, 2, 4	–	0.5	4
Feature				
n_lags	12, 24, 48, 72	24	24	24
datetime_encodings[a]	0, 1	0	1	1
temperature_feature	0, 1	1	1	1
heat_wave_binary	0, 1	0	0	0

[a] These include the trigonometric and one-hot encoding of hour of the day, day of the week, and week of the year mentioned in Table 3.

deciding on the number of decision trees (i.e. *n_estimators* and *max_depth* to which each tree is grown, while in neural networks this type of hyperparameter is represented by the width of linear layers or the dimension of hidden state. The second type of hyperparameter regards regularization, which effectively constraints the number of parameters used to learn the data. In tree-based methods examples are *reg_lambda* and *min_child_weight*, while in neural network the dropout parameter is typically employed. For a more detailed study on hyperparameter tuning refer to Probst et al. (2019).

2.3.5. Model inference

In this study model performance is evaluated over the horizons of 1, 8, 24, and 48 h. Therefore for each model and test dataset a forecast with the training horizon (48 h) was generated at each time step $t \in T$, and subsequently truncated and compared to the ground for the forecast horizon at question. Fig. 4 illustrates this procedure.

It should be noted that we perform inference at each timestep of the time series for the entire 48 horizon. In other words, this means that the last inference call for each time series is 48 h before its final timestep, ensuring a full 48 h of ground truth to be compared to the prediction. In this setup the final error score is the mean of error scores across all timesteps in the test set. For an arbitrary metric D its error score $\epsilon^{(D)}$ is calculated as shown in Eq. (8):

$$\epsilon^{(D)} = \frac{1}{T}\sum_{t}^{T} \epsilon_t^{(D)} \tag{8}$$

2.4. Forecast performance evaluation & benchmarking

The error metrics applied in this work consist of four statistical error metrics, and a novel application-driven error score, termed the *Net Load Error*.

2.4.1. Statistical error metrics

The statistical error metrics include the root mean squared error (RMSE), mean absolute error (MAE), mean absolute percentage error (MAPE), and the R^2. Their mathematical expressions for a single forecast series with horizon H at timestep t are given by Eqs. (9)–(12).

$$RMSE_t = \sqrt{\frac{1}{H}\sum_{j=0}^{H}\left(y_j - \hat{y}_j\right)^2} \tag{9}$$

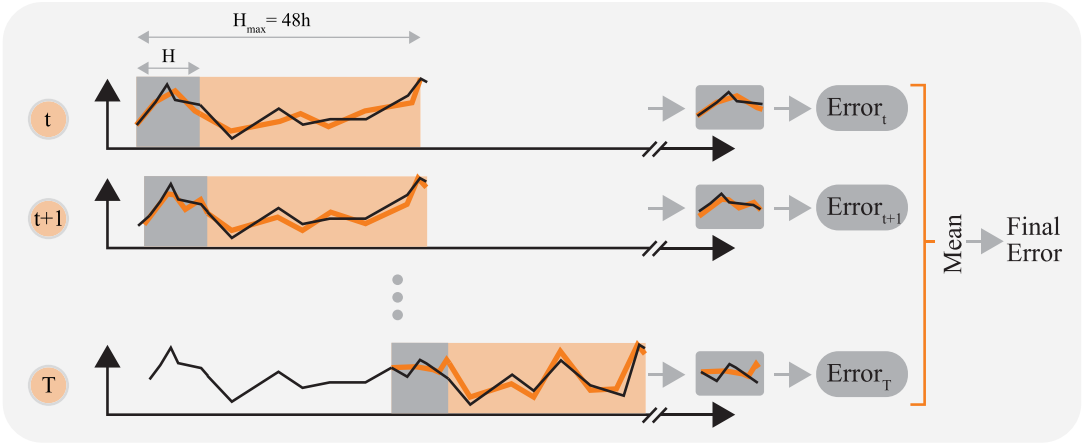


Fig. 4. Output of the Forecasting Models is at each timestep t for the maximum (training) horizon and then truncated to the horizon at question. [single column, print full colors].

$$MAE_t = \frac{1}{H}\sum_{j=0}^{N} | y_j - \hat{y}_j | \tag{10}$$

$$MAPE_t = \frac{100\%}{H}\sum_{j=0}^{H}\left|\frac{y_j - \hat{y}_j}{y_j}\right| \tag{11}$$

$$R_t^2 = 1 - \frac{\sum_{j=0}^{H}(y_j - \hat{y}_j)^2}{\sum_{j=0}^{H}(y_j - \bar{y})^2} \tag{12}$$

where,

H = number of point forecasts within the horizon
y_j = ground truth based on measurement data
$\hat{y}_j$ = prediction of the forecasting model
$\bar{y}_j$ = mean of the ground truth

2.4.2. Benchmarking

We selected a rudimentary Linear Regression (Hastie et al., 2009) algorithm in its direct implementation (see Appendix 5), using 48 lagged measurement values and datetime encoding, as described in 2.3.2, as the benchmark algorithm. The resulting benchmark models took around 0.5 s to train, which is considerably faster than the models associated with the examined algorithms.

Regarding statistical benchmarking, once horizon-based error scores (see Fig. 4) of forecasts were averaged across all timesteps across the test dataset, the skill scores relative to the error score of the corresponding[5] Linear Regression model were calculated (see Eq. (13)):

$$S_M^{(D)} = 1 - \frac{D_M}{D_{LR}} \tag{13}$$

where,

$S_M^{(D)}$ = Skill Score w.r.t. to an error metric D and specific model M
D_M = Root Mean Squared Error of model M
D_{LR} = Root Mean Squared Error of the Linear Regression (benchmark model)

[5] Trained on the same training dataset and inference for the same forecast horizon.

3. Results

The results section is structured as a three-part analysis to report on the performance of different algorithms on the examined datasets: First, we examine the qualitative performance of the models to see if they are generally viable for the task of short-term load forecasting. This includes showcasing side-by-side comparisons of model forecasts on specific days for select datasets. We also group algorithms by type, *i.e.* tree-based and NNs, and benchmark them with the base Linear Regression model. Secondly, we show how factors like seasons and forecast horizons affect the model performance, drawing on error distribution analyses for a clearer understanding. The third part of our analysis is focused on peak load prediction, validating previous observations with our proposed Net Load Error score. Note that the following parts merely report on the results and the interpretation and discussion of results is presented in Section 4.

3.1. Overall performance

To provide a qualitative comparison of the performance of the algorithms, Fig. 5 shows 24-hours ahead forecasts of tree-based models (left) and NNs (right) on days of extreme temperatures[6] for one time series dataset per spatial scale. Besides the depicted forecasts, a general observation regarding the ground truth (gray) is that the timeseries increase in volatility as spatial scale decreases, moving from the county-level to the building-level dataset. This is intuitive, as higher levels of aggregation include more diverse load types and consumers. In terms of forecasts (dashed), it is evident that models are better able to learn timeseries that aggregate more consumer types, *i.e.* higher levels of spatial aggregation.

Further examining the qualitative differences between the forecasts of the two algorithm types, Fig. 5 shows that tree-based models are well-suited for repeating trends of load timeseries, but tend to under-predict peaks, whereas NNs exhibit increased ability in reaching the peaks, however with the downside of overestimating days with low load (e.g. *building_2* or *neighborhood_1*). Furthermore, the forecasts of NNs exhibit sharper edges, *i.e.* less auto-correlated trajectories, than those of tree-based models.

While we find the above statements to be consistent with the ensuing quantitative analysis, it should be noted that much more nuance is required to make reliable choices in terms of forecasting algorithm for a given time series. The next sections will quantitatively underpin these observations and provide detailed results on the performance of algorithms for short-term forecasting tasks across various levels of aggregation.

[6] Results for other days can be found in the WandB repository.

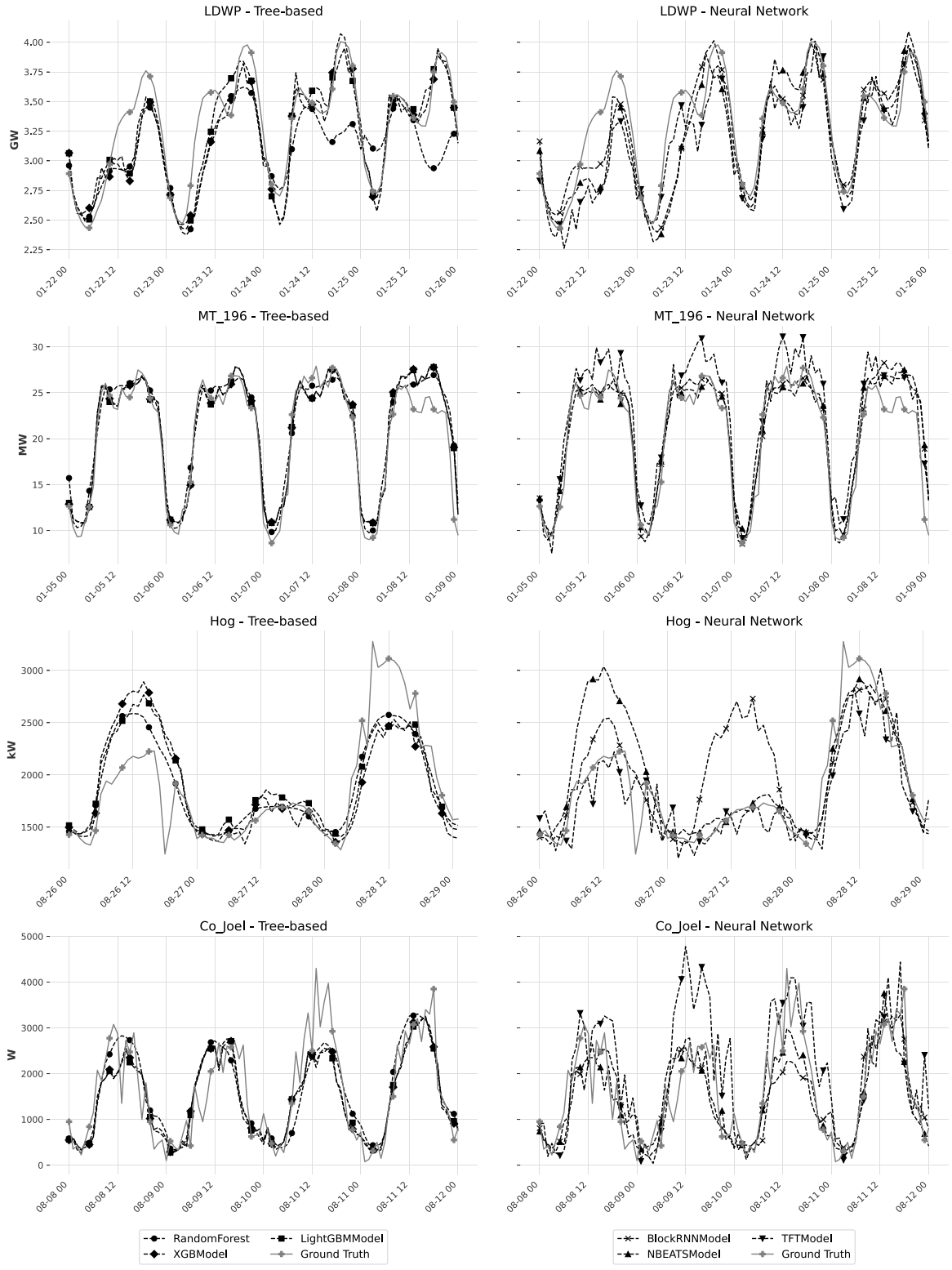


Fig. 5. Qualitative comparison of algorithm types on spatial scales for 24 h ahead forecasts [double column].

3.1.1. Tree-based vs. Neural networks vs. Benchmark

To quantitatively compare models we benchmark them with models based on a Linear Regression (see Section 2.4.2). Fig. 6 provides the best models by algorithm type for all spatial scales and forecast horizons. This is performed by grouping each model by its algorithmic type, and then averaging the error scores of the groups per horizon and scale.

