HYPERPARAMETER SENSITIVITY ANALYSIS OF AUTOFORMER FOR SHORT-TERM ELECTRIC LOAD FORECASTING

TUAN ANH NGUYEN[1]*, THANH NGOC TRAN[2], THANH THUAN NGUYEN[3]
[1,2,3] *Faculty of Electrical Engineering Technology, Industrial University of Ho Chi Minh City, Ho Chi Minh City, Vietnam,*
** Corresponding author: nguyenanhtuan@iuh.edu.vn*
DOIs: https://www.doi.org/ 10.46242/jstiuh.v83i5. 6183

Abstract. Short-term electric load forecasting is a key task in power system operation, electricity market scheduling, energy management, and demand-side planning. Accurate short-term forecasts help operators anticipate demand variations, reduce reserve requirements, improve economic dispatch, and maintain system reliability. In recent years, Transformer-based forecasting models have attracted increasing attention due to their ability to capture temporal dependencies in time-series data. Autoformer is one of the representative Transformer variants designed for time-series forecasting through series decomposition and an auto-correlation mechanism. However, the performance of Autoformer is highly affected by hyperparameter settings, especially those controlling model representation capacity, attention structure, and optimization behavior. This paper investigates the sensitivity of Autoformer performance to three key hyperparameters: hidden size, number of attention heads, and learning rate. A grid-based experimental framework is developed for short-term load forecasting using the New South Wales electricity demand dataset with 30-minute temporal resolution. The forecasting task is configured for a one-day-ahead prediction, with the model forecasting 48 future time steps using a historical input window of 7 days. Each hyperparameter is evaluated at 5 candidate values, yielding 125 experimental trials. Model performance is assessed using mean absolute error, mean absolute percentage error, mean squared error, and root mean squared error. Among the 125 designed configurations, 100 were completed successfully, while 25 with hidden_size = 512 failed due to GPU memory limitations. The best configuration, consisting of hidden_size = 128, n_head = 4, and learning_rate = 0.0005, achieved an MAE of 372.70 MW, a MAPE of 4.9978%, an MSE of 289,690.53, and an RMSE of 538.23 MW. The sensitivity results indicate that learning_rate has the strongest marginal influence, while the effects of hidden_size and n_head depend substantially on their interactions with the other hyperparameters. The proposed framework provides a systematic basis for selecting Autoformer configurations while considering forecasting accuracy, variability, and computational feasibility.

Keywords. Autoformer, electric load forecasting, hyperparameter sensitivity, grid search, time-series forecasting.

1 INTRODUCTION

Electric load forecasting is an important task in the planning and operation of modern power systems. Electricity demand changes continuously over time and is influenced by many factors, including customer consumption behavior, weather conditions, working days and holidays, industrial activities, and seasonal factors. Therefore, accurate load forecasting helps system operators properly schedule generation resources, reduce operating costs, avoid unnecessary reserve margins, and maintain power system stability.

Among different forecasting horizons, short-term load forecasting plays a particularly important role because it directly supports operational decisions from several minutes to several days ahead. Forecasting results are used for power dispatch, operation planning, electricity market management, and supply–demand balancing. When forecasting errors are large, the system may need to maintain a higher reserve level or face operational instability risks. Therefore, improving the accuracy of short-term load forecasting remains a problem of both practical and scientific significance.

Many methods have been proposed for electric load forecasting. Traditional methods are mainly based on statistical and mathematical models, such as moving average, exponential smoothing, linear regression, AR, MA, ARMA, ARIMA, and SARIMA [1-3]. These methods have the advantages of simple structure, easy implementation, low computational cost, and relatively clear interpretability. However, they often assume linear relationships or a certain level of data stationarity. As a result, their forecasting performance

© 2026 Industrial University of Ho Chi Minh City

may decrease when the load series is nonlinear, non-stationary, noisy, and varies across multiple time scales.

In addition, machine learning methods such as Support Vector Regression [4-5], Decision Tree [6-7], Random Forest [8-9], k-Nearest Neighbors [10], Gradient Boosting [11-12], and XGBoost [13-14] have been widely used to improve the ability to model nonlinear relationships in load data. These methods are more flexible than traditional statistical models and can exploit various input features. However, their performance depends heavily on manual feature construction and feature selection. Moreover, conventional machine learning models still have limitations in automatically learning temporal dependencies, especially long-term dependencies in load time series.

Recently, deep learning models have attracted increasing attention in electric load forecasting due to their ability to automatically learn features from data. Representative models include Multilayer Perceptron [15-16], Convolutional Neural Network [17-18], Recurrent Neural Network [19-21], Long Short-Term Memory [22], Gated Recurrent Unit [23-24], and Transformer-based models [25-27]. MLP can approximate nonlinear relationships but does not effectively exploit the sequential structure of time-series data. CNNs are effective at extracting local features and short-term variation patterns, but they may be limited when modeling long-term dependencies. RNNs, LSTMs, and GRUs are suitable for sequential data; however, their training can be time-consuming and difficult when the input sequence is long. Meanwhile, Transformer uses an attention mechanism to learn relationships across different time points in a sequence, allowing it better to capture long-term dependencies and repeated patterns in load data.

Although deep learning models have achieved promising results, several research gaps remain. Models such as CNNs, LSTMs, and GRUs can learn short-term features effectively, but they may not fully capture long-term trends, seasonality, and recurring cycles simultaneously. Traditional Transformer models can learn long-term dependencies, but they often incur high computational costs and are not specifically designed for the trend–seasonality decomposition of time-series data. In addition, many studies evaluate models using only a fixed hyperparameter configuration, while the influence of hyperparameters on forecasting accuracy has not been sufficiently analyzed.

In this context, Autoformer [28-30] is considered a suitable approach for short-term electric load forecasting. It is an improved Transformer-based architecture specifically designed for time-series forecasting. Autoformer integrates a decomposition mechanism to separate trend and seasonal components, while using an auto-correlation mechanism to identify periodic relationships. These characteristics are well-suited to electric load data, which often exhibit daily and weekly cycles and recurring consumption patterns. However, the performance of Autoformer depends significantly on hyperparameter selection. Parameters such as hidden size, number of attention heads, and learning rate can directly affect the model's representation capacity, ability to learn temporal relationships, training stability, and computational cost. Therefore, selecting hyperparameters solely on experience or trial-and-error may lead to suboptimal results and undermine the reliability of experimental conclusions.

Based on these issues, this study selects Autoformer as the model for short-term electric load forecasting and focuses on evaluating the influence of key hyperparameters on forecasting performance. Specifically, three hyperparameters, namely hidden_size, n_head, and learning_rate, are investigated using a grid-search strategy. The New South Wales electricity demand dataset is used to evaluate the model in a one-day-ahead forecasting task, with an input window of 7 days of historical load data and an output of the next 48 time steps. This approach aims to clarify Autoformer's sensitivity to different hyperparameter configurations and to provide an experimental basis for selecting suitable configurations for short-term load forecasting. The objective of this study is not to establish the superiority of Autoformer over other forecasting architectures. Instead, the study conducts a controlled within-model evaluation to isolate the effects of hidden_size, n_head, and learning_rate on forecasting accuracy, computational time, and memory feasibility. A cross-model benchmark would require separate model-specific architectures, hyperparameter spaces, and optimization protocols, which are outside the scope of the present hyperparameter-sensitivity analysis. The main contributions of this study can be summarized as follows.

- First, an Autoformer-based short-term electric load forecasting framework is developed for the one-day-ahead forecasting task.

- Second, the effects of three important hyperparameters, including hidden_size, n_head, and learning_rate, are systematically analyzed.
- Third, the experimental procedure is designed to be reproducible, in which trial-level results are stored for further analysis and comparison.
- Fourth, the study identifies the accuracy–cost–feasibility trade-offs among Autoformer configurations without claiming performance superiority over other forecasting architectures.

2 METHODOLOGY

2.1 Autoformer Model and Architecture

In this study, the Autoformer model is employed for short-term electric load forecasting in a multi-step forecasting manner. The model is implemented using the Nixtla NeuralForecast library, where the input data are organized using three main fields: unique_id, ds, and y. The unique_id field identifies the time series, ds represents the observation timestamp, and y denotes the electric load value to be forecasted. In the experimental code, the load sequence is sampled at a 30-minute frequency. The model uses seven days of historical load data as input and predicts the next 48 time steps, corresponding to the load profile of the following day. The main model settings include h = 48, input_size = 336, freq = '30min', and the model is initialized through the Autoformer class in the NeuralForecast library.

Let the historical electric load sequence be denoted as:

$$Y = \{y_1, y_2, \ldots, y_t, \ldots, y_N\} \tag{1}$$

Where y_t is the electric load value at time step t, and $N\ is$ the total number of observations in the time series. To formulate the supervised forecasting problem, a historical input window with length p is used as the model input:

$$X_t = [y_{t-p+1}, y_{t-p+2}, \ldots, y_t] \tag{2}$$

where p is the number of time steps in the input sequence. Since the data have a 30-minute resolution, one day contains 48 observations. In this study, the model uses seven historical days; therefore, the input window length is defined as: $p = 48 \times 7 = 336$

The objective of Autoformer is to map the input sequence $X_t$into a future load sequence:

$$\hat{Y}_{t+1:t+H} = [\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+H}] \tag{3}$$

Where H is the forecasting horizon. For one-day-ahead forecasting, the forecasting horizon is set to $H = 48$. Thus, the forecasting mapping of the model can be generally expressed as:

$$\hat{Y}_{t+1:t+H} = f_{\text{Autoformer}}(X_t; \theta) \tag{4}$$

where $f_{\text{Autoformer}}(\cdot)$ denotes the forecasting function learned by the Autoformer model, and θ represents the set of model parameters and hyperparameters. In this study, the investigated hyperparameters include hidden_size, n_head, and learning_rate, while the main forecasting settings, such as h, input_size, and max_steps, are kept fixed during the experiments.

In terms of architecture, Autoformer is a Transformer-based model designed for time-series forecasting. The main difference between Autoformer and the standard Transformer is that Autoformer does not rely only on point-wise attention across time steps. Instead, it incorporates a series decomposition mechanism and an autocorrelation mechanism. This design is suitable for electric load data because load demand typically exhibits long-term trends, recurring daily or weekly patterns, and local fluctuations driven by changes in electricity consumption behavior.

