International Journal of Robotics and Control Systems
Vol. 6, No. 2, 2026, pp. 1325-1351
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

Investigating the Impact of Hyperparameters on N-BEATS for Load Forecasting

Thanh Ngoc Tran [a,1], Tuan Anh Nguyen [b,2,*]

[a] Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City 700000, Vietnam
[1] tranthanhngoc@iuh.edu.vn; [2] nguyenanhtuan@iuh.edu.vn
* Corresponding Author

ARTICLE INFO

Article history
Received March 20, 2026
Revised April 10, 2026
Accepted April 25, 2026

Keywords
Short-Term Load Forecasting;
N-BEATS;
Hyperparameter;
Deep Learning

ABSTRACT

This study investigates how key architectural and training hyperparameters influence the forecasting accuracy of the N-BEATS deep learning model for short-term load forecasting. Rather than proposing a new automated optimization algorithm, the study addresses the need for a controlled, interpretable sensitivity analysis of N-BEATS within a fixed experimental setting. The research contribution is to identify which hyperparameter groups most strongly affect forecasting performance and to determine practical configuration ranges for N-BEATS-based load forecasting. Experiments are conducted on the Tasmania daily peak-load dataset from AEMO, which contains 2,120 daily observations. A fixed split is used, with 364 days for training and a 7-day forecasting horizon for testing. The evaluation is carried out in two stages: 45 architectural configurations are first examined by varying stack variant, number of stacks, and number of blocks per stack, and then 125 training-hyperparameter configurations are evaluated by varying maximum training steps, learning rate, and MLP width. The results show that N-BEATS performance is strongly affected by both architectural and training hyperparameters. Within the evaluated search space, the best-observed configuration achieves a MAPE of 3.1300% and an RMSE of 51.35, compared with 4.2839% and 64.32 for the default configuration, corresponding to a 26.9% reduction in MAPE. Moderate block depths perform better than overly shallow or deeper settings. In addition, max_steps = 500, learning rates from 1×10^{-4} to 5×10^{-4}, and an MLP width of [512, 512] provide the most favorable accuracy–stability trade-off. These findings provide practical guidance for configuring N-BEATS in load-forecasting applications.

1. Introduction

Load forecasting, which aims to predict future electricity demand, plays a fundamental role in modern power system planning, operation, and control. Accurate forecasts support critical tasks such as generation scheduling, unit commitment, economic dispatch, demand-side management, maintenance planning, and market operation. In practical power systems, reliable load forecasts help maintain balance between supply and demand, improve operational security, reduce reserve uncertainty, and lower overall operating costs. However, electricity demand is influenced by many

factors, including historical consumption patterns, weather conditions, seasonal effects, calendar variables, and socio-economic activities. As a result, load forecasting remains a challenging problem, especially when the load series exhibits nonlinear, nonstationary, and complex temporal dynamics.

Early studies on load forecasting mainly relied on traditional statistical methods, such as autoregressive integrated moving average (ARIMA) [1]–[4], SARIMA [5]–[8], and Holt–Winters exponential smoothing [9], [10]. These approaches are attractive because of their clear mathematical formulation, good interpretability, and effectiveness when the underlying time series is approximately linear and stationary. Nevertheless, their practical performance often depends on substantial preprocessing, including detrending, differencing, seasonal adjustment, and parameter identification. When these methods are applied to modern electricity load data with stronger nonlinearities, irregular fluctuations, and evolving temporal patterns, their predictive capability may become limited.

To address some of these limitations, machine-learning methods have been increasingly explored for load forecasting. Representative techniques include support vector regression (SVR) [11]–[18], random forest [19]–[23], XGBoost [24]–[29], and LightGBM [30]–[34]. Compared with conventional statistical models, these methods are generally better at learning nonlinear relationships between input variables and electricity demand. In many cases, they can achieve improved forecasting performance, particularly when external variables and complex interactions are involved. However, their effectiveness still depends considerably on feature engineering, data representation, and handcrafted input design. In addition, although these methods can model nonlinear mappings effectively, they do not exploit sequential temporal dependencies as directly as deep-learning-based forecasting architectures.

More recently, deep learning has emerged as a powerful approach for load forecasting because it can automatically learn latent temporal representations from raw historical data. A wide range of deep-learning architectures has been investigated, including multilayer perceptrons (MLPs) [35]–[37], recurrent neural networks (RNNs) [38]–[42], long short-term memory networks (LSTMs) [43]–[48], gated recurrent units (GRUs) [49]–[51], convolutional neural networks (CNNs) [52]–[55], sequence-to-sequence frameworks, and Transformer-based models [56]–[60]. These approaches have shown promising forecasting performance in many time-series applications by reducing reliance on manual feature extraction and enabling end-to-end learning. However, their improved modeling capability is accompanied by greater architectural and training complexity. In practice, the forecasting performance of deep learning models is often highly sensitive to hyperparameter selection, making model design, tuning, and interpretation significantly more difficult.

In this context, N-BEATS [61]–[64] has emerged as a competitive deep learning architecture for univariate time series forecasting. Unlike recurrent or convolutional models, N-BEATS is built on a pure multilayer perceptron backbone and organizes fully connected layers into stacks and blocks connected through residual learning. It supports two main variants: an interpretable variant that explicitly models trend and seasonality via basis expansions, and a generic variant that uses a data-driven identity basis for greater flexibility. Because of its simple feedforward structure, strong forecasting capability, and potential interpretability, N-BEATS is a promising candidate for electricity load forecasting.

In this study, N-BEATS is selected not for cross-model benchmarking, but because it provides a suitable, well-structured framework for within-model hyperparameter sensitivity analysis. Its architecture combines configurable stack variants, the number of stacks, the number of blocks, and MLP width within a unified forecasting framework, enabling examination of how structural and training choices affect forecasting performance without the additional confounding factors that arise when fundamentally different model families are compared. In other words, the objective of this study is not to claim that N-BEATS is universally superior to ARIMA, LSTM, or Transformer-based models, but rather to understand, in a focused and interpretable manner, how the performance of N-BEATS itself changes under different hyperparameter settings.

Despite its advantages, N-BEATS's practical performance depends heavily on hyperparameter selection. Forecasting accuracy can vary substantially with architectural choices, such as stack variant,

number of stacks, number of blocks, and hidden-layer size, as well as with training-related settings such as learning rate and training duration. In many existing applications, these hyperparameters are selected using ad hoc trial-and-error procedures or default library settings. Consequently, there remains limited systematic understanding of which hyperparameters most strongly affect N-BEATS forecasting accuracy and which configuration ranges are more suitable for a specific load-forecasting task. This research gap is important because insufficient understanding of hyperparameter sensitivity can hinder both practical deployment and the development of more principled optimization strategies for N-BEATS-based forecasting systems.

To address this gap, this study conducts a systematic sensitivity analysis of N-BEATS hyperparameters for short-term load forecasting. Rather than proposing a new automated hyperparameter optimization algorithm, the objective is to evaluate, in a controlled and transparent manner, how key structural and training hyperparameters influence forecasting performance. Although automated hyperparameter optimization methods are effective for searching high-performing configurations, they are not always the most suitable choice when the main research objective is to interpret the relative influence of different hyperparameter groups. In the present study, the priority is not only to identify high-performing settings but also to clarify which architectural and training choices contribute most strongly to improvements in forecasting accuracy. For this reason, a flowchart-based experimental framework is adopted in which one hyperparameter group is examined while the remaining settings are held fixed. This design enables clearer effect isolation and more transparent interpretation of architectural and training sensitivities, although possible interaction effects are acknowledged as a limitation of the controlled analysis. Therefore, the proposed framework should be understood as being more suitable for explanatory sensitivity analysis than as a universal replacement for automated hyperparameter optimization methods.

A case study is carried out using the Tasmania electricity load dataset. This dataset is selected because it is a real-world public load series that exhibits meaningful temporal variability in daily peak demand, including local fluctuations and changes in magnitude over time. Such characteristics make it a relevant case for sensitivity analysis, since the forecasting task is not purely regular and may respond differently to alternative N-BEATS variants and hyperparameter settings. In this sense, the dataset provides a practical setting for examining how different N-BEATS configurations affect forecasting accuracy and model behavior in a realistic load-forecasting scenario. The use of Tasmania data also supports the study's objective by allowing the proposed analysis to be demonstrated on an actual electricity-demand series rather than an artificial or simplified benchmark.

The research contribution of this study is to provide a controlled and interpretable sensitivity-analysis framework for understanding how key N-BEATS hyperparameters affect short-term load forecasting performance. More specifically, the contributions of this study are as follows:

- First, a structured experimental framework is developed to analyze N-BEATS's sensitivity to both architectural and training hyperparameters in short-term load forecasting.
- Second, a comprehensive empirical evaluation is conducted across multiple N-BEATS configurations, including stack variants, the number of stacks, the number of blocks per stack, the learning rate, the maximum training steps, and the MLP hidden-layer size.
- Third, the study identifies the hyperparameters with the strongest influence on forecasting accuracy and highlights practical configuration ranges that provide favorable accuracy–robustness trade-offs for N-BEATS-based load forecasting.

2. Method

2.1. N-BEATS Model and Architecture

N-BEATS (Neural Basis Expansion Analysis for Time Series) is a deep feedforward architecture designed specifically for univariate time-series forecasting. The model consists of multiple blocks organized into one or more stacks, where each block performs two tasks simultaneously:

reconstructing the historical input through a backcast component and producing future predictions through a forecast component. A key mechanism of N-BEATS is backward residual learning, in which each successive block focuses on residual information not explained by preceding blocks, thereby progressively refining the forecast.

Let $x \in \mathbb{R}^H$ denote the input window of length H, and let $y \in \mathbb{R}^O$ denote the future values to be predicted over a forecasting horizon of length O. At block b, the input is denoted by $x^{(b)}$. The block first applies a multi-layer perceptron (MLP) to extract a latent representation:

$$\theta^b = f_{MLP}^{(b)}(x^{(b)}) \tag{1}$$

where $\theta^{(b)} \in \mathbb{R}^d$ is the latent feature vector produced by the MLP, and d denotes the dimensionality of the hidden representation.

Based on $\theta^{(b)}$, the block produces two outputs: A backcast $\hat{x}^{(b)} \in \mathbb{R}^H$, which approximates the input sequence, and a forecast $\hat{y}^{(b)} \in \mathbb{R}^O$, which predicts the future values. These two outputs are defined as:

$$\hat{x}^{(b)} = B^{(b)}(\theta^{(b)}), \hat{y}^{(b)} = F^{(b)}(\theta^{(b)}) \tag{2}$$

where $B^{(b)}(\cdot)$ and $F^{(b)}(\cdot)$ are mappings from the latent representation to the backcast and forecast spaces, respectively.

The input to the next block is computed using a residual formulation:

$$x^{(b+1)} = x^{(b)} - \hat{x}^{(b)} \tag{3}$$

This residual connection allows each block to focus on the remaining unexplained component of the input sequence. The final forecast of the entire N-BEATS model is obtained by summing the forecast outputs from all blocks:

$$\hat{y} = \sum_{b=1}^{B} \hat{y}^{(b)} \tag{4}$$

where B is the total number of blocks across all stacks. This design enables N-BEATS to refine its predictions through multiple residual learning steps iteratively.

Blocks are grouped into stacks, and each stack's role depends on the architectural variant. In the generic variant, the backcast and forecast mappings are implemented as simple linear projections:

$$\hat{x}^{(b)} = W_b^{(x)}\theta^{(b)}, \hat{y}^{(b)} = W_b^{(y)}\theta^{(b)} \tag{5}$$

where $W_b^{(x)}$ and $W_b^{(y)}$ are learned weight matrices. This variant is fully data-driven and imposes no explicit assumptions about trend or seasonality.

In the interpretable variant, N-BEATS uses structured basis functions to model trend and seasonality separately. For example, a trend block may use a polynomial basis of degree p:

$$\hat{y}_{\text{trend}}^{(b)}(t) = \sum_{i=0}^{p} \theta_i^{(b)} \left(\frac{t}{O}\right)^i, t = 1, \ldots, O \tag{6}$$

where p is the degree of the polynomial basis and $\theta_i^{(b)}$ are the corresponding trend coefficients.

