International Journal of Robotics and Control Systems
Vol. 6, No. 1, 2026, pp. 285-304
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

# Optimizing N-BEATS for Short-Term Load Forecasting Via Random Search–TPE and Genetic Algorithm

Tuan Anh Nguyen [a,1,*], Trung Dung Nguyen [a,2]

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

ARTICLE INFO

**Article history**
Received December 12, 2025
Revised January 11, 2026
Accepted January 28, 2026

**Keywords**
NBEATS;
Short-Term Load Forecasting;
Hyperparameter Optimization;
Random Search;
TPE;
Genetic Algorithm

ABSTRACT

Accurate short-term electricity load forecasting is crucial for secure and economical power system operation; however, deep forecasting models can degrade markedly when hyperparameters are poorly chosen. This paper presents a systematic and reproducible hyperparameter-optimization framework for the N-BEATS (Neural Basis Expansion Analysis for Time Series) model to improve one-day-ahead load forecasting under a fixed experimental setup. A unified tuning protocol-using the same objective, search space, and evaluation budget-optimizes key N-BEATS hyperparameters (stack_types, n_blocks, mlp_units, and learning_rate). Three strategies are compared fairly: Random Search, Tree-structured Parzen Estimator (TPE) Bayesian optimization, and a Genetic Algorithm (GA), each with 100 evaluations (100 trials). Experiments use two Australian National Electricity Market regional datasets (New South Wales and Queensland), each with 122,735 half-hourly demand records; the most recent 28 days are used for training and the final day for testing, with a 7-day input window (336 points) forecasting 48 steps (24 hours) ahead (max_steps = 1000). Performance is assessed using MSE, RMSE, MAE, and MAPE. All tuned configurations outperform the default N-BEATS baseline. Using MAPE as the optimization objective, TPE performs best in both regions. In NSW, MAPE decreases from 2.56% to 1.29% (49.6% reduction), and MAE decreases from 180.0 MW to 88.7 MW (50.7% reduction). In QLD, MAPE decreases from 2.80% to 1.79% (36.1% reduction), while also yielding the lowest MSE/RMSE/MAE. These results confirm the value of standardized hyperparameter tuning for N-BEATS and suggest that the most effective strategy can be region- and metric-dependent.



## 1. Introduction

Accurate load forecasting is a fundamental requirement for operating power systems in a reliable, flexible, and economical manner. Core operational decisions-such as unit commitment, day-ahead scheduling, market dispatch, and capacity planning-depend heavily on the quality of forecasting. Because electricity demand varies across multiple time scales (daily, weekly, and seasonal cycles) and is also affected by irregular events (such as holidays and sudden demand spikes), practical forecasting

models must capture multi-scale temporal patterns while remaining robust to distribution shifts across different regions.

Load forecasting methods have evolved through three major "generations," reflecting differences in modeling assumptions, nonlinear representation capacity, and the degree of automated feature extraction. In the first generation, statistical approaches such as ARIMA [1]-[4] and SARIMA [5]-[8] models utilize autoregressive and moving-average structures, which can be effective for stationary series (or series rendered stationary through differencing) and provide interpretable trend/seasonal components. Related methods, such as Holt–Winters/ETS, seasonal regression, and state-space formulations, are often adopted to capture periodicities with low computational cost and straightforward deployment. However, their linearity assumptions and limited flexibility can restrict their ability to represent complex nonlinear relationships, and performance may degrade under abrupt variations or when regional demand characteristics differ.

In the second generation, machine learning methods improve nonlinear modeling by learning mappings from input features to target demand values. Representative models include Support Vector Regression (SVR/SVM) [9]-[10], Random Forest regression [11]-[13], and gradient-boosting families such as Gradient Boosting [14]-[17], XGBoost [18], [19], LightGBM [20]-[23], and CatBoost [24]-[27]. These approaches can outperform purely statistical baselines and can incorporate diverse features; nevertheless, their effectiveness typically depends on feature engineering and careful hyperparameter tuning. This reliance often increases deployment effort and can introduce sensitivity to task configurations and regional differences.

In the third generation, deep learning architectures aim to learn temporal representations directly from raw time series. Sequential models such as RNN [28], LSTM [29]-[33], and GRU [34]-[38] are designed to model long-range dependencies, while convolutional architectures such as 1D-CNN [39]-[41] and Temporal Convolutional Networks (TCN) [42]-[45] exploit time-domain receptive fields with parallel training advantages. Attention-based sequence models, including Transformers [46]-[49], further strengthen long-range dependency modeling and multi-periodic representation. Although deep models can achieve strong accuracy, they are often sensitive to architectural choices and hyperparameters, which motivates the development of standardized optimization and evaluation protocols to attain stable performance in real-world deployments.

Among modern deep forecasting architectures, N-BEATS [50]-[53] has attracted considerable attention due to its deep fully connected block design, backward–forward forecasting mechanism, and its capability to represent trend and seasonality components. However, N-BEATS performance can vary substantially across datasets and regions when hyperparameters are not selected appropriately. Key factors, such as stack composition, architectural depth, MLP width, and learning rate, directly influence model capacity and optimization dynamics. Suboptimal settings may lead to degraded peak tracking and higher percentage errors. Despite the potential of N-BEATS [50]-[53], standardized and reproducible hyperparameter optimization procedures explicitly tailored to short-term load forecasting remain limited; consequently, many studies still rely on default configurations or manual tuning, which can yield unstable accuracy across regions and forecasting setups. In contrast to ad-hoc or model-agnostic tuning practices, this work emphasizes a unified and reproducible tuning protocol for N-BEATS in short-term load forecasting, enabling fair comparison across optimization strategies under the same experimental setup.

To address this gap, this study proposes an optimization-oriented framework to systematically tune N-BEATS using three complementary hyperparameter search strategies: Random Search (RS) [54]-[57], the Tree-structured Parzen Estimator (TPE) [58], and the Genetic Algorithm (GA) [59]-[62]. The tuning process focuses on four influential hyperparameters-stack_types, n_blocks, mlp_units, and learning_rate-while keeping the remaining settings fixed to ensure fair comparison. Experiments are conducted on Australian regional load datasets from New South Wales (NSW) and Queensland (QLD), using MAPE, MAE, MSE, and RMSE as evaluation metrics, along with forecast plots to assess peak tracking and stability. The research contributions are summarized as follows:

- We propose a systematic N-BEATS tuning procedure for load forecasting that focuses on four influential hyperparameters (stack_types, n_blocks, mlp_units, and learning_rate).
- We conduct a fair and comprehensive comparison of three hyperparameter optimization strategies (RS, TPE, GA) under a unified evaluation protocol and the same evaluation budget.
- We establish a standardized experimental setup to ensure reproducibility across optimization strategies and datasets.
- We demonstrate that hyperparameter optimization significantly reduces forecasting errors compared with the default N-BEATS configuration, and that the best-performing strategy can be region- and metric-dependent.
- We provide practical insights to support real-world deployment of the proposed tuning framework in power system operation.

## 2. Method

### 2.1. NBEAT Model

N-BEATS (Neural Basis Expansion Analysis for Time Series) is a deep neural architecture for time-series forecasting built as a sequence of forecasting blocks. Each block is implemented as a Multi-Layer Perceptron (MLP) and produces two outputs: a backcast, which explains part of the input window and is subtracted to form a residual for the next block, and a forecast, which is accumulated to generate the final prediction. This backcast–forecast residual mechanism enables progressive refinement of the input signal across blocks without requiring explicit feature engineering. N-BEATS can be configured in Generic mode, where basis functions are learned from data, or Interpretable mode, where blocks are organized to represent trend and seasonality components.