The grouped models outperform the benchmark models for all scales and on most horizons. Specifically, the benchmark models only beat NNs and tree-based models on horizons shorter than 24-hours for the highest level of spatial aggregation, *i.e.* the county scale. This is the case for all examined error metrics and makes intuitive sense, when considering that county-level data smooth and therefore relatively easy to forecast (see Fig. 5). Between the two model types, tree-based

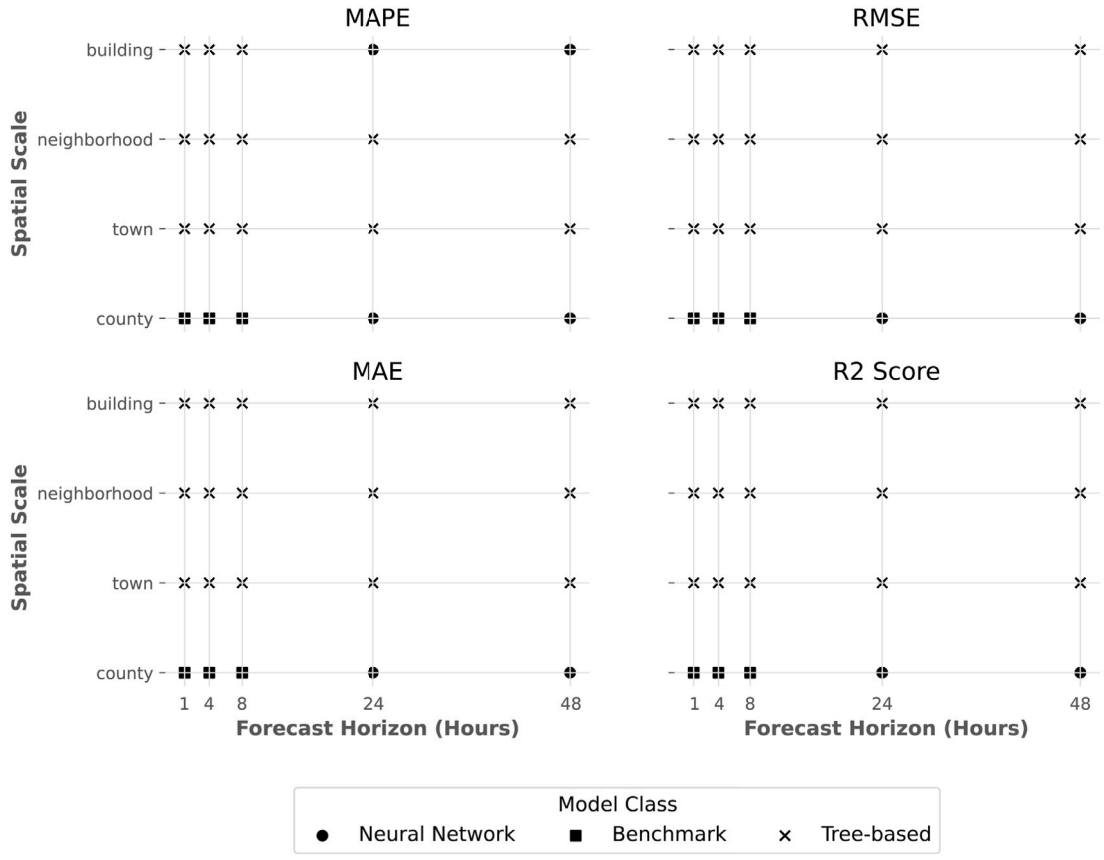


Fig. 6. Best model type per spatial scale and forecast horizon [double column, print full colors].

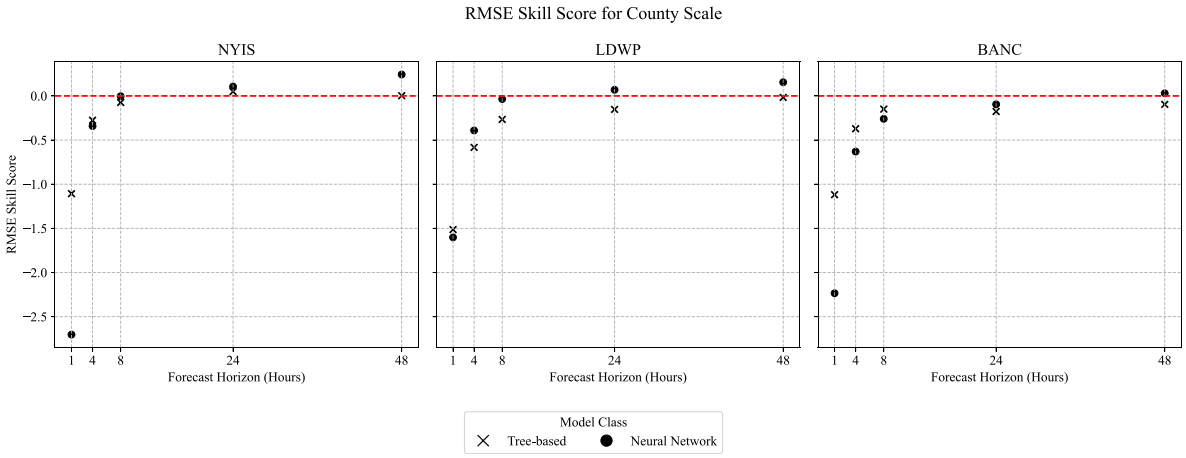


Fig. 7. Skill score w.r.t benchmark model (red dashed line) [double column, print full colors].

algorithms tend to yield lower error scores than NNs across the board. Exceptions were observed at the county scale, when looking at RMSE, MAE, and R^2, and building scale when measuring by MAPE. While shown for completeness, it should be noted that for datasets with measurements close to or equal to zero (the building-level data), the MAPE should be interpreted with caution. For more nuanced insights, the relative gain through the RMSE skill score is shown in Fig. 7 for the county level timeseries.

For the 3 datasets on the county level, the studied algorithms outperform the benchmark model for horizons longer than 24 h. While this result is expected, as tree-based and NN models have the potential to learn more complex relationships between features and targets than a linear regression, it is noteworthy how the skill scores of the same are significantly negative for horizons of 1 and 4 h.[7] The NNs perform particularly poorly on these short horizons. This trend is representative of the other studied datasets on the various spatial scales, in that the skill score improves for increasing forecast horizons.[8] However, as depicted in Fig. 6, aggregate skill scores of tree-based models are non-negative

[7] For reference: A skill score of 1 is equivalent to a 100% improvement in terms of the underlying metric.

[8] Note that the above plot does not imply that forecasting performance increases with forecast horizon (the opposite is true), solely that the examined

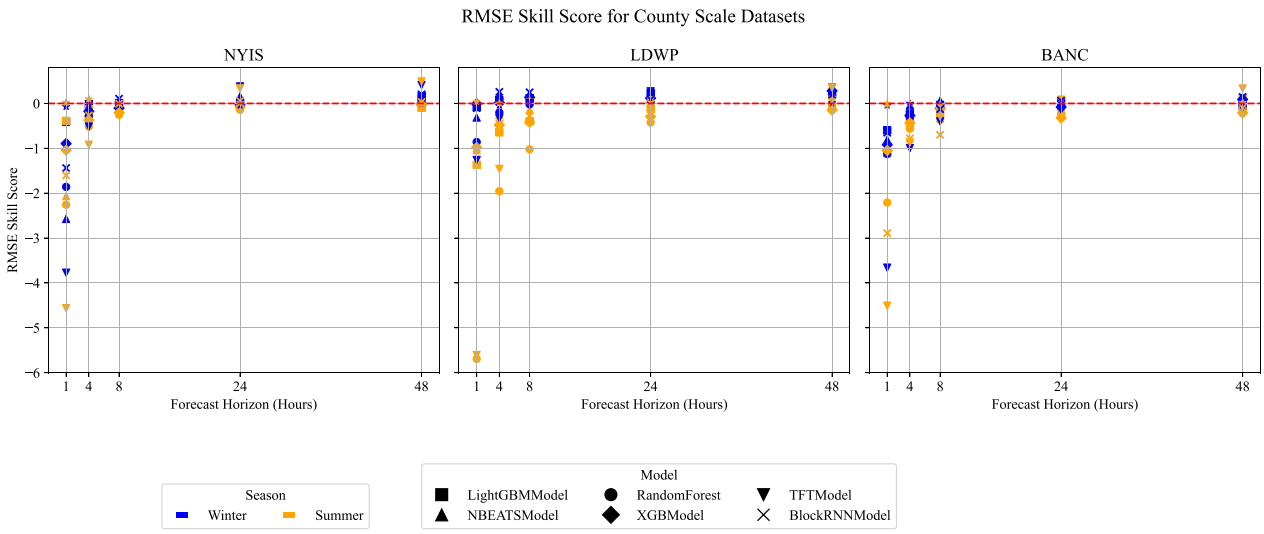


Fig. 8. RMSE Skill Scores of Models per Season [double column, print full colors].

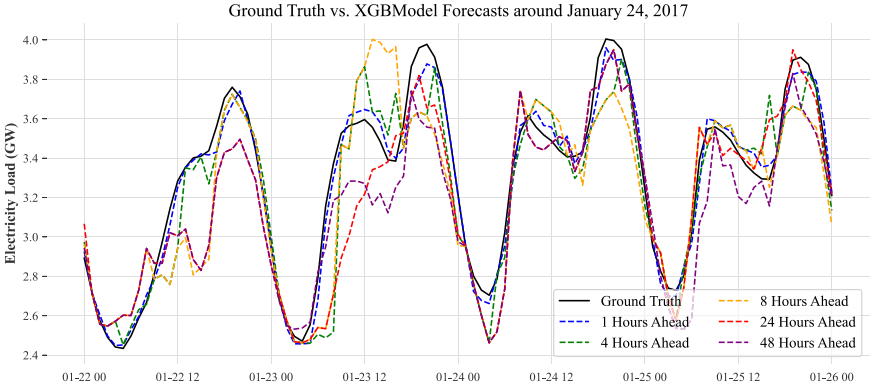


Fig. 9. XGBoost Forecasts on LDWP in Winter [print full colors].

for the other three spatial scales. Looking at Fig. 7 more closely, at the 24-hour horizon mark NNs outperform not only the benchmark, but also tree-based models. As further discussed in Section 3.3, this has can have significant implications for peak load management.

3.2. Zooming in: Specific models & the impact of season on performance

Expanding on Fig. 7, we now report RMSE skill scores of specific models, additionally differentiated by the two seasons in Fig. 8. Discernibly, compared to the benchmark, ML models tend to underperform during the summer months, with the TFT models decisively yielding the lowest skill score for all datasets.

The finding that the summer month is more difficult to forecast than the winter month is consistent across all spatial scales, except the building-level, where seasonal differences in the data are less pronounced. To further illustrate this phenomenon, we provide the forecasts of the XGBoost Model, trained on the Los Angeles (LDWP) county scale data, for various horizons during Summer and Winter in Figs. 9 and 10. In this example it is shown that during the summer the model under-estimates the peaks, while in winter load peaks are forecast accurately for all horizons.

The seasonal differences in performance are also evident by the error[9] distributions shown in Fig. 11. We find that models tend to bias towards under-predictions during summer, given by the distribution's long tail into the positive domain for all 6 models during summer, compared to the less skewed error distributions in winter. We also find this effect for shorter forecast horizons, though not as pronounced as shown in Fig. 11.

To underpin these findings quantitatively, we provide Table 5, which shows the skewness and Fisher's kurtosis (see DeCarlo, 1997) for summer and winter months grouped by spatial scale and algorithm type. With one exception, it is evident that error distributions during summer are significantly right-skewed, confirming systematic under-predictions, whereas during the winter month the predictions are left-skewed, however at a lower absolute value. Between NNs and tree-based algorithms it is evident that the forecasts of tree-based models are less skewed in absolute terms. However, from the error distributions in Fig. 11, it is also visible that the TFT model has the least skewed distribution out of all models for LDWP. Looking at kurtosis, *i.e.* the measure quantifying tail width of distributions, we also observe a higher values in summer than in winter. This indicates that models exhibit more frequent extreme forecasting errors during summer months than in winter. To verify that the hyperparameter optimization performed on the summer data did not lead to overfitting, we conducted an additional

models are better at forecasting load for longer horizons than the Linear Regression models.

[9] The errors are calculated in the standard fashion, *i.e.* by subtracting the forecast from the ground truth ($\epsilon = y - \hat{y}$).