Conceptually, a load time series can be decomposed as follows:

$$x_t = T_t + S_t + e_t \tag{5}$$

where T_t represents the trend component, S_t represents the seasonal or periodic component, and e_t denotes the residual or random fluctuation at time step t. In Autoformer, the series decomposition is commonly implemented using a moving average to separate the trend component from the original sequence. Given an input sequence X, the trend component can be estimated as:

$$X_{\text{trend}} = \text{AvgPool}(X) \tag{6}$$

where AvgPool(·) denotes the moving average operation over the input sequence, the remaining component, which generally represents seasonal or periodic variation, is obtained by:

$$X_{\text{seasonal}} = X - X_{\text{trend}} \tag{7}$$

Therefore, the input sequence can be represented as the sum of two main components:

$$X = X_{\text{trend}} + X_{\text{seasonal}} \tag{8}$$

This decomposition mechanism allows Autoformer to handle trend variations and periodic fluctuations separately. For electric load forecasting, this is important because load series often exhibit repeated intraday patterns while also containing gradual changes and irregular variations over time.

In addition to series decomposition, Autoformer employs an auto-correlation mechanism to capture repeated temporal patterns in time-series data. Instead of measuring the relationship between every pair of time steps as in conventional self-attention, the auto-correlation mechanism focuses on identifying time delays with high similarity. Given the query Q, key K, and value V representations the auto-correlation coefficient at delay τcan be expressed as:

$$R_{Q,K}(\tau) = \frac{1}{L}\sum_{t=1}^{L} Q_t\, K_{t-\tau} \tag{9}$$

where L is the sequence length, τ is the time delay, Q_t is the query representation at time step t, and $K_{t-\tau}$ is the delayed key representation. The delays with the highest correlation values are selected to identify important repeated patterns in the sequence:

$$\{\tau_1, \tau_2, \ldots, \tau_k\} = \text{TopK}(R_{Q,K}) \tag{10}$$

After the significant delays are identified, the output of the auto-correlation mechanism is obtained by aggregating the value sequences shifted according to these delays:

$$\text{AutoCorrelation}(Q, K, V) = \sum_{i=1}^{k} \alpha_i \cdot \text{Roll}(V, \tau_i) \tag{11}$$

where α_i is the weight associated with delay τ_i, and $\text{Roll}(V, \tau_i)$ denotes the operation of shifting the value sequence V by the corresponding delay. This mechanism enables Autoformer to learn periodic dependencies more effectively, especially when load data contains repeated consumption patterns every 48 time steps, corresponding to one day.

In the Autoformer architecture, the auto-correlation mechanism can be extended into a multi-head structure, allowing the model to learn different types of temporal dependencies in parallel. For the i-th head, the query, key, and value representations are computed as:

$$Q_i = XW_i^Q, K_i = XW_i^K, V_i = XW_i^V \tag{12}$$

where W_i^Q, W_i^K, and $W_i^V$are trainable weight matrices. The output of each head is obtained as:

$$head_i = \text{AutoCorrelation}(Q_i, K_i, V_i) \tag{13}$$

The outputs of all heads are then combined to form the final representation:

$$\text{MultiHead}(X) = \text{Concat}(head_1, head_2, \ldots, head_m)W^O \tag{14}$$

where m is the number of auto-correlation heads, and W^O is the output projection matrix. In the experimental code, the number of heads is controlled by the hyperparameter n_head, while the model's internal representation dimension is controlled by hidden_size. These two hyperparameters directly influence the model's representation capacity and its ability to capture temporal dependencies. In addition, learning_rate controls the update step for the parameter during training.

The final output of Autoformer is formed by combining the predicted trend and seasonal components:

$$\hat{Y} = \hat{Y}_{\text{trend}} + \hat{Y}_{\text{seasonal}} \tag{15}$$

where $\hat{Y}_{\text{trend}}$ is the predicted trend component, and $\hat{Y}_{\text{seasonal}}$ is the predicted seasonal or periodic component. With this structure, Autoformer can model both long-term trends and recurring temporal patterns in electric load data. Therefore, it is suitable for short-term load forecasting problems in which the data exhibit strong periodic characteristics, particularly the daily cycle, with 48 observations per day.

2.2 Hyperparameters of the Autoformer Model in Nixtla

In this study, Autoformer is implemented using the Nixtla NeuralForecast library. The library provides a practical implementation of Autoformer with a set of configurable hyperparameters related to forecasting setup, model architecture, data processing, and training optimization. These hyperparameters determine how the model receives historical information, represents temporal dependencies, performs decomposition, and updates its parameters during training.

The first group of hyperparameters pertains to the forecasting configuration. The forecasting horizon, denoted as h, defines the number of future time steps to be predicted. The input_size parameter determines the length of the historical autoregressive input window used by the model. In the context of short-term load forecasting, these two parameters are directly related to the forecasting task definition. For half-hourly load data, a one-day-ahead forecasting horizon corresponds to h = 48, while using seven days of historical load data corresponds to input_size = 336.

The second group consists of architecture-related hyperparameters. The hidden_size parameter controls the dimensionality of the embedding and encoder representations. A larger hidden size generally increases the model's representation capacity, but it may also increase the number of trainable parameters and the computational cost. The n_head parameter controls the number of heads in the multi-head auto-correlation or attention structure. Multiple heads allow the model to learn different temporal dependency patterns in parallel. Other architectural parameters include encoder_layers, decoder_layers, conv_hidden_size, dropout, factor, activation, and MovingAvg_window. These parameters affect the depth of the encoder-decoder structure, the convolutional feature representation, the regularization behavior, and the moving-average decomposition used to separate trend and seasonal components.

The third group concerns training and optimization. The learning_rate parameter determines the step size of parameter updates during training and directly influences convergence behavior. A small learning rate may lead to slow convergence, whereas a large learning rate may cause unstable training. Other optimization-related parameters include max_steps, batch_size, windows_batch_size, loss, valid_loss, num_lr_decays, early_stop_patience_steps, optimizer, and lr_scheduler. These parameters control the number of training iterations, the amount of data processed per batch, the objective function, the validation strategy, and the learning rate scheduling.

Therefore, the Autoformer implementation in Nixtla NeuralForecast provides a flexible hyperparameter space for adapting the model to different forecasting problems. In this study, these hyperparameters are first reviewed from the implementation perspective, while the experimental section focuses on a selected subset of key parameters for systematic evaluation. Table 1 presents the main Autoformer hyperparameters in the Nixtla NeuralForecast library and their default values.

Table 1: Main Autoformer hyperparameters in the Nixtla NeuralForecast library

Group	Hyperparameter	Description	Default value
Forecasting configuration	h	Forecasting horizon, i.e., the number of future time steps to be predicted.	Required
	input_size	Length of the historical input sequence used by the model.	Required
	step_size	Step size between temporal windows.	1
	scaler_type	Type of scaler used for temporal input normalization.	identity
Model architecture	hidden_size	Number of units used in embeddings and encoder representations.	128

	n_head	Number of heads in the multi-head attention/autocorrelation structure.	4
	dropout	Dropout rate is used throughout the Autoformer architecture.	0.05
	factor	A factor controlling the sparse dependency selection mechanism.	3
	conv_hidden_size	Number of channels in the convolutional encoder.	32
	activation	Activation function used in the model.	gelu
	encoder_layers	Number of encoder layers.	2
	decoder_layers	Number of decoder layers.	1
	MovingAvg_window	Window size of the moving average filter for series decomposition.	25
Training and optimization	loss	Training loss function.	MAE()
	valid_loss	Validation loss function.	None
	max_steps	Maximum number of training steps.	5000
	learning_rate	Learning rate used for parameter updates.	0.0001
	num_lr_decays	Number of learning rate decays across training steps.	-1
	early_stop_patience_steps	Number of validation checks before early stopping.	-1
	val_check_steps	Number of training steps between validation checks.	100
	batch_size	Number of time series processed in each batch.	32
	windows_batch_size	Number of windows sampled in each training batch.	1024
	random_seed	Random seed for model initialization and data processing.	1

2.3 Workflow of the Hyperparameter Impact Evaluation

Fig. 1 presents the overall algorithmic flowchart for evaluating Autoformer hyperparameters in the short-term load forecasting task. The diagram illustrates the complete experimental procedure, from the original load-time series data through data preprocessing, dataset construction, and train/test splitting. It then shows how the forecasting problem is formulated using the training and testing sets, together with the predefined Autoformer configuration.

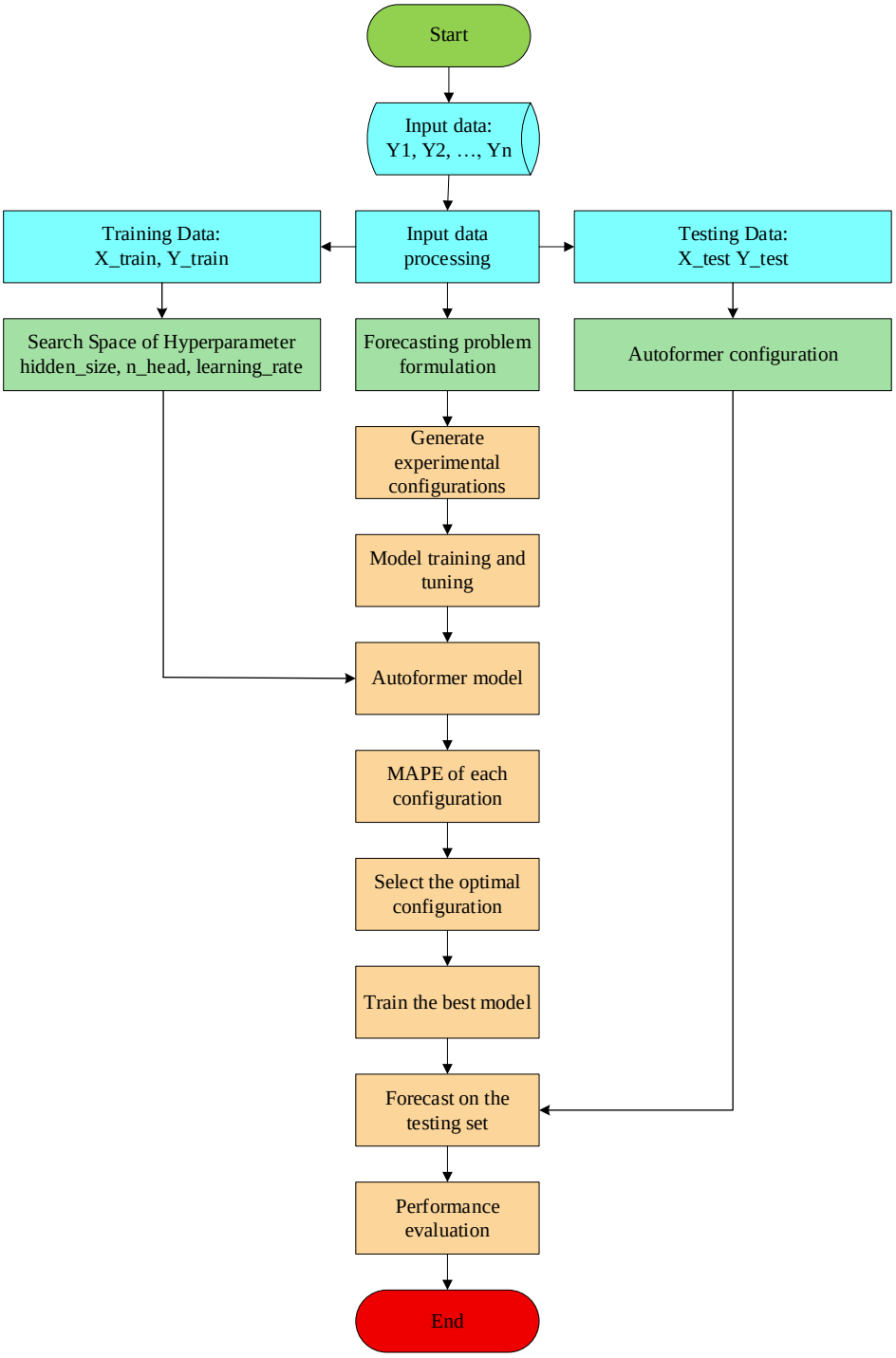


Figure 1: Experimental workflow for Autoformer hyperparameter sensitivity analysis.