Backcast and forecast estimation for each block:

$$(\hat{y}_t^{b,(i)}, \hat{y}_t^{f,(i)}) = f_{\theta_i}(x_t^{(i)}) \tag{1}$$

Residual update for the next block:

$$x_t^{(i+1)} = x_t^{(i)} - \hat{y}_t^{b,(i)} \tag{2}$$

Final forecast aggregated across all blocks:

$$\hat{y}_t^{f} = \sum_{i=1}^{K} \hat{y}_t^{f,(i)} \tag{3}$$

Training loss (Mean Squared Error):

$$L(\theta) = \frac{1}{N}\sum_{t=1}^{N} (y_t - \hat{y}_t)^2 \tag{4}$$

These equations describe the sequential residual learning mechanism through which N-BEATS refines the remaining error at each block and constructs the final forecast via backcast–forecast decomposition. In practice, forecasting performance is sensitive to architectural and training hyperparameters that control model capacity and optimization dynamics. In this study, we focus on four influential hyperparameters-stack_types, n_blocks, mlp_units, and learning_rate-because they directly determine stack composition, model depth, MLP width, and training stability. The remaining settings (max_steps, batch size, and dropout) are kept fixed to ensure fair comparison across optimization strategies.

According to the default configuration in Nixtla's NeuralForecast library, N-BEATS typically uses 512 units per hidden layer, a learning rate of 0.001, a maximum of 1000 training steps, and 3–4 blocks, depending on the model mode. However, these default settings may be suboptimal for electric load forecasting, where daily–weekly periodicity and seasonal peak variations require tailored hyperparameters. Therefore, systematic hyperparameter tuning is crucial for reliable and accurate forecasting performance. Table 1 presents the default hyperparameters used as the baseline configuration before applying hyperparameter optimization in this study.

**Table 1.** Default hyperparameters of the model

| Hyperparameter | Description | Default Value |
|---|---|---|
| MLP units | Number of neurons in hidden layers | 512 |
| Learning rate | Weight update step size | 0.001 |
| Max steps | Maximum training iterations | 1000 |
| Batch size | Samples per batch | 32 |
| Dropout rate | Dropout to mitigate overfitting | 0.1 |
| Number of blocks | Depth of the N-BEATS architecture | 3 |
| Stack expansion | Width expansion factor | 2 |

## 2.2. Hyperparameter Optimization Algorithms

This study evaluates three widely used hyperparameter optimization strategies-Random Search (RS), Tree-structured Parzen Estimator (TPE) Bayesian optimization, and a Genetic Algorithm (GA)-under a unified and reproducible tuning protocol. All strategies are optimized over the same search space and constraints to ensure a fair comparison. Random Search (RS). RS randomly samples hyperparameter configurations from a predefined search space. It is simple to implement, easy to parallelize, and can be effective when only a small subset of hyperparameters strongly influences forecasting accuracy. However, RS does not exploit information from previous trials and may require a larger evaluation budget to locate high-quality regions of the search space.

Tree-structured Parzen Estimator (TPE). TPE is a Bayesian optimization approach that uses outcomes from previous trials to guide subsequent sampling. It models promising and non-promising regions of the search space using Parzen estimators and proposes new configurations with greater expected improvement. Compared with purely random exploration, TPE typically improves sampling efficiency by concentrating evaluations in regions that are more likely to reduce the objective value. Genetic Algorithm (GA). GA is a population-based evolutionary method that searches for high-quality hyperparameter settings through selection, crossover, and mutation. In each generation, each individual represents one hyperparameter configuration. Individuals are evaluated using a fitness function (validation error), and the best-performing candidates are retained as parents to generate offspring for the next generation. This mechanism encourages exploration and helps reduce the risk of premature convergence to suboptimal regions.

To ensure fairness and direct comparability, all methods were executed under identical experimental constraints, including the same optimization objective (minimizing validation MAPE), a fixed random seed (42), and an equal evaluation budget of 100 evaluations (100 trials). All optimizers tune the same four influential N-BEATS hyperparameters-stack_types, n_blocks, mlp_units, and learning_rate-while keeping the remaining settings fixed (max_steps = 1000) to isolate the effect of the search strategy. Importantly, MAPE is computed on the validation set during tuning, and the test set is reserved only for final performance reporting, thereby preventing data leakage. The learning rate is explored using a log-uniform range of [1e−4, 5e−3] to balance stable training convergence and sufficient adaptation capacity for deep MLP-based forecasting. Overall, RS provides a strong, assumption-free baseline; TPE enables informed sampling via probabilistic modeling; and GA offers population-based exploration through elitism, crossover, and mutation. Table 2 summarizes the experimental configurations of the three hyperparameter optimization methods-RS, TPE, and GA-applied to the N-BEATS model. All methods use the same stopping rule and training configuration to ensure that performance differences reflect search efficiency rather than training cost.

**Table 2.** Configuration summary of hyperparameter optimization methods for the N-BEATS model

| Configuration | RS | TPE | GA |
|---|---|---|---|
| Trials | 100 | 100 | 100(10×10) |
| Objective | Minimize MAPE | Minimize MAPE | Minimize MAPE |
| Seed | 42 | 42 | 42 |
| Search space | stack_types, n_blocks, mlp_units, learning_rate | stack_types, n_blocks, mlp_units, learning_rate | stack_types, n_blocks, mlp_units, learning_rate |
| LR range | log-uniform 1e−4 → 5e−3 | log-uniform 1e−4 → 5e−3 | log-uniform 1e−4 → 5e−3 |
| Algorithm | Random sampling | TPE sampler | GA (pop=10, gen=2, elite=2, pc=0.9, pm=0.2) |

## 2.3. Flowchart of the Proposed Method Evaluation

Fig. 1 illustrates the complete workflow of the proposed hyperparameter optimization framework for the N-BEATS model, integrating three optimization strategies: Random Search (RS), Tree-structured Parzen Estimator (TPE), and Genetic Algorithm (GA).

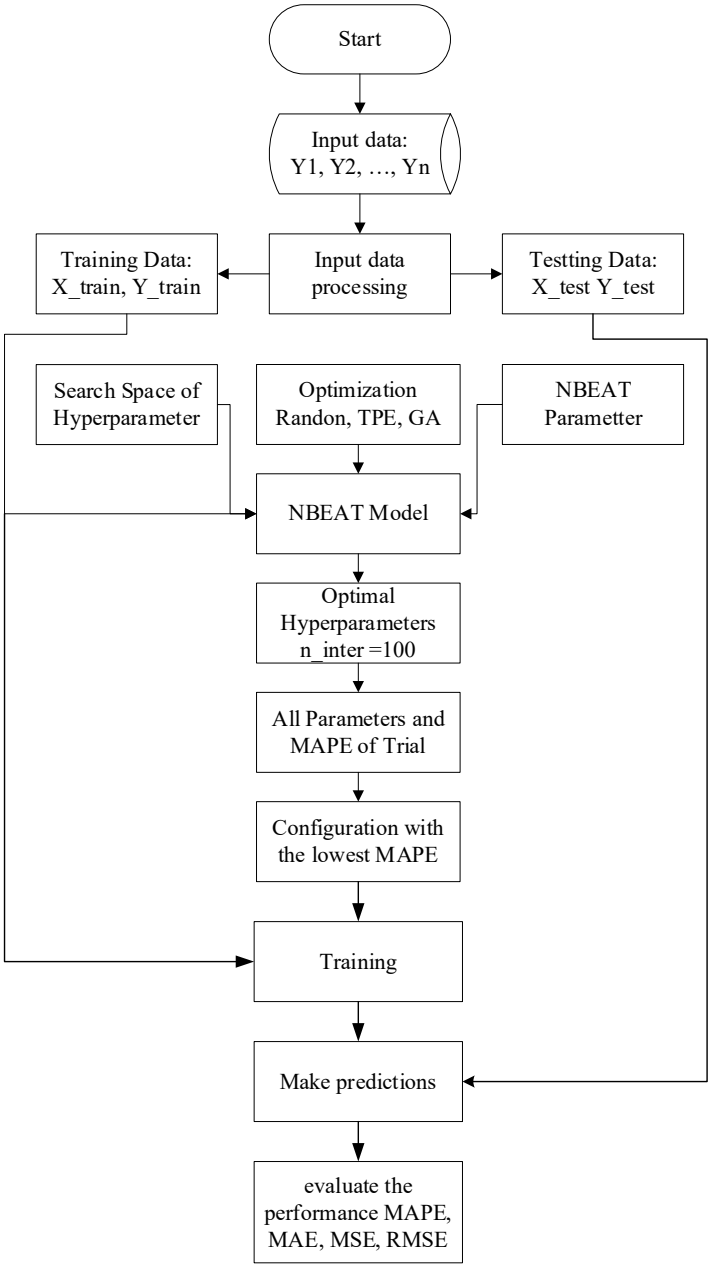


**Fig. 1.** NBEATS model optimization and evaluation flow

The process begins with the input load time series Y1, Y2, ..., Yn. The data are then processed and transformed into supervised learning samples, and subsequently split chronologically into a training/development set (X_train, Y_train) and a test set (X_test, Y_test). This chronological split preserves the temporal structure of the load series, ensuring that the model is evaluated on unseen data. Next, the hyperparameter search space is defined consistently with Table 2, focusing on four key hyperparameters that strongly influence the representational capacity and learning dynamics of N-BEATS: stack_types, n_blocks, mlp_units, and learning_rate. The remaining settings are kept fixed (max_steps = 1000) to ensure a fair comparison. Under the same experimental constraints and evaluation budget, RS, TPE, and GA independently generate candidate configurations and train the N-BEATS model accordingly. For each trial, model performance is assessed using time-series cross-validation (cv = 3) on (X_train, Y_train), and the objective is defined as the mean CV MAPE across the three folds. The process is repeated until 100 trials (n_iter = 100) are completed, after which the configuration with the lowest mean CV MAPE is selected as the optimal hyperparameter set.