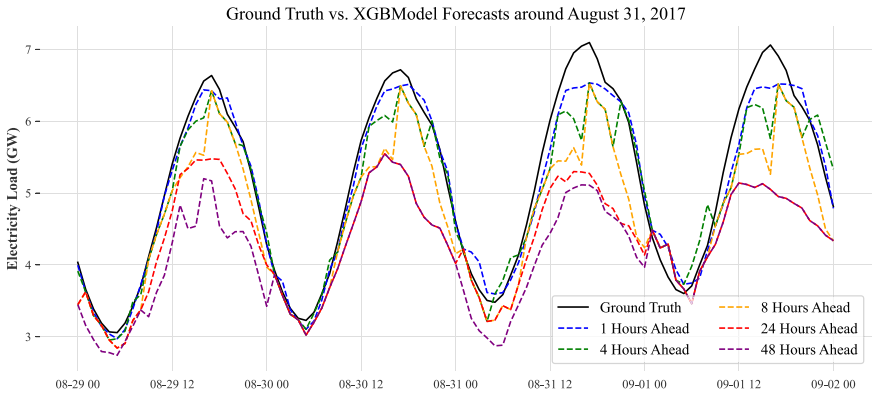


Fig. 10. XGBoost Forecasts on LDWP in Summer (on Validation Summer dataset) [print full colors].

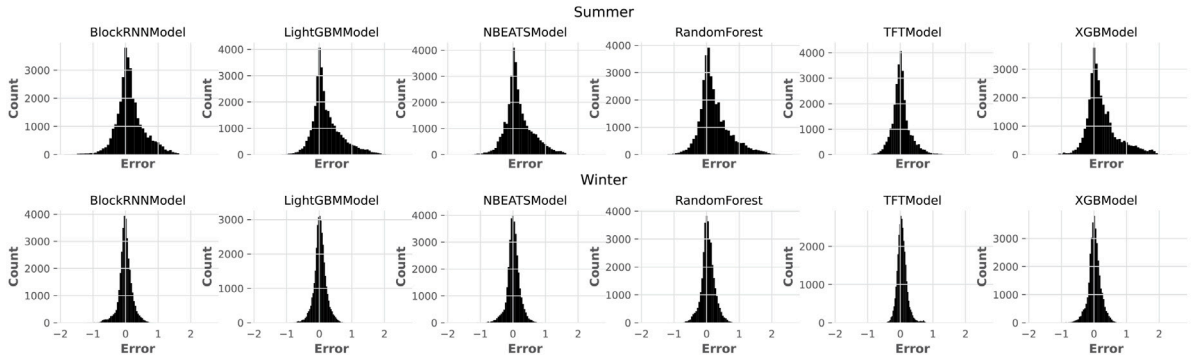


Fig. 11. Error distributions of all models on LDWP: Winter vs. Summer [double column, print gray scale].

Table 5
Comparison of Skewness and Kurtosis of Validation Summer, Test Summer, and Test Winter Forecast Errors. The inclusion of the unseen Test Summer confirms the robustness of the models outside the validation set.

Scale	Type	Skewness			Kurtosis		
		Val. Summer	Test Summer	Test Winter	Val. Summer	Test Summer	Test Winter
County	Neural network	0.83	0.88	−0.06	9.36	9.39	7.07
	Tree-based	0.12	0.13	0.04	−9.67	−10.10	9.85
Town	Neural network	0.45	0.47	−0.43	3.87	4.03	2.66
	Tree-based	0.36	0.38	0.16	−7.31	−7.64	8.70
Neighborhood	Neural network	1.48	1.55	−0.07	16.74	17.49	5.55
	Tree-based	0.56	0.58	−0.38	−9.82	−10.26	4.13
Building	Neural network	−0.01	0.02	−0.09	1.35	1.41	4.25
	Tree-based	0.17	0.18	−0.22	1.71	1.79	3.22

evaluation on a strictly unseen summer month. As shown in Table 5, the error distributions remain consistent between the validation and test sets. Furthermore, detailed performance metrics (RMSE and MAE) for this robustness check are provided in Appendix D (Table A.3), which confirms that the predictive accuracy on the unseen summer data is comparable to that of the validation set.

Based on this statistical performance analysis, we conclude that all examined algorithms are effective choices to forecast electricity load time series in the short-term. Models based on the Linear Regression algorithm remain competitive choices and are only significantly outperformed by more complex machine learning algorithms for horizons exceeding eight hours. Between the two examined model types, *i.e.* tree-based algorithms and NNs, the former yield lower error scores and less skewed error distributions. The season in which the forecasts are issued was identified as a major factor of performance, with examined models offering smaller skill improvements over the Linear Regression during summer than in winter.

3.3. Net load error results

The completed statistical comparison and error analysis of algorithms contributes to the academic discussion on the adequacy of machine learning algorithms for multi-step ahead forecasting, and specifically to the debate regarding the use of tree-based algorithms over NNs (see Hong et al., 2014; Miller et al., 2022; Januschowski et al., 2022; Shwartz-Ziv and Armon, 2022). In this sub-section we push the discourse further by evaluating models from an application-driven perspective, namely their effectiveness in forecasting peak load.

Firstly we provide an empirical example of the rationale of the newly proposed error score, regarding Fig. 12 which inspects the NLE

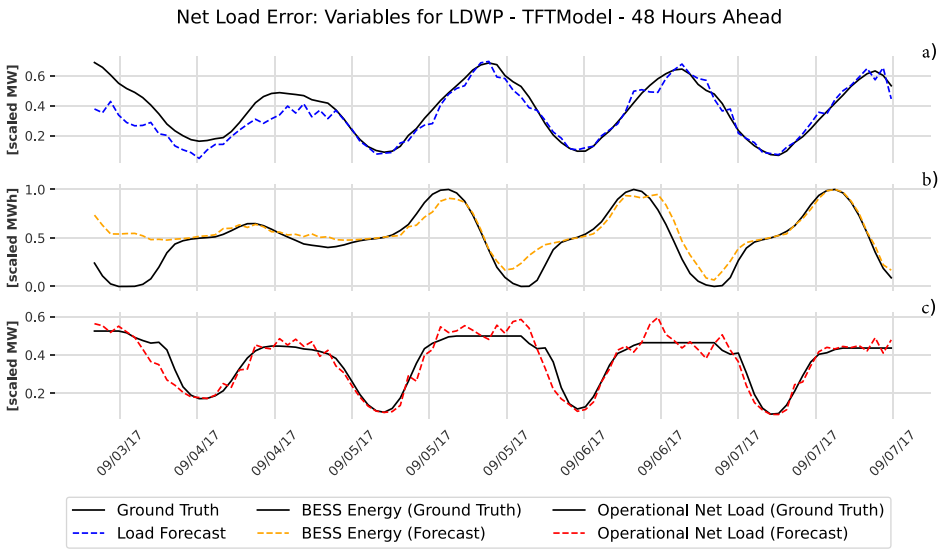


Fig. 12. Net Load Error Inspection (on Validation Summer dataset): (a) Electric Load Ground Truth (black) vs. Forecasts (blue), (b) BESS stored energy based on ground truth (black) vs. forecasts (yellow), (c) operational net load based on ground truth (black) vs. forecasts (red) [double column, print full colors].

variables for a 48-hour ahead forecast for the TFTModel on the LDWP dataset. In Fig. 12a) the truncated[10] load forecast and the ground truth are shown. Operating the BESS based on these timeseries to minimize the maximum load, yields the BESS energy as shown in Fig. 12(b). Unrolling the MPC, results in the operational net load shown in Fig. 12, which effectively shaves the peak load to below 0.6 for both the ground truth and the forecasts.

The NLE score arises directly from the fact that the operational net load based on forecasts yields higher daily peaks than the one based on the ground truth. Moreover, a closer look at Fig. 12(c), for example 09/06/17, reveals that the peaks in operational net load happen exactly at times of under-prediction in Fig. 12(c).

In Fig. 13, we provide a comparison between model type groups based on NLE. Unlike comparisons based on existing error metrics (see Figs. 7, 8), measuring forecasting performance with the NLE paints a different picture in two crucial aspects. Firstly, NN models consistently yield lower NLE scores than benchmark and tree-based models for county datasets. While this was already partially indicated in Fig. 8, where, for forecast horizons of more than 8 h, NNs overtook the others in performance marginally, the NLE further provides evidence that NNs are apt models to capture load peaks. This becomes especially clear when examining the results of the town-level datasets, where by statistical error metrics tree-based models were the most performant. In terms of NLE, however, NNs often yield the lowest score. As depicted in Fig. 13, there is a visible relationship between forecast horizon and score. Specifically, we observe that while scores derived from statistical error metrics increase with forecast horizon, NLE scores do the opposite. A final observation is that the difference between model groups increases when forecasting horizon increases.

4. Discussion of results & limitations

4.1. Tree-based models vs. Neural networks

In this study tree-based algorithms outperform NNs on the large majority of datasets for statistical error metrics. To interpret these results, two important differences between the models of these groups should be emphasized. Firstly, tree-based models tended to be smaller (fewer tunable parameters) than NNs. In the qualitative analysis we observed more volatile forecast trajectories for NNs. This is indicative of over-fitting, *i.e.* a lack of regularization in the training process for NNs. Indeed we find that the "dropout" hyperparameter in the NNs reached its upped bound (0.2) for 40% of the best performing models, while the most important regularization hyperparameter "reg_lambda" was only maxed out 20% for the tree-based models (see Appendix C). Here it should be noted that unlike the studied NNs, gradient-based tree-based algorithms, such as XGBoost and LightGBM, do not only have one regularization parameter and therefore offer more flexibility to strike the right balance between over- and under-fitting. Nevertheless, it could be worth-while repeating the hyperparameter tuning with NNs for a higher dropout upper-bound (e.g. 0.3) and see if the erratic forecasts persist. This was not performed in this study, due to time constraints.

The other salient difference between the NNs and tree-based models in this study is their implementation strategy. As highlighted in Table 2, all NNs were implemented in MIMO fashion, whereas tree-based algorithms are implemented as direct forecasting models. Referring to Ben Taieb et al. (2012) and Appendix A, MIMO methods calculate the loss for the entire forecast horizon, and inference is performed in one-shot, whereas direct methods train a separate model for each lead time within the horizon. The attribution of performance difference to NN vs. tree-based or MIMO vs. direct is therefore difficult. Still, as selecting and comparing readily available algorithms was a major objective of this study, and as in today's literature NNs are never implemented with direct strategies, we leave a less confounded comparison up to future research. It could be that a NN model implemented in either the direct or recursive fashion, could yield improved performance. More research is needed to confirm this, as apart from Ben Taieb et al. (2012), there has been limited work connecting the algorithmic differences to their performance.

4.2. The impact of seasonality on performance: Summers vs. Winters

Results showed that models generally under-performed on the test dataset in summer. Specifically, skill scores relative to the Linear Regression were observed to be consistently lower during the summer dataset, compared to the winter dataset. The error analysis corroborated these findings, showing heavily skewed error distributions during summer, while models seemed less biased in winter. At first glance, this is surprising, because the summer datasets were used as development

[10] Note that only for plotting purposes this is shown as the concatenation of $\hat{y}_{j=0;t} \forall t$. The actual MPC uses $\hat{y}_{j=0:H;t} \ \forall t \in T$.