The hyperparameter search space, including hidden_size, n_head, and learning_rate, is used to generate different experimental configurations for model training and tuning. For each configuration, the Autoformer model is trained, evaluated, and compared using MAPE. Finally, the configuration with the lowest MAPE is selected as the optimal setting, and the corresponding model is used for forecasting and final performance evaluation. This structure ensures that all candidate configurations are assessed under the same data conditions and forecasting horizon. In addition, separating training and test data helps prevent information leakage during model evaluation. The flowchart also highlights the role of the test set in evaluating the predictive performance of the selected model. Overall, the procedure provides a systematic and reproducible framework for analyzing the impact of Autoformer hyperparameters on forecasting accuracy.

As shown in Fig. 1, the process starts with the input load time-series data, denoted by Y1, Y2, ..., Yn. The data are first processed to ensure that they are suitable for time-series forecasting. After preprocessing, the dataset is divided into two parts: the training set $(X_{\text{train}}, Y_{\text{train}})$ and the testing set $(X_{\text{test}}, Y_{\text{test}})$. The training set is used to build and tune the Autoformer model, while the testing set is reserved for evaluating the forecasting performance of the selected model.

The flowchart also shows that the hyperparameter search space is constructed using three selected hyperparameters: hidden_size, n_head, and learning_rate. These parameters are used to generate different experimental configurations. At the same time, the Autoformer configuration and the forecasting problem formulation are defined before model training. Each generated configuration is then used to train and tune an Autoformer model, and the MAPE value of each configuration is recorded. Based on these trial results, the configuration with the lowest MAPE is selected as the optimal configuration.

After the optimal configuration is identified, the best Autoformer model is trained and then used to generate forecasts on the testing set. Finally, the forecasting performance is evaluated using standard error metrics, including MAPE, MAE, MSE, and RMSE. Overall, the flowchart clearly illustrates the complete experimental procedure used in this study, from data preparation and hyperparameter search to optimal model selection and final performance evaluation.

3 EXPERIMENTAL SETUP

3.1 Dataset Description and Preprocessing

The experimental dataset used in this study is the New South Wales electricity demand dataset. The dataset contains chronological electricity load observations recorded at 30-minute intervals. It consists of two main columns: SETTLEMENTDATE, which represents the timestamp of each observation, and TOTALDEMAND, which represents the total electricity demand in megawatts. The full dataset contains 122,735 observations, covering the period from 1 January 2015 at 00:30 to 31 December 2021 at 23:30. The demand values range from 4,316.63 MW to 13,985.87 MW, with an average load of approximately 7,890.88 MW.

Before model training, the dataset was preprocessed to ensure consistency with the time-series forecasting task. First, the column names were standardized by removing unnecessary spaces. The SETTLEMENTDATE column was then converted to the datetime format using the month/day/year hour:minute format. The observations were then sorted in ascending chronological order to preserve the temporal structure of the load sequence. This step is important because random shuffling is not suitable for time-series forecasting and can lead to information leakage between the training and test sets.

For the experimental design, the last 29 days of data were selected. The first 28 days, comprising 1,344 half-hourly observations, were used as the training set, while the final day, with 48 observations, was used as the test set. The training period extended from 3 December 2021 at 00:00 to 30 December 2021 at 23:30, whereas the testing period corresponded to 31 December 2021 from 00:00 to 23:30. The supervised samples were generated using a chronological rolling-window scheme. With input_size = 336 and h = 48, each sample consisted of the previous seven days of half-hourly load observations and the subsequent 48 observations to be forecasted. Using a window step size of 1, the 1,344-observation training segment could form $1{,}344 - 336 - 48 + 1 = 961$ candidate supervised windows. This sample-generation procedure preserved the temporal order of the data and prevented future testing observations from being included in the training inputs.

Figure 2 illustrates the chronological distribution of the NSW electricity demand dataset and the train/test split used in this study. The full load series is shown over the period from 2015 to 2021, while the training and testing segments are highlighted at the end of the time series.

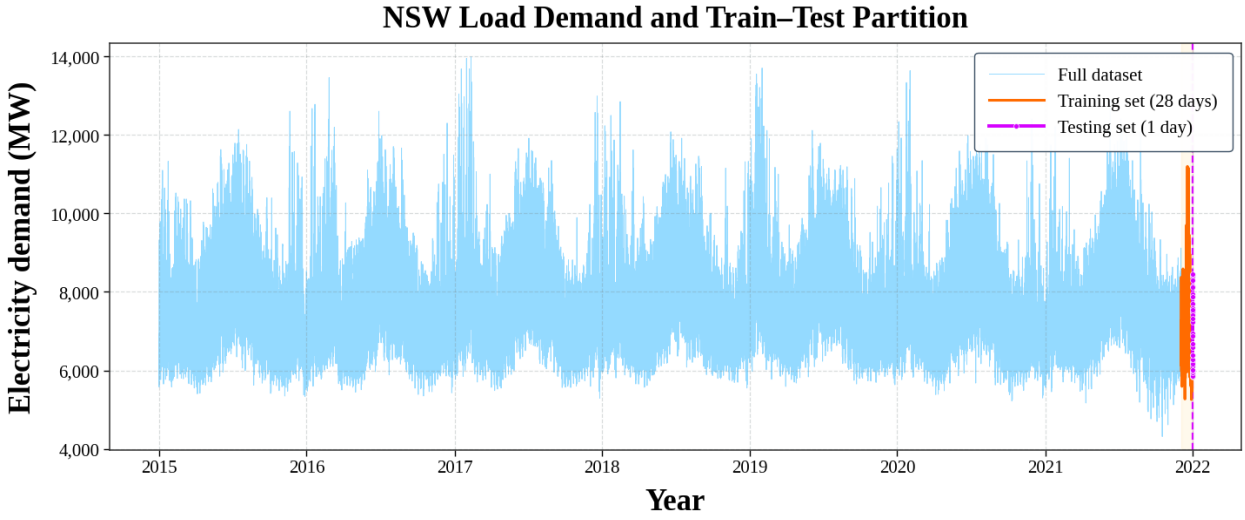


Figure 2: Chronological NSW load series and the final 29-day train–test partition.

As shown in Figure 2, the NSW electricity demand series exhibits clear temporal variation over the entire observation period. The load values fluctuate widely, reflecting daily consumption patterns, seasonal effects, and changes in electricity demand over time. The blue curve represents the complete dataset, the orange segment represents the 28-day training period, and the green segment represents the one-day testing period. Since both training and testing data are selected from the end of the time series, the split preserves the temporal order required for one-day-ahead forecasting. This design allows the model to learn from the most recent load behavior and avoids information leakage from future observations.

For compatibility with the Nixtla NeuralForecast library, the data were transformed into the required input format with three fields: unique_id, ds, and y. The unique_id field identifies the time series and was set to NSW for all observations. The ds field stores the timestamp, while the y field contains the target electricity demand value. This formatted dataset was then used as the input for the Autoformer model. Table 2 summarizes the main characteristics of the NSW electricity demand dataset and the experimental data split. In this study, only the historical load series y was used as the predictive input. The ds field preserved the chronological timestamp, whereas unique_id identified the single NSW load series required by the NeuralForecast framework; therefore, these two fields were not treated as explanatory variables. This univariate input design was intentionally selected to isolate the effects of hidden_size, n_head, and learning_rate without introducing additional variability from exogenous-feature selection or feature engineering. The seven-day input window was selected to include seven complete daily cycles, with 48 half-hourly observations per day, thereby representing both intraday periodicity and short-term weekly load regularities. Weather, holiday, and other calendar-related variables were excluded to maintain a controlled hyperparameter-sensitivity experiment.

Table 2: Summary of the NSW electricity demand dataset

Item	Description
Dataset	New South Wales electricity demand
Time column	SETTLEMENTDATE
Target column	TOTALDEMAND
Time resolution	30 minutes
Total observations	122,735
Full data period	01/01/2015 00:30 – 31/12/2021 23:30
Minimum demand	4,316.63 MW
Maximum demand	13,985.87 MW
Mean demand	7,890.88 MW
Training period	03/12/2021 00:00 – 30/12/2021 23:30
Testing period	31/12/2021 00:00 – 31/12/2021 23:30
Training size	1,344 observations

Testing size	48 observations
Forecasting horizon	48 time steps
Input window	336 time steps

This preprocessing procedure ensures that the input data are chronologically ordered, properly formatted, and consistent with the forecasting configuration used in the Autoformer experiment.

3.2 Configuration of the Hyperparameter Search Space

The hyperparameter search space was designed to assess the Autoformer model's sensitivity to various architectural and optimization settings. In this study, three representative hyperparameters were selected for systematic investigation: hidden_size, n_head, and learning_rate. These parameters were chosen because they directly affect two important aspects of the model: the Autoformer's representation capacity and the stability of the training process.

The first selected hyperparameter is hidden_size, which controls the dimensionality of the Autoformer model's internal hidden representations. This parameter is closely related to the model's capacity to learn complex temporal patterns from historical load data. A smaller hidden size may reduce computational cost, but may not provide sufficient representation capacity to capture nonlinear load variations. In contrast, a larger hidden size can improve the model's ability to represent complex daily and weekly demand patterns. Still, it may also increase the number of trainable parameters, computational time, and the risk of overfitting. Therefore, hidden_size was selected to examine whether increasing the capacity of the internal representation improves short-term load forecasting accuracy.

The second selected hyperparameter is n_head, which determines the number of parallel attention or auto-correlation heads used in the Autoformer architecture. In time-series forecasting, different heads can learn different temporal dependency patterns from the input sequence. For electric load data, such patterns may include intraday variations, peak-load behavior, low-load periods, and repeated daily cycles. A small number of heads may limit the diversity of temporal features learned by the model, whereas too many heads may increase computational complexity without necessarily improving accuracy. Thus, n_head was included in the search space to analyze how the number of parallel temporal dependency representations affects the forecasting performance of Autoformer.