Finally, the N-BEATS model is retrained from scratch using the selected hyperparameters on the whole training/development set, and forecasts are generated for the test period. The final test performance is reported using multiple evaluation metrics, including MAPE, MAE, MSE, and RMSE, which provide a comprehensive assessment of both relative and absolute forecasting errors. Overall, Fig. 1 encapsulates a systematic and reproducible workflow that enables a fair comparison of RS, TPE, and GA for tuning N-BEATS in short-term load forecasting, while ensuring that the test set is used only for final reporting to prevent data leakage.

### 2.4. General Optimization Procedure for NBEATS Tuning

INPUT: time-series df, H, INPUT_SIZE, TRAIN_SIZE, TEST_SIZE, search space S (conditional), optimizer OPT∈{RS,TPE,GA}, budget B

1) Data preparation (common)
   - Sort df by time
   - Split: train = last TRAIN_SIZE points, test = last TEST_SIZE points
   - Convert to NeuralForecast format: (unique_id, ds, y)
2) Define search space S (standard)
   - stack_types ∈ {A, B, C}
   - n_blocks, mlp_units depend on stack_types
   - learning_rate ~ log-uniform
   - dropout fixed (not optimized)
3) Optimization loop (standard)
   - Initialize OPT_state, best_score=+∞, best_params=None
   - For k = 1..B:

   params = PROPOSE(OPT, OPT_state, S)  # differs by RS/TPE/GA

   Train NBEATS(params) on train

   Predict on the test horizon

   score = MAPE(y_true, y_pred)

   Update best if score improves

   OPT_state = UPDATE(OPT, OPT_state, params, score)

OUTPUT: best_params, best_score

## 3. Experimental Setup

### 3.1. Dataset Description

The experimental analysis in this study utilizes two regional electricity demand datasets from Australia's National Electricity Market (NEM), specifically those corresponding to New South Wales (NSW) and Queensland (QLD). Each dataset contains 122,735 records with two main fields: SETTLEMENTDATE, representing the timestamp of demand measurements sampled at 30-minute intervals starting from January 1, 2015, and TOTALDEMAND, representing the aggregated electrical load in megawatts (MW). The 30-minute resolution (48 observations per day) provides high-granularity temporal information suitable for one-day-ahead short-term load forecasting under the fixed forecasting setup used in this study. Table 3 summarizes the datasets.

**Table 3.** Dataset summary for NSW and QLD

| Region | Number of Records | Variables | Time Interval | Target Variable |
|---|---|---|---|---|
| NSW | 122,735 | Settlement date, total demand | 30 minutes | Totaldemand |
| QLD | 122,735 | Settlement date, total demand | 30 minutes | Totaldemand |

In preprocessing, the raw CSV files are imported, and the SETTLEMENTDATE field is parsed into a standard datetime format. The data is then sorted chronologically to preserve the integrity of the time series. Because each entry represents a 30-minute interval, the data form a high-resolution daily load profile with 48 points per day.

For model development and evaluation, the data are split chronologically to prevent information leakage. Specifically, the most recent 28 days are used as the development window (X_train, Y_train), and the final day is reserved as the held-out test set (X_test, Y_test). During hyperparameter tuning, model performance is assessed using time-series cross-validation with cv = 3 within the 28-day development window (an expanding-window split, with no shuffling). For each trial, the model is trained on earlier segments and validated on subsequent segments, and the objective value is defined as the mean CV MAPE across the three folds. After identifying the best configuration, the N-BEATS model is retrained from scratch on the whole 28-day development window and evaluated once on the held-out test day, ensuring that the test set is used only for final reporting.

Fig. 2 and Fig. 3 display the NSW and QLD electricity demand series at a 30-minute resolution, along with the chronological boundary between the 28-day development window and the final-day test period. The separation confirms that the test set is entirely unseen during model selection and tuning. Overall, both regions exhibit clear daily periodic patterns, with the amplitude and variability differing across areas, which supports a meaningful cross-regional evaluation under the same forecasting protocol (48-step, one-day-ahead prediction at 30-minute sampling).

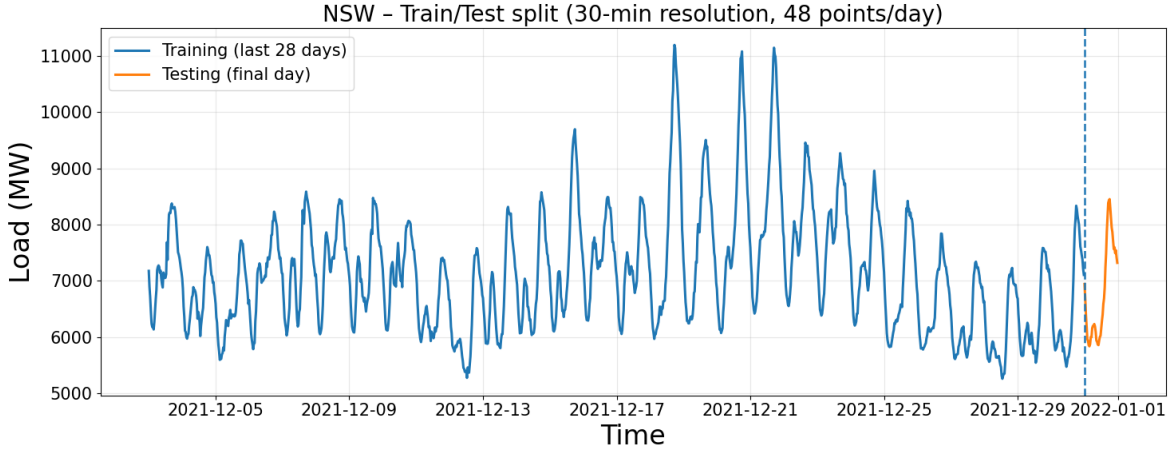


**Fig. 2.** NSW electricity demand series with chronological development–test split


**Fig. 3.** QLD electricity demand series with chronological development–test split

Regarding data characteristics, Fig. 2 (NSW) exhibits a clear daily periodic pattern, while several pronounced demand spikes appear around mid-December, indicating substantial temporal variability and potential short-term influences. In contrast, Fig. 3 (QLD) also shows a distinct daily cycle with relatively more regular oscillations, where peak-trough patterns repeat more consistently throughout the 28-day development window. The held-out test segment remains fully separated after the split boundary, ensuring that evaluation is performed on unseen data. Overall, the two figures confirm that the chronological split is consistent with the experimental design: the model is developed using the most recent 28 days and evaluated on the final day, matching the forecasting setup of predicting 48 steps (one day) under a 30-minute sampling interval. During hyperparameter tuning, performance is assessed within the 28-day development window using time-series cross-validation (cv = 3), while the test day is reserved only for final reporting to prevent data leakage.

### 3.2. NBEAT Model Configuration

In this study, the N-BEATS model is implemented using the NeuralForecast library under a controlled and reproducible experimental setup. To ensure a fair and consistent basis for subsequent comparisons, the baseline N-BEATS model follows the library's default configuration, which is summarized in Table 1. In other words, before applying any hyperparameter optimization, the internal model architecture and training settings are kept at their default values, and only the task-specific parameters required by the dataset characteristics and forecasting objective are explicitly specified.