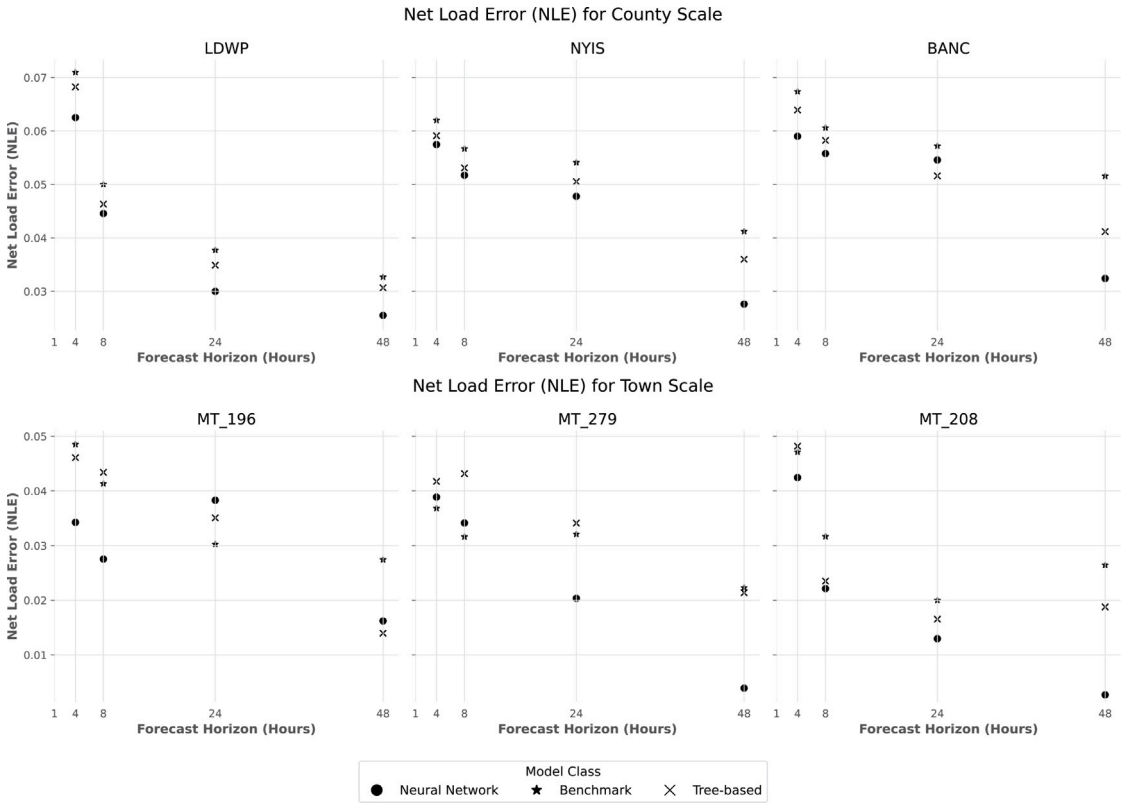


Fig. 13. Net load error scores for county and town level datasets for various forecast horizons [double column, print gray scale].

datasets during hyperparameter tuning, as detailed in Section 2.3.3. As a result, the hyperparameters should be better adapted to predict higher peaks. However, when examining the load measurement distributions of the training data, a possible explanation for this phenomenon can be found. Shown in Fig. 14, the load data for the county-level datasets BANC and LDWP are distributed strongly non-normally in summer, while the distribution in Winter is more symmetric, despite it having two modes. For the NYIS dataset, however, distributions in summer and winter are very similarly quasi-normal. This is directly reflected in Fig. 8, where summer and winter performance is similar for the NYIS datasets, and very distinct for the other two. Although, the distributions of other datasets (on town, neighborhood, building scales) do not follow the same shape as shown in Fig. 14, strong differences between summer and winter were found in 7 of the 9 remaining datasets studied datasets. As described in Section 2.3.2, a Boxcox transform was performed to normalize the data. This was performed on the entire dataset, *i.e.* the same transformation coefficient was used for summer and winter data, which, from this point of view, was a costly simplification. We therefore conjecture that it would be more effective to independently pre-process load time series data when distributions differ significantly, and only then train the forecasting model, or train a separate model per sub-timeseries.

4.2.1. Peak load management & NLE design

The lack of existing literature linking statistical model performance to real energy system-related applications motivated the development of the Net Load Error. Evaluating the forecasts on the 12 timeseries datasets in this way, provided important insights that run counter to those gleaned from statistical error metrics like RMSE and MAPE. Specifically, NNs, which were prone to over-fitting, still showed competence in peak load reduction. This can be explained by NNs' ability to reach adequate peak height (as shown in Fig. 5) and improved relative performance with increasing forecast horizon. Regarding this inverse relationship of horizon and score, the following consideration provides an explanation: Since the optimizer attempts to flatten the load for a given forecast series, a shorter horizon, has a higher probability of not including actual daily peak. In other words, a more myopic operation of the BESS in the NLE framework, causes sub-optimal set-points relative to the daily demand charge, which is used in ex-post evaluation. Re-examining Fig. 13, the difference of scores between model types is of a similar magnitude as between forecasting horizons, regardless of model type. This implies that stakeholders attempting to curb peak load are well-advised to use forecasts with long forecasting horizons, which could potentially even offset lower accuracy.

While these results provide additional nuance to the selection of models, they also elicit three important discussion points on the adequacy of the NLE on higher spatial scales (e.g., county or town).

1. **Evaluating Load Forecasts:** Common grid stability maintenance operations by grid operators differ significantly from the peak shaving objective posited in the NLE framework. In today's grids, operating reserves are procured through market-based mechanisms, drawing from a diverse array of technologies with variable marginal costs, capacities, and technical constraints. If operating reserves are procured dynamically (see De Vos et al., 2019), short-term load forecasts would not directly be used to schedule these resources, but would instead be used alongside non-dispatchable (renewable) generation forecasts to predict system imbalances. The application of the proposed metric on load forecasts instead of predicted imbalances for grid operations is thus the first limitation of the NLE.
2. **Asymmetry:** Linked to the previous point, we find that, since the NLE only penalizes positive peaks it is inherently asymmetric, and would need adjustment to evaluate system imbalances soundly. Specifically, in its current form, the metric penalizes under-predictions more than over-predictions. In this sense, load forecasts with NNs, which exhibited a tendency towards over-prediction, would likely fare less effective in the context of

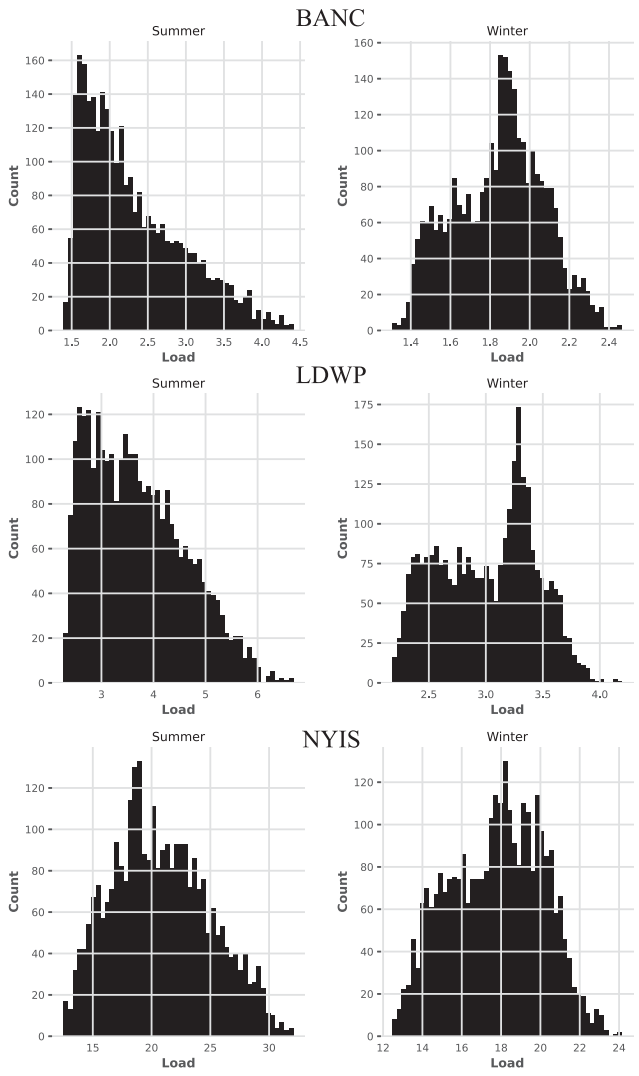


Fig. 14. Data distributions for county-scale datasets (Summer vs. Winter)[single column, print gray scale].

imbalance peak prediction, than suggested by the presented analysis of the NLE. While prior research (see Koch and Hirth, 2019; Koch, 2022) suggests that historical imbalances have exhibited skewness towards shortage in some bidding zones, efficient markets tend to evolve towards symmetric imbalances over time. In this scenario the potential for strategic bidding or biased forecasting models would be mitigated.

3. **Spatial Aspects:** Explicit peak load shaving, while relevant for grid operators during specific periods (such as extreme weather events), is primarily pertinent regarding the capacities of specific power lines and appliances. In other words, there is a locational aspect to peak load management that the NLE design overlooks. Decoupling load time series from physical network constraints, as implemented in the NLE framework, simplifies the model but deviates from real-world applications, and is the third major limitation of the proposed metric. However, we emphasize that any practical application-driven metric must necessarily abstract to some degree for tractability and comparability.

From this discussion it is clear that the NLE system setup only directly resembles situations comparable to a single consumer with an owned flexibility (e.g., a household with a BESS), optimizing against a demand charge tariff. However, in this study we have used it to make statements about load time series at higher levels of spatial aggregation. We argue that, while the proposed metric is imperfect in its representation of real grid operation activities, even for datasets on the highest spatial scales (county-level) the expressivity of the proposed error score is still notably higher than existing Euclidean metrics (e.g. RMSE, MAPE). This is because the double-penalty effect is effectively circumvented by the recursive nature of the model predictive controller. Nevertheless, we see substantial opportunities in developing the metric further, and hope to open the seam for the development application-driven metrics by the research community.

4.3. Further limitations

We now highlight some general limitations inherent to the design of this study. The first limitation of this work is that to include all relevant spatial scales, potentially not enough time could be allocated to the various hyperparameter optimization runs. For instance, it could be the case that certain algorithms would clearly outperform others across multiple scales if more computational resources had been dedicated to their calibration. With that being said, as described in Section 4.1, even though all algorithms had an equal number of training runs during hyperparameter tuning, in terms of time spent neural networks model took much longer to train and still yielded lower performance than tree-based models. A second limitation of this study is that due to the large amounts of models, datasets and horizons considered, the extent of the presentation of results is inherently limited. We attempted to provide both birds-eye-view results, e.g. comparing neural networks to tree-based, and nuanced observations on specific datasets, however the reader should be aware that the full breadth of insights can only be obtained when going through the WandB results repository, which is offered as supplementary material. Next, we find the use of measured weather data, instead of forecast weather data as a limitation to our work. While only outdoor ambient temperature was used as a variable, which is usually forecast quite accurately, an additional source of uncertainty would further lower the perceived performance. We refer to previous work by Wang et al. (2021), who demonstrated that including weather predictions during the training process increased real-life performance.

5. Conclusions, recommendations & future work

This paper presents a comprehensive analysis of machine learning algorithms for short-term electricity load forecasting across various spatial scales. The study's in-depth benchmarking of six machine learning algorithms over twelve electricity load time series reveals significant insights into their performance and applicability. Tree-based models generally outperformed neural networks in terms of statistical metrics, exhibiting better overall ability in capturing load patterns, though they tended to under-predict peak loads. Neural networks demonstrated strengths in peak load prediction quantified by the Net Load Error, but overestimated low-load periods and noticeably failed to perform well in terms of statistical error metrics. Seasonality emerged as a critical factor of performance, with models under-performing during summer months due to strongly skewed data distributions. This observation underscores the necessity for algorithms to adapt to seasonal variations, especially for applications in grid operation where accurate peak load prediction is crucial. The introduction of the novel "Net Load Error" score is a significant contribution of this work. By quantifying the relationship between forecast errors and peak load reduction, it bridges the gap between theoretical model accuracy and practical economic effectiveness in energy systems. This error score provides a more nuanced understanding of a model's applicability in real-world scenarios, particularly in managing peak load and its associated costs.

Derived from our results we provide three concrete recommendations to stakeholders of the electricity system (e.g. grid operators), which we hope will provide guidance in implementing forecasting strategies in the real-world.

1. **Develop a reliable baseline model:** Results of this study show that for certain spatial scales and forecast horizons, sophisticated machine learning models do not produce significant performance gains when compared to a simple Linear Regression. Practitioners are therefore well-advised to first implement a simple and interpretable model, and see if it is fit for the given application.
2. **Choose or develop multiple error metrics that capture the objective of the application of the forecasting model:** The proposed Net Load Error is an example specifically tailored for peak load reduction. Custom metrics as such yield additional insights, which may run counter to the results from conventional metrics.
3. **Examine seasonal differences in the timeseries data:** Despite considering outdoor temperature as a feature, model performance may vary greatly between the different seasons. To avoid this consider analyzing the data distributions by season, performing diligent preprocessing, and training a model per distinct distribution (e.g., one for the summer months and one for the winter months).