The third selected hyperparameter is learning_rate, which controls the step size of parameter updates during model training. This parameter strongly influences convergence behavior and training stability. A very small learning rate may result in slow convergence and insufficient learning within the fixed training budget. Conversely, a very large learning rate may cause unstable optimization and poor forecasting accuracy. Since the objective of this study is to compare different Autoformer configurations in a controlled experimental setting, it is necessary to investigate the effect of the learning rate on final forecasting performance.

A grid search strategy was adopted to define the experimental search space. Each selected hyperparameter was assigned five candidate values. The values were chosen to cover small, moderate, and large configurations, allowing the experiment to observe how the forecasting accuracy changes when the model capacity, temporal dependency modeling, and optimization step size are varied. In the implementation, the hyperparameter grid contains five values each for hidden_size, n_head, and learning_rate, as shown in Table 3. The grid is generated using all possible combinations of these values, and each configuration is trained and evaluated independently under the same forecasting setting. The code defines the forecasting horizon as h = 48, the input window as input_size = 48 × 7 = 336, and the training limit as max_steps = 500, while the three selected hyperparameters are varied across the grid search.

Table 3: Hyperparameter search space used in the Autoformer experiment

Hyperparameter	Candidate values	Reason for selection
hidden_size	{32, 64, 128, 256, 512}	Controls the dimensionality of internal representations and reflects the model capacity

n_head	{1, 2, 4, 8, 16}	Controls the number of parallel attention/auto-correlation heads for learning temporal dependencies
learning_rate	{1e-5, 5e-5, 1e-4, 5e-4, 1e-3}	Controls the step size of parameter updates and affects convergence stability

The total number of experimental configurations is calculated as 125.

Where N_{trials} denotes the total number of Autoformer runs. For each trial, only hidden_size, n_head, and learning_rate are varied, while the remaining experimental settings are kept fixed. This controlled design ensures that the observed differences in forecasting accuracy and computational time are mainly due to the selected hyperparameters rather than to changes in the dataset, forecasting horizon, or training setup. In the implementation, all trial results, prediction details, error values, and execution time are stored for later ranking and sensitivity analysis.

3.3 Google Colab Environment and Nixtla Library

The experiments were conducted in the Google Colab environment, which provides a convenient cloud-based platform for implementing and testing deep learning models. Google Colab was selected because it supports Python-based data processing, model training, and result visualization without requiring a local high-performance computing setup. In this study, the implementation uses common scientific computing libraries, including pandas for data manipulation, numpy for numerical computation, matplotlib for visualization, itertools for generating hyperparameter combinations, and time for recording the execution time of each experimental run. The implementation also uses garbage collection and GPU memory clearing after each trial to reduce memory accumulation during repeated model training.

The Autoformer model was implemented using the Nixtla NeuralForecast library. This library provides a unified framework for deep learning-based time-series forecasting models and allows the Autoformer model to be initialized with user-defined forecasting and training parameters. In the experimental implementation, the model is imported from neuralforecast.models, while the forecasting pipeline is managed using the NeuralForecast class. The input data are formatted according to the NeuralForecast requirement, where unique_id identifies the time series, ds stores the timestamp, and y represents the target load value. In this study, the unique_id is set to NSW, the ds column corresponds to settlementdate, and the y column corresponds to totaldemand.

For each hyperparameter configuration, an Autoformer model is initialized with a forecasting horizon of 48 and an input window length of input_size = 336. These settings correspond to one-day-ahead forecasting using seven previous days of half-hourly load data. The model is trained using the formatted training dataset and then used to generate the next 48 predicted load values. The frequency parameter is set to 30 minutes to match the temporal resolution of the NSW electricity demand dataset. The three investigated hyperparameters, namely hidden_size, n_head, and learning_rate, are passed directly into the Autoformer constructor for each trial, while max_steps is fixed at 500 to limit the computational burden of the grid-search experiment.

After each training process, the model produces a forecast dataframe. The predicted values are then aligned with the test data using both the series identifier and the timestamp. This alignment step ensures that each predicted load value corresponds exactly to the correct actual observation in the testing period. If missing predicted values are detected after merging, the implementation raises an error, because incomplete forecast alignment would make the subsequent error calculation invalid. This procedure helps ensure consistency and reliability in the evaluation of all Autoformer configurations.

3.4 Evaluation Metrics

The forecasting performance of each Autoformer configuration was evaluated using four standard error metrics: mean absolute error, mean absolute percentage error, mean squared error, and root mean squared error. These metrics were selected because they provide complementary perspectives on forecasting accuracy. MAE and RMSE express the forecasting error in the original load unit, while MAPE represents the relative error as a percentage. MSE and RMSE are also useful because they penalize large forecasting

errors more strongly. In the implementation, MAE, MAPE, and MSE are computed using functions from the sklearn. metrics module, while RMSE is calculated as the square root of MSE.

Let y_i and $\hat{y}_i$denote the actual and predicted load values at the i-th time step of the testing horizon, respectively. Let n be the total number of predicted points. In this study, $n = 48$, corresponding to the 48 half-hourly observations of the testing day. The mean absolute error is defined as:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} | y_i - \hat{y}_i | \tag{16}$$

MAE measures the average magnitude of the forecasting error, ignoring its direction. A smaller MAE indicates that the predicted values are closer to the actual load values on average.

The mean absolute percentage error is calculated as:

$$MAPE = \frac{100}{n}\sum_{i=1}^{n} | \frac{y_i - \hat{y}_i}{y_i} | \tag{17}$$

MAPE expresses the forecasting error as a percentage of the actual value. This metric is useful for comparing forecasting performance in relative terms and is used as the main ranking criterion in this study.

The mean squared error is given by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{18}$$

MSE emphasizes large forecasting errors because the error terms are squared. Therefore, it is useful for identifying configurations that may produce large deviations at certain time steps.

The root mean squared error is defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{19}$$

RMSE has the same unit as the original load data and is sensitive to large errors. In this study, RMSE is used alongside MAE and MAPE to provide a more comprehensive evaluation of forecasting performance.

In addition to these aggregate metrics, point-wise error values are also calculated for each of the 48 forecasted time steps. The error, absolute error, and absolute percentage error are computed as:

$$Error_i = y_i - \hat{y}_i \tag{20}$$

$$Abs_Error_i = | y_i - \hat{y}_i | \tag{21}$$

$$APE_i = | \frac{y_i - \hat{y}_i}{y_i} | \times 100 \tag{22}$$

These point-wise error values are stored together with the actual and predicted load values for each trial. They support detailed analysis of forecasting performance across the 48 half-hourly intervals, enabling identification of whether errors are concentrated during peak, low, or transition periods. The implementation also records additional statistical information, including the mean error, the standard deviation of error, the minimum and maximum absolute errors, and the execution time for each trial.

4 RESULTS AND DISCUSSION

The experimental results were analyzed to evaluate the forecasting performance of Autoformer under different hyperparameter configurations for one-day-ahead short-term load forecasting. The grid search experiment consisted of 125 configurations generated from three selected hyperparameters: hidden_size, n_head, and learning_rate. Among these configurations, 100 trials were successfully trained and evaluated, while 25 trials failed due to CUDA out-of-memory errors. The failed trials were associated with hidden_size = 512, indicating that this configuration required higher computational resources than those available in the experimental environment. Among the successful trials, the best forecasting performance was obtained in Run 64 with hidden_size = 128, n_head = 4, and learning_rate = 0.0005. This configuration achieved MAE

= 372.70 MW, MAPE = 4.9978%, MSE = 289,690.53, and RMSE = 538.23 MW.

To provide an overall view of the best-performing configurations, Fig. 3 presents the top 10 Autoformer configurations ranked by MAPE. This figure helps identify which combinations of hidden_size, n_head, and learning_rate produced the lowest forecasting errors.

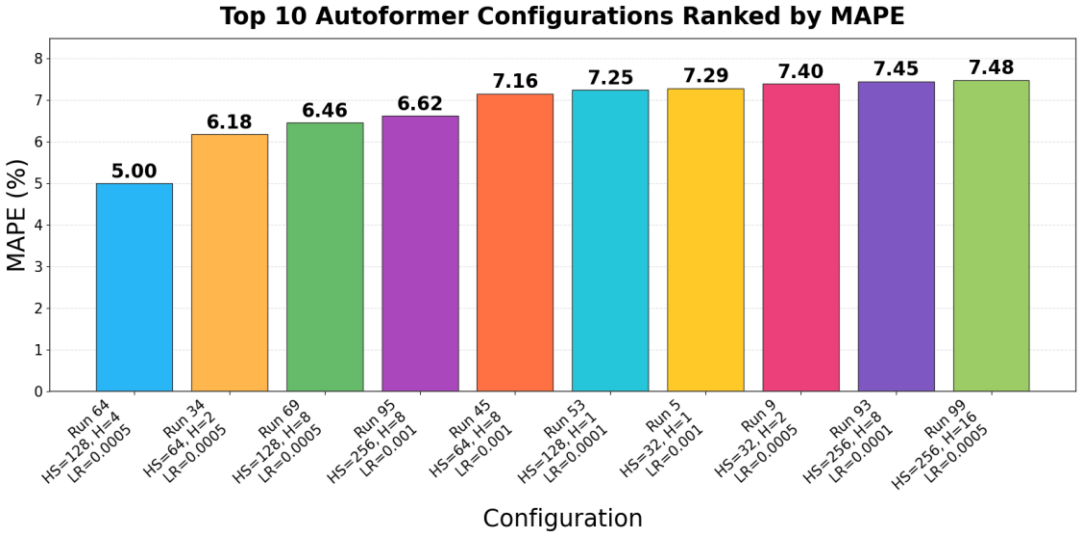


Figure 3: Ten best-performing Autoformer configurations ranked by MAPE.

As shown in Fig. 3, the lowest MAPE is achieved with hidden_size = 128, n_head = 4, and learning_rate = 0.0005. This result indicates that the best forecasting performance does not necessarily require the largest model size. Instead, a moderate hidden representation size, a suitable number of heads, and an appropriate learning rate can achieve better forecasting accuracy. Several top-ranked configurations also use a learning rate of 0.0005, suggesting that this value provides a favorable balance between convergence speed and training stability. Although some configurations with hidden_size = 256 also appear among the top results, they generally require higher computational time. Therefore, the ranking result suggests that both forecasting accuracy and computational cost should be considered when selecting Autoformer hyperparameters.

After identifying the best configuration, the predicted load profile was compared with the actual testing data. Fig. 4 presents the actual and predicted load values of the best Autoformer configuration over the 48 half-hourly intervals of the testing day.