Given that the electricity demand data are sampled every 30 minutes, the data frequency is set to freq = '30min'. Under this sampling interval, one day corresponds to 48 time steps; therefore, the forecasting horizon is defined as h = 48, which represents a one-day-ahead (24-hour) multi-step forecast. To provide sufficient historical context and capture weekly temporal patterns that commonly appear in electricity demand (weekday–weekend differences), the input window is set to input_size = 48 × 7 = 336. This setting means that, at each forecast origin, the model receives the previous seven days (336 points) as input and produces the following 48 points as output, matching the fixed forecasting protocol used throughout the experiments.

To support reproducibility and enable direct comparison across repeated runs and across optimization strategies, the random seed is fixed at random_seed = 42. This ensures that any stochastic components in training (parameter initialization and mini-batch sampling) are controlled as much as possible under the same experimental conditions. Overall, this configuration establishes a standardized baseline aligned with the default N-BEATS setup (Table 1), while explicitly defining the forecasting-task parameters (freq, h, and input_size) according to the 30-minute resolution data and the one-day-ahead forecasting target. This baseline is then used as the reference point for evaluating the impact of hyperparameter optimization in the subsequent sections.

### 3.3. Hyperparameter Range

The hyperparameter search space was constructed to investigate four influential N-BEATS parameters that directly affect forecasting accuracy and training dynamics: stack_types, n_blocks, mlp_units, and learning_rate. These four parameters were selected because they control (i) the stack composition (identity/trend/seasonality), (ii) the model depth through the block configuration, (iii) the representational capacity of the MLP blocks, and (iv) the convergence behavior during optimization. All remaining settings (batch size, max_steps, optimizer, and other NeuralForecast defaults) were kept fixed to isolate the effect of the tuned parameters and to maintain a controlled comparison across optimization strategies. To address reproducibility and prevent data leakage during tuning, all strategies (RS, TPE, and GA) optimize the same search space under the same evaluation budget (100 trials) and the same objective, defined as minimizing mean validation MAPE computed using time-series cross-validation (cv = 3) on the development window. At the same time, the held-out test day is reserved only for final reporting. The candidate values/ranges in Table 4 were chosen to cover practical configurations of N-BEATS while maintaining stable training under the fixed setup (freq = '30min', h = 48, input_size = 336, max_steps = 1000). Table 4 summarizes the unified search space used consistently in the N-BEATS code for RS, TPE, and GA.

This search space (Table 4) was designed to quantify the contribution of each tuned hyperparameter while ensuring a fair and stable training setup by keeping all remaining settings at their default values. The candidate stack_types configurations were selected to compare different degrees of decomposition (identity only, trend–seasonality, and their combination), which directly affects how N-BEATS represents daily and weekly structures in electricity demand. The n_blocks candidates were restricted to a small set of shallow-to-moderate depths to examine the effect of model depth and block granularity without introducing excessive complexity under the fixed development window. The mlp_units range (256–1024) was chosen to cover moderate to high representational capacity for the MLP blocks, enabling an explicit comparison of under-parameterized versus higher-capacity settings while remaining computationally feasible. Finally, the learning rate was sampled from a log-uniform range of [1e-4, 5e-3] to balance stable convergence (smaller steps) and sufficient adaptation speed (larger steps) for deep MLP-based optimization. Overall, these choices provide a controlled yet representative search space for evaluating RS, TPE, and GA under identical constraints.

### 3.4. Evaluation Metrics

To quantitatively assess the forecasting performance of the NBEATS model, four widely used statistical error metrics were employed: Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE). These metrics offer complementary insights into model accuracy, robustness, and sensitivity to large deviations. The formulas for each metric are defined as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{5}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{6}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{7}$$

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \tag{8}$$

Where $y_t$ denotes the actual load value t, end sub, is the corresponding forecasted value, and $N$ is the total number of test samples.

The MAPE expresses the average relative prediction error in percentage form, making it easily interpretable and suitable for comparing across datasets. The MAE captures the average magnitude of prediction errors, disregarding their direction, and provides a robust measure against outliers. The MSE penalizes large deviations more heavily due to the squaring operation. At the same time, the RMSE, as its square root, retains the same unit as the target variable, allowing for an intuitive interpretation of forecast deviations in megawatts (MW). Together, these metrics ensure a comprehensive and balanced evaluation of model accuracy and stability in electric load forecasting.

**Table 4.** Hyperparameter search space for N-BEATS optimization

| Hyperparameter | Search Space | Type |
|---|---|---|
| **stack_types** | {identity, trend+seasonality, identity+trend+seasonality} | Categorical |
| **n_blocks** | Structured block settings depending on stack type ([1], [2], [3], [4]; [1,1], [2,2], [3,3]; [1,1,1], [2,2,2], [3,3,3]) | Discrete |
| **mlp_units** | Predefined MLP configurations for each stack type (256–1024 neurons per layer) | Discrete |
| **learning_rate** | 1e−4 to 5e−3 (log-uniform) | Continuous |

### 3.5. Experimental Environment

All experiments were conducted on Google Colab, a cloud-based platform that provides a flexible environment for developing and evaluating deep learning models. The runtime was configured with Python 3 using the standard Colab environment. All computations were performed on the CPU, with a clock speed of 1.71 GHz and a high-RAM mode enabled to ensure sufficient memory for model training and data processing, thereby maintaining consistency and reproducibility across runs. Although Colab provides optional GPU/TPU acceleration, this study intentionally used the CPU setting to ensure that the reported results can be reproduced under a consistent and widely accessible configuration. The main software libraries employed in the experiments included NeuralForecast (for implementing NBEATS and managing the training and forecasting pipeline), NumPy and Pandas (for data preprocessing and manipulation), Matplotlib (for visualization), and Scikit-learn (for computing evaluation metrics such as MAPE, MAE, MSE, and RMSE). This configuration ensured a stable and reproducible computational environment for all experiments in this study.

## 4. Results and Discussion

Table 5 presents the default and optimized hyperparameter configurations of the N-BEATS model for the NSW and QLD datasets, corresponding to the three optimization methods Random Search (RS), Tree-structured Parzen Estimator (TPE), and Genetic Algorithm (GA).

Table 5 presents the default and optimized N-BEATS hyperparameter configurations for the NSW and QLD datasets, obtained through Random Search (RS), Tree-structured Parzen Estimator (TPE), and Genetic Algorithm (GA) methods. Following the reviewers' requests for clarity and reproducibility, the table focuses only on the tuned architectural and training parameters (stack_types, n_blocks, mlp_units, learning_rate) together with max_steps. At the same time, all other settings are kept at their NeuralForecast defaults to isolate the effect of hyperparameter tuning. Additionally, the optimized configurations are selected under the revised tuning protocol, utilizing 100 trials per optimizer with validation-based model selection (time-series cross-validation with cv = 3). The held-out test day is reserved only for final reporting to avoid data leakage.

For NSW, the default configuration uses two stacks, ['trend', 'seasonality'], with n_blocks = [3, 3], mlp_units = 512, learning_rate = 0.00100, and max_steps = 1000. After optimization, RS selects a three-stack configuration ['identity', 'trend', 'seasonality'] with n_blocks = [2, 2, 2] and two-layer MLPs for each stack (mlp_units = [[512, 512], [512, 512], [512, 512]]), together with a lower learning_rate of 0.00014. TPE maintains the two-stack trend–seasonality structure but reduces the depth to n_blocks = [1, 1] with mlp_units = [[512, 512], [512, 512]] and a learning rate of 0.00025.

GA selects a single-stack identity configuration ['identity'] with n_blocks = [4], increases the MLP width to mlp_units = [[1024, 1024]], and uses learning_rate = 0.00057, while max_steps remains fixed at 1000 for all methods.