Overall, this study enhances the understanding of the efficacy of various machine learning techniques in short-term electricity load forecasting. It highlights the importance of considering both statistical accuracy and practical application in model selection, paving the way for more informed and effective use of these technologies in the energy sector. Building on this work, several promising future research directions emerge.

First, to address the challenge of seasonal performance degradation, future work could investigate more sophisticated data preprocessing methods. Rather than applying a single transformation to an entire dataset, one could train separate models for distinct seasonal periods (e.g., summer and winter months) or implement adaptive preprocessing techniques that account for the non-normal data distributions observed during high-demand periods like summer. Additionally, a formal comparison between Multi-Input-Multi-Output (MIMO) and direct forecasting strategies for neural networks could clarify whether the observed performance differences stem from the algorithm type or the implementation strategy.

Second, while this study used historical weather data, a significant enhancement would be to incorporate weather *forecasts* as input features during model training. This would more accurately simulate real-world operational conditions and account for the uncertainty inherent in weather predictions, providing a more realistic assessment of model performance. Integrating other diverse data sources, such as consumer behavior patterns, could further improve forecasting accuracy, particularly during critical peak load periods.

Third, the proposed Net Load Error (NLE) metric opens a new avenue for application-driven evaluation. A crucial next step is to adapt and apply this metric to system imbalances, which involves assessing the net difference between load and renewable generation forecasts against their ground truth values. This would require modifying the NLE to be symmetric, penalizing both over-predictions and under-predictions to reflect the costs associated with both positive and negative imbalances in electricity markets. Developing other tailored, application-driven scores for different energy sector use cases, such as frequency regulation or optimal dispatch, would also provide deeper insights into the practical utility of various forecasting models. Furthermore, the NLE or similar application-driven metrics could be used to compared a wider range of model categories and architectures (*e.g.,* as Support Vector Regression).

Lastly, expanding the geographical scope of the analysis would bolster the generalizability of the findings. Future studies should include diverse regions with varying climatic conditions, consumption patterns, and levels of renewable energy penetration to test the robustness of the models and recommendations. Such a broad analysis would contribute significantly to the wider application and operationalization of these advanced forecasting methods in real-world grid operations globally.

CRediT authorship contribution statement

Nikolaus Houben: Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Data curation, Conceptualization. **Miguel Heleno:** Writing – review & editing, Methodology. **Han Li:** Writing – review & editing, Methodology, Conceptualization. **Tianzhen Hong:** Writing – review & editing, Supervision, Conceptualization. **Hans Auer:** Writing – review & editing, Supervision. **Amela Ajanovic:** Writing – review & editing, Supervision. **Reinhard Haas:** Writing – review & editing, Supervision.

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.

Appendix A. Mathematical details of machine learning algorithms

Linear Regression is a classical statistical method for modeling linear relationships between variables, and it has been in use for well over a century. While it is neither a decision tree-based method nor a neural network, this work includes its prediction as a benchmark (see Section 2.4). The rudimentary mathematical expression is given by Eq. (14):

$$\hat{y} = \beta + \beta_1 x_1 + \beta 2 x_2 + \cdots + \beta_n x_n \tag{14}$$

where,

$\hat{y}$ = target vector (predictions)
x_i = feature vectors (variables)
β_i = learnable parameters

The training process consists of finding the optimal combination of parameters β to linearly transform the feature vectors into the target vector. This can be achieved by solving a system of n equations, or as performed in sklearn's implementation, through gradient descent (Pedregosa et al., 2011).

Random Forest (Breiman, 2001) is a decision tree ensemble algorithm, which for the context of forecasting means that predictions are obtained by the averaging of K functions (Classification and Regression Trees) as shown in Eq. (15).

$$\hat{y}_j = \sum_{k=1}^{K} p_k(x_j) \; ; \; p_k \in P \tag{15}$$

where,

p_k = a classification and regression tree (CART)
P = the space of regression trees
x_j = vector of features for a single example at forecast timestep j

For each regression tree p_k the covariate space is recursively partitioned based on discrete or Boolean values into subsets to minimize a cost criterion. Fig. A.1 exemplifies this, where the top-most node – the *parent node* – splits the data into two more groups, the *child nodes*. This process is repeated until the splits are found that optimally separate the data based on the cost criterion and other hyperparameters, such as maximum tree depth. The order of the features and the numerical value by which the data are split are systematically chosen to minimize the gap between the forecasts and the observed values. If a split for a particular branch does not reduce the cost criterion (by a significant margin), the process is stopped. In this way, terminal nodes, or *leaf nodes* are created, which can later be used for forecasting.

The random forest algorithm is trained through *bagging* - short for *boostrap aggregation* - which involves creating subsets of the original dataset by randomly selecting samples with replacement. Each subset

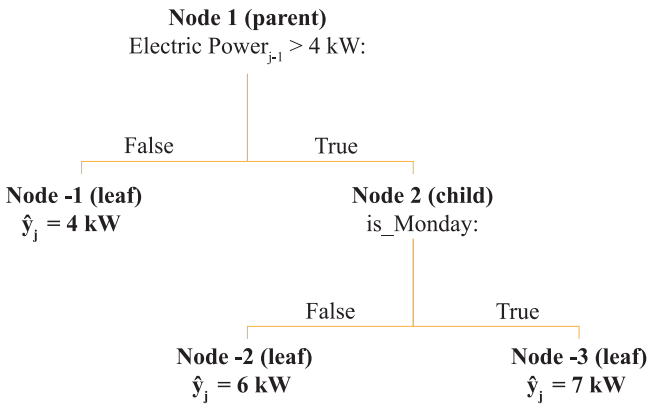


Fig. A.1. Regression tree p_k for time series forecasting: a simple example of how a regression tree would split features for electrical load forecasting. [single column; print full colors].

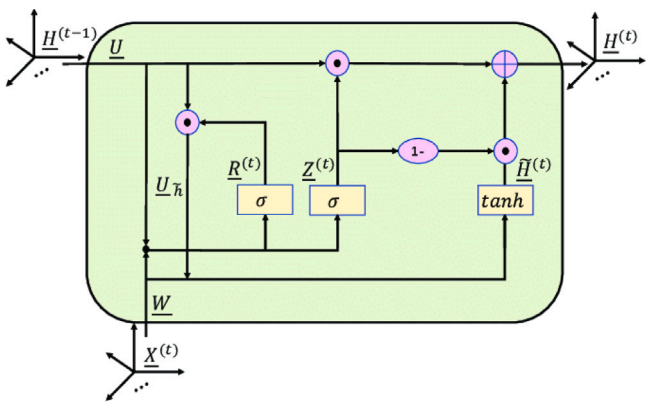


Fig. A.2. Architecture of a Gated Recurrent Unit taken from Wu et al. (2021) [single column; print full colors].

is used to train an individual tree (Breiman, 1996). The objective of this method is to introduce diversity among the trees and improve the model's ability to generalize.

XGBoost & LightGBM Like Random Forest, Extreme Gradient Boost (XGBoost) algorithm (Chen and Guestrin, 2016) and the Light Gradient Boosting Machine (LightGBM) are tree-based algorithms. However unlike Random Forest, which builds trees independently, these two algorithms build trees sequentially, where each tree is built to correct the errors of the next. Furthermore, XGBoost and LightGBM leverage a technique called *gradient tree-boosting*. In the XGBoost algorithm, the cost criterion used to learn the tree set P is a *regularized* cost function given by Eq. (16).

$$L(\phi) = \sum_j l(\hat{y}_j, y_j) + \sum_k \Omega(p_k) \tag{16}$$

Here l is a differentiable loss function (e.g., mean squared error) that measures the distance between the forecast $\hat{y}_j$ and the target y_j. Furthermore, the regularization term $\Omega(f)$ is given by Eq. (17), and penalizes the complexity of the model. This is achieved by considering the number of leaf nodes T and the dimensionality of their respective weights w.

$$\Omega(p) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{17}$$

where,

γ = pruning hyperparameter
λ = smoothing hyperparameter
T = number of leaf nodes in a tree
w = leaf weights

Gradient Tree Boosting: In XGBoost, the Eq. (16) is solved in an iterative manner. Formally, let $\hat{y}_j^z$ be the forecast of the jth training example at the zth iteration. Implied by *boosting*, the algorithm greedily adds a component f_z to minimize the objective given by Eq. (18).

$$L^z = \sum_j l(\hat{y}_j^{z-1}, y_j + p_z(x_j)) + \Omega(p_z) \tag{18}$$

In practice, the Eq. (18) is approximated by bootstrapping the first and second-order derivatives of the loss function of the previous iteration to the next. Formally, the Eq. (19) gives:

$$L^z \simeq \sum_j \left[l(\hat{y}_j^{z-1}, y_j) + g_j p_z(x_j) + \frac{1}{2} h_j p_z^2(x_j) \right] + \Omega(p_z) \tag{19}$$

where,

$$g_j = \partial_{\hat{y}^{z-1}} l(\hat{y}_j^{z-1}, y_j)$$
$$h_j = \partial^2_{\hat{y}^{z-1}} l(\hat{y}_j^{z-1}, y_j)$$

Omitting the constant terms and defining $I_e = \{j | f(x_j) = e\}$ as the set of all data samples of leaf node e, Eq. (19) is rewritten as:

$$L^z \simeq \sum_{j=1}^{T} [(\sum_{j \in I_e} g_j) w_e + \frac{1}{2} (\sum_{j \in I_e} h_j + \lambda) w_e^2] + \gamma T \tag{20}$$

To find the optimal weight of a leaf e, the XGBoost algorithm differentiates Eq. (20) w.r.t w_e:

$$w_e^* = -\frac{\sum_{j \in I_e} g_j}{\sum_{j \in I_e} h_j + \lambda} \tag{21}$$

Substituting w_e in Eq. (20) with Eq. (21), the quality of a fixed tree can be calculated according to Eq. (22).

$$L^z \simeq \tilde{L}^{(z)}(p) = -\sum_{e=j}^{T} \frac{(\sum_{i \in I_e} g_j)^2}{\sum_{j \in I_e} h_j + \lambda} + \gamma T \tag{22}$$

Incremental Algorithm: As it is intractable to enumerate all possible trees and then select the one with the minimum $\tilde{L}(f)$, a greedy algorithm that starts from a single parent node and iteratively adds branches is used instead. Let I_L and I_R be the training sample sets of left and right nodes after a split. With $I = I_L \cup I_R$, the incremental loss reduction, or the gain G, can be calculated by Eq. (23)

$$G = -\frac{1}{2}\left[\frac{(\sum_{j \in I_L} g_j)^2}{\sum_{j \in I_L} h_j + \lambda} + \frac{(\sum_{j \in I_R} g_j)^2}{\sum_{j \in I_R} h_j + \lambda} - \frac{(\sum_{j \in I} g_j)^2}{\sum_{j \in I} h_j + \lambda} \right] - \gamma \tag{23}$$

While LightGBM follows similar principles as XGBoost, the major difference between the two algorithms is the tree growth strategy. Specifically, LightGBM chooses the leaf node that has the maximum reduction in the loss function (or impurity) at each step. The algorithm selects the best split based on the feature that contributes the most to reducing the loss, leading to a more aggressive search for informative features. This approach can lead to deeper trees with fewer leaf nodes. For further reading regarding the mathematical difference compared to XGBoost, the original manuscript by Ke et al. (2017) is recommended.

Gated Recurrent Unit (GRU) is a Deep Neural Network, specifically a recurrent neural network (RNN). These were developed to model sequential, time-dependent data. Since RNNs' inception in the late 1980s (see Elman, 1990), the Long Short Term Memory (LSTM) model by Hochreiter and Schmidhuber (1997) has been a popular recurrent architecture. With the introduction of gates, LSTM addressed the issue of vanishing gradients and thus improved the gradient-flow during backpropagation. The Gated Recurrent Unit (GRU) (Chung et al., 2014) is a simpler variant of the LSTM, omitting the memory gate.