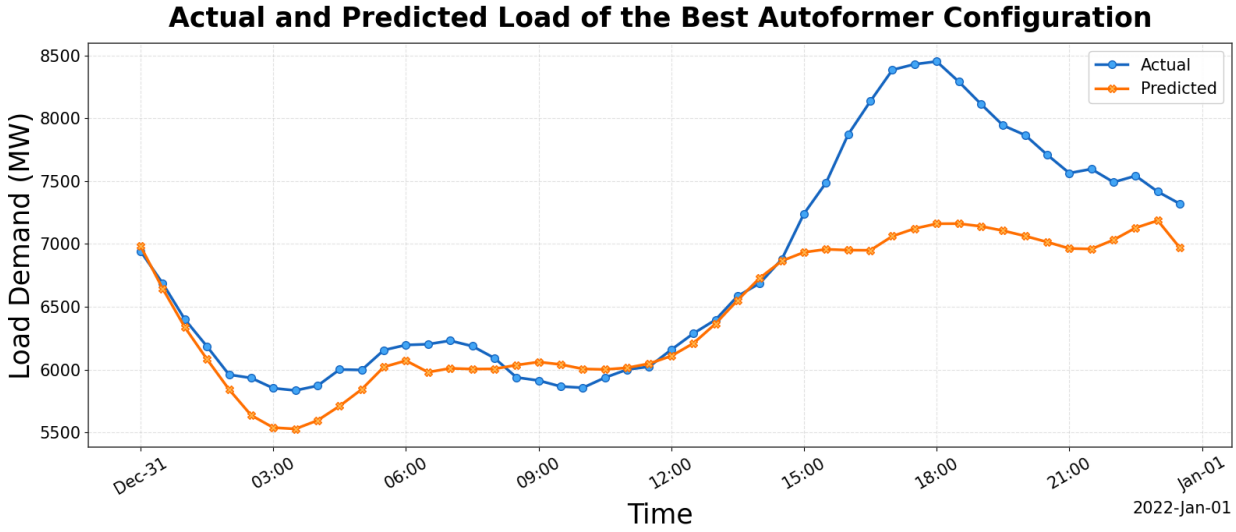


Figure 4: Test-day actual and forecasted load profiles for the selected Autoformer configuration.

Fig. 4 shows that the predicted load profile generally follows the temporal pattern of the actual electricity demand. The model captures the main daily variation of the load curve and provides reasonable predictions across most time intervals. This result confirms that Autoformer can learn useful temporal dependencies

from the previous seven days of historical load data. However, some deviations remain, especially during periods of stronger load changes. These deviations indicate that although the model can represent the general shape of the load profile, forecasting errors may still occur during rapid transitions or peak-load intervals.

To examine forecasting deviations in more detail, Fig. 5 presents the point-wise forecasting errors for the best configuration. The error is calculated as the difference between the actual and predicted values at each forecasting step.

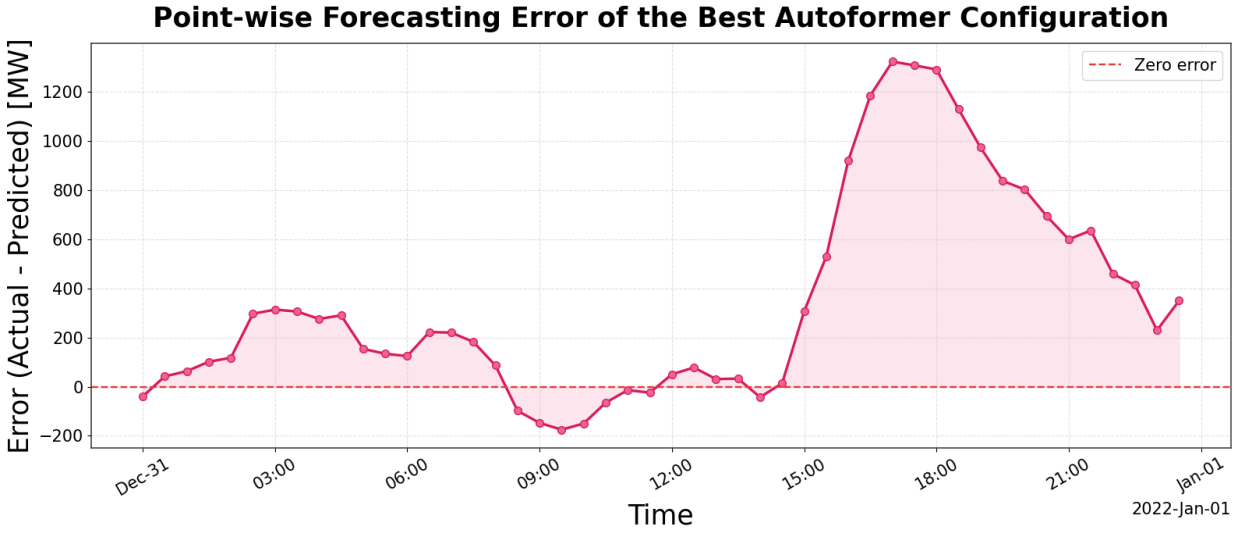


Figure 5: Signed forecasting errors over the 48-step test horizon.

As illustrated in Fig. 5, the forecasting error is not uniformly distributed across the testing horizon. Some time intervals have relatively small errors, while others show larger deviations. The largest absolute error occurs at 17:00, where the actual load is 8385.26 MW and the predicted value is 7061.72 MW, resulting in an absolute error of 1323.54 MW. This indicates that the model underestimates the load during this period. The results suggest that the forecasting task becomes more challenging during periods of rapid demand changes or peak-load conditions. Therefore, although the best configuration achieves good average accuracy, further improvement may require additional explanatory variables such as temperature, day type, holiday indicators, or other calendar-related information.

In addition to absolute error, the relative forecasting error was also examined. Fig. 6 presents the absolute percentage error profile of the best Autoformer configuration across the 48 forecasting steps.

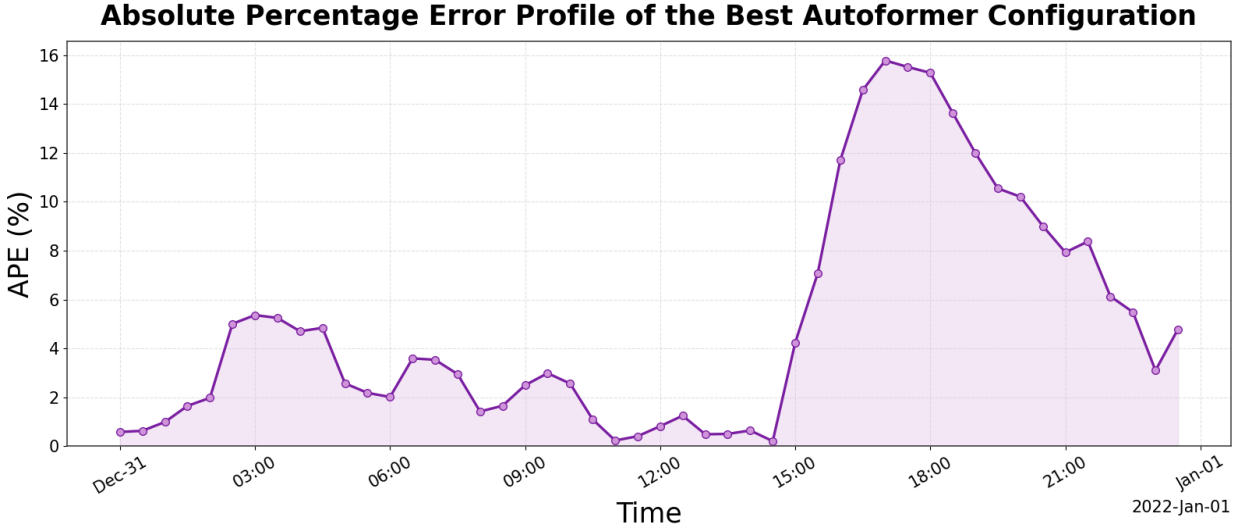


Figure 6: Absolute percentage errors over the 48-step test horizon.

From Fig. 6, most forecasting steps have moderate percentage errors, which explains the overall MAPE of 4.9978%. However, the maximum APE reaches approximately 15.78%, occurring around the same period

as the largest absolute error. This confirms that the main forecasting difficulty is concentrated in a limited number of time intervals rather than being evenly distributed throughout the day. Therefore, the point-wise APE analysis provides additional information beyond aggregate metrics such as MAE, MAPE, MSE, and RMSE.

A closer examination of Figs. 4–6 shows that the largest forecasting errors are concentrated during the late-afternoon high-demand period rather than being uniformly distributed throughout the testing day. From approximately 15:00 onward, the actual demand increases rapidly, whereas the predicted profile rises more slowly and remains below the actual load. The maximum absolute error occurs at 17:00, when the actual demand is 8,385.26 MW and the predicted demand is 7,061.72 MW. The resulting underestimation is 1,323.54 MW, corresponding to an absolute percentage error of approximately 15.78%. In contrast, the errors are generally smaller during the low- and moderate-demand intervals earlier in the day. These results indicate that the selected Autoformer configuration captures the overall daily pattern but has greater difficulty representing rapid load ramping and the magnitude of the late-afternoon peak.

The effect of the first selected hyperparameter, hidden_size, was then analyzed. Fig. 7 presents the mean MAPE obtained for different values of hidden_size. This figure shows whether increasing the Autoformer's internal representation dimension improves forecasting accuracy.

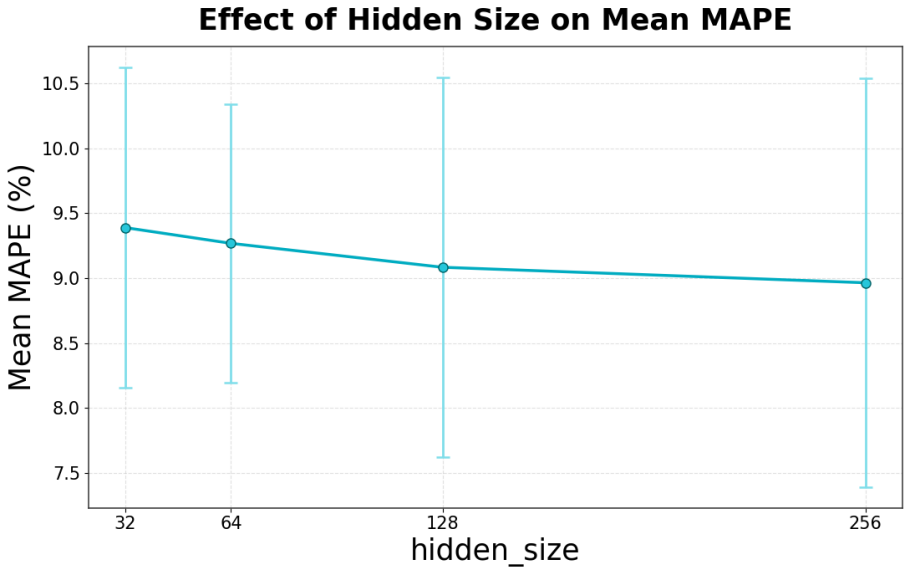


Figure 7: Effect of hidden size on mean MAPE. Error bars represent ±1 standard deviation across all successful configurations sharing the same hidden_size..