A seasonality block may use harmonic basis functions:

$$\hat{y}_{\text{season}}^{(b)}(t) = \sum_{k=1}^{K} \left[a_k^{(b)} \cos\left(\frac{2\pi kt}{O}\right) + b_k^{(b)} \sin\left(\frac{2\pi kt}{O}\right) \right], t = 1, \ldots, O \quad (7)$$

where K is the number of harmonics, and $a_k^{(b)}$ and $b_k^{(b)}$ are the seasonal coefficients learned by block b.

By combining trend and seasonality stacks, the interpretable variant can provide a meaningful decomposition of temporal patterns while maintaining strong forecasting performance.

The model is trained by minimizing a loss function between the actual target y and the predicted output $\hat{y}$. Standard loss functions in load forecasting include MSE, MAE, and MAPE. For example, the mean squared error (MSE) loss is defined as:

$$\mathrm{L} = \frac{1}{N} \sum_{n=1}^{N} \| y^{(n)} - \hat{y}^{(n)} \|_2^2 \quad (8)$$

where N is the number of training samples, $y^{(n)}$ is the actual target of sample n, and $\hat{y}^{(n)}$ is the corresponding forecast.

Thanks to its pure MLP architecture, residual block design, and flexible stack formulation, N-BEATS can be adapted to different load-forecasting scenarios while preserving forecasting capability and, in the interpretable variant, a degree of component-level interpretability.

2.2. Hyperparameters of the N-BEATS Model

2.2.1. Architectural Hyperparameters

The forecasting performance of N-BEATS is strongly influenced by several architectural hyperparameters that determine the model's structure, depth, and representational capacity. In this study, the architectural hyperparameters are selected because they directly affect how the model represents temporal patterns and how effectively it refines the forecasting task through its residual-learning mechanism.

N-BEATS supports three main stack types. The Identity, or Generic, stack is fully data-driven and suitable for complex or irregular load patterns because it imposes no explicit assumptions about the time series structure. In contrast, the Trend stack uses polynomial basis functions to model long-term trend behavior, while the Seasonality stack employs harmonic basis functions to capture periodic fluctuations. Depending on the forecasting task and the dataset's characteristics, a model may include one or more stacks arranged in different structural patterns.

Another important architectural hyperparameter is the number of stacks, denoted by S. This hyperparameter determines how many stacks are used in the overall model architecture. Increasing the number of stacks can enable deeper hierarchical decomposition of the forecasting problem and may improve the model's ability to capture complex temporal patterns. However, excessively large values of S also increase model complexity, training time, and computational cost, and may raise the risk of overfitting.

The number of blocks per stack, denoted by N, is also a key factor in N-BEATS design. Each stack contains N blocks that progressively refine the residual representation from previous blocks. A larger value of N can improve predictive capability by allowing more iterative refinement within each stack. Nevertheless, increasing the number of blocks also makes the network deeper, which may increase computational burden and reduce training stability if the architecture becomes overly complex.

The MLP size, also called MLP units, describes the hidden-layer structure within each block. It can be expressed as:

$$MLP\ units = [u_1, u_2, \ldots, u_L] \quad (8)$$

where L is the number of hidden layers and u_i is the number of neurons in the i-th layer. Larger MLPs generally offer greater representational capacity and may improve forecasting accuracy by learning richer latent features. However, they also require more memory and may increase the risk of overfitting. On the other hand, smaller MLPs produce lighter models with lower computational demand, although this may come at the expense of reduced predictive performance.

2.2.2. Training Hyperparameters

In addition to architectural design, N-BEATS' forecasting performance is also influenced by several training hyperparameters that determine how the model is optimized during learning. In this study, the training hyperparameters are considered because they directly affect convergence behavior, training stability, and the model's final forecasting accuracy.

One of the most important training hyperparameters is the learning rate, denoted by η. The learning rate controls the step size used in updating model parameters during optimization. If η is too large, the optimization process may become unstable and fail to converge properly. In contrast, if η is too small, training becomes slower and may lead to suboptimal solutions due to insufficient progress during parameter updates. Therefore, selecting an appropriate learning rate is essential for achieving a balance between convergence speed and training stability.

Another key hyperparameter is the maximum number of training steps, denoted max_steps, which sets an upper bound on the number of parameter-update iterations during training. Increasing max_steps can allow the model to learn more thoroughly and may improve forecasting accuracy. However, excessively large values also increase computational cost and training time. They may raise the risk of overfitting, especially when the model begins to fit noise rather than underlying temporal patterns.

Batch size is also an important training hyperparameter because it determines how many samples are used in each gradient update. Smaller batch sizes usually produce noisier gradient estimates, which may help the optimizer escape poor local minima and improve generalization. On the other hand, larger batch sizes provide smoother gradient updates. They can improve computational efficiency, although they require more memory and may reduce the stochastic regularization effect of mini-batch training. As a result, batch size influences both optimization dynamics and hardware efficiency.

The loss function further shapes the model's training behavior by defining how prediction errors are measured during optimization. Common loss functions in load forecasting include mean absolute error (MAE), mean squared error (MSE), and mean absolute percentage error (MAPE). Each loss function emphasizes errors differently, and the selected loss function can therefore influence the training process's sensitivity, particularly when load values vary substantially over time. For this reason, the choice of loss function is an important factor in the overall training configuration of N-BEATS.

2.2.3. Proposed Method for Analyzing Hyperparameter Impact

In this study, a systematic methodology is adopted to quantify the effects of hyperparameters on N-BEATS' forecasting performance. The main objective is not to propose a new automated hyperparameter optimization algorithm, but to perform a controlled sensitivity analysis that allows the effects of each hyperparameter group to be examined in a transparent and interpretable manner. The core principle of the methodology is to vary one hyperparameter group while keeping the remaining settings fixed, so that changes in forecasting accuracy can be more directly attributed to the parameter group under investigation. This design facilitates effect isolation and helps identify the hyperparameters that exert the strongest influence on model performance.

Consistent with the experimental flowchart, the hyperparameters are organized into two groups. The first group contains the architectural hyperparameters, including stack configuration (identity-only, trend–seasonality-only, or hybrid identity–trend–seasonality), the number of stacks S, and the

number of blocks per stack N. The second group contains the training hyperparameters, including max_steps, learning rate, and mlp_units. Other settings, such as batch size, optimizer, and loss function, are kept constant throughout the experiments to ensure a fair comparison across configurations and to avoid confounding effects from changes outside the targeted search space.

The evaluation is conducted in two stages. In the first stage, the training hyperparameters are fixed, while architectural configurations are explored within a predefined search space. For each configuration, an independent N-BEATS model is trained and evaluated using MAE, MAPE, MSE, and RMSE. This stage is intended to assess the influence of stack variants, stack depth, and block depth, and to identify a strong reference architecture for the next stage. In the second stage, the selected reference architecture is fixed, and the training hyperparameters max_steps, learning rate, and mlp_units are examined according to the one-factor-at-a-time principle, with the model retrained for each tested setting.

It should be noted that this controlled design emphasizes interpretability of hyperparameter effects rather than exhaustive joint optimization over the full configuration space. Consequently, possible interactions or couplings between architectural and training hyperparameters may not be fully captured, and the identified best configuration should therefore be interpreted as the best-performing setting within the evaluated search space rather than as a guaranteed global optimum. Nevertheless, this staged framework provides a practical, reproducible way to analyze hyperparameter sensitivity and derive empirical guidance for configuring N-BEATS in load-forecasting applications.

In addition, the forecasting horizon is kept fixed at 7 days throughout the experiments to isolate the effect of hyperparameter changes within a consistent short-term evaluation setting.

2.3. Flowchart of the Hyperparameter Impact Evaluation Process

Fig. 1 illustrates the overall experimental framework used to evaluate the impact of hyperparameters on the N-BEATS model's forecasting performance via a controlled grid search. The framework is organized into four main stages: data preprocessing and partitioning; defining the hyperparameter search space and fixed settings; model training for each tested configuration; and performance evaluation using standard forecasting metrics. In the first stage, the original daily time series is prepared for forecasting by performing data cleaning, handling missing values, applying normalization, and converting the processed series into the format required by the model. After preprocessing, the dataset is divided into a training set and a test set. The training set contains the historical observations used for model fitting, whereas the test set contains the final forecasting window reserved for out-of-sample evaluation. This setup is designed to simulate a short-term load-forecasting scenario while ensuring that all tested configurations are evaluated under the same protocol.

In the second stage, the search space is defined by grouping the model settings into two categories: structural hyperparameters and training hyperparameters. The structural group includes the stack configuration, the number of stacks, and the number of blocks per stack, all of which directly determine the architectural depth and composition of the N-BEATS model. The training group includes the maximum number of training steps, the learning rate, and the size of the hidden layers in the MLP blocks, which together influence optimization behavior, convergence stability, and model capacity. To preserve fairness and interpretability in the comparison, the remaining training settings are held constant throughout the experiments. This separation of hyperparameters into two groups is consistent with the study's purpose: to analyze their effects in a controlled and transparent manner rather than to perform unrestricted joint optimization over all possible combinations.

In the third stage, two complementary experiments are carried out within the same framework. The first experiment focuses on architectural analysis. In this stage, the structural hyperparameters are varied while the training hyperparameters are kept fixed. The purpose is to investigate how differences in stack type, stack depth, and block depth affect forecasting accuracy, and to identify a strong reference architecture for the subsequent experiment. The second experiment focuses on training-related analysis. After selecting the reference architecture from the first stage, the architectural settings

are fixed, and the training hyperparameters are varied systematically. For each tested configuration, an independent N-BEATS model is trained on the training set and then used to generate forecasts for the test period. The results obtained from all configurations are stored and organized for later comparison, allowing the effects of individual hyperparameter groups to be examined more clearly.

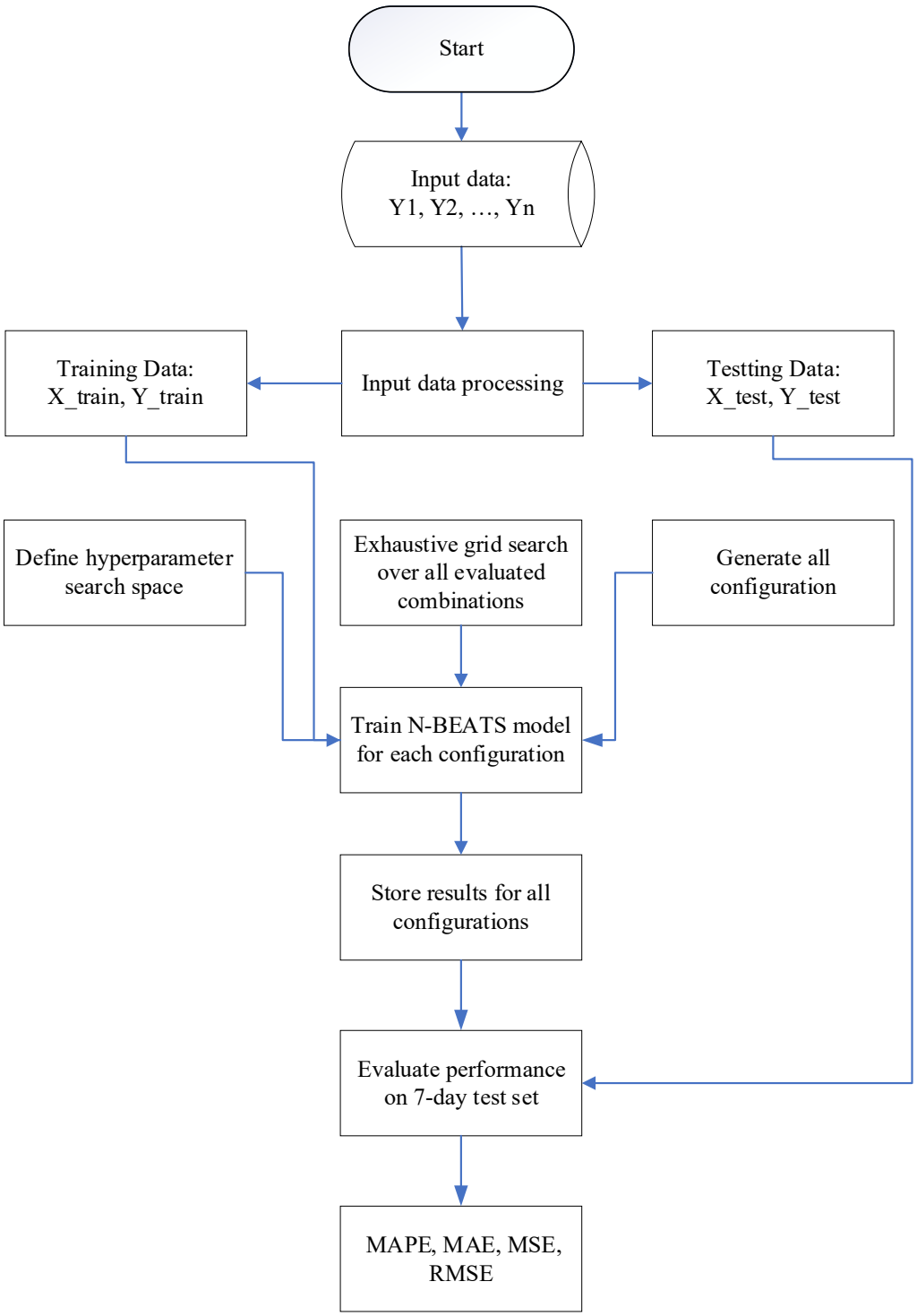

Fig. 1. Flowchart of the hyperparameter impact evaluation procedure for N-BEATS

In the final stage, forecasting performance is evaluated using four standard error metrics: MAPE, MAE, MSE, and RMSE. These metrics provide complementary views of prediction quality, enabling comparisons across configurations from multiple perspectives rather than relying on a single measure. By following this staged procedure, the framework provides a transparent, systematic, and

reproducible approach to examining how different architectural and training hyperparameters affect N-BEATS forecasting accuracy. At the same time, because the experiments are designed to isolate the influence of selected hyperparameter groups under fixed settings, the framework should be interpreted primarily as a sensitivity-analysis procedure. Therefore, its main value lies in clarifying the relative influence of hyperparameters and providing empirical guidance for model configuration, rather than claiming to identify a guaranteed global optimum across the entire configuration space.