As shown in Fig. A.2, each GRU cell takes as input the hidden state S_{t-1} from the previous cell and the input X_t. It computes the hidden state S_t based on a combination of gating mechanisms: the reset gate

R_t and the update gate Z_t. These gates determine how the information flows through the cell and which information is stored or forgotten.

$$R_j = \sigma(W_r \cdot [S_{j-1}, X_j] + b_r)$$

$$Z_j = \sigma(W_z \cdot [S_{j-1}, X_j] + b_z)$$

$$S_j = (1 - Z_j) \odot \tanh(W \cdot [R_j \odot S_{j-1}, X_j] + b) + Z_j \odot S_{j-1}$$

where,

R_j = reset gate at time j
Z_j = update gate at time j
S_j = hidden state at time j
X_j = input at time j
W_r = weight matrix for the reset gate
W_z = weight matrix for the update gate
W = weight matrix for the hidden state
b_r = bias term for the reset gate
b_z = bias term for the update gate
b = bias term for the hidden state
σ = sigmoid activation function
tanh = hyperbolic tangent function
$\odot$ = element-wise multiplication

This architecture allows GRUs to learn long-term dependencies while being more efficient than LSTMs due to the reduced number of gates. The gating mechanisms, specifically the reset and update gates, allow the GRU to capture various time-scale patterns in the data, making them suitable for tasks like time-series prediction.

Note that in most Recurrent Neural Network (RNN) implementations for time series forecasting, the model is designed to predict one step into the future and then relies on its own predictions to forecast further. This method is known as the *recursive* multi-step ahead forecasting (see). However, in the work presented here, a different approach for the GRU is applied, termed the "BlockRNN", which represents a *MIMO* formulation of the forecasting task. The BlockRNN does not recursively unroll into the future up to the desired forecast horizon. Instead, at prediction time j, the hidden layer S_j is directly mapped to the forecast using a final linear layer, as shown in Eq. (24).

$$\hat{y}_{j:j+H} = W_f \cdot S_j + b_f \tag{24}$$

where,

$y_{j:\hat{j+H}}$ = forecast vector with H elements
W_f = final weight matrix of size $(batch\_size, hidden\_size, H)$
b_f = final bias term of dimensions $(batch\_size, H)$

N-BEATS is a deep neural network architecture that was specifically developed for time series forecasting, presented in the context of beating common benchmarks on the M3, M4 and TOURISM datasets (Oreshkin et al., 2019). It has also been effectively used in energy forecasting (e.g., Putz et al., 2021; Oreshkin et al., 2021). Although it is not a recurrent architecture, it achieves remarkable depth through stacking multiple blocks of fully-connected neural networks, each predicting the future outputs and their contribution to the decomposition of the input (backcast). As shown in Fig. A.3, each block consists of a sequence of fully connected layers, each using the historic residual of the previous as input. As the authors write in Oreshkin et al. (2021): "the architecture runs a residual recursion over the entire input window and sums block outputs to make its final forecast".

Formally, assume there are R residual blocks with L hidden layers each, and l and r denote the superscripts of a particular layer in a block. The fully connected layer with weights $W^{r,l}$ and bias term $b^{r,l}$ can be expressed by Eq. (25).

$$FC^{r,l}\left(h^{r,l-1}\right) = ReLu\left(W^{r,l} h^{r,l-1} + b^{r,l}\right) \tag{25}$$

Then the recursive operation of N-BEATS is given by Eq. (26).

$$\begin{aligned} x^r &= \mathrm{ReLu}[x^{r-1} - \hat{x}^{r-1}], \\ h^{r,1} &= FC^{r,1}\left(x^r\right), \\ &\vdots \\ h^{r,L} &= FC^{r,L}\left(h^{r,L-1}\right), \\ \hat{x}^r &= B^r h^{r,L} \end{aligned} \tag{26}$$

Note that even though the algorithm performs a recursive step, it strictly belongs to the MIMO type as...

Temporal Fusion Transformer (TFT) by Lim et al. (2021) is a variant of the original transformer neural network (as presented in Vaswani et al. (2017)). It combines recurrent neural network layers of the GRU (Chung et al., 2014) and the self-attention mechanism.

The architecture, shown in Fig. A.4, consists of the following differentiable building blocks:

1. *Gated Residual Network (GRN)*: Multi-layer perceptron (see Hornik et al., 1989) with two layers and a residual connection (see He et al., 2016). The residual connection allows the network to skip over any unused components of the architecture. As shown in Fig. A.4, the GRN component is used throughout the architecture, *i.e.* in the Variable Selection Network and before and after the self-attention layer.
2. *Variable selection networks (VSN)*: This component combines multiple instances of the GRN. Particularly, a given variable ξ is flattened over all considered timesteps to Ξ to produce weights v_{χ_t}, shown in Eq. (27).

   $$v_{\chi_t} = softmax(GRN_{v_\chi}(\Xi_t)) \tag{27}$$

   As the decoder rolls out over time, each feature vector $\xi^{(j)}$ is input into its own GRN and weighted with the current v_{χ_t}. Note that lags $x_{t-k:t}$ and future features $\chi_{t:t+H}$ have their own VSNs, but are shared weights across all timesteps. As static features are not used in this work, the context vector c can be ignored in Fig. A.4.
3. *Temporal Fusion Decoder*: This component combines LSTM encoder layers (see Hochreiter and Schmidhuber, 1997) with a modified self-attention layer (see Vaswani et al., 2017). The idea of the authors in Lim et al. (2021) is that the self-attention layer, which is effective in modeling long-term dependencies in a sequence, can benefit from the localized enhancement of the LSTM. The authors use the scaled dot-product attention, as given by Eq. (28).

   $$Attention(Q, K, V) = softmax(Q, K^T / \sqrt{d_{attn}})V \tag{28}$$

   where,

   Q, K, V = linear projects of the transformed feature vectors $\bar{\xi}$ called queries, keys and values.
   d_{attn} = width dimension of the key tensor, which is a model choice

   As proposed in the original transformer paper, the attention mechanism is framed in a multi-head setup (see Vaswani et al., 2017), however, here with the additive aggregation of attention heads to improve interpretability, see Eq. (29).

   $$MultiHead(Q, K, V) = \frac{1}{m_H} \sum_{h=1}^{m_H} Attention(QW_Q^h, KW_K^h, VW_V) \tag{29}$$

   where,

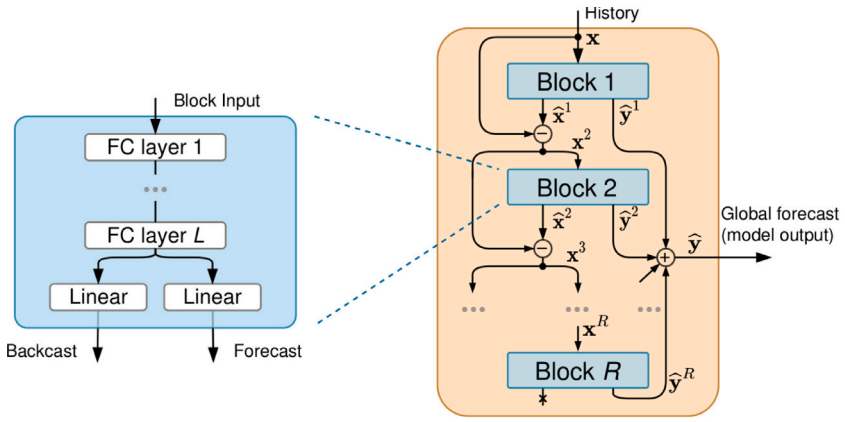


Fig. A.3. Architecture of N-BEATS with exogenous feature support taken from Olivares et al. (2023) [double column; print full colors].

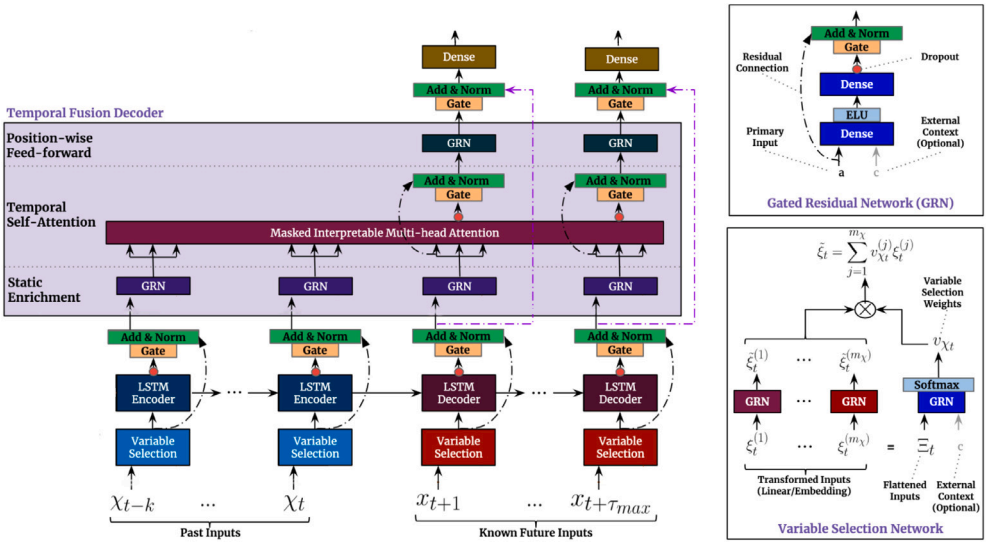


Fig. A.4. Architecture of the deterministic temporal fusion transformer, adapted from Lim et al. (2021) [double column; print full colors].

Q, K, V = linear projects of the transformed feature vectors $\tilde{\xi}$ called queries, keys and values.

W_Q, W_K, W_V = weights to combine query, keys and value projects across attention heads.

m_H = number of attention heads

Note that while the original implementation of the TFT is designed for probabilistic forecasting, this work uses the deterministic version.

Details on the implementation of forecasting algorithms

This study evaluates forecasts for time horizons up to 48 h at an hourly temporal resolution. As such, all the algorithms under consideration are implemented for the task of multi-step ahead forecasting. Formally, this task is about predicting the subsequent H values, represented as $[\hat{y}_{j+1}, \dots, \hat{y}_{j+H}]$, from a historical time series $[y_1, \dots, y_T]$ containing T observations, where $H > 1$ specifies the forecast horizon. There are several methodologies for multi-step ahead forecasting documented in the literature, among which the recursive, multi-input-multi-output (MIMO), and direct strategies are foundational. Although machine learning algorithms can, in theory, be adapted to any forecasting approach, the implementations of algorithms in this study are either the direct and MIMO approach (as displayed in Table 2).

The direct approach involves training separate sub-models for each lead-time h within forecast horizon H. In other words, if the aim is to forecast H steps ahead, then H models would be constructed, each producing a scalar value for a specific lead-time h. Formally, for a given lead-time h and a look-back window l, the sub-model f^h is formulated as shown in Eq. (30)

$$f^h(y_{j-l}, \dots, y_j) \rightarrow \hat{y}_{j+h} \tag{30}$$

To arrive at the multi-step ahead forecast for horizon H, the output of each sub-model is concatenated, as given by Eq. (31):

$$[\hat{y}_{j+1}, \hat{y}_{j+2}, \dots, \hat{y}_{j+H}] = [f^1(y_{j-l}, \dots, y_j), f^2(y_{j-l}, \dots, y_j), \dots, f^H(y_{j-l}, \dots, y_j)] \tag{31}$$

In contrast, the MIMO approach trains a single model to predict all required future time steps simultaneously. To forecast up to a horizon of H, the model is designed to output all values from $[y_{j+1}, \dots, y_{j+H}]$ at once. The MIMO approach can be represented as:

$$f(y_{j-l}, \dots, y_j) \rightarrow [\hat{y}_{j+1}, \dots, \hat{y}_{j+H}] \tag{32}$$

Note that in this study, all neural networks were executed as MIMO forecasting models, while tree-based models were implemented as direct forecasting models.

Appendix B. Details on the Net Load Error

Mathematical details

The (dis-)charging action $u_{t,j=0}$ at each timestep $t+1$ is obtained by solving the constrained linear optimization problem described here.