Fig. 7 shows that the mean MAPE gradually decreases from 9.3890% at hidden_size = 32 to 9.2677%, 9.0836%, and 8.9643% at hidden_size values of 64, 128, and 256, respectively. However, the error bars are large and substantially overlap across the four successful hidden-size levels. Therefore, the decreasing curve should be interpreted as an average marginal tendency rather than evidence that every larger hidden_size consistently produces a lower MAPE. The wide dispersion indicates that configurations sharing the same hidden_size can still produce substantially different results because of changes in n_head and learning_rate. Thus, the effect of hidden_size should be interpreted jointly with the remaining hyperparameters. Furthermore, although hidden_size = 256 achieves the lowest group-average MAPE, it requires considerably greater execution time, while all configurations with hidden_size = 512 fail because of GPU memory limitations.

The second investigated hyperparameter is n_head, which controls the number of parallel attention or auto-correlation heads in Autoformer. Fig. 8 presents the effect of n_head on the mean MAPE.

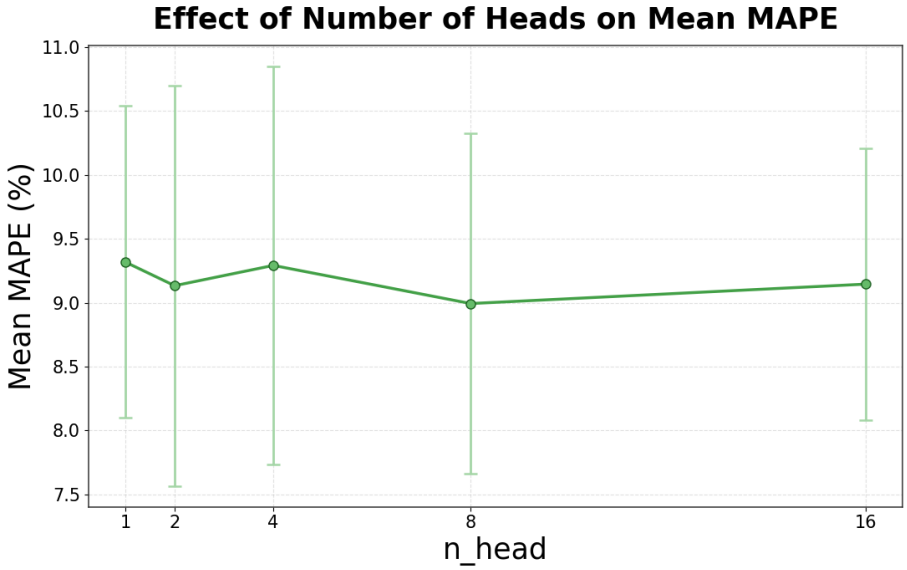


Figure 8: Effect of number of heads on mean MAPE. Error bars represent ±1 standard deviation across all successful configurations sharing the same n_head..

As shown in Fig. 8, n_head = 8 provides the lowest group-average MAPE of 8.9931%, while n_head = 2 and n_head = 16 produce mean MAPE values of 9.1330% and 9.1451%, respectively. Nevertheless, the error bars overlap substantially across all tested head numbers. This overlap indicates that the differences among the group means are relatively small compared with the variability caused by the accompanying hidden_size and learning_rate values. Therefore, n_head = 8 should not be interpreted as uniformly superior for every configuration. In fact, the best individual trial uses n_head = 4, whereas n_head = 8 provides the best marginal average. This difference confirms that the effect of the number of heads depends on its interaction with the other hyperparameters.

The third investigated hyperparameter is learning_rate, which directly affects the optimization process during model training. Fig. 9 presents the mean MAPE obtained for different learning rate values.

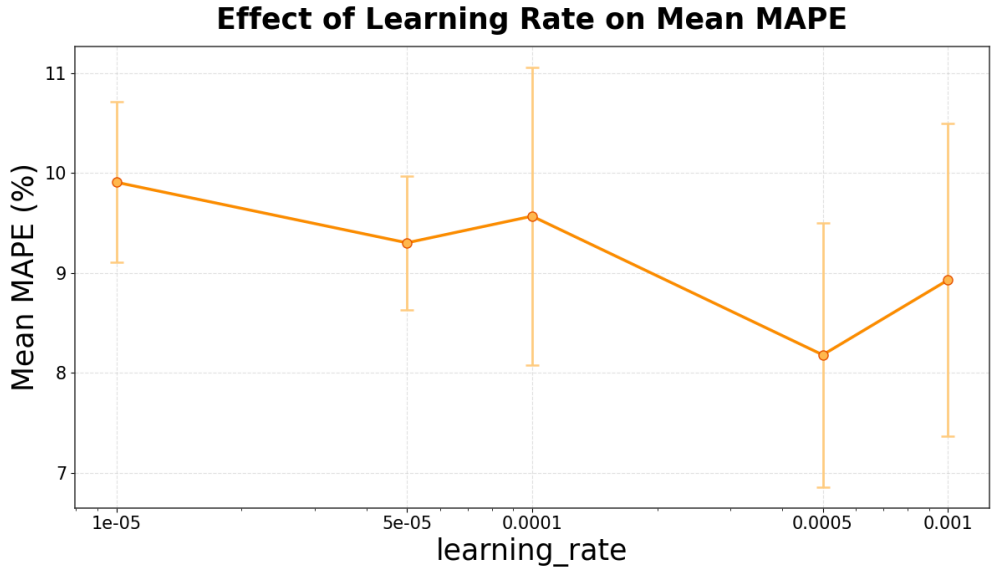


Figure 9: Effect of learning rate on mean MAPE. Error bars represent ±1 standard deviation across all successful configurations sharing the same learning_rate.

Fig. 9 shows that learning_rate = 0.0005 yields the lowest group-average MAPE of 8.1793%, whereas learning_rate = 1e-5 produces the highest mean MAPE of 9.9062%. Although the error bars overlap to some extent, the variation in the group means is more pronounced than that observed for n_head. The relatively poor result obtained with very small learning rates suggests that the model may not converge sufficiently within the fixed training budget of max_steps = 500. Conversely, increasing the learning rate

to 0.001 does not further improve the average performance, possibly because larger parameter updates reduce optimization stability. Therefore, learning_rate exhibits the strongest marginal effect among the three investigated hyperparameters. Nevertheless, the remaining error-bar overlap indicates that its effect is also influenced by hidden_size and n_head.

After examining the individual effects of each hyperparameter, the interaction between architectural parameters was analyzed. Fig. 10 presents the mean MAPE values for different combinations of hidden_size and n_head.

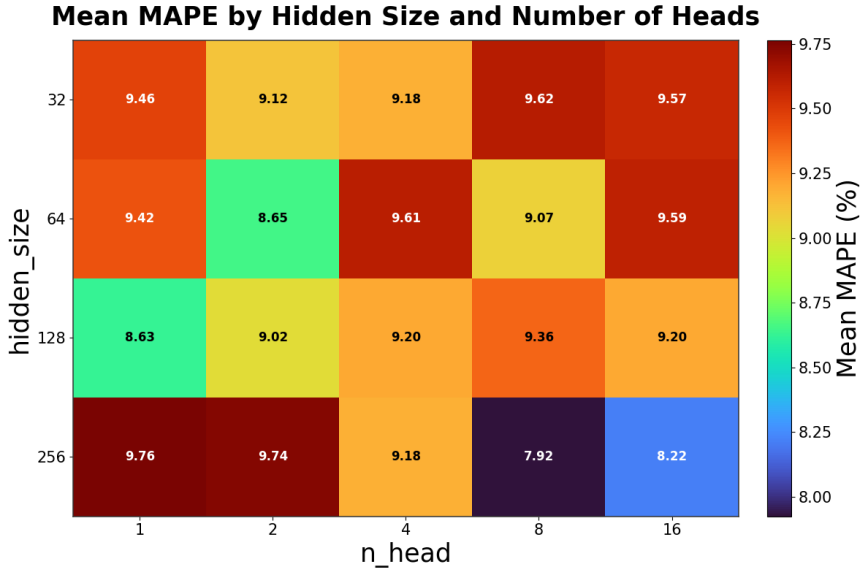


Figure 10: Interaction between hidden size and the number of heads in terms of mean MAPE.

Fig. 10 indicates that the relationship between hidden_size and n_head is not strictly monotonic. Increasing one parameter alone does not always lead to improved forecasting accuracy. Some combinations with moderate hidden size and suitable number of heads yield lower MAPE values than larger configurations. This suggests that Autoformer performance depends on the balance between representation capacity and temporal dependency modeling. Therefore, architectural hyperparameters should not be selected independently; their interaction should also be considered when configuring the model.

The interaction between hidden_size and learning_rate is further examined in Fig. 11. This analysis is important because the previous results show that the learning rate has a strong effect on forecasting accuracy.

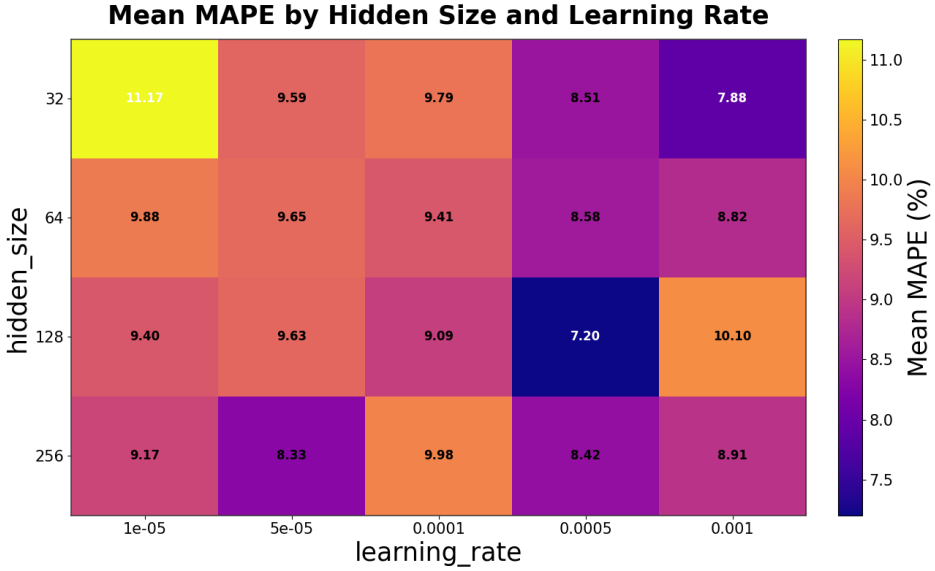


Figure 11: Interaction between hidden size and learning rate in terms of mean MAPE.