**Table 5.** Default and optimized hyperparameters across optimization methods (N-BEATS)

| Region | Method | stack_types | n_blocks | mlp_units | learning_rate | max_steps |
|---|---|---|---|---|---|---|
| NSW | Default | ['trend', 'seasonality'] | [3,3] | 512 | 0.00100 | 1000 |
| | RS | ['identity', 'trend', 'seasonality'] | [2, 2, 2] | [[512, 512], [512, 512], [512, 512]] | 0.00014 | 1000 |
| | TPE | ["trend", "seasonality"] | [1, 1] | [[512, 512], [512, 512]] | 0.00025 | 1000 |
| | GA | ['identity'] | [4] | [[1024, 1024]] | 0.00057 | 1000 |
| QLD | Default | ['trend','seasonality'] | [3,3] | 512 | 0.00100 | 1000 |
| | RS | ['trend', 'seasonality'] | [2, 2] | [[512, 512], [512, 512]] | 0.00124 | 1000 |
| | TPE | ["identity", "trend", "seasonality"] | [2, 2, 2] | [[256, 256], [256, 256], [256, 256]] | 0.00070 | 1000 |
| | GA | ['identity', 'trend', 'seasonality'] | [1, 1, 1] | [[512, 512], [512, 512], [512, 512]] | 0.00043 | 1000 |

For QLD, the default configuration is again ['trend', 'seasonality'] with n_blocks = [3, 3], mlp_units = 512, learning_rate = 0.00100, and max_steps = 1000. After optimization, RS retains the trend–seasonality stacks and reduces depth to n_blocks = [2, 2] with mlp_units = [[512, 512], [512, 512]], and selects a learning rate of 0.00124. TPE selects the three-stack configuration ['identity', 'trend', 'seasonality'] with n_blocks = [2, 2, 2] and smaller MLP widths (mlp_units = [[256, 256], [256, 256], [256, 256]]), with learning_rate = 0.00070. GA also uses ['identity', 'trend', 'seasonality'] but with a shallower depth n_blocks = [1, 1, 1], mlp_units = [[512, 512], [512, 512], [512, 512]], and learning_rate = 0.00043. Notably, all optimized configurations in Table 5 are selected from the unified search space in Table 4 using the same validation protocol, while the held. The out-of-test day is used only for final reporting; the resulting forecasting performance is summarized in Table 6.

**Table 6.** Forecasting performance for NSW and QLD

| Data | Model | MSE | RMSE | MAE | MAPE |
|---|---|---|---|---|---|
| NSW | Default | 47719.7 | 218.4 | 180.0 | 2.56 |
| | RS | 21882.6 | 147.9 | 110.7 | 1.58 |
| | TPE | 16811.1 | 129.7 | 88.7 | 1.29 |
| | GA | 14259.6 | 119.4 | 92.0 | 1.31 |
| QLD | Default | 44430.0 | 210.8 | 162.0 | 2.80 |
| | RS | 21634.1 | 147.1 | 108.3 | 1.84 |
| | TPE | 19499.4 | 139.6 | 106.6 | 1.79 |
| | GA | 24153.2 | 155.4 | 129.3 | 2.19 |

Table 6 presents the forecasting performance of the N-BEATS model for both NSW and QLD, evaluated using MSE, RMSE, MAE, and MAPE. The default configuration is compared with three hyperparameter optimization methods: Random Search (RS), Tree-structured Parzen Estimator (TPE), and Genetic Algorithm (GA). For NSW, all optimized configurations outperform the default model across all metrics. In particular, TPE achieves the lowest percentage error (MAPE = 1.29) and the lowest MAE (88.7 MW), reducing MAPE from 2.56 to 1.29 (approximately 49.6%) and MAE from 180.0 to 88.7 MW (approximately 50.7%) compared to the default. GA yields the lowest squared-error metrics for NSW (MSE = 14,259.6 and RMSE = 119.4), corresponding to reductions of approximately 70.1% in MSE and 45.3% in RMSE compared with the default (MSE = 47,719.7, RMSE = 218.4). These results indicate that, for NSW, the best configuration is metric-dependent: TPE performs best for MAPE/MAE, whereas GA performs best for MSE/RMSE.

For QLD, the optimized models also consistently improve upon the default baseline (MSE = 44430.0, RMSE = 210.8, MAE = 162.0, MAPE = 2.80). Among the three strategies, TPE delivers the best overall performance across all reported metrics (MSE = 19499.4, RMSE = 139.6, MAE = 106.6, MAPE = 1.79), reducing MAPE by approximately 36.1% and MSE by approximately 56.1% compared to the default. RS achieves competitive improvements (MAPE = 1.84 and MSE = 21634.1), while GA exhibits smaller gains for QLD compared to RS and TPE (MAPE = 2.19 and MSE = 24153.2). Overall, Table 6 confirms that standardized hyperparameter optimization improves N-BEATS forecasting accuracy in both regions, while the most effective optimization strategy can differ across regions and evaluation metrics. The MAPE comparisons for NSW and QLD are further visualized in Fig. 4 based on the results reported in Table 6.

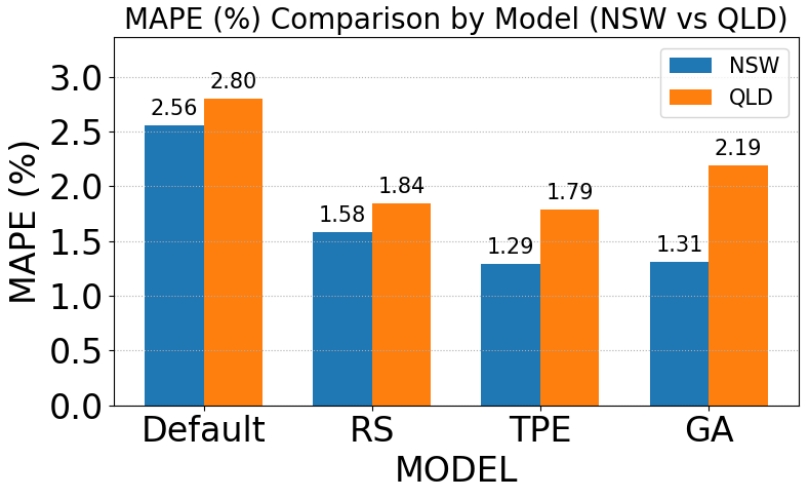


**Fig. 4.** MAPE comparison (NSW vs QLD)

To better understand how these error reductions manifest over time, Fig. 5 and Fig. 6 zoom in on the NSW test period and compare the temporal evolution of the forecasts against the actual load. Fig. 5 shows the last seven training days together with one test day and the corresponding forecasts for NSW. The training segment highlights the strong daily seasonality and variability of the load. In contrast, the zoomed test segment reveals that all optimized models (RS, TPE, and GA) track the actual test curve more closely than the default N-BEATS configuration. In particular, the optimized forecasts better capture both the timing and amplitude of the rising and falling ramps during the test period, whereas the default model exhibits visibly larger deviations around the peak and trough regions. Fig. 6 further magnifies the 24-hour test day, providing a more precise comparison between the actual load and the four forecast trajectories. Throughout the entire day, the RS, TPE, and GA curves remain tightly clustered around the ground truth, particularly during the morning and evening peak hours. In contrast, the default model tends to either overestimate or underestimate demand in these high-load intervals. These plots visually confirm the numerical improvements reported in Table 6 and indicate that the hyperparameter-optimized configurations not only reduce aggregated error metrics but also yield more faithful reconstructions of the intraday load profile for NSW.

While Fig. 5 and Fig. 6 focus on the NSW test period, Fig. 7 and Fig. 8 present the corresponding zoomed views for QLD, allowing a visual assessment of how the different N-BEATS configurations behave on this region.

Fig. 7 illustrates the last seven training days together with one test day and the corresponding forecasts for QLD. Similar to NSW, the training segment exhibits clear daily cyclic patterns, but with a slightly lower overall demand level. In the zoomed test portion, all optimized models again follow the shape of the actual load more closely than the default configuration. The RS, TPE, and GA forecasts, in particular, align well with the timing and magnitude of the rapid increase in demand at the end of the test day, whereas the default N-BEATS model shows more noticeable deviations around the rising ramp and the peak.