Note that for readability the index t has been omitted in the following equations. The weighted cost function given by Eq. (33) is composed of a term for the cost of the peak and a term for the terminal cost, which is assumed as the squared deviation between the final and the initial SOC of the BESS. A weight value for the terminal cost is assumed to be $w_H = 0.1$.

$$Cost = C_{peak} \cdot (1 - w_H) + C_H \cdot w_H = p^h \cdot (1 - w_H) + (SOC_{init} - SOC_{j=H})^2 \cdot w_H \tag{33}$$

where,

C_H = terminal costs
C_{peak} = peak costs
w_H = terminal cost weight
p^h = horizon peak

Note that the horizon peak p_h is set as an upper bound of the optimized net load $l^{opt}_{j=0:H}$, as shown in Eq. (34).

$$p^h \geq l^{opt}_{j=0:H} \tag{34}$$

The overall energy balance, relating the optimized net load to the charging action is given by Eq. (35)

$$\hat{y}_j = l^{opt}_j + u_j \tag{35}$$

The terminal costs in the objective function represent the future value of stored energy at the end of the horizon. When imposed, this cost guarantees that there are no net-energy gains within each optimization horizon. This terminal costs term is implemented by the squared difference of the initial state-of-charge SOC_{init} and the terminal state-of-charge $SOC_{j=H}$ (see Risbeck and Rawlings, 2019; Shi et al., 2022 for an alternative formulations).

For simplicity, the BESS is modeled linearly, as an energy storage without losses regarding storage and round-trip efficiency. Depending on the timestep t the initial energy in the BESS for a given optimization run is given by Eqs. (36), (37).

$$E_{j=0} = E_{init} \quad t = 0 \tag{36}$$

$$E_{j=0;t+1} = E_{j=1;t} \quad \forall t > 0 \tag{37}$$

For $j > 1$, the update constraints are given by Eq. (38), while the remaining necessary technical constraints are given by:

$$E_{j+1} = E_j + u_j \quad \forall j > 1 \tag{38}$$

$$-C_{rate} \cdot \overline{BESS} \leq u_j \leq C_{rate} \cdot \overline{BESS} \tag{39}$$

$$0 \leq E_j \leq \overline{BESS} \tag{40}$$

where,

E_j = energy in the BESS ($SOC_j \cdot \overline{BESS}$)
$\overline{BESS}$ = capacity of the BESS
u_j = (dis-)charging action

NLE data assumptions

The NLE metric depends as much on the forecast errors as on the assumptions regarding the stylized energy stem. The goal in devising the system setup, was to provide a transparent and simple energy system that is transferable to real grid operations. To make results comparable across datasets, the pricing scheme and BESS specifications are parameterized by the scaled load time series (see Eq. (41)).

$$y^{scaled}_t = \frac{y_t - min(y_t)}{max(y_t) - min(y_t)} \tag{41}$$

Based on this scaled load time series the price signals and storage are parameterized as a daily demand charge $P_{DC} = 1$, a BESS C-Rate and Capacity of 1.

The linear optimization problem was solved with the CPLEX solver, leading to a total calculation time of the net load error of 2 min for a month of forecasts in hourly temporal resolution.

Table A.1
Dropout values for the best performing neural network models.

Scale	Location	GRU	N-BEATS	TFT
County	Los_Angeles	0.0	0.0	0.2
	New_York	0.2	0.1	0.2
	Sacramento	0.2	0.0	0.0
Town	town_0	0.2	0.1	0.2
	town_1	0.0	0.0	0.0
	town_2	0.1	0.0	0.2
Neighborhood	neighborhood_0	0.1	0.2	0.2
	neighborhood_1	0.1	0.2	0.1
	neighborhood_2	0.0	0.0	0.1
Building	building_1	0.0	0.2	0.1
	building_2	0.2	0.1	0.2
	building_3	0.1	0.1	0.2

Table A.2
Mean training runtimes per time series. Note that the reported times are dependent on the used hardware and hyperparameters.

Algorithm	Hyperparameter setting	Mean training time
Tree-based models		
Random Forest	n_estimators = 100	4 s
	n_estimators = 1000	32 s
XGBoost	n_estimators = 100	5 s
	n_estimators = 1000	38 s
LightGBM	n_estimators = 100	2 s
	n_estimators = 1000	11 s
Neural network models		
GRU (BlockRNN)	batch_size = 32	3 m 15 s
	batch_size = 256	1 m 25 s
N-BEATS	batch_size = 32	4 m 10 s
	batch_size = 256	2 m 05 s
TFT	batch_size = 32	11 m 45 s
	batch_size = 256	6 m 30 s

Appendix C. *Weights & biases* repository supplementary

The complete hyperparameters tuning runs, training runs and results of this study are available in the various *Weights & Biases* repositories. In these repository, we have opted for different acronyms mapping to the dataset names as follows: 'LDWP' to 'Los_Angeles', 'BANC' to 'Sacramento', 'NYIS' to 'New_York', 'MT_196' to 'town_0', 'MT_279' to 'town_1', 'MT_208' to 'town_2', 'Bull' to 'neighborhood_0', 'Hog' to 'neighborhood_1', 'Bobcat' to 'neighborhood_2', 'Be_Sandy' to 'building_0', 'Be_Millie' to 'building_1', and 'Co_Joel' to 'building_2' (see Table A.1).

Computational cost

See Table A.2.

Appendix D. Supplementary performance evaluation on unseen summer data

To address potential selection bias regarding the use of summer months for hyperparameter tuning, we evaluated the models on an additional summer month that was not used during the development phase. Table A.3 compares the RMSE and MAE of the original validation month (Summer Val.) with the unseen test month (Summer Test). The similarity in metrics between the validation and unseen test sets confirms that the models have not overfit to the validation data and maintained robustness.

Data availability

The data is openly available and linked to in the manuscript.

Table A.3
Robustness check: Comparison of RMSE and MAE between the original Summer Validation set and the unseen Summer Test set.

Dataset	Algorithm type	RMSE		MAE	
		Summer Val.	Summer Test	Summer Val.	Summer Test
LDWP (County)	Tree-based	0.27	0.28	0.18	0.19
	Neural network	0.28	0.29	0.19	0.20
MT_196 (Town)	Tree-based	2.23	2.34	1.70	1.73
	Neural network	2.32	2.44	1.77	1.91
Hog (Neighborhood)	Tree-based	276.11	289.92	197.23	207.09
	Neural network	287.61	301.99	205.45	217.72
Co_Joel (Building)	Tree-based	353.50	371.18	274.65	284.38
	Neural network	368.23	386.64	286.09	298.39

References

Bahdanau, Dzmitry, Cho, Kyunghyun, Bengio, Yoshua, 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Behm, Christian, Nolting, Lars, Praktiknjo, Aaron, 2020. How to model European electricity load profiles using artificial neural networks. Appl. Energy 277, 115564, Publisher: Elsevier.

Ben Taieb, Souhaib, Bontempi, Gianluca, Atiya, Amir F., Sorjamaa, Antti, 2012. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Syst. Appl. 39 (8), 7067–7083, URL https://www.sciencedirect.com/science/article/pii/S0957417412000528.

Ben Taieb, Souhaib, Hyndman, Rob J., 2014. A gradient boosting approach to the Kaggle load forecasting competition. Int. J. Forecast. 30 (2), 382–394, URL https://www.sciencedirect.com/science/article/pii/S0169207013000812.

Biewald, Lukas, 2020. Experiment tracking with weights and biases. URL https://wandb.ai/site/research, http://wandb.ai/site/research.

Boßmann, T., Staffell, I., 2015. The shape of future electricity demand: Exploring load curves in 2050s Germany and Britain. Energy 90, 1317–1333, URL https://www.sciencedirect.com/science/article/pii/S0360544215008385.

Breiman, Leo, 1996. Bagging predictors. Mach. Learn. 24, 123–140, Publisher: Springer.

Breiman, Leo, 2001. Random forests. Mach. Learn. 45, 5–32, Publisher: Springer.

Chen, Tianqi, Guestrin, Carlos, 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794, URL http://arxiv.org/abs/1603.02754. arXiv:1603.02754 [cs].

Chen, Tianqi, He, T., Benesty, M., 2018. XGBoost documentation. Dmlc XGBoost.

Chung, Junyoung, Gulcehre, Caglar, Cho, KyungHyun, Bengio, Yoshua, 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. http://dx.doi.org/10.48550/arXiv.1412.3555, URL http://arxiv.org/abs/1412.3555. arXiv:1412.3555 [cs].

De Vos, Kristof, Stevens, Nicolas, Devolder, Olivier, Papavasiliou, Anthony, Hebb, Bob, Matthys-Donnadieu, James, 2019. Dynamic dimensioning approach for operating reserves: Proof of concept in Belgium. Energy Policy 124, 272–285, Publisher: Elsevier.

Deb, Chirag, Zhang, Fan, Yang, Junjing, Lee, Siew Eang, Shah, Kwok Wei, 2017. A review on time series forecasting techniques for building energy consumption. Renew. Sustain. Energy Rev. 74, 902–924, Publisher: Elsevier.

DeCarlo, Lawrence T., 1997. On the meaning and use of kurtosis. Psychol. Methods 2 (3), 292, Publisher: American Psychological Association.

Drgoňa, Ján, Arroyo, Javier, Figueroa, Iago Cupeiro, Blum, David, Arendt, Krzysztof, Kim, Donghun, Ollé, Enric Perarnau, Oravec, Juraj, Wetter, Michael, Vrabie, Draguna L, 2020. All you need to know about model predictive control for buildings. Annu. Rev. Control. 50, 190–232, Publisher: Elsevier.

Elattar, Ehab E., Sabiha, Nehmdoh A., Alsharef, Mohammad, Metwaly, Mohamed K., Abd-Elhady, Amr M., Taha, Ibrahim B.M., 2020. Short term electric load forecasting using hybrid algorithm for smart cities. Appl. Intell. 50 (10), 3379–3399.

Elman, Jeffrey L., 1990. Finding structure in time. Cogn. Sci. 14 (2), 179–211, Publisher: Wiley Online Library.

Falkner, Stefan, Klein, Aaron, Hutter, Frank, 2018. BOHB: Robust and Efficient Hyperparameter Optimization at Scale. PMLR, pp. 1437–1446.

Gokhale, Gargya, Van Gompel, Jonas, Claessens, Bert, Develder, Chris, 2023. Transfer Learning in Transformer-Based Demand Forecasting For Home Energy Management System. pp. 458–462.

Haben, Stephen, Arora, Siddharth, Giasemidis, Georgios, Voss, Marcus, Vukadinović Greetham, Danica, 2021. Review of low voltage load forecasting: Methods, applications, and recommendations. Appl. Energy 304, 117798, URL https://www.sciencedirect.com/science/article/pii/S0306261921011326.

Haben, Stephen, Giasemidis, Georgios, Ziel, Florian, Arora, Siddharth, 2019. Short term load forecasting and the effect of temperature at the low voltage level. Int. J. Forecast. 35 (4), 1469–1484, URL https://www.sciencedirect.com/science/article/pii/S0169207018301870.

Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome, Hastie, Trevor, Tibshirani, Robert, Friedman, Jerome, 2009. Linear methods for regression. Elements Stat. Learn.: Data Min. Inference, Predict. 43–99, Publisher: Springer.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, Sun, Jian, 2016. Deep Residual Learning for Image Recognition. pp. 770–778.

Herzen, Julien, Lässig, Francesco, Piazzetta, Samuele Giuliano, Neuer, Thomas, Tafti, Léo, Raille, Guillaume, Van Pottelbergh, Tomas, Pasieka, Marek, Skrodzki, Andrzej, Huguenin, Nicolas, 2022. Darts: User-friendly modern machine learning for time series. J. Mach. Learn. Res. 23 (1), 5442–5447, Publisher: JMLRORG.

Hochreiter, Sepp, Schmidhuber, Jürgen, 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780, Publisher: MIT press.

Hong, Tao, 2010. Short Term Electric Load Forecasting. North Carolina State University.

Hong, Tao, Pinson, Pierre, Fan, Shu, 2014. Global energy forecasting competition 2012. Int. J. Forecast. 30 (2), 357–363, Publisher: Elsevier.

Hong, Tao, Pinson, Pierre, Wang, Yi, Weron, Rafał, Yang, Dazhi, Zareipour, Hamidreza, 2020. Energy forecasting: A review and outlook. IEEE Open Access J. Power Energy 7, 376–388, URL https://ieeexplore.ieee.org/document/9218967/.

Hong, Tao, Xie, Jingrui, Black, Jonathan, 2019. Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. Int. J. Forecast. 35 (4), 1389–1399, URL https://www.sciencedirect.com/science/article/pii/S016920701930024X.