3. Results and Discussion

3.1. Dataset Description and Experimental Setup

This study uses the public load dataset published by the Australian Energy Market Operator (AEMO) for the Tasmania region. The dataset was selected because it provides a real-world, publicly accessible load series with sufficient temporal variability for evaluating N-BEATS's sensitivity under practical forecasting conditions. The raw data are recorded at a 30-minute resolution and include the variables *settlementdate* and *totaldemand*. To prepare the series for short-term load forecasting, the timestamps are converted to a datetime format, missing records are removed, and the settlement date is set as the time index. The series is then resampled to a daily frequency by taking the maximum total demand value for each day, thereby constructing a daily peak-load series. This transformation is consistent with the study's objective, which focuses on forecasting daily peak demand rather than sub-daily load variation. For compatibility with the NeuralForecast framework, the timestamp column is renamed as *ds*, the peak-load values are assigned to *y*, and a *unique_id* field is added to represent the time series in the required input format.

In all experiments, the input window length (lookback) was kept fixed at 28 observations. Since the data were prepared at a daily frequency, this corresponds to a 28-day sliding input window. This choice was made to provide the same historical context for all tested configurations and to ensure that differences in forecasting performance could be attributed more directly to the evaluated N-BEATS hyperparameters rather than to changes in the input-window design. Therefore, the present study does not investigate the effect of the lookback/forecast-horizon ratio itself. Instead, the lookback is treated as a fixed experimental setting, while the analysis focuses on the sensitivity of the selected architectural and training hyperparameters.

Fig. 2 presents the daily peak-load time series of Tasmania over the study period from 2020 to 2025 after preprocessing and daily resampling. The figure is intended to provide an overall visual overview of the temporal behavior of the processed input data used in this study and to illustrate the variability of the daily peak-demand series before model training and evaluation.

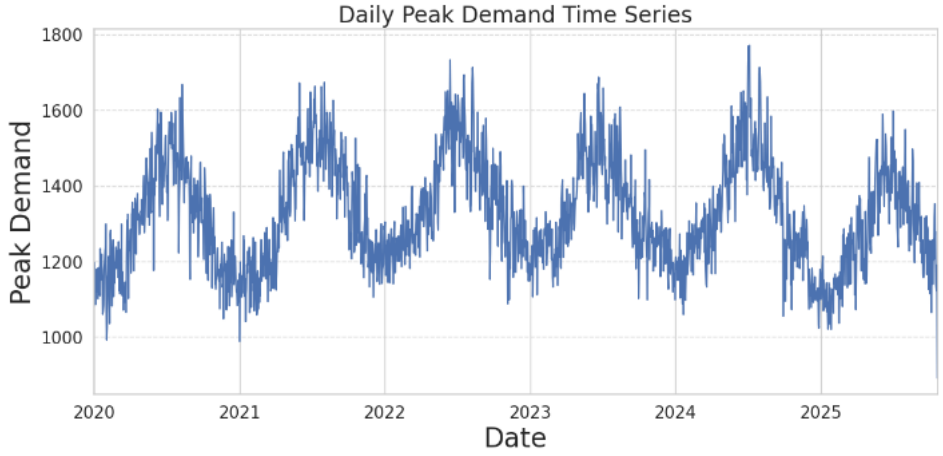

Fig. 2. Tasmania daily peak (2020–2025)

As shown in Fig. 2, the Tasmania daily peak-load series exhibits noticeable fluctuations over time, with recurrent rises and declines appearing across different years. The series does not remain at

a fixed level; instead, it exhibits both medium-term variation and short-term local peaks, indicating that the load behavior is neither static nor purely regular. Such characteristics suggest that the dataset reflects realistic operational variability and therefore provides a suitable case for examining how different N-BEATS hyperparameter settings affect forecasting accuracy. In particular, the visible changes in magnitude and pattern over time support the need for a systematic hyperparameter sensitivity analysis rather than relying solely on default model settings.

While Fig. 2 provides a visual overview of the temporal variation in the Tasmania daily peak-load series, Table 1 presents the corresponding descriptive statistics to quantify its overall distribution and variability. Specifically, the table reports the number of samples, minimum value, quartiles, median, maximum value, mean, and standard deviation of the processed dataset.

Table 1. Reports descriptive statistics, Overall of Tasmania, Daily Peak

Data	Samples	Min	Q1 (25%)	Median (Q2)	Q3 (75%)	Max	Mean	Std
Overall	2120	891.81	1214.04	1304.83	1430.77	1771.02	1323.50	143.14

The results in Table 1 show that the processed daily peak-load series contains 2,120 observations, indicating a sufficiently large dataset for experimental evaluation. The minimum and maximum values are 891.81 and 1771.02, respectively, which reveals a relatively wide load range. The mean is 1323.50, while the median is 1304.83, indicating that the series's central tendency lies around 1300. In addition, the first and third quartiles are 1214.04 and 1430.77, showing that a large proportion of the observations is distributed within a clearly defined interval. The standard deviation of 143.14 further indicates noticeable dispersion in the series. Overall, these descriptive statistics are consistent with the visual observations in Fig. 2 and confirm that the Tasmania daily peak-load dataset has sufficient variability to support the analysis of N-BEATS hyperparameter sensitivity in short-term load forecasting.

To further clarify how the processed series is used in the forecasting experiments, Fig. 3 and Fig. 4 illustrate the train–test partition adopted in this study. Specifically, the dataset is divided using a fixed daily sliding window, with the final forecasting window reserved for testing and the preceding observations used for model training. This partition is intended to reflect a short-term load-forecasting scenario under a consistent, reproducible evaluation protocol.

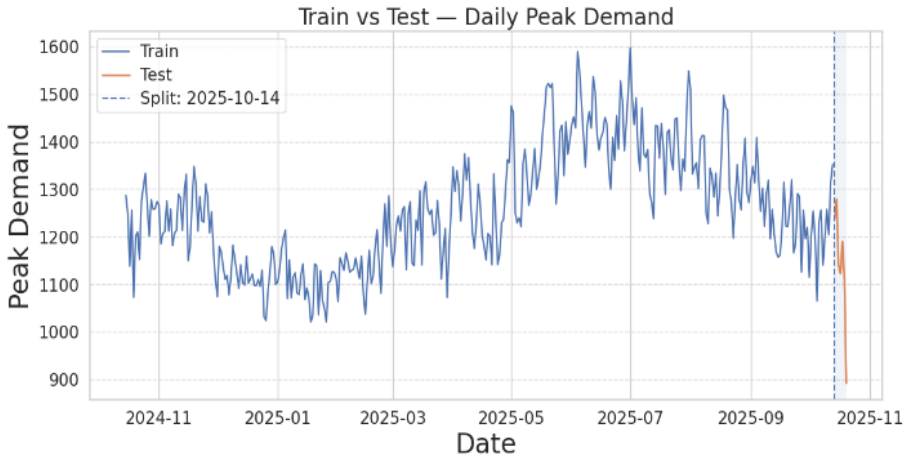

Fig. 3. Train vs Test — Daily Peak Demand (Tasmania)

Fig. 3 presents the complete train–test split of the Tasmania daily peak-load series. The figure shows the overall temporal partition used in the experiments, with the training segment covering the historical observations used for model fitting, and the test segment corresponding to the final forecast window reserved for out-of-sample evaluation. A dashed vertical line marks the cutoff point between the two subsets, making the selected evaluation setup visually explicit.

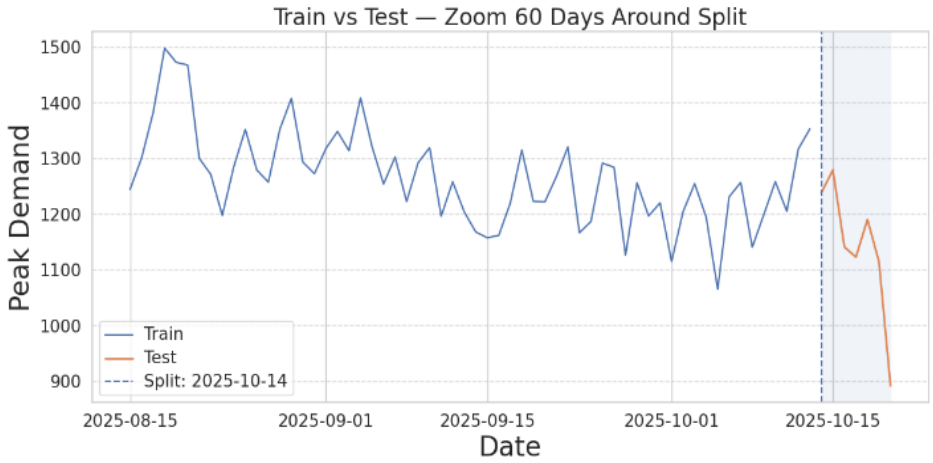

Fig. 4. Train vs Test — Zoom Around Split (60 Days)

As shown in Fig. 3, the training set includes historical daily peak-demand observations before the final evaluation period. In contrast, the test set corresponds to the last seven days of the series. This visualization clearly shows the temporal continuity of the data. It confirms that the forecasting task is performed under a strict chronological split, in which future values are predicted solely from past observations. Such a setup is appropriate for load forecasting because it preserves the natural time order of the series and avoids information leakage from the future into the training stage. In addition, the figure provides a clear overview of the final forecast window's position relative to the preceding training data.

To provide a more detailed view of the selected evaluation window, Fig. 4 zooms in on the 60 days surrounding the split point. This close-up visualization is included to show the local temporal context immediately before and during the forecast week, thereby making the short-term forecasting setting easier to interpret.

Fig. 4 shows the short-term behavior of the load series near the train–test boundary in greater detail. Compared with Fig. 3, this zoomed view makes it easier to observe the most recent training observations and the evolution of the seven test days. The figure shows that the forecast window follows the latest available training data, consistent with the study's short-term forecasting objective. It also helps illustrate the local variation in load values immediately before the test period, thereby providing useful visual context for interpreting the prediction errors reported in the subsequent analysis.

While Fig. 3 and Fig. 4 provide a visual representation of the train–test partition, Table 2 presents the descriptive statistics for the two subsets to quantify their main characteristics. Specifically, the table reports the number of samples, minimum, quartiles, median, maximum, mean, and standard deviation for both the training and test sets.