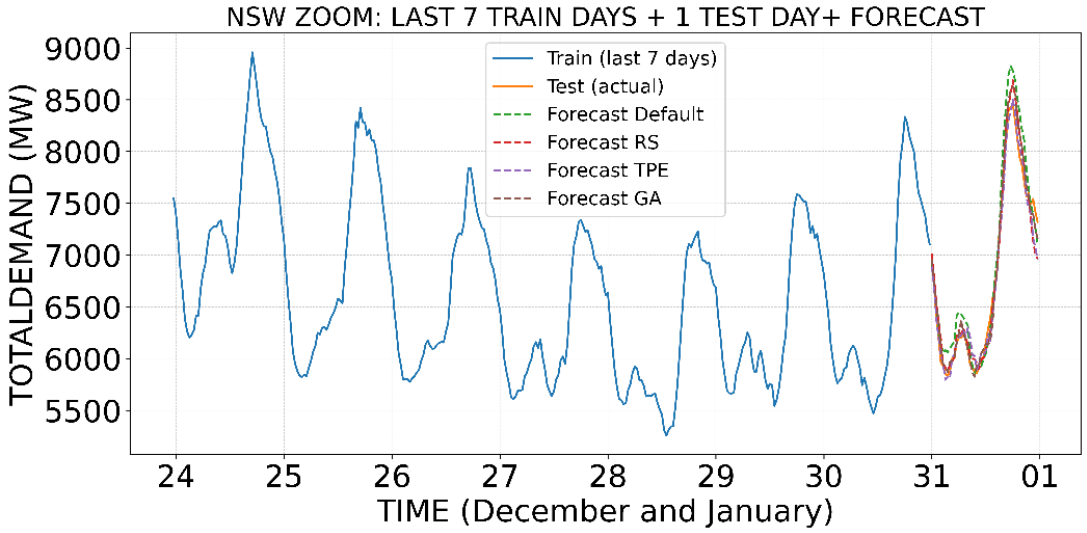


**Fig. 5.** NSW forecast zoom

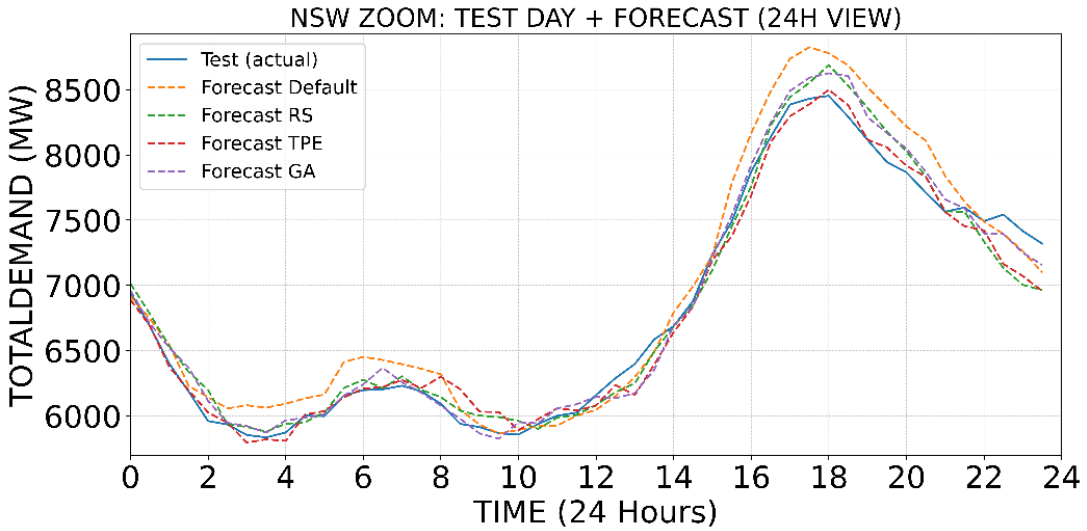


**Fig. 6.** NSW 24h view

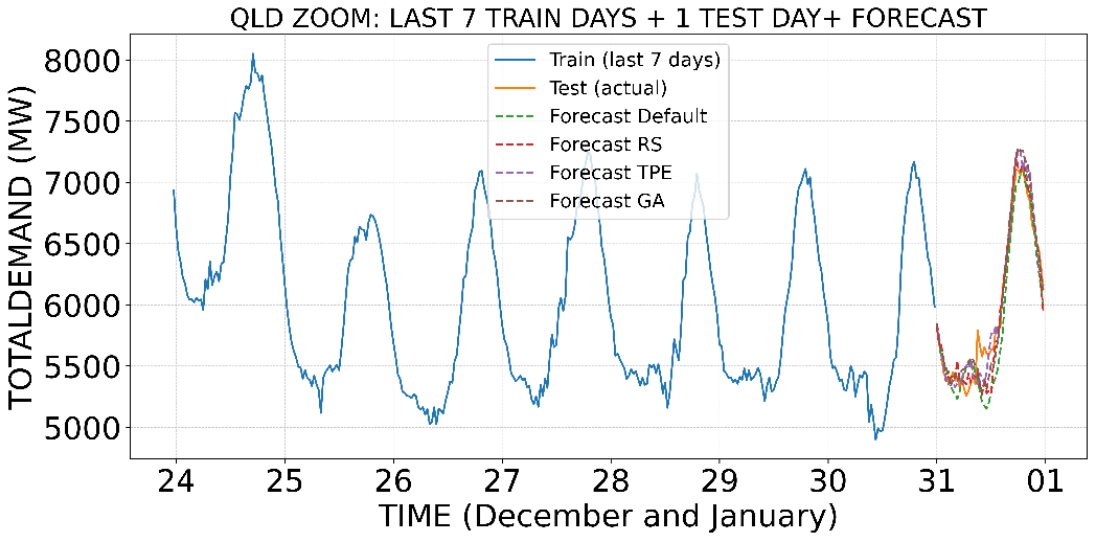


**Fig. 7.** QLD forecast zoom

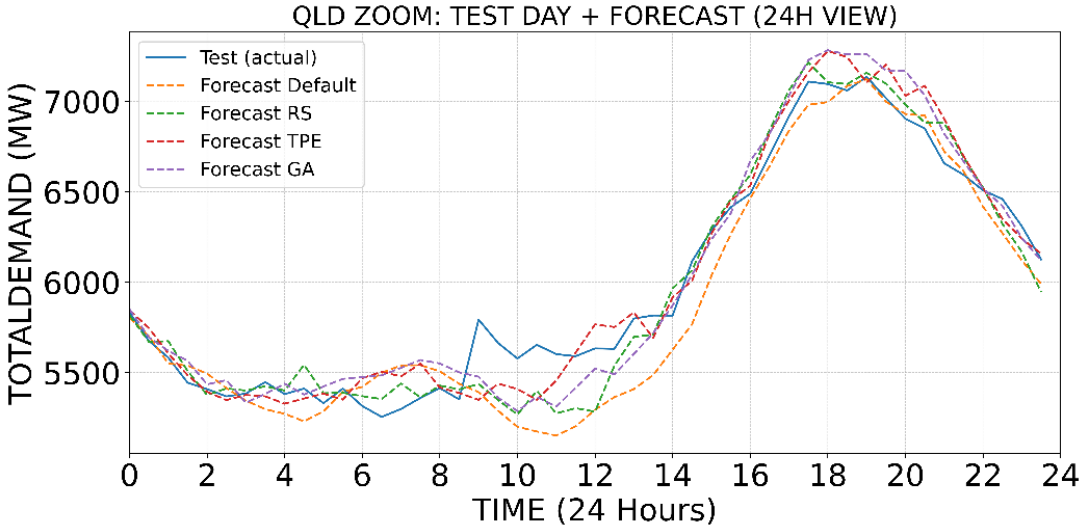


**Fig. 8.** QLD 24h view

Fig. 8 provides a 24-hour view of the QLD test day, enabling a detailed comparison between the actual load and the four forecast trajectories. Throughout the day, the optimized models remain closer to the ground truth than the default model, with RS generally offering the best match to both the morning recovery and the evening peak, consistent with the quantitative results in Table 6. TPE and GA also capture the overall intraday pattern but tend to slightly overestimate or underestimate the load in some intervals. At the same time, the default model exhibits the most considerable underestimation during the midday-to-afternoon ramp. Taken together, Fig. 7 and Fig. 8 visually confirm that the hyperparameter-optimized configurations not only reduce aggregate error metrics but also provide more realistic reconstructions of the hourly load profile in this region. Next, Fig. 9 presents the runtime of RS, TPE, and GA on NSW and QLD, thereby clarifying the computational cost and the accuracy–runtime trade-off when selecting an optimization strategy.