Hopf, Konstantin, Hartstang, Hannah, Staake, Thorsten, 2023. Meta-regression analysis of errors in short-term electricity load forecasting. In: Companion Proceedings of the 14th ACM International Conference on Future Energy Systems. ACM, Orlando FL USA, pp. 32–39, URL https://dl.acm.org/doi/10.1145/3599733.3600248.

Hornik, Kurt, Stinchcombe, Maxwell, White, Halbert, 1989. Multilayer feedforward networks are universal approximators. Neural Netw. 2 (5), 359–366, Publisher: Elsevier.

Houben, Nikolaus, Cosic, Armin, Stadler, Michael, Mansoor, Muhammad, Zellinger, Michael, Auer, Hans, Ajanovic, Amela, Haas, Reinhard, 2023. Optimal dispatch of a multi-energy system microgrid under uncertainty: A renewable energy community in Austria. Appl. Energy 337, 120913, Publisher: Elsevier.

Hu, Maomao, Stephen, Bruce, Browell, Jethro, Haben, Stephen, Wallom, David CH, 2023. Impacts of building load dispersion level on its load forecasting accuracy: Data or algorithms? Importance of reliability and interpretability in machine learning. Energy Build. 285, 112896, Publisher: Elsevier.

IEA, Outlook for electricity – World Energy Outlook 2022 – Analysis.

Januschowski, Tim, Wang, Yuyang, Torkkola, Kari, Erkkilä, Timo, Hasson, Hilaf, Gasthaus, Jan, 2022. Forecasting with trees. Int. J. Forecast. 38 (4), 1473–1481, Publisher: Elsevier.

Kazmi, Hussain, Fu, Chun, Miller, Clayton, 2023. Ten questions concerning data-driven modelling and forecasting of operational energy demand at building and urban scale. Build. Environ. 239, 110407, URL https://www.sciencedirect.com/science/article/pii/S0360132323004341.

Ke, Guolin, Meng, Qi, Finley, Thomas, Wang, Taifeng, Chen, Wei, Ma, Weidong, Ye, Qiwei, Liu, Tie-Yan, 2017. Lightgbm: A highly efficient gradient boosting decision tree. Adv. Neural Inf. Process. Syst. 30.

Keil, Christian, Craig, George C., 2009. A displacement and amplitude score employing an optical flow technique. Weather. Forecast. 24 (5), 1297–1308, Publisher: American Meteorological Society.

Koch, Christopher, 2022. Intraday imbalance optimization: incentives and impact of strategic intraday bidding behavior. Energy Syst. 13 (2), 409–435.

Koch, Christopher, Hirth, Lion, 2019. Short-term electricity trading for system balancing: An empirical analysis of the role of intraday trading in balancing Germany's electricity system. Renew. Sustain. Energy Rev. 113, 109275, URL https://www.sciencedirect.com/science/article/pii/S1364032119304836.

Krizhevsky, Alex, Sutskever, Ilya, Hinton, Geoffrey E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.

Kunz, Manuel, Birr, Stefan, Raslan, Mones, Ma, Lei, Li, Zhen, Gouttes, Adele, Koren, Mateusz, Naghibi, Tofigh, Stephan, Johannes, Bulycheva, Mariia, 2023. Deep Learning based Forecasting: A case study from the online fashion industry. arXiv preprint arXiv:2305.14406.

Kuster, Corentin, Rezgui, Yacine, Mourshed, Monjur, 2017. Electrical load forecasting models: A critical systematic review. Sustain. Cities Soc. 35, 257–270, Publisher: Elsevier.

Lechner, Mathias, Hasani, Ramin, Neubauer, Philipp, Neubauer, Sophie, Rus, Daniela, 2022. PyHopper–Hyperparameter optimization. arXiv preprint arXiv:2210.04728.

Li, Han, Heleno, Miguel, Zhang, Wanni, Sun, Kaiyu, Rodriguez Garcia, Luis, Hong, Tianzhen, 2025. A cross-dimensional analysis of data-driven short-term load forecasting methods with large-scale smart meter data. Energy Build. 344, 115909, URL https://www.sciencedirect.com/science/article/pii/S0378778825006395.

Lim, Bryan, Arık, Sercan Ö, Loeff, Nicolas, Pfister, Tomas, 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. Int. J. Forecast. 37 (4), 1748–1764, Publisher: Elsevier.

Lusis, Peter, Khalilpour, Kaveh Rajab, Andrew, Lachlan, Liebman, Ariel, 2017. Short-term residential load forecasting: Impact of calendar effects and forecast granularity. Appl. Energy 205, 654–669, URL https://www.sciencedirect.com/science/article/pii/S0306261917309881.

Miller, Clayton, Hao, Liu, Fu, Chun, 2022. Gradient boosting machines and careful pre-processing work best: ASHRAE Great Energy Predictor III lessons learned. arXiv preprint arXiv:2202.02898.

Miller, Clayton, Kathirgamanathan, Anjukan, Picchetti, Bianca, Arjunan, Pandarasamy, Park, June Young, Nagy, Zoltan, Raftery, Paul, Hobson, Brodie W., Shi, Zixiao, Meggers, Forrest, 2020. The building data genome project 2, energy meter data from the ASHRAE great energy predictor III competition. Sci. Data 7 (1), 368, URL https://www.nature.com/articles/s41597-020-00712-x. Number: 1 Publisher: Nature Publishing Group.

Nti, Isaac Kofi, Teimeh, Moses, Nyarko-Boateng, Owusu, Adekoya, Adebayo Felix, 2020. Electricity load forecasting: A systematic review. J. Electr. Syst. Inf. Technol. 7 (1), 13.

Olivares, Kin G., Challu, Cristian, Marcjasz, Grzegorz, Weron, Rafał, Dubrawski, Artur, 2023. Neural basis expansion analysis with exogenous variables: Forecasting electricity prices with NBEATSx. Int. J. Forecast. 39 (2), 884–900, URL https://www.sciencedirect.com/science/article/pii/S0169207022000413.

Open-Meteo, Free Open-Source Weather API | Open-Meteo.com.

Oreshkin, Boris N, Carpov, Dmitri, Chapados, Nicolas, Bengio, Yoshua, 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. arXiv preprint arXiv:1905.10437.

Oreshkin, Boris N., Dudek, Grzegorz, Pełka, Paweł, Turkina, Ekaterina, 2021. N-BEATS neural network for mid-term electricity load forecasting. Appl. Energy 293, 116918, URL https://www.sciencedirect.com/science/article/pii/S0306261921003986.

Pardo, Angel, Meneu, Vicente, Valor, Enric, 2002. Temperature and seasonality influences on Spanish electricity load. Energy Econ. 24 (1), 55–70, Publisher: Elsevier.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, 2019. Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. 32.

Pedregosa, Fabian, Varoquaux, Gaël, Gramfort, Alexandre, Michel, Vincent, Thirion, Bertrand, Grisel, Olivier, Blondel, Mathieu, Prettenhofer, Peter, Weiss, Ron, Dubourg, Vincent, 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830, Publisher: JMLR. org.

Pérez-Arriaga, Ignacio, Knittle, Christopher, 2016. Utility of the Future: An Mit Energy Iniative Response to an Industry in Transition. MIT Energy Initiative.

Petropoulos, Fotios, Apiletti, Daniele, Assimakopoulos, Vassilios, Babai, Mohamed Zied, Barrow, Devon K, Taieb, Souhaib Ben, Bergmeir, Christoph, Bessa, Ricardo J, Bijak, Jakub, Boylan, John E, 2022. Forecasting: Theory and practice. Int. J. Forecast. 38 (3), 705–871, Publisher: Elsevier.

Pinheiro, Marco G., Madeira, Sara C., Francisco, Alexandre P., 2023. Short-term electricity load forecasting—A systematic approach from system level to secondary substations. Appl. Energy 332, 120493, URL https://www.sciencedirect.com/science/article/pii/S0306261922017500.

Probst, Philipp, Boulesteix, Anne-Laure, Bischl, Bernd, 2019. Tunability: Importance of hyperparameters of machine learning algorithms. J. Mach. Learn. Res. 20 (53), 1–32.

Putz, Dominik, Gumhalter, Michael, Auer, Hans, 2021. A novel approach to multi-horizon wind power forecasting based on deep neural architecture. Renew. Energy 178, 494–505, URL https://www.sciencedirect.com/science/article/pii/S0960148121009654.

Putz, Dominik, Gumhalter, Michael, Auer, Hans, 2023. The true value of a forecast: Assessing the impact of accuracy on local energy communities. Sustain. Energy, Grids Networks 33, 100983, URL https://www.sciencedirect.com/science/article/pii/S2352467722002284.

Ranaweera, Damitha K., Karady, George G., Farmer, Richard G., 1997. Economic impact analysis of load forecasting. IEEE Trans. Power Syst. 12 (3), 1388–1392, Publisher: IEEE.

Rawlings, James B., 2000. Tutorial overview of model predictive control. IEEE Control Syst. Mag. 20 (3), 38–52, Publisher: IEEE.

Risbeck, Michael J., Rawlings, James B., 2019. Economic model predictive control for time-varying cost and peak demand charge optimization. IEEE Trans. Autom. Control 65 (7), 2957–2968, URL https://ieeexplore.ieee.org/abstract/document/8825493/. Publisher: IEEE.

Ruggles, Tyler H., Farnham, David J., Tong, Dan, Caldeira, Ken, 2020. Developing reliable hourly electricity demand data through screening and imputation. Sci. Data 7 (1), 155, URL https://www.nature.com/articles/s41597-020-0483-x. Number: 1 Publisher: Nature Publishing Group.

Sakia, Remi M., 1992. The Box-Cox transformation technique: A review. J. R. Stat. Soc.: Ser. D (the Statistician) 41 (2), 169–178, Publisher: Wiley Online Library.

Shi, Jie, Ye, Zuzhao, Gao, H. Oliver, Yu, Nanpeng, 2022. Lyapunov optimization in online battery energy storage system control for commercial buildings. IEEE Trans. Smart Grid 14 (1), 328–340, Publisher: IEEE.

Shwartz-Ziv, Ravid, Armon, Amitai, 2022. Tabular data: Deep learning is not all you need. Inf. Fusion 81, 84–90, Publisher: Elsevier.

Trindade, Artur, 2015. ElectricityLoadDiagrams20112014. http://dx.doi.org/10.24432/C58C86, URL https://archive-beta.ics.uci.edu/dataset/321.

Turner, Ryan, Eriksson, David, McCourt, Michael, Kiili, Juha, Laaksonen, Eero, Xu, Zhen, Guyon, Isabelle, 2021. Bayesian Optimization Is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. PMLR, pp. 3–26.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, Polosukhin, Illia, 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Voss, Marcus, 2020. Permutation-based residential short-term load forecasting in the context of energy management optimization objectives. In: Proceedings of the Eleventh ACM International Conference on Future Energy Systems. In: e-Energy '20, Association for Computing Machinery, New York, NY, USA, pp. 231–236, URL https://dl.acm.org/doi/10.1145/3396851.3397731.

Wang, Zhe, Hong, Tianzhen, Li, Han, Ann Piette, Mary, 2021. Predicting city-scale daily electricity consumption using data-driven models. Adv. Appl. Energy 2, 100025, URL https://www.sciencedirect.com/science/article/pii/S2666792421000184.

Wang, Dan, Zheng, Wanfu, Wang, Zhe, Wang, Yaran, Pang, Xiufeng, Wang, Wei, 2023. Comparison of reinforcement learning and model predictive control for building energy system optimization. Appl. Therm. Eng. 228, 120430, URL https://www.sciencedirect.com/science/article/pii/S1359431123004593.

Wen, Qingsong, Zhou, Tian, Zhang, Chaoli, Chen, Weiqi, Ma, Ziqing, Yan, Junchi, Sun, Liang, 2022. Transformers in time series: A survey. arXiv preprint arXiv:2202.07125.

Wu, Qing, Jiang, Zhe, Hong, Kewei, Huazhong, Liu, Yang, Laurence, Ding, Jihong, 2021. Tensor-based recurrent neural network and multi-modal prediction with its applications in traffic network management. IEEE Trans. Netw. Serv. Manag. PP, 1.

Yildiz, Baran, Bilbao, Jose I., Sproul, Alistair B., 2017. A review and analysis of regression and machine learning models on commercial building electricity load forecasting. Renew. Sustain. Energy Rev. 73, 1104–1122, Publisher: Elsevier.