Table 2. Reports descriptive statistics train and test

Data	Samples	Min	Q1 (25%)	Median (Q2)	Q3 (75%)	Max	Mean	Std
Train	364	1020.30	1158.47	1254.88	1344.38	1597.25	1259.07	124.01
Test	7	891.81	1118.25	1140.42	1213.86	1278.74	1139.31	125.09

The results in Table 2 show that the training set contains 364 daily observations, whereas the test set contains 7 observations corresponding to the selected forecasting horizon. The mean value of the training set is 1259.07. In contrast, the test set has a lower mean of 1139.31, indicating that the evaluation window lies at a relatively lower demand level than the average of the training period. The minimum value in the test set is also lower than that of the training set. In contrast, the standard deviations of the two subsets are similar, suggesting that although the forecast window is short, it still exhibits noticeable variation. Overall, these statistics provide a quantitative reference for

understanding the distributional characteristics of the training and evaluation data, and they help contextualize the forecasting performance obtained under the selected experimental setup.

The 7-day test horizon was selected to represent a short-term daily peak-load forecasting scenario under a strict and consistent evaluation setting. In the context of this study, the main objective is not to optimize performance for all possible forecasting horizons, but to examine how changes in hyperparameter settings affect N-BEATS behavior under a clearly defined short-term task. Using the same 7-day out-of-sample window for all evaluated configurations also ensures direct comparability of forecasting errors across experiments. Nevertheless, the findings should be interpreted within this short-horizon setting, and broader validation over longer horizons remains an important direction for future work.

3.2. Configuration of the Hyperparameter Search Space

Table 3 summarizes the hyperparameter search space considered in the N-BEATS experiments. For clarity, the parameters are organized into two groups: (i) architectural hyperparameters, which determine the model structure, and (ii) training hyperparameters, which govern the optimization process. This separation is consistent with the study's objective, namely, to perform a controlled sensitivity analysis in which the influence of each hyperparameter group can be examined more transparently.

Table 3. Configuration of the hyperparameter search space

Parameter	Symbol	Search Space
Stack type	variant	{G, D, H}
Number of stacks per variant	S	G: {1, 2, 3}; D: {2, 4, 6}; H: {3, 6, 9}
Number of blocks in each stack	N_blocks	{1, 2, 3, 4, 5}
Training steps	max_steps	{250, 500, 750, 1000, 1250}
Learning rate	learning_rate	$\{5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2.5\times10^{-3}\}$
Number of neurons per MLP layer	mlp_units	{[128,128], [256,256], [512,512], [640,640], [768,768]}

The selected search space was designed to cover representative low, moderate, and relatively high settings for the main N-BEATS hyperparameters while keeping the experiments computationally feasible. Rather than aiming at exhaustive global optimization, the intention was to investigate how forecasting performance changes across a structured range of architectural and training choices. Therefore, the tested values were chosen to provide meaningful variation in model depth, model capacity, and optimization behavior, while avoiding an excessively large configuration space that would hinder transparent analysis.

For the architectural group, three stack variants were considered: the Generic variant (G), the Trend–Seasonality variant (D), and the Hybrid variant (H). These variants were included because they represent different structural assumptions about how temporal patterns are modeled in N-BEATS. In particular, G relies on identity blocks and is fully data-driven, D alternates trend and seasonality blocks, and H repeats the identity–trend–seasonality pattern. For each variant, the number of stacks starts with startwith$S$ws varied according to its structural composition, namely $G = \{1,2,3\}$, $D = \{2,4,6\}$, and $H = \{3,6,9\}$. This choice ensures that each variant is examined at comparable levels of architectural depth while respecting its internal stack design. In addition, the number of blocks per architecture was varied from 1 to 5 to examine the effect of block depth within each architecture. Taken together, these settings generate 45 architectural configurations (3 variants $\times$ 3 stack levels $\times$ 5 block levels), spanning shallow to deeper structures and enabling a focused analysis of architectural sensitivity.

For the training group, three key hyperparameters were examined: maximum training steps, learning rate, and MLP width. The range of max_steps $\{250, 500, 750, 1000, 1250\}$was selected to cover shorter, moderate, and longer training durations, thereby allowing the study to assess whether additional optimization iterations improve forecasting accuracy or instead reduce stability. The learning-rate values $\{5\times10^{-5}, 1\times10^{-4}, 5\times10^{-4}, 1\times10^{-3}, 2.5\times10^{-3}\}$were chosen to span conservative to relatively aggressive update scales, so that the experiments could capture the trade-off

between stable convergence and faster but potential ally less reliable optimization. Similarly, the MLP-with settings open bracket 128,128 close bracket, open bracket 256,256 close bracket, open bracket 512,512 close bracket, open bracket 640,640 close bracket, open bracket 768,768 close bracket were selected to represent increasing levels of representational capacity within each N-BEATS block. Because both hidden layers in each block share the same size, this design also facilitates a more interpretable comparison of model capacity across configurations.

The Cartesian product of the three training hyperparameters yields 125 configurations. However, combining these 125 settings with all 45 architectural configurations would yield 5,625 total combinations. Although such a full search could be explored in principle, it would substantially increase computational cost and would make the interpretation of individual hyperparameter effects less transparent. For this reason, a two-phase strategy was adopted. In the first phase, the training hyperparameters were held fixed while 45 architectural configurations were evaluated to identify a strong reference architecture. In the second phase, this selected reference architecture was fixed, and the 125 training-hyperparameter configurations were examined. This staged design reduces computational burden while still allowing the relative influence of architectural and training hyperparameters to be analyzed in a controlled, reproducible manner.

It should be emphasized that this search-space design was intended to support sensitivity analysis rather than to claim a guaranteed global optimum over all possible N-BEATS configurations. Because the experiments were organized sequentially, possible interactions between architectural and training hyperparameters may not have been fully captured. Nevertheless, the selected search space provides a practical balance between breadth, interpretability, and computational feasibility. It is sufficient to reveal the main performance trends of N-BEATS across different hyperparameter settings in the present study.

3.3. Google Colab Environment and Nixtla Library

All experiments in this study were conducted in the Google Colab environment, which provides a cloud-based Python workflow with GPU support and convenient integration with Google Drive. This environment was selected because it provides a practical, reproducible platform for running a large number of deep learning experiments under consistent computational conditions. In the present study, the forecasting models were trained on an NVIDIA A100 GPU available in Colab. At the same time, the input dataset was loaded from a mounted Google Drive directory, and the intermediate outputs, including generated forecasts and evaluation tables, were saved back to Drive. This workflow enabled managing experiment results in an organized manner and preserving the outputs of each tested configuration for later verification and comparison.

The computational pipeline was implemented in Python 3. Data preprocessing and tabular manipulation were carried out using Pandas and NumPy, while Matplotlib was used for visualization. The forecasting error metrics, including MAE, MSE, RMSE, and MAPE, were computed using standard scientific Python tools, including scikit-learn, where appropriate. The deep learning training process was implemented using PyTorch, which served as the backend framework for GPU-optimized model optimization. To implement the forecasting models, this study used Nixtla's NeuralForecast library, which provides a unified framework for training and evaluating modern neural forecasting architectures, including N-BEATS. In accordance with the library requirements, the processed time series was formatted using the standard columns unique_id, ds, and y, where unique_id identifies the series, ds denotes the timestamp, and y represents the target load value. This format was used consistently across all experiments to ensure compatibility and consistency in the training pipeline.

To improve reproducibility, the experiments were executed under fixed random seeds, and deterministic operations were enabled whenever supported by the underlying framework and hardware environment. This design was adopted to reduce uncontrolled variation across repeated runs and to ensure that the observed performance differences could be attributed more reliably to the tested hyperparameter settings rather than to random initialization effects. In addition, the use of a unified Colab-based notebook workflow, along with fixed preprocessing steps and a consistent data format, helped maintain a consistent implementation pipeline across all experiments. Therefore, the reported

results should be interpreted as obtained under a controlled, reproducible experimental setting, while acknowledging that minor run-to-run variation may persist in GPU-based deep learning environments.

4. Results and Discussion

4.1. Selected N-BEATS Architecture and Baseline Performance

Fig. 5 illustrates the distribution of MAPE across all 45 experimental configurations and compares it with the target configuration (H, S = 3, N = 1). In the figure, the "Overall" group is represented by a box-and-whisker plot summarizing the MAPE distribution of all 45 runs, whereas the "Target" group is marked by a red dot corresponding to the MAPE value of the target configuration. This presentation provides a clear visual reference for positioning the target configuration within the overall experimental space.

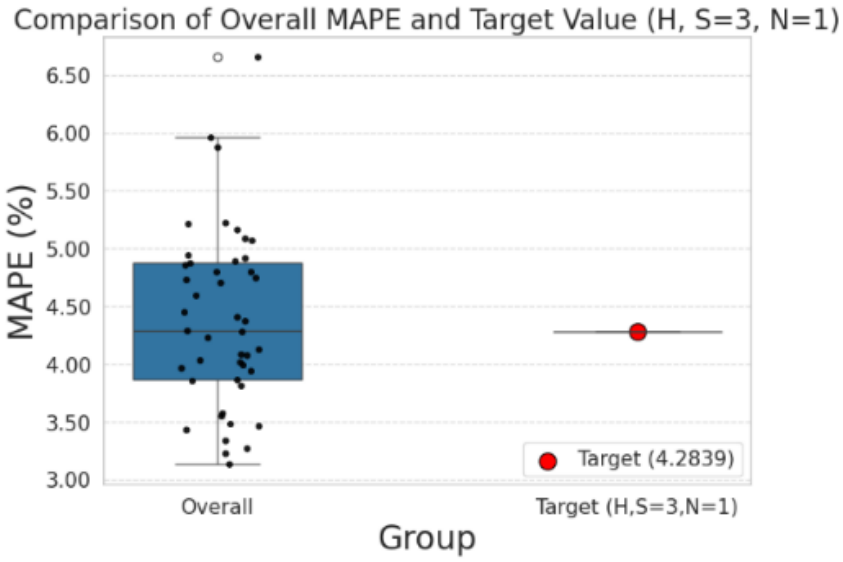

Fig. 5. Boxplot of MAPE for 45 Runs with Target Highlighted (H, S=3, N=1)

As shown in Fig. 5, the MAPE values exhibit moderate dispersion, with a median of approximately **4.29%** and an interquartile range from about **3.86% to 4.87%**. Several upper outliers exceed **6.65%**, suggesting configurations with substantially poorer predictive performance. By contrast, the minimum value of approximately **3.13%** suggests considerable room for improvement if a more suitable configuration is selected. The target configuration achieves a MAPE of **4.2839%**, which lies very close to the median and within the interquartile range, indicating that its performance is representative rather than optimal.

To complement the visual interpretation in Fig. 5, the distributional characteristics of MAPE across the 45 configurations are summarized in Table 4. This table reports the main descriptive statistics and provides a numerical basis for assessing the central tendency and variability of the results.

Table 4. Descriptive Statistics of MAPE Across 45 Configurations

Samples	Min	Q1 (25%)	Median (Q2)	Q3 (75%)	Max	Mean	Std
45	3.1300	3.8648	4.2852	4.8714	6.6544	4.3872	0.7689

As reported in Table 4, the mean MAPE is **4.3872%**, which is close to the median value of **4.2852%**, suggesting that most configurations are concentrated around the **4.3–4.4%** range. Nevertheless, the standard deviation of **0.7689%** indicates a noticeable degree of variability among configurations. Moreover, the wide gap between the minimum (**3.1300%**) and maximum (**6.6544%**) values further confirms that model configuration has a substantial impact on forecasting accuracy.

To further examine the influence of specific architectural choices, Table 5 compares three representative configurations: the default, best-performing, and worst-performing. In addition to MAPE, the table also reports **MAE, MSE, and RMSE**, thereby enabling a more comprehensive evaluation of model performance.

Table 5. MAPE and Error Metrics

Config	Variant	S	N	MAPE	MAE	MSE	RMSE
Default	H	3	1	4.2839	51.69	4136.68	64.32
Best (MAPE min)	G	2	1	3.1300	37.64	2636.39	51.35
Worst (MAPE max)	D	4	1	6.6544	79.93	9656.23	98.27

Table 5 shows that the default configuration **(H, 3, 1)** yields a MAPE of **4.2839%**, which is consistent with its near-median position in Fig. 5. Compared with the default setting, the best-performing configuration **(G, 2, 1)** reduces the MAPE to **3.1300%**, corresponding to an improvement of approximately **26.9%**. This improvement is also accompanied by substantial reductions in **MAE, MSE, and RMSE**, indicating a consistent enhancement across multiple error metrics. In contrast, the worst-performing configuration **(D, 4, 1)** produces a MAPE of **6.6544%**, representing a degradation of approximately **55.4%** relative to the default configuration, along with markedly higher MAE, MSE, and RMSE values. These results clearly demonstrate that architectural and parameter choices materially affect predictive accuracy, while also reinforcing the observations from **Figure 5** and **Table 4** that the target configuration reflects typical rather than optimal performance.