As shown in Fig. 11, the combination of hidden_size = 128 and learning_rate = 0.0005 provides the lowest

mean MAPE among the tested combinations in this two-parameter view. This result is consistent with the best individual configuration identified in the ranking analysis. The heatmap also shows that the same hidden size can produce different forecasting errors depending on the learning rate. Therefore, the benefit of increasing model capacity can only be fully realized when the optimization setting is appropriate. This confirms that learning_rate is a critical hyperparameter for Autoformer training and should be tuned together with architectural parameters.

Besides forecasting accuracy, computational cost was also considered. Fig. 12 presents the average execution time for different values of hidden_size.

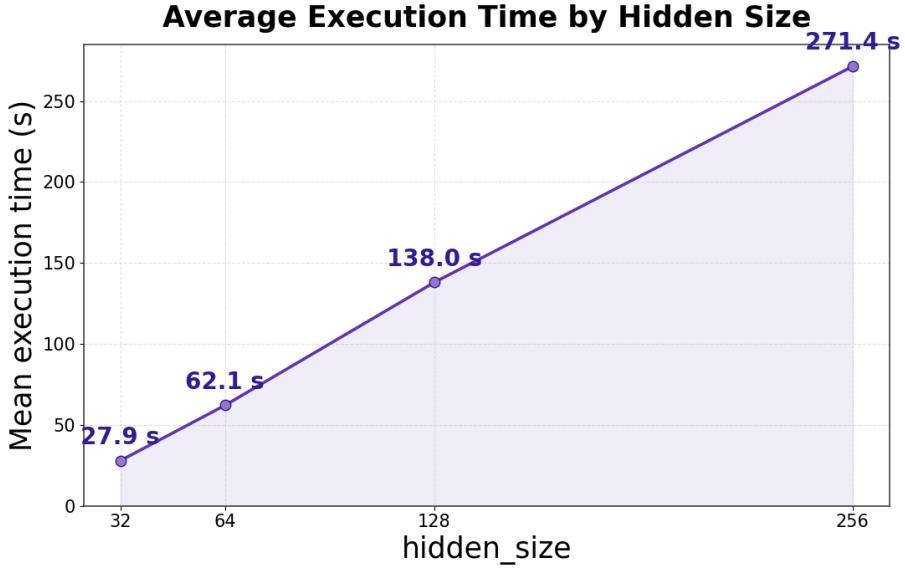


Figure 12: Average execution time across hidden-size levels.

Fig. 12 shows that execution time increases rapidly as hidden_size increases. The average execution time is approximately 27.95 seconds for hidden_size = 32, 62.11 seconds for hidden_size = 64, 138.01 seconds for hidden_size = 128, and 271.40 seconds for hidden_size = 256. Although hidden_size = 256 achieves the lowest average MAPE among the successful hidden-size groups, its computational cost is much higher than that of hidden_size = 128. Since the best individual configuration uses hidden_size = 128, this value is a more balanced trade-off between forecasting accuracy and execution time.

Finally, Fig. 13 presents the failed trials grouped by hidden size. This figure is included to show the computational feasibility of different Autoformer configurations in the experimental environment.

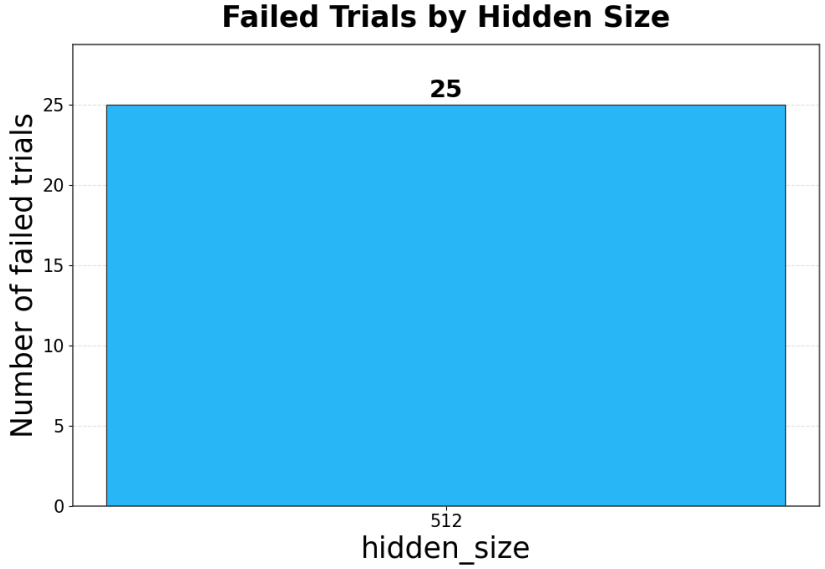


Figure 13: Distribution of unsuccessful trials across hidden-size levels.

Fig. 13 shows that all 25 failed trials occur when hidden_size = 512. This confirms that very large Autoformer configurations may exceed available GPU memory during repeated grid searches. Therefore, increasing model capacity can improve forecasting performance to some extent, but excessively large configurations may become impractical. Overall, the experimental results show that the configuration with hidden_size = 128, n_head = 4, and learning_rate = 0.0005 provides the best forecasting accuracy for the NSW short-term load forecasting task. The findings also demonstrate that careful hyperparameter selection is essential for effectively applying Autoformer, as forecasting accuracy depends not only on model capacity but also on optimization behavior and computational feasibility.

5 CONCLUSION

This study presented a systematic hyperparameter sensitivity analysis of the Autoformer model for short-term electric load forecasting. The forecasting task was formulated as a one-day-ahead prediction problem using the New South Wales electricity demand dataset with 30-minute resolution. In the experimental setup, seven days of historical load data were used as the input window to forecast the next 48 time steps. Three important hyperparameters, namely hidden_size, n_head, and learning_rate, were investigated using a grid-search strategy.

The experimental results show that hyperparameter settings strongly influence Autoformer performance. Among the 125 designed configurations, 100 trials were completed, while 25 trials failed due to memory limitations, mainly when hidden_size = 512. The best forecasting performance was obtained with hidden_size = 128, n_head = 4, and learning_rate = 0.0005, achieving MAE = 372.70 MW, MAPE = 4.9978%, MSE = 289,690.53, and RMSE = 538.23 MW. The predicted load profile generally followed the actual daily demand pattern, indicating that Autoformer can effectively capture temporal dependencies and repeated consumption behavior from historical load data. Nevertheless, the point-wise analysis showed that the most challenging period occurred during the late-afternoon demand peak. At 17:00, the model underestimated the actual load by 1,323.54 MW, corresponding to an absolute percentage error of approximately 15.78%

The sensitivity analysis further indicates that increasing the hidden size can improve forecasting accuracy to some extent, but it also increases computational cost and memory requirements. The number of heads affects forecasting performance, although its influence is less dominant than that of the learning rate. Among the investigated parameters, learning_rate shows the strongest effect on model accuracy and training stability. These findings confirm that careful hyperparameter selection is essential when applying Autoformer to short-term load forecasting.

Although the present experimental framework provides useful insights into Autoformer configuration for electric load forecasting, several limitations remain. The experiment was conducted using only the NSW dataset and a one-day testing period. In addition, the model used only historical load values without incorporating external factors such as temperature, humidity, day type, or holiday information. The present study focuses on within-model hyperparameter sensitivity rather than cross-model benchmarking. Therefore, the reported results should be interpreted as evidence of performance differences among Autoformer configurations and not as evidence that Autoformer is superior to other forecasting architectures. Future work will extend the evaluation to other regions, longer testing periods, repeated rolling-origin experiments, and multivariate forecasting settings. Additional benchmark models will also be implemented under a common experimental protocol, together with more computationally efficient hyperparameter optimization strategies.

REFERENCES

[1] N. T. N. Anh, N. N. Anh, T. N. Thang, V. K. Solanki, R. G. Crespo, and N. Q. Dat, "Online SARIMA applied for short-term electricity load forecasting," *Appl. Intell.*, vol. 54, no. 1, pp. 1003–1019, 2024, doi: 10.1007/s10489-023-05230-y.

[2] G. R. Alonso Brito, A. Rivero Villaverde, A. Lau Quan, and M. E. Ruíz Pérez, "Comparison between SARIMA and Holt–Winters models for forecasting monthly streamflow in the western region of Cuba," *SN Appl. Sci.*, vol. 3, no. 6, 2021, doi: 10.1007/s42452-021-04667-5.

[3] U. M. Sirisha, M. C. Belavagi, and G. Attigeri, "Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting : A Comparison," *IEEE Access*, vol. 10, no. December, pp. 124715–124727, 2022, doi: 10.1109/ACCESS.2022.3224938.

[4] Z. Zhang and W. Hong, "Electric Load Forecasting by Hybrid Self- Recurrent Support Vector Regression Model With Variational Mode Decomposition and Improved Cuckoo Search Algorithm," vol. 8, 2020.

[5] M. Safari, A. H. Rabiee, and J. Joudaki, "Developing a Support Vector Regression (SVR) Model for Prediction of Main and Lateral Bending Angles in Laser Tube Bending Process," *Materials (Basel)*, vol. 16, no. 8, 2023, doi: 10.3390/ma16083251.

[6] N. C. Yang and K. L. Sung, "Non-Intrusive Load Classification and Recognition Using Soft-Voting Ensemble Learning Algorithm With Decision Tree, K-Nearest Neighbor Algorithm and Multilayer Perceptron," *IEEE Access*, vol. 11, no. September, pp. 94506–94520, 2023, doi: 10.1109/ACCESS.2023.3311641.

[7] D. Chowdhury, M. Sarkar, M. Z. Haider, and T. Alam, "Zone-Wise Hourly Load Prediction Using Regression Decision Tree Model," *2018 Int. Conf. Innov. Eng. Technol. ICIET 2018*, pp. 1–6, 2018, doi: 10.1109/CIET.2018.8660781.

[8] M. K. Madhu Malini, B. Iswariya, H. Prasad, and T. D. Sudhakar, "Load Forecasting using Random Forest Regression Algorithm in Machine Learning," *Proc. 2024 Int. Conf. Sci. Technol. Eng. Manag. ICSTEM 2024*, pp. 1–6, 2024, doi: 10.1109/ICSTEM61137.2024.10560982.

[9] U. Singh and S. Vadhera, "Random Forest and XGBoost Technique for Short-Term Load Forecasting," *2022 1st Int. Conf. Sustain. Technol. Power Energy Syst.*, pp. 1–6, 2022, doi: 10.1109/STPES54845.2022.10006635.