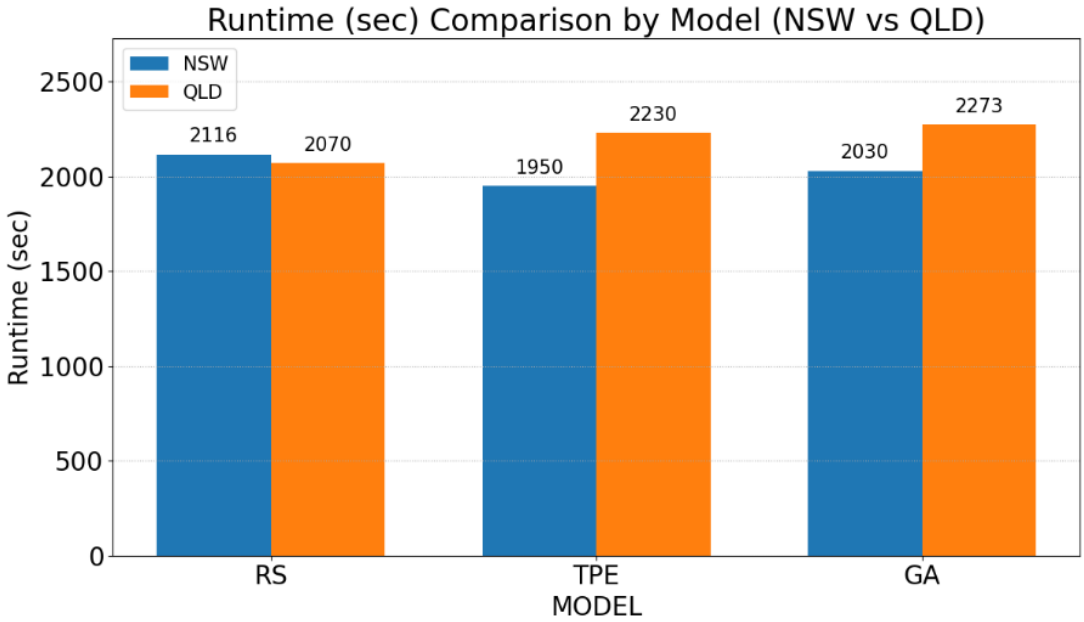


**Fig. 9.** Runtime (sec) comparison of RS, TPE, and GA on NSW vs QLD

Fig. 9 compares the runtime of RS, TPE, and GA on NSW and QLD under the same evaluation budget. For NSW, TPE achieves the lowest runtime (1950s), followed by GA (2030s) and RS (2116s). In contrast, QLD exhibits a consistent increase in runtime from RS (2070 s) to TPE (2230 s) and GA

(2273 s). Overall, the runtime gaps remain moderate (within 166 s for NSW and 203 s for QLD), indicating that the three optimization strategies are all computationally feasible in practice. Notably, GA tends to require the highest runtime on QLD, whereas TPE is the most time-efficient on NSW, suggesting that computational cost may vary with regional data characteristics; nevertheless, the observed differences are not significant enough to dominate the choice of optimizer when accuracy is the primary objective.

The results in Table 6 and the forecasting figures indicate that the performance of RS, TPE, and GA is **both region-dependent (NSW/QLD) and metric-dependent**, meaning there is no single "universally best" optimizer for all cases. This discrepancy may arise from differences in data variability and error distribution across regions, where metrics such as MAPE/MAE emphasize relative errors over the entire series, whereas MSE/RMSE are more sensitive to large deviations; consequently, the configuration that is optimal under one metric may not be optimal under another. In terms of search behavior, Random Search broadly explores the parameter space and can yield competitive results when the optimization landscape is not overly complex. In contrast, TPE performs guided sampling and therefore tends to be more stable under the same evaluation budget. GA leverages evolutionary mechanisms to exploit promising parameter combinations, which may provide advantages in specific settings but can also be sensitive to GA configurations and computational cost. Therefore, the choice of an optimization method should be aligned with the **target metric** and the **regional data characteristics**, rather than relying on a single fixed approach.

## 5. Conclusion

This study examined how three hyperparameter optimization strategies-Random Search (RS), Tree-structured Parzen Estimator (TPE), and Genetic Algorithm (GA)-affect the one-day-ahead forecasting performance of the N-BEATS model on two Australian NEM regions (NSW and QLD). Using a unified and reproducible tuning protocol, the optimizers searched key N-BEATS hyperparameters (stack_types, n_blocks, mlp_units, and learning_rate) while keeping the remaining training settings fixed (max_steps = 1000). The resulting configurations (Table 5) demonstrate that competitive performance can be achieved by appropriately balancing stack composition, model depth, MLP capacity, and learning rate under the same training budget, highlighting the practical value of standardized hyperparameter tuning for N-BEATS load forecasting.

Across both regions, hyperparameter tuning yields substantial error reductions compared with the default baseline (Table 6). For NSW, the best configuration depends on the evaluation metric: TPE provides the lowest percentage and absolute errors (MAPE decreases from 2.56% to 1.29% and MAE from 180.0 MW to 88.7 MW), while GA achieves the lowest squared-error metrics (MSE decreases from 47,719.7 to 14,259.6 and RMSE from 218.4 to 119.4). For QLD, TPE delivers the best overall results across all reported metrics, reducing MAPE from 2.80% to 1.79%, MAE from 162.0 MW to 106.6 MW, and MSE from 44,430.0 to 19,499.4. The zoomed forecast plots (Fig. 5, Fig. 6, Fig. 7, Fig. 8) visually support these numerical results, showing that optimized configurations track the intraday load profile more closely, particularly around ramping and peak periods.

Despite these improvements, this study has several limitations. First, the experiments are conducted in a univariate setup and with a fixed one-day-ahead horizon under a fixed development window, and results are reported for two regions only. Therefore, generalization to other areas, longer horizons, and multivariate feature sets requires further investigation. Second, although the tuning protocol is standardized, the evaluation budget remains finite, and no formal statistical significance testing is included. From a practical standpoint, the results suggest using TPE as a strong default choice when the goal is consistent improvements across regions and metrics. At the same time, GA may be preferred when minimizing squared-error criteria is prioritized (as observed for NSW).

Future work will extend this framework to include multivariate inputs (such as weather and calendar effects), longer forecasting horizons, and broader regional testing. It will also incorporate additional robustness checks, such as repeated runs with uncertainty reporting, runtime–accuracy

trade-off analysis, and statistical significance testing, to strengthen reliability for real-world deployment.

**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] R. K. Jagait, M. N. Fekri, K. Grolinger and S. Mir, "Load Forecasting Under Concept Drift: Online Ensemble Learning With Recurrent Neural Network and ARIMA," *IEEE Access*, vol. 9, pp. 98992-99008, 2021, https://doi.org/10.1109/ACCESS.2021.3095420.

[2] Y. -C. Jin *et al*., "Models for COVID-19 Data Prediction Based on Improved LSTM-ARIMA Algorithms," *IEEE Access*, vol. 12, pp. 3981-3991, 2024, https://doi.org/10.1109/ACCESS.2023.3347403.

[3] Y. -C. Jin, Q. Cao, K. -N. Wang, Y. Zhou, Y. -P. Cao and X. -Y. Wang, "Prediction of COVID-19 Data Using Improved ARIMA-LSTM Hybrid Forecast Models," *IEEE Access*, vol. 11, pp. 67956-67967, 2023, https://doi.org/10.1109/ACCESS.2023.3291999.

[4] 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, pp. 124715-124727, 2022, https://doi.org/10.1109/ACCESS.2022.3224938.

[5] 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," *Applied Intelligence*, vol. 54, no. 1, pp. 1003-1019, 2024, https://doi.org/10.1007/s10489-023-05230-y.

[6] G. R. A. Brito, A. R. 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. 671, 2021, https://doi.org/10.1007/s42452-021-04667-5.

[7] 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.

[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," *2023 IEEE Colombian Conference on Applications of Computational Intelligence (ColCACI)*, pp. 1-6, 2023, https://doi.org/10.1109/ColCACI59285.2023.10225803.

[9] 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*, vol. 16, no. 8, p. 3251, 2023, https://doi.org/10.3390/ma16083251.

[10] C. Yıldız, "An integrated model based on deep kernel extreme learning machine and variational mode decomposition for day-ahead electricity load forecasting," *Neural Computing and Applications*, vol. 35, no. 25, pp. 18763-18781, 2023, https://doi.org/10.1007/s00521-023-08702-x.

[11] M. M. M. K, I. B, H. Prasad and S. TD, "Load Forecasting Using Random Forest Regression Algorithm in Machine Learning," *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, pp. 1-6, 2024, https://doi.org/10.1109/ICSTEM61137.2024.10560982.

[12] M. Sayadlou, M. S. Naderi, M. Abedi, S. Esmaeili and M. Amini, "A Comprehensive Deep Learning Method for Short-Term Load Forecasting," *2022 30th International Conference on Electrical Engineering (ICEE)*, pp. 1074-1078, 2022, https://doi.org/10.1109/ICEE55646.2022.9827325.

[13] H. Hou *et al*., "Load Forecasting Combining Phase Space Reconstruction and Stacking Ensemble Learning," *IEEE Transactions on Industry Applications*, vol. 59, no. 2, pp. 2296-2304, 2023, https://doi.org/10.1109/TIA.2022.3225516.

[14] 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 Forecas," *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pp. 297-302, 2021, https://doi.org/10.1109/RTEICT52294.2021.9573988.

[15] 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 for Science and Engineering*, vol. 47, no. 6, pp. 7367-7385, 2022, https://doi.org/10.1007/s13369-022-06560-8.

[16] W. Niu, X. Song, H. Wang and J. Chen, "Forecasting the load flow of Engine Driven Pump based on Light Gradient Boosting Machine Model," *2021 IEEE International Conference on Computer Science, Electronic Information Engineering and Intelligent Control Technology (CEI)*, pp. 771-774, 2021, https://doi.org/10.1109/CEI52496.2021.9574553.

[17] Z. Qinghe, X. Wen, H. Boyan, W. Jong, and F. Junlong, "Optimised extreme gradient boosting model for short-term electric load demand forecasting of regional grid system," *Scientific Reports*, vol. 12, no. 1, pp. 1-12, 2022, https://doi.org/10.1038/s41598-022-22024-3.

[18] M. Mohamed, F. E. Mahmood, M. A. Abd, M. Rezkallah, A. Hamadi and A. Chandra, "Load Demand Forecasting Using eXtreme Gradient Boosting (XGboost)," *2023 IEEE Industry Applications Society Annual Meeting (IAS)*, pp. 1-7, 2023, https://doi.org/10.1109/IAS54024.2023.10406613.

[19] M. M. Hasan, N. El-Tazi, R. Moawad and A. H. B. Eissa, "TSB-Forecast: A Short-Term Load Forecasting Model in Smart Cities for Integrating Time Series Embeddings and Large Language Models," *IEEE Access*, vol. 13, pp. 141694-141716, 2025, https://doi.org/10.1109/ACCESS.2025.3597421.

[20] Z. Gao, S. Hu, H. Sun, Z. Wang, S. Liu, and F. Yang, "Day-ahead dynamic thermal line rating forecasting and power transmission capacity calculation based on ForecastNet," *Electric Power Systems Research*, vol. 220, p. 109350, 2023, https://doi.org/10.1016/j.epsr.2023.109350.

[21] S. Park, S. Jung, S. Jung, S. Rho, and E. Hwang, "Sliding window-based LightGBM model for electric load forecasting using anomaly repair," *The Journal of Supercomputing*, vol. 77, no. 11, pp. 12857-12878, 2021, https://doi.org/10.1007/s11227-021-03787-4.

[22] S. Lei *et al*., "A Short-term Net Load Forecasting Method Based on Two-stage Feature Selection and LightGBM with Hyperparameter Auto-Tuning," *2023 IEEE/IAS 59th Industrial and Commercial Power Systems Technical Conference (I&CPS)*, pp. 1-6, 2023, https://doi.org/10.1109/ICPS57144.2023.10142095.

[23] Y. Tan, Z. Teng, C. Zhang, G. Zuo, Z. Wang and Z. Zhao, "Long-Term Load Forecasting Based on Feature fusion and LightGBM," *2021 IEEE 4th International Conference on Power and Energy Applications (ICPEA)*, pp. 104-109, 2021, https://doi.org/10.1109/ICPEA52760.2021.9639313.

[24] X. Yang and Z. Chen, "A Hybrid Short-Term Load Forecasting Model Based on CatBoost and LSTM," *2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP)*, pp. 328-332, 2021, https://doi.org/10.1109/ICSP51882.2021.9408768.

[25] 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.

[26] T. N. Tran, "Research on the Impact of the Differencing Operator on Ensemble Learning Algorithms in the Case of Peak Load Forecasting," *Arabian Journal for Science and Engineering*, vol. 50, pp. 5633–5646, 2024, https://doi.org/10.1007/s13369-024-09460-1.

[27] W. Zhang, "Short-term Load Forecasting of Power Model Based on CS-Catboost Algorithm," *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, pp. 2295-2299, 2022, https://doi.org/10.1109/ITAIC54216.2022.9836483.

[28] 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, 2022, https://doi.org/10.1016/j.epsr.2022.108635.

[29] W. G. Buratto, R. N. Muniz, A. Nied and G. V. González, "Seq2Seq-LSTM With Attention for Electricity Load Forecasting in Brazil," *IEEE Access*, vol. 12, pp. 30020-30029, 2024, https://doi.org/10.1109/ACCESS.2024.3365812.

[30] A. Kathirgamanathan, A. Patel, A. S. Khwaja, B. Venkatesh, and A. Anpalagan, "Performance comparison of single and ensemble CNN, LSTM, and traditional ANN models for short-term electricity load forecasting," *The Journal of Engineering*, vol. 2022, no. 5, pp. 550-565, 2022, https://doi.org/10.1049/tje2.12132.

[31] S. M. Praminta, H. Aji and E. Y. Ikhsan, "Emerging and Evaluating Long Short Term Memory (LSTM) Network for Load Forecast in Java Bali System," *2023 4th International Conference on High Voltage Engineering and Power Systems (ICHVEPS)*, pp. 797-801, 2023, https://doi.org/10.1109/ICHVEPS58902.2023.10257331.

[32] S. H. Rafi, Nahid-Al-Masood, S. R. Deeba and E. Hossain, "A Short-Term Load Forecasting Method Using Integrated CNN and LSTM Network," *IEEE Access*, vol. 9, pp. 32436-32448, 2021, https://doi.org/10.1109/ACCESS.2021.3060654.

[33] H. Liang, G. Li, L. Xu, X. Liu and Q. Liu, "Short-Term Load Forecasting for A Power Supplying District Based on CEEMDAN-WPE-LSTM-Stacking Methods," *2024 7th International Conference on Energy, Electrical and Power Engineering (CEEPE)*, pp. 436-441, 2024, https://doi.org/10.1109/CEEPE62022.2024.10586323.

[34] H. Eskandari, M. Imani, and M. P. Moghaddam, "Best-tree wavelet packet transform bidirectional GRU for short-term load forecasting," *The Journal of Supercomputing*, vol. 79, no. 12, pp. 13545-13577, 2023, https://doi.org/10.1007/s11227-023-05193-4.

[35] Y. Li, Y. Ye, Y. Xu, L. Li, X. Chen, and J. Huang, "Two-stage forecasting of TCN-GRU short-term load considering error compensation and real-time decomposition," *Earth Science Informatics*, vol. 17, pp. 5347-5357, 2024, https://doi.org/10.1007/s12145-024-01456-7.

[36] R. Liu, J. Shi, G. Sun, S. Lin, and F. Li, "A Short-term net load hybrid forecasting method based on VW-KA and QR-CNN-GRU," *Electric Power Systems Research*, vol. 232, p. 110384, 2024, https://doi.org/10.1016/j.epsr.2024.110384.

[37] H. Hua, M. Liu, Y. Li, S. Deng, and Q. Wang, "An ensemble framework for short-term load forecasting based on