Following the configuration-level comparison in Table 5, model performance is further examined at the **variant level** through Fig. 6. This figure compares the three variants, **G, D, and H**, against the default baseline configuration **(H, S = 3, N = 1; MAPE = 4.2839%)**, thereby revealing differences in both the central tendency and dispersion of forecasting error across variants.

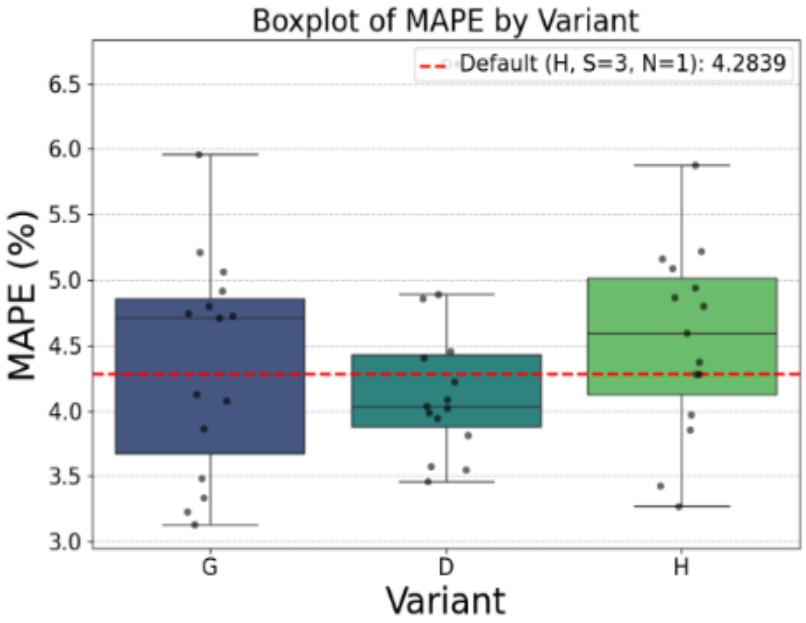

Fig. 6. Boxplot of MAPE by N-BEATS Variant ($G/D/H$) with Default Baseline ($H, S = 3, N = 1$)

As shown in Fig. 6, variant D exhibits the most favorable central tendency, with a median of 4.0378% and a mean of 4.2655%, both slightly lower than the baseline. Variant G achieves the lowest minimum MAPE (3.1300%), indicating its potential to deliver very high accuracy in some cases; however, its higher median and wider spread suggest lower consistency. By contrast, variant H shows a more concentrated distribution, but its higher mean and median indicate weaker overall accuracy than those of the other two variants. To further quantify the visual observations from Fig. 6, the detailed descriptive statistics for each variant are summarized in Table 6.

Table 6. Detailed MAPE statistics for each variant (D, G, H)

Statistic	D	G	H
Number of samples	15.0000	15.0000	15.0000
Mean	4.2655	4.3600	4.5360
Standard deviation	0.7868	0.8319	0.7117
Minimum	3.4634	3.1300	3.2719
Q1 (25%)	3.8797	3.6755	4.1262
Median (50%)	4.0378	4.7092	4.5952
Q3 (75%)	4.4307	4.8566	5.0175
Maximum	6.6544	5.9652	5.8804

As reported in Table 6, variant **D** provides the best typical performance, as reflected by its lowest mean and median among the three variants. However, its maximum value of **6.6544%** indicates that large forecasting errors may still occur in some cases. Variant **G** yields the best minimum value, but also has the largest standard deviation (**0.8319**), indicating greater variability across runs. In contrast, variant **H** has the smallest standard deviation (**0.7117**), suggesting relatively higher stability, but its larger mean and median imply poorer characteristic accuracy. Overall, variant **D** offers the strongest typical performance, **G** can achieve very good results but with lower consistency, and **H** is relatively stable but less accurate overall.

Following the variant-level comparison in Fig. 6 and Table 6, the analysis is further extended to examine the effect of the **number of blocks per stack (N)** on forecasting performance. Fig. 7 compares the MAPE distributions for different values of **N** relative to the default baseline configuration **(variant H, S = 3, N = 1; MAPE = 4.2839%)**, thereby illustrating how model depth influences both accuracy and variability.

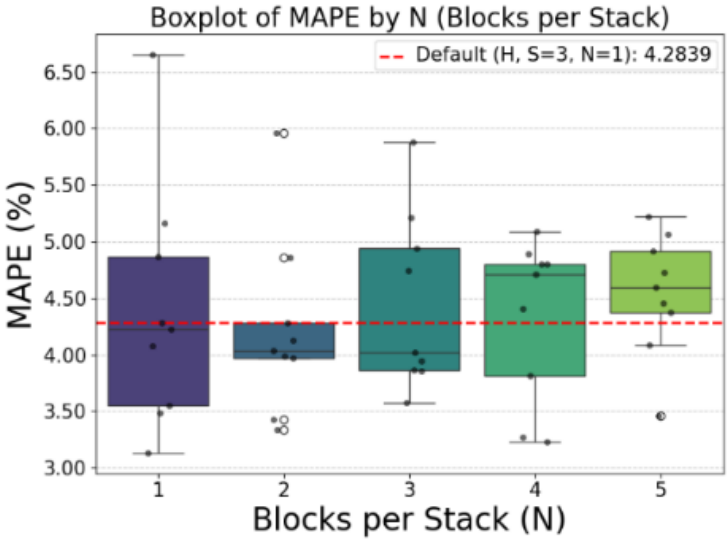

Fig. 7. Boxplot of MAPE by Number of Blocks per Stack (N), with Default Baseline

As shown in Fig. 7, the results suggest a clear trade-off between model depth and predictive performance. Moderate depths, particularly **N = 2** and **N = 3**, achieve lower median MAPE values (**4.0378%** and **4.0178%**, respectively) than the baseline, indicating better typical accuracy. In contrast, deeper settings (**N = 4** and **N = 5**) show higher medians and means, suggesting diminishing returns as the number of blocks increases. The baseline setting **N = 1** exhibits the widest spread, including the highest maximum value, indicating greater performance variability despite achieving the lowest minimum error.

To quantify these visual observations, the detailed descriptive statistics for each depth setting are summarized in Table 7. As reported in Table 7, **N = 2** and **N = 3** provide the most favorable balance between accuracy and robustness, with relatively low median and mean MAPE values. Although **N = 1** achieves the best minimum result (**3.1300%**), it also has the largest standard deviation (**1.0726**) and

the highest maximum (**6.6544%**), indicating unstable performance across runs. By comparison, **N = 4** and **N = 5** produce higher central error values, implying that excessive depth does not improve forecasting accuracy and may even reduce it. Overall, the results indicate that **moderate stack depth**, especially **N = 2–3**, is more effective and reliable than either shallow or overly deep configurations.

Table 7. Detailed MAPE Statistics by N (Transposed Form)

Statistic	*N = 1*	*N = 2*	*N = 3*	*N = 4*	*N = 5*
Number of samples	9.0000	9.0000	9.0000	9.0000	9.0000
Mean	4.3830	4.2225	4.4495	4.3354	4.5455
Standard deviation	1.0726	0.7919	0.7791	0.7143	0.5393
Minimum	3.1300	3.3393	3.5772	3.2299	3.4634
Q1 (25%)	3.5513	3.9684	3.8648	3.8154	4.3732
Median (50%)	4.2259	4.0378	4.0178	4.7092	4.5952
Q3 (75%)	4.8714	4.2852	4.9442	4.7983	4.9150
Maximum	6.6544	5.9652	5.8804	5.0908	5.2235

4.2. Impact of N-BEATS Hyperparameters on Forecasting Accuracy

Following the architectural analysis presented above, the effects of N-BEATS hyperparameters on forecasting accuracy are further examined. In this regard, Fig. 8 compares the overall MAPE distribution across all 125 hyperparameter configurations with the specific result for the target setting **max_steps = 1000** *(MAPE = 4.1025%)*. This comparison helps determine whether the target setting is optimal or simply a typical outcome within the broader hyperparameter space.

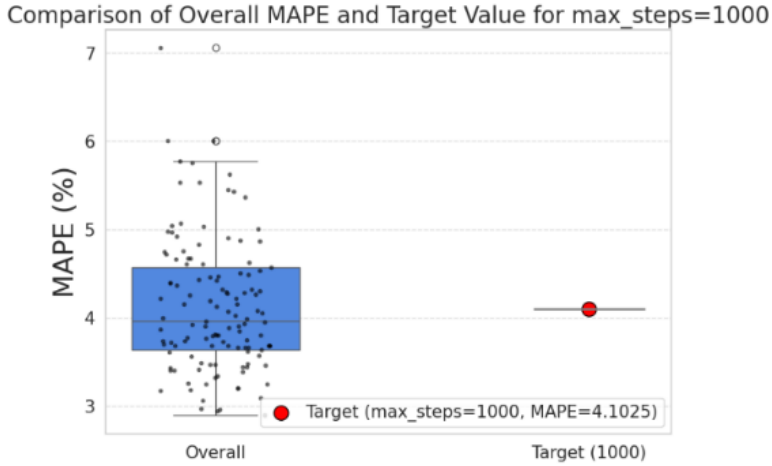

Fig. 8. Comparison of Overall MAPE and Target Value for max_steps = 1000)

As shown in Fig. 8, the target MAPE of **4.1025%** lies within the interquartile range and remains close to the median of the overall distribution. This indicates that **max_steps = 1000** provides representative rather than optimal performance. It is not associated with the best-performing configurations near the lower tail, nor does it fall among the clearly inferior cases in the upper tail.

To support the visual interpretation from Fig. 8, the descriptive statistics for all configurations are summarized in Table 8. As reported in Table 8, the overall MAPE distribution has a median of **3.9634%** and a mean of **4.1298%**, with moderate variability (**Std = 0.7513**). The target value of **4.1025%** is therefore slightly above the median but still very close to the distribution's center. Overall, these results suggest that **max_steps = 1000** is a stable and reasonable default setting, although it does not achieve the best forecasting accuracy among the 125 configurations tested.

Table 8. Detailed statistics for max_steps (default=1000)

Samples	Min	Q1	Q2	Q3	Max	Mean	Std	Target MAPE
125	2.8930	3.6368	3.9634	4.5723	7.0568	4.1298	0.7513	4.1025

Fig. 9 illustrates the distribution of MAPE (%) across different values of the *max_steps* parameter during model optimization. As shown in Fig. 9, the overall MAPE values are centered around approximately 4% across all tested settings. Among them, max_steps = 500 provides the best overall performance, with a relatively low central tendency and a compact distribution. The setting max_steps = 250 also performs competitively, showing a low median and limited dispersion, although it appears slightly less stable than 500. In contrast, larger values, including 750, 1000, and 1250, are associated with wider spreads and more frequent upper-tail errors, suggesting reduced robustness.

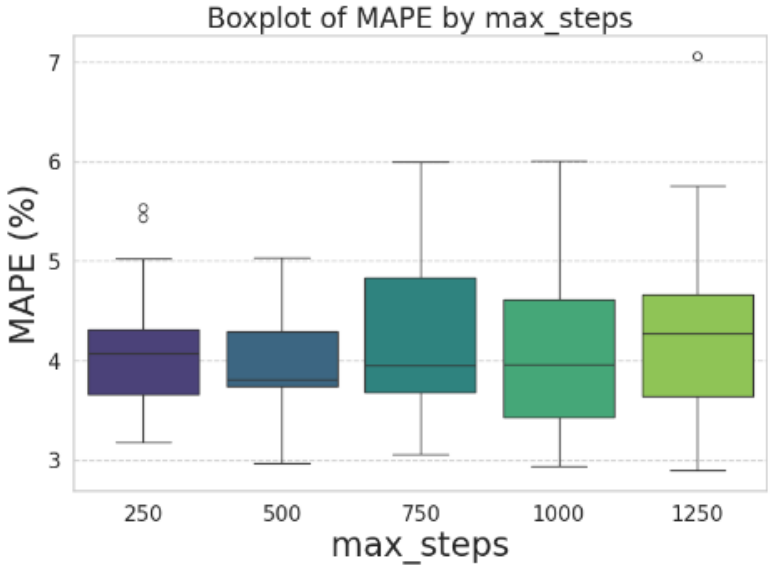

Fig. 9. Boxplot of MAPE (%) for different values of the max_steps parameter

To complement the visual comparison in Fig. 9, the descriptive statistics for each *max_steps* setting are summarized in Table 9. As reported in Table 9, **max_steps = 500** achieves the lowest mean MAPE (**3.9908%**), the smallest standard deviation (0.5054), and the narrowest interquartile range (**0.5506**), indicating the most concentrated and reliable error distribution among the tested settings. Although **max_steps = 250** also performs well, its mean and variability are slightly higher. By comparison, increasing **max_steps** to **750**, **1000**, and **1250** results in higher mean errors and noticeably greater dispersion, as reflected in wider IQRs and larger standard deviations. Overall, these results suggest that increasing the number of training steps does not necessarily improve forecasting accuracy and may even reduce stability; therefore, **max_steps = 500** is the most suitable choice among the evaluated settings.