[10] D. A. Khan, A. Arshad, and Z. Ali, "Performance Analysis of Machine Learning Techniques for Load Forecasting," *ICET 2021 - 16th Int. Conf. Emerg. Technol. 2021, Proc.*, pp. 1–6, 2021, doi: 10.1109/ICET54505.2021.9689903.

[11] M. Mohamed, F. E. Mahmood, M. A. Abd, M. Rezkallah, and A. Hamadi, "Load Demand Forecasting Using eXtreme Gradient Boosting (XGboost)," *2023 IEEE Ind. Appl. Soc. Annu. Meet.*, pp. 1–7, 2023, doi: 10.1109/IAS54024.2023.10406613.

[12] S. Singh and M. M. Tripathi, "A Comparative Analysis of Extreme Gradient Boosting Technique with Long Short-Term Memory and Layered Recurrent Neural Network for Electricity Demand Forecasting," *2021 6th Int. Conf. Recent Trends Electron. Information, Commun. Technol. RTEICT 2021*, pp. 297–302, 2021, doi: 10.1109/RTEICT52294.2021.9573988.

[13] Q. Yang, Y. Lin, S. Kuang, and D. Wang, "A novel short-term load forecasting approach for data-poor areas based on K-MIFS-XGBoost and transfer-learning," *Electr. Power Syst. Res.*, vol. 229, no. January, p. 110151, 2024, doi: 10.1016/j.epsr.2024.110151.

[14] Z. Mustaffa and M. H. Sulaiman, "Advanced forecasting of building energy loads with XGBoost and metaheuristic algorithms integration," *Energy Storage Sav.*, pp. 0–55, 2025, doi: 10.1016/j.enss.2025.03.005.

[15] Y. Xie, Y. Ueda, and M. Sugiyama, "A two-stage short-term load forecasting method using long short-term memory and multilayer perceptron," *Energies*, vol. 14, no. 18, pp. 1–17, 2021, doi: 10.3390/en14185873.

[16] A. P. Wibawa, A. B. P. Utama, H. Elmunsyah, U. Pujianto, F. A. Dwiyanto, and L. Hernandez, "Time-series analysis with smoothed Convolutional Neural Network," *J. Big Data*, vol. 9, no. 1, 2022, doi: 10.1186/s40537-022-00599-y.

[17] 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 Intell. Syst.*, vol. 9, no. 4, pp. 4249–4261, 2023, doi: 10.1007/s40747-022-00952-x.

[18] S. R. Fahim *et al.*, "Forecasting EV Charging Demand : A Graph Convolutional Neural Network-Based Approach," *2024 4th Int. Conf. Smart Grid Renew. Energy*, pp. 1–6, doi: 10.1109/SGRE59715.2024.10428726.

[19] R. Gong, A. Jiang, H. Hu, D. Liu, X. Wu, and S. Zhang, "Short-term multi-featured power load forecasting model based on GBKA-VMD-Mambaformer with noise separation error correction under scarce load data," *Electr. Power Syst. Res.*, vol. 247, no. February, p. 111846, 2025, doi: 10.1016/j.epsr.2025.111846.

[20] K. Aurangzeb, M. Alhussein, K. Javaid, and S. I. Haider, "A Pyramid-CNN based deep learning model for power load forecasting of similar-profile energy customers based on clustering," *IEEE Access*, vol. 9, pp. 14992–15003, 2021, doi: 10.1109/ACCESS.2021.3053069.

[21] Y. Chen, M. Wang, Y. Wei, X. Huang, and S. Gao, "Multi-Encoder Spatio-Temporal Feature Fusion Network for Electric Vehicle Charging Load Prediction," *J. Intell. Robot. Syst. Theory Appl.*, vol. 110, no. 3, 2024, doi: 10.1007/s10846-024-02125-z.

[22] S. Cantillo-Luna, R. Moreno-Chuquen, and J. A. Lopez Sotelo, "Intra-day Electricity Price Forecasting Based on a Time2Vec-LSTM Neural Network Model," *2023 IEEE Colomb. Conf. Appl. Comput. Intell. ColCACI 2023 - Proc.*, pp. 1–6, 2023, doi: 10.1109/ColCACI59285.2023.10225803.

[23] X. He, W. Zhao, Z. Gao, Q. Zhang, and W. Wang, "A hybrid prediction interval model for short-term electric

load forecast using Holt-Winters and Gate Recurrent Unit," *Sustain—Energy, Grids Networks*, vol. 38, no. February 2023, p. 101343, 2024, doi: 10.1016/j.segan.2024.101343.

[24] H. Shi, L. Wang, R. Scherer, M. Wozniak, P. Zhang, and W. Wei, "Short-Term Load Forecasting Based on Adabelief Optimized Temporal Convolutional Network and Gated Recurrent Unit Hybrid Neural Network," *IEEE Access*, vol. 9, pp. 66965–66981, 2021, doi: 10.1109/ACCESS.2021.3076313.

[25] K. Qu, G. Si, Z. Shan, Q. Wang, X. Liu, and C. Yang, "Forwardformer : Efficient Transformer With Multi-Scale Forward Self-Attention for Day-Ahead Load Forecasting," *IEEE Trans. Power Syst.*, vol. 39, no. 2, pp. 1421–1433, 2024, doi: 10.1109/TPWRS.2023.3266369.

[26] O. Bouhamed, M. Dissem, M. Amayri, and N. Bouguila, "Engineering Applications of Artificial Intelligence Transformer-based deep probabilistic network for load forecasting," *Eng. Appl. Artif. Intell.*, vol. 152, no. April, p. 110781, 2025, doi: 10.1016/j.engappai.2025.110781.

[27] C. Xu and G. Chen, "Interpretable transformer-based model for probabilistic short-term forecasting of residential net load," *Int. J. Electr. Power Energy Syst.*, vol. 155, no. PB, p. 109515, 2024, doi: 10.1016/j.ijepes.2023.109515.

[28] F. Cheng and H. Liu, "Multi-step electric vehicles charging loads forecasting: An autoformer variant with feature extraction, frequency enhancement, and error correction blocks," *Appl. Energy*, vol. 376, no. PB, p. 124308, 2024, doi: 10.1016/j.apenergy.2024.124308.

[29] W. Wang, X. Li, and P. Yan, "A Multi-Scaler Hybrid Autoformer for Enhanced Time Series Forecasting in Energy Consumption," *IEEE Access*, vol. 12, no. December, pp. 196347–196363, 2024, doi: 10.1109/ACCESS.2024.3522563.

[30] S. Cen and C. G. Lim, "Multi-Task Learning of the PatchTCN-TST Model for Short-Term Multi-Load Energy Forecasting Considering Indoor Environments in a Smart Building," *IEEE Access*, vol. 12, no. December 2023, pp. 19553–19568, 2024, doi: 10.1109/ACCESS.2024.3355448.

PHÂN TÍCH ĐỘ NHẠY SIÊU THAM SỐ CỦA MÔ HÌNH AUTOFORMER TRONG DỰ BÁO PHỤ TẢI ĐIỆN NGẮN HẠN

NGUYỄN ANH TUẤN, TRẦN THANH NGỌC, NGUYỄN THANH THUẬN
Khoa Công Nghệ Điện – Trường Đại học Công Nghiệp Thành Phố Hồ Chí Minh
** Tác giả liên hệ: nguyenanhtuan@iuh.edu.vn*

Tóm tắt. Dự báo phụ tải điện ngắn hạn là một nhiệm vụ quan trọng trong vận hành hệ thống điện, lập kế hoạch thị trường điện, quản lý năng lượng và hoạch định phía nhu cầu. Các dự báo ngắn hạn có độ chính xác cao giúp đơn vị vận hành dự đoán trước sự biến động của nhu cầu phụ tải, giảm yêu cầu dự phòng, nâng cao hiệu quả điều độ kinh tế và duy trì độ tin cậy của hệ thống. Trong những năm gần đây, các mô hình dự báo dựa trên Transformer ngày càng thu hút sự quan tâm nhờ khả năng nắm bắt các mối quan hệ phụ thuộc theo thời gian trong dữ liệu chuỗi thời gian. Autoformer là một trong những biến thể Transformer tiêu biểu được thiết kế cho bài toán dự báo chuỗi thời gian thông qua cơ chế phân rã chuỗi và cơ chế tự tương quan. Tuy nhiên, hiệu suất của Autoformer chịu ảnh hưởng đáng kể bởi việc thiết lập các siêu tham số, đặc biệt là những siêu tham số kiểm soát năng lực biểu diễn của mô hình, cấu trúc attention và hành vi tối ưu hóa.

Bài báo này nghiên cứu độ nhạy của hiệu suất Autoformer đối với ba siêu tham số quan trọng, gồm kích thước ẩn (hidden size), số lượng đầu attention (number of attention heads) và tốc độ học (learning rate). Một khung thực nghiệm dựa trên tìm kiếm lưới được xây dựng cho bài toán dự báo phụ tải điện ngắn hạn sử dụng bộ dữ liệu nhu cầu điện của bang New South Wales với độ phân giải thời gian 30 phút. Bài toán dự báo được thiết lập theo dạng dự báo trước một ngày, trong đó mô hình dự báo 48 bước thời gian tương lai dựa trên cửa sổ dữ liệu lịch sử dài 7 ngày. Mỗi siêu tham số được khảo sát tại 5 giá trị ứng viên, tạo thành tổng cộng 125 thử nghiệm.

Hiệu suất của mô hình được đánh giá bằng các chỉ số sai số tuyệt đối trung bình (MAE), sai số phần trăm tuyệt đối trung bình (MAPE), sai số bình phương trung bình (MSE) và căn bậc hai của sai số bình phương trung bình (RMSE). Trong tổng số 125 cấu hình được thiết kế, 100 cấu hình đã được thực hiện thành công, trong khi 25 cấu hình có hidden_size = 512 không thể hoàn thành do giới hạn bộ nhớ GPU. Cấu hình tốt nhất gồm hidden_size = 128, n_head = 4 và learning_rate = 0.0005, đạt MAE bằng 372,70 MW, MAPE bằng 4,9978%, MSE bằng 289.690,53 và RMSE bằng 538,23 MW.

Kết quả phân tích độ nhạy cho thấy learning_rate có ảnh hưởng biên mạnh nhất đến hiệu suất mô hình, trong khi tác động của hidden_size và n_head phụ thuộc đáng kể vào sự tương tác của chúng với các siêu tham số còn lại. Khung phương pháp được đề xuất cung cấp một cơ sở có hệ thống cho việc lựa chọn cấu hình Autoformer, đồng thời xem xét đồng thời độ chính xác dự báo, mức độ biến thiên của kết quả và tính khả thi về mặt tính toán.

Từ khóa: Autoformer, Dự báo phụ tải điện, Độ nhạy siêu tham số, Tìm kiếm lưới, Dự báo chuỗi thời gian.

Received on August 25 – 2026
Revised on October 08 – 2026