Table 9. Descriptive statistics of MAPE (%) for different values of the max_steps parameter

max_steps	Count	Mean	Std	Min	Q1	Median	Q3	IQR	Max
250	25.0000	4.0802	0.6353	3.1801	3.6593	4.0703	4.3168	0.6574	5.5318
500	25.0000	3.9908	0.5054	2.9697	3.7393	3.8075	4.2899	0.5506	5.0377
750	25.0000	4.2334	0.8397	3.0631	3.6805	3.9468	4.8302	1.1497	6.0023
1000	25.0000	4.1025	0.7984	2.9394	3.4335	3.9634	4.6101	1.1766	6.0065
1250	25.0000	4.2419	0.9311	2.8930	3.6368	4.2722	4.6599	1.0232	7.0568

Fig. 10 illustrates the overall distribution of MAPE (%) across all tested configurations and highlights the specific result obtained at the target learning rate of **0.001**. As shown in Fig. 10, the target setting learning_rate = 0.001 yields a MAPE of 4.5225%, which lies in the upper portion of the overall distribution. This value is higher than both the median and the mean, indicating that the default learning rate provides acceptable but not optimal performance. Although it is not an outlier, it is clearly less competitive than many other tested configurations, suggesting that better learning-rate settings exist within the evaluated range.

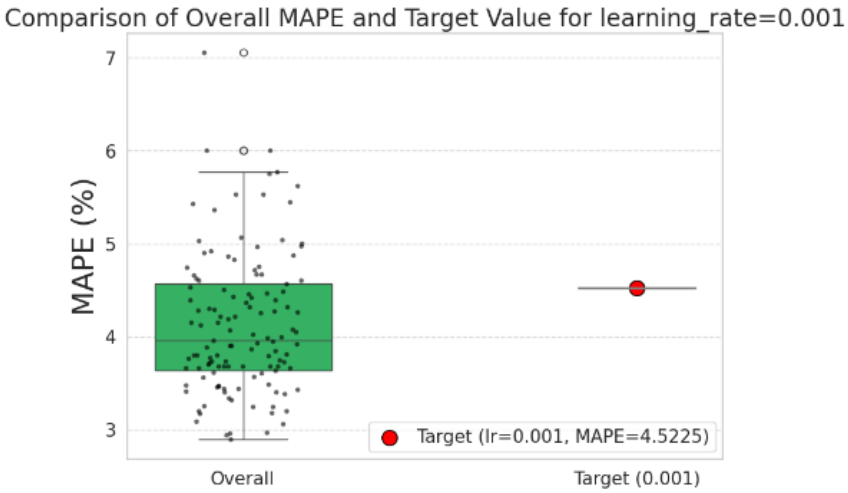

Fig. 10. Comparison of the overall MAPE distribution and the target MAPE value at learning_rate

To complement the visual comparison in Fig. 10, the descriptive statistics for the overall distribution and the target setting are summarized in Table 10. As **reported** in Table 10, the overall MAPE ranges from **2.8930%** to **7.0568%**, with a median of **3.9634%** and an interquartile range from **3.6368%** to **4.5723%**. The mean value is **4.1298%**, with a standard deviation of **0.7513%**, indicating moderate variability across configurations. In this context, the target MAPE of **4.5225%** is above both the median and the mean, confirming that **learning_rate = 0.001** lies in the upper-middle region of the distribution rather than the best-performing area. Overall, these results suggest that the default learning rate is workable, but suboptimal for achieving the highest forecasting accuracy.

Table 10. Detailed statistics for learning_rate (default=0.001)

Samples	Min	Q1	Median	Q3	Max	Mean	Std	MAPE
125	2.8930	3.6368	3.9634	4.5723	7.0568	4.1298	0.7513	4.5225

Fig. 11 illustrates the distribution of MAPE (%) across different values of the **learning_rate** hyperparameter during model optimization. As shown in Fig. 11, smaller learning rates, particularly in the range of 5×10^{-5} to 5×10^{-4}, generally produce lower MAPE values than larger ones. Among them, 1×10^{-4} achieves the lowest mean MAPE and maintains a relatively compact distribution, indicating favorable and stable performance. The setting 5×10^{-4} yields a slightly higher mean, but its narrower spread suggests the highest consistency across runs. In contrast, larger learning rates, especially 0.001 and 0.0025, are associated with higher central errors and greater dispersion, indicating reduced robustness.

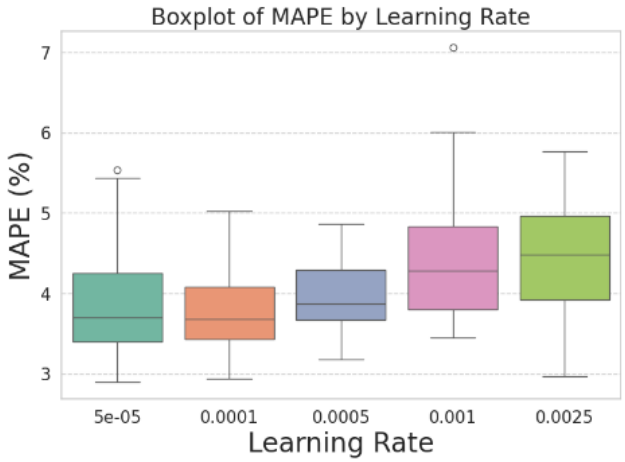

Fig. 11. Boxplot of MAPE (%) across different learning rate values

To complement the visual comparison in Fig. 11, the descriptive statistics for each learning_rate setting are summarized in Table 11. As reported in Table 11, **learning_rate = 0.0001** achieves the lowest mean MAPE (**3.8221%**) with relatively low dispersion, indicating the best overall balance between accuracy and stability. Although **learning_rate = 0.0005** has a slightly higher mean (**3.9644%**), it produces the smallest standard deviation (**0.4385**) and one of the narrowest interquartile ranges, reflecting the most consistent performance across trials. By comparison, increasing the learning rate to **0.001** and **0.0025** results in higher mean MAPE values and substantially larger variability, as evidenced by wider IQRs, larger standard deviations, and upper outliers extending to **7.0568%**. Overall, these findings suggest that the range 0.0001 to 0.0005 is most appropriate for this problem, providing a better balance between forecasting accuracy and stability than larger learning rates.

Table 11. MAPE statistics by learning_rate

learning_rate	Count	Mean	Std	Min	Q1	Median	Q3	IQR	Max
0.00005	25.0000	3.8730	0.6785	2.8930	3.4033	3.6966	4.2516	0.8483	5.5318
0.00010	25.0000	3.8221	0.5867	2.9394	3.4335	3.6852	4.0785	0.6450	5.0292
0.00050	25.0000	3.9644	0.4385	3.1801	3.6671	3.8672	4.2899	0.6228	4.8680
0.00100	25.0000	4.5225	0.9179	3.4452	3.8060	4.2839	4.8302	1.0242	7.0568
0.00250	25.0000	4.4670	0.7759	2.9697	3.9198	4.4851	4.9663	1.0465	5.7695

Fig. 12 illustrates the overall distribution of MAPE (%) across all tested configurations and highlights the specific result obtained for the target MLP setting **[512, 512]**. As shown in Fig. 12, the target MLP configuration [512, 512] yields a MAPE of 3.9680%, which is slightly below both the overall mean and the median. Its position in the lower part of the distribution indicates that this setting performs better than most of the tested configurations. This suggests that the [512, 512] architecture provides a favorable balance between model capacity and forecasting accuracy.

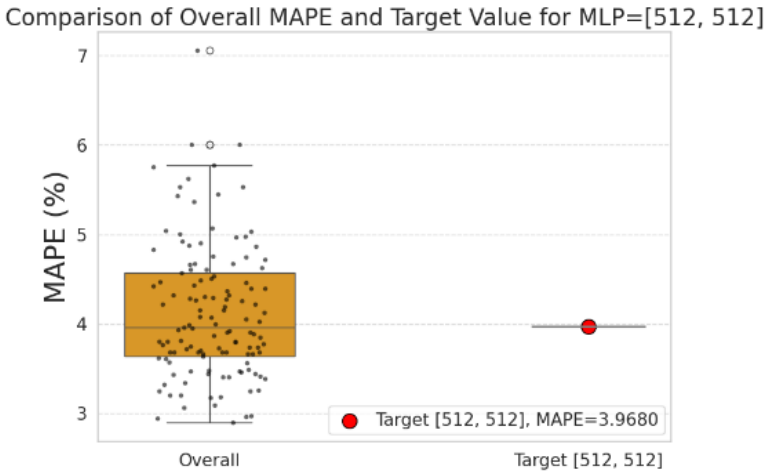

Fig. 12. Comparison of the overall MAPE distribution and the target MAPE value for the MLP

To complement the visual comparison in Fig. 12, the descriptive statistics for the overall sample and the target MLP setting are summarized in Table 12. As **reported** in Table 12, the overall MAPE ranges from **2.8930%** to **7.0568%**, with a median of **3.9634%** and an interquartile range from **3.6368%** to **4.5723%**, indicating moderate variability across configurations. In this context, the target MAPE of **3.9680%** is very close to the median and remains below the mean. Its proximity to the lower half of the distribution further confirms that the **[512, 512]** MLP setting is a competitive and effective choice, outperforming many alternatives without introducing excessive variability.

Table 12. Descriptive statistics of MAPE (%) for the overall sample

Samples	Min	Q1	Q2	Q3	Max	Mean	Std	MAPE
125	2.8930	3.6368	3.9634	4.5723	7.0568	4.1298	0.7513	3.9680

Fig. 13 illustrates the distribution of MAPE (%) across different MLP architectures, defined by the number of hidden units per layer. As shown in Fig. 13, forecasting performance varies noticeably across MLP widths. The configurations [512, 512] and [768, 768] achieve the lowest mean MAPE values, indicating better overall accuracy than the other tested settings. Among them, [512, 512] appears more favorable because it combines low error with a relatively moderate spread. In contrast, [128, 128], [256, 256], and [640, 640] show higher central error values, while [256, 256] also exhibits more pronounced upper-tail variation. These patterns suggest that intermediate-to-large MLP widths can improve performance, although excessive width may reduce stability.

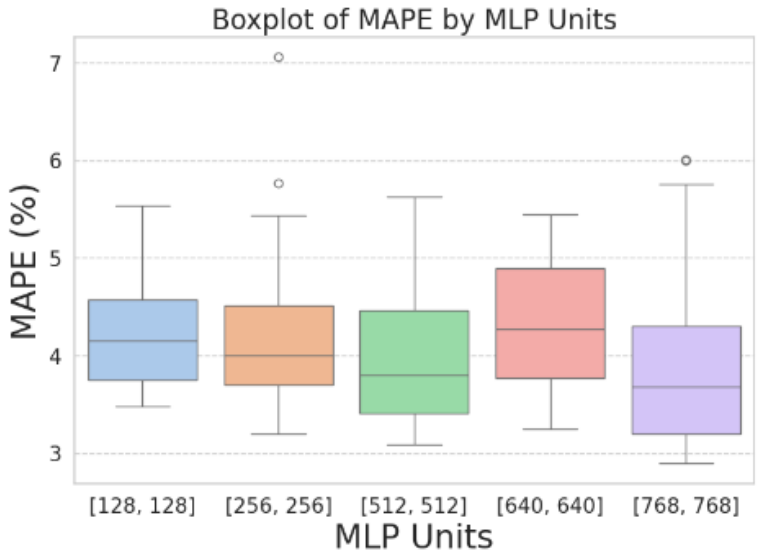

Fig. 13. Boxplot of MAPE (%) for different MLP hidden-unit configurations

To complement the visual comparison in Fig. 13, the descriptive statistics for each MLP configuration are summarized in Table 13. As **reported** in Table 13, **[512, 512]** yields the lowest median MAPE (**3.8060%**) and one of the lowest mean values (**3.9680%**), while maintaining moderate variability (**Std = 0.6338**). Although **[768, 768]** achieves a similar mean (**3.9699%**), its substantially larger standard deviation (**0.9803**) indicates lower consistency across runs. By comparison, **[128, 128]**, **[256, 256]**, and **[640, 640]** all produce higher mean errors, and **[256, 256]** shows the largest maximum value (**7.0568%**), reflecting a greater risk of poor outcomes. Overall, these results suggest that **[512, 512]** offers the most favorable balance between forecasting accuracy and robustness among the evaluated MLP configurations.

Table 13. Descriptive statistics of MAPE (%) for each MLP hidden-unit configuration

mlp_str	Count	Mean	Std	Min	Q1	Median	Q3	IQR	Max
[128, 128]	25	4.1741	0.5281	3.4849	3.7469	4.1499	4.5723	0.8255	5.5318
[256, 256]	25	4.2769	0.8719	3.2020	3.6966	3.9968	4.5096	0.8130	7.0568
[512, 512]	25	3.9680	0.6338	3.0862	3.4135	3.8060	4.4636	1.0501	5.6249
[640, 640]	25	4.2600	0.6468	3.2529	3.7659	4.2722	4.8995	1.1336	5.4470
[768, 768]	25	3.9699	0.9803	2.8930	3.1976	3.6835	4.3029	1.1053	6.0065

5. Conclusion

This study presented a systematic, controlled sensitivity-analysis framework to examine how architectural and training hyperparameters influence N-BEATS' forecasting performance in short-term load forecasting. Rather than proposing a new forecasting architecture or conducting cross-model benchmarking, the study focused on clarifying the relative effects of key N-BEATS design choices under a transparent two-stage experimental procedure.

Within the evaluated search space, the results show that N-BEATS performance is strongly affected by both architectural and training hyperparameters. For the architectural group, the Generic variant exhibited the most favorable performance profile for the considered Tasmania dataset, while moderate block depths (N_blocks = 2 or 3) generally provided a better balance between accuracy and stability than shallower or deeper settings. For the training group, max_steps = 500 yielded the most stable overall behavior, learning rates in the range of 1×10^(-4) to 5×10^(-4) provided the best trade-off between convergence and robustness, and an MLP width of [512, 512] offered a favorable balance between predictive accuracy and model stability. These findings indicate that careful hyperparameter selection can substantially improve N-BEATS forecasting performance relative to default settings.

From a practical perspective, the results provide a useful deployment guideline for practitioners who wish to configure a new N-BEATS forecasting node. For a similar daily peak-load forecasting task, a reasonable starting point is to adopt the Generic variant, use a moderate block depth, initialize training with max_steps = 500, test learning rates between 1×10^(-4) and 5×10^(-4), and select a medium-to-large hidden-layer width such as [512, 512]. These settings should be interpreted as empirically supported starting ranges rather than universally optimal values, and they should be validated on the target dataset before operational deployment.

Several limitations of the present study should also be acknowledged. First, the analysis was conducted on a single public dataset from Tasmania, which limits the generalizability of the findings to other regions and load characteristics. Second, the forecasting evaluation was based on a short 7-day test horizon, so the reported results should be interpreted within this specific short-term setting. In addition, the input window length was kept fixed throughout the experiments, so the effect of alternative lookback settings was not examined in the present study. Third, a staged sensitivity analysis design was chosen to improve interpretability. Still, it does not fully capture all possible interaction effects between architectural and training hyperparameters and therefore does not guarantee a global optimum over the full configuration space. In addition, the purpose of this work was not to compare N-BEATS against other forecasting model families, but to understand the internal sensitivity of N-BEATS itself.

Future research may extend this framework by evaluating multiple datasets, using rolling or broader validation schemes, investigating longer forecasting horizons, and examining multivariate load-forecasting settings. Further work may also incorporate formal statistical significance testing and automated hyperparameter optimization methods after the main sensitivity patterns have first been established through controlled analysis. Overall, this study provides practical, interpretable evidence on how key hyperparameters shape N-BEATS forecasting behavior and lays a foundation for more robust configuration and tuning of N-BEATS in future load-forecasting applications.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] S. Karamolegkos and D. E. Koulouriotis, "Advancing short-term load forecasting with decomposed Fourier ARIMA: A case study on the Greek energy market," *Energy*, vol. 325, p. 135854, Jun. 2025, https://doi.org/10.1016/j.energy.2025.135854.

[2] J. A. C. Dias, P. H. D. V. Guimarães, W. G. S. Pereira, L. D. O. Tamasauskas, M. S. Gomes, A. B. S. Corrêa, K. Figueiredo, G. Costa, G. B. Costa, F. A. R. Costa, and M. C. D. R. Seruffo, "Enhanced carbon flux forecasting via STL decomposition and hybrid ARIMA-ES-LSTM model in Amazon forest," *IEEE Access*, vol. 13, pp. 84713–84726, 2025, https://doi.org/10.1109/ACCESS.2025.3561166.

[3] A. Gupta and A. Kumar, "Mid term daily load forecasting using ARIMA, wavelet-ARIMA and machine learning," in *2020 IEEE International Conference on Environment and Electrical Engineering and 2020 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 2020, pp. 1–5, https://doi.org/10.1109/EEEIC/ICPSEurope49358.2020.9160563.

[4] U. Samal and A. Kumar, "Enhancing software reliability forecasting through a hybrid ARIMA-ANN model," *Arabian Journal for Science and Engineering*, vol. 49, no. 5, pp. 7571–7584, 2024, https://doi.org/10.1007/s13369-023-08486-1.

[5] G. R. A. Brito, A. Rivero Villaverde, A. L. Quan, and M. E. R. Pérez, "Comparison between SARIMA and Holt–Winters models for forecasting monthly streamflow in the western region of Cuba," *SN Applied Sciences*, vol. 3, no. 6, p. 671, 2021, https://doi.org/10.1007/s42452-021-04667-5.

[6] V. Gayathry, D. Kaliyaperumal, and S. R. Salkuti, "Seasonal solar irradiance forecasting using artificial intelligence techniques with uncertainty analysis," *Scientific Reports*, vol. 14, no. 1, pp. 1–19, 2024, https://doi.org/10.1038/s41598-024-68531-3.

[7] A. P. Piyal, S. Ahmed, K. F. Rahman, and A. S. M. Mohsin, "Energy demand forecasting using machine learning perspective in Bangladesh," in *2023 IEEE IAS Global Conference on Renewable Energy and Hydrogen Technologies (GlobConHT)*, 2023, pp. 1–5, https://doi.org/10.1109/GlobConHT56829.2023.10087679.

[8] S. Cantillo-Luna, R. Moreno-Chuquen, and J. A. Lopez Sotelo, "Intra-day electricity price forecasting based on a Time2Vec-LSTM neural network model," in *2023 IEEE Colombian Conference on Applied Computational Intelligence (ColCACI)*, 2023, pp. 1–6, https://doi.org/10.1109/ColCACI59285.2023.10225803.

[9] N. Drop and A. Bohdan, "Application of the Holt–Winters model in the forecasting of passenger traffic at Szczecin–Goleniów Airport (Poland)," *Sustainability*, vol. 17, no. 14, p. 6407, 2025, https://doi.org/10.3390/su17146407.

[10] J. C. García-Díaz and Ó. Trull, Eds., *Advanced Methods of Power Load Forecasting*. Basel, Switzerland: MDPI, 2022, https://doi.org/10.3390/books978-3-0365-4217-1.

[11] P. F. Pai and W. C. Hong, "Forecasting regional electricity load based on recurrent support vector machines with genetic algorithms," *Electric Power Systems Research*, vol. 74, no. 3, pp. 417–425, 2005, https://doi.org/10.1016/j.epsr.2005.01.006.

[12] A. Moradzadeh, S. Zakeri, M. Shoaran, B. Mohammadi-Ivatloo, and F. Mohammadi, "Short-term load forecasting of microgrid via hybrid support vector regression and long short-term memory algorithms," *Sustainability*, vol. 12, no. 17, p. 7076, 2020, https://doi.org/10.3390/su12177076.

[13] X. Qiu, L. Zhang, Y. Ren, P. N. Suganthan, and G. Amaratunga, "Ensemble deep learning for regression and time series forecasting," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2014, pp. 1–6, https://doi.org/10.1109/CIEL.2014.7015739.

[14] L. A. Abad, S. M. Sarabia, J. M. Yuzon, and M. C. Pacis, "A short-term load forecasting algorithm using support vector regression and artificial neural network method (SVR-ANN)," in *Proceedings of the 11th IEEE Control Systems Graduate Research Colloquium (ICSGRC)*, 2020, pp. 138–143, https://doi.org/10.1109/ICSGRC49013.2020.9232630.

[15] J. Luo, Y. Zheng, T. Hong, A. Luo, and X. Yang, "Fuzzy support vector regressions for short-term load forecasting," *Fuzzy Optimization and Decision Making*, vol. 23, no. 3, pp. 363–385, 2024, https://doi.org/10.1007/s10700-024-09425-x.

[16] Z. Zhang, Q. Zhang, H. Liang, and B. Gorbani, "Optimizing electric load forecasting with support vector regression/LSTM optimized by flexible gorilla troops algorithm and neural networks: A case study," *Scientific Reports*, vol. 14, no. 1, pp. 1–20, 2024, https://doi.org/10.1038/s41598-024-73893-9.

[17] S. Sreekumar, J. Verma, A. Sujil, and R. Kumar, "One day ahead forecasting of hourly electrical load using genetically tuned support vector regression for smart grid frame work," in *Proceedings of the 2nd International Conference on Recent Advances in Engineering and Computational Sciences (RAECS)*, 2015, pp. 1–6, https://doi.org/10.1109/RAECS.2015.7453405.

[18] Y. Lu and G. Wang, "A load forecasting model based on support vector regression with whale optimization algorithm," *Multimedia Tools and Applications*, vol. 82, no. 7, pp. 9939–9959, 2023, https://doi.org/10.1007/s11042-022-13462-2.

[19] S. Deng, X. Dong, L. Tao, J. Wang, Y. He, and D. Yue, "Multi-type load forecasting model based on random forest and density clustering with the influence of noise and load patterns," *Energy*, vol. 307, p. 132635, 2024, https://doi.org/10.1016/j.energy.2024.132635.

[20] C. Wang, J. Zhang, L. Tian, L. Xue, Y. Zheng, and L. Liu, "Short-term load forecasting based on k-prototypes clustering and random forest," in *Proceedings of the 5th IEEE Conference on Energy Internet and Energy System Integration (EI2)*, 2021, pp. 1226–1230, https://doi.org/10.1109/EI252483.2021.9712977.

[21] T. Kavzoglu and A. Teke, "Predictive performances of ensemble machine learning algorithms in landslide susceptibility mapping using random forest, extreme gradient boosting (XGBoost) and natural gradient boosting (NGBoost)," *Arabian Journal of Science and Engineering*, vol. 47, no. 6, pp. 7367–7385, 2022, https://doi.org/10.1007/s13369-022-06560-8.

[22] U. Singh and S. Vadhera, "Random forest and XGBoost technique for short-term load forecasting," in *Proceedings of the 1st International Conference on Sustainable Technology in Power and Energy Systems (STPES)*, 2022, pp. 1–6, https://doi.org/10.1109/STPES54845.2022.10006635.

[23] V. Veeramsetty, K. R. Reddy, M. Santhosh, A. Mohnot, and G. Singal, "Short-term electric power load forecasting using random forest and gated recurrent unit," *Electrical Engineering*, vol. 104, no. 1, pp. 307–329, 2022, https://doi.org/10.1007/s00202-021-01376-5.

[24] M. Abdurohman and A. G. Putrada, "Forecasting model for lighting electricity load with a limited dataset using XGBoost," *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control*, vol. 8, no. 2, pp. 571–580, 2023, https://doi.org/10.22219/kinetik.v8i2.1687.

[25] D. Barochiner, R. Lado, L. Carletti, and F. Pintar, "A machine learning approach to address 1-week-ahead peak demand forecasting using the XGBoost algorithm," in *Proceedings of the IEEE Biennial Congress of Argentina (ARGENCON)*, 2022, pp. 1–5, https://doi.org/10.1109/ARGENCON55245.2022.9939986.

[26] S. Lei, X. Liang, X. Xia, H. Dai, C. Zhang, X. Ge, and F. Wang, "A two-stage short-term load forecasting method based on comprehensive similarity day selection and CEEMDAN-XGBoost error correction," in *2023 International Conference on Future Energy Solutions (FES)*, 2023, pp. 1–6, https://doi.org/10.1109/FES57669.2023.10183307.

[27] D.-J. Bae, B.-S. Kwon, and K.-B. Song, "XGBoost-based day-ahead load forecasting algorithm considering behind-the-meter solar PV generation," *Energies*, vol. 15, no. 1, p. 128, 2022, https://doi.org/10.3390/en15010128.

[28] L. Zhang and D. Jánošík, "Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches," *Expert Systems with Applications*, vol. 241, p. 122686, 2024, https://doi.org/10.1016/j.eswa.2023.122686.

[29] N. Shabbir, R. Ahmadiahangar, A. Rosin, M. Jawad, J. Kilter, and J. Martins, "XGBoost-based short-term electrical load forecasting considering trends and periodicity in historical data," in *Proceedings of the IEEE International Conference on Energy Technologies for Future Grids (ETFG)*, 2023, pp. 1–6, https://doi.org/10.1109/ETFG55873.2023.10407926.

[30] Y. Miao, J. Zhu, H. Dong, Z. Chen, S. Li, and X. Wen, "Short-term load forecasting based on echo state network and LightGBM," in *Proceedings of the IEEE International Conference on Predictive Control of Electrical Drives and Power Electronics (PRECEDE)*, 2023, pp. 1–6, https://doi.org/10.1109/PRECEDE57319.2023.10174609.

[31] S. Lei, X. Liang, X. Wang, J. Ding, X. Ge, F. Wang, and J. Feng, "A short-term net load forecasting method based on two-stage feature selection and LightGBM with hyperparameter auto-tuning," in *2023 IEEE/IAS 59th Industrial and Commercial Power Systems Technical Conference (I&CPS)*, 2023, pp. 1–6, https://doi.org/10.1109/ICPS57144.2023.10142095.

[32] X. Liang, Y. Feng, J. Jiang, W. Wang, X. Liu, and Z. Gong, "Short-term load forecasting of a technology park based on a LightGBM-LSTM fusion algorithm," in *2022 IEEE 5th International Conference on*

Automation, Electronics and Electrical Engineering (AUTEEE), 2022, pp. 151–155, https://doi.org/10.1109/AUTEEE56487.2022.9994355.

[33] X. Yao, X. Fu, and C. Zong, "Short-term load forecasting method based on feature preference strategy and LightGBM-XGBoost," *IEEE Access*, vol. 10, pp. 75257–75268, 2022, https://doi.org/10.1109/ACCESS.2022.3192011.

[34] S. Lei, X. Liang, X. Xia, H. Dai, J. Ding, X. Ge, and F. Wang, "Net load segmented forecasting method for data center based on GS-LightGBM model," in *2023 IEEE IAS Global Conference on Renewable Energy and Hydrogen Technologies (GlobConHT)*, 2023, pp. 1–6, https://doi.org/10.1109/GlobConHT56829.2023.10087415.

[35] J. H. Kim, B. S. Lee, and C. H. Kim, "A study on the development of long-term hybrid electrical load forecasting model based on MLP and statistics using massive actual data considering field applications," *Electric Power Systems Research*, vol. 221, p. 109415, Aug. 2023, https://doi.org/10.1016/j.epsr.2023.109415.

[36] A. I. Arvanitidis, D. Bargiotas, A. Daskalopulu, D. Kontogiannis, I. P. Panapakidis, and L. H. Tsoukalas, "Clustering informed MLP models for fast and accurate short-term load forecasting," *Energies*, vol. 15, no. 4, p. 1295, 2022, https://doi.org/10.3390/en15041295.

[37] T. Bashir, H. Wang, M. Tahir, and Y. Zhang, "Wind and solar power forecasting based on hybrid CNN-ABiLSTM and CNN-transformer-MLP models," *Renewable Energy*, vol. 239, p. 122055, Feb. 2025, https://doi.org/10.1016/j.renene.2024.122055.

[38] A. Ajitha, M. Goel, M. Assudani, S. Radhika, and S. Goel, "Design and development of residential sector load prediction model during COVID-19 pandemic using LSTM-based RNN," *Electric Power Systems Research*, vol. 212, p. 108635, Nov. 2022, https://doi.org/10.1016/j.epsr.2022.108635.

[39] J. Mo, R. Wang, M. Cao, K. Yang, X. Yang, and T. Zhang, "A hybrid temporal convolutional network and Prophet model for power load forecasting," *Complex & Intelligent Systems*, vol. 9, no. 4, pp. 4249–4261, 2023, https://doi.org/10.1007/s40747-022-00952-x.

[40] A. Ali and E. A. Jasmin, "Deep learning networks for short-term load forecasting," in *2023 International Conference on Control, Communication and Computing (ICCC)*, 2023, pp. 1–5, https://doi.org/10.1109/ICCC57789.2023.10165216.

[41] N. A. Mohammed and A. Al-Bazi, "An adaptive backpropagation algorithm for long-term electricity load forecasting," *Neural Computing and Applications*, vol. 34, no. 1, pp. 477–491, 2022, https://doi.org/10.1007/s00521-021-06384-x.

[42] N. Bacanin, V. Simic, M. Zivkovic, M. Alrasheedi, and A. Petrovic, "Cloud computing load prediction by decomposition reinforced attention long short-term memory network optimized by modified particle swarm optimization algorithm," *Annals of Operations Research*, 2023, https://doi.org/10.1007/s10479-023-05745-0.

[43] O. Rubasinghe, X. Zhang, T. K. Chau, Y. H. Chow, T. Fernando, and H. H. C. Iu, "A novel sequence to sequence data modeling based CNN-LSTM algorithm for three years ahead monthly peak load forecasting," *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1932–1947, 2024, https://doi.org/10.1109/TPWRS.2023.3271325.

[44] J.-W. Xiao, X.-Y. Cui, X.-K. Liu, H. Fang, and P.-C. Li, "Improved 3D LSTM: A video prediction approach to long sequence load forecasting," *IEEE Transactions on Smart Grid*, vol. 16, no. 2, pp. 1885-1896, 2024, https://doi.org/10.1109/TSG.2024.3458989.

[45] H. Liu, Z. Li, C. Li, L. Shao, and J. Li, "Research and application of short-term load forecasting based on CEEMDAN-LSTM modeling," *Energy Reports*, vol. 12, pp. 2144–2155, 2024, https://doi.org/10.1016/j.egyr.2024.08.035.

[46] X. Chen, M. Yang, Y. Zhang, J. Liu, and S. Yin, "Load prediction model of integrated energy system based on CNN-LSTM," in *2023 3rd International Conference on Energy Engineering and Power Systems (EEPS)*, 2023, pp. 248–251, https://doi.org/10.1109/EEPS58791.2023.10257124.

[47] M. Abumohsen, A. Y. Owda, and M. Owda, "Electrical load forecasting using LSTM, GRU, and RNN algorithms," *Energies*, vol. 16, no. 5, pp. 1–31, 2023, https://doi.org/10.3390/en16052283.

[48] F. Y. Li and J. F. Xiao, "Multi-channel LSTM-CNN power load multi-step prediction based on feature fusion," in *2022 5th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)*, 2022, pp. 591–594, https://doi.org/10.1109/WCMEIM56910.2022.10021531.

[49] I. Jrhilifa, H. Ouadi, A. Jilbab, and N. Mounir, "Forecasting smart home electricity consumption using VMD-Bi-GRU," *Energy Efficiency*, vol. 17, no. 4, p. 35, 2024, https://doi.org/10.1007/s12053-024-10205-0.

[50] G. Emmanouilidis, M. Tzelepi, and A. Tefas, "Short-term electric load demand forecasting on Greek energy market using deep learning: A comparative study," in *2022 Panhellenic Conference on Electronics and Telecommunications (PACET)*, 2022, pp. 1–4, https://doi.org/10.1109/PACET56979.2022.9976351.

[51] Q. Hua, Z. Fan, W. Mu, J. Cui, R. Xing, H. Liu, and J. Gao, "A short-term power load forecasting method using CNN-GRU with an attention mechanism," *Energies*, vol. 18, no. 1, p. 106, 2025, https://doi.org/10.3390/en18010106.

[52] B. Chen, W. Yang, B. Yan, and K. Zhang, "An advanced airport terminal cooling load forecasting model integrating SSA and CNN-Transformer," *Energy and Buildings*, vol. 309, p. 114000, 2024, https://doi.org/10.1016/j.enbuild.2024.114000.

[53] M. Alhussein, K. Aurangzeb, and S. I. Haider, "Hybrid CNN-LSTM model for short-term individual household load forecasting," *IEEE Access*, vol. 8, pp. 180544–180557, 2020, https://doi.org/10.1109/ACCESS.2020.3028281.

[54] L. Wensheng, W. Kuihua, F. Liang, L. Hao, W. Yanshuo, and C. Can, "A region-level integrated energy load forecasting method based on CNN-LSTM model with user energy label differentiation," in *2020 5th International Conference on Power and Renewable Energy (ICPRE)*, 2020, pp. 154–159, https://doi.org/10.1109/ICPRE51194.2020.9233226.

[55] J. Ran, Y. Gong, Y. Hu, and J. Cai, "EV load forecasting using a refined CNN-LSTM-AM," *Electric Power Systems Research*, vol. 238, p. 111091, 2025, https://doi.org/10.1016/j.epsr.2024.111091.

[56] Q. Zhang, S. Zhou, B. Xu, and X. Li, "TCAMS-Trans: Efficient temporal-channel attention multi-scale transformer for net load forecasting," *Computers and Electrical Engineering*, vol. 118, p. 109415, 2024, https://doi.org/10.1016/j.compeleceng.2024.109415.

[57] L. Li, X. Su, X. Bi, Y. Lu, and X. Sun, "A novel transformer-based network forecasting method for building cooling loads," *Energy and Buildings*, vol. 296, p. 113409, 2023, https://doi.org/10.1016/j.enbuild.2023.113409.

[58] A. Ahmad, X. Xiao, H. Mo, and D. Dong, "TFTformer: A novel transformer based model for short-term load forecasting," *International Journal of Electrical Power and Energy Systems*, vol. 166, p. 110549, 2025, https://doi.org/10.1016/j.ijepes.2025.110549.

[59] Z. Tian, W. Liu, W. Jiang, and C. Wu, "CNNs-Transformer based day-ahead probabilistic load forecasting for weekends with limited data availability," *Energy*, vol. 293, p. 130666, 2024, https://doi.org/10.1016/j.energy.2024.130666.

[60] O. Bouhamed, M. Dissem, M. Amayri, and N. Bouguila, "Transformer-based deep probabilistic network for load forecasting," *Engineering Applications of Artificial Intelligence*, vol. 152, p. 110781, 2025, https://doi.org/10.1016/j.engappai.2025.110781.

[61] A. B. Habib, M. G. R. Alam, and M. Z. Uddin, "AUNET (Attention-Based Unified Network): Leveraging attention-based N-BEATS for enhanced univariate time series forecasting," *IEEE Access*, vol. 13, pp. 95184–95217, 2025, https://doi.org/10.1109/ACCESS.2025.3574459.

[62] P. Nedić, I. Djurović, M. Ćalasan, S. Kovačević, and K. Pavlović, "Electrical energy load forecasting using a hybrid N-BEATS-CNN approach: Case study Montenegro," *Electric Power Systems Research*, vol. 247, p. 111749, 2025, https://doi.org/10.1016/j.epsr.2025.111749.

[63] A. Motavali, K.-C. Yow, N. Hansmeier, and T.-C. Chao, "DSA-BEATS: Dual self-attention N-BEATS model for forecasting COVID-19 hospitalization," *IEEE Access*, vol. 11, pp. 137352–137365, 2023, https://doi.org/10.1109/ACCESS.2023.3318931.

[64] F. C. de Lima Duarte, P. S. G. de Mattos Neto, and P. R. A. Firmino, "A hybrid recursive direct system for multi-step mortality rate forecasting," *The Journal of Supercomputing*, vol. 80, no. 13, pp. 18430–18463, 2024, https://doi.org/10.1007/s11227-024-06182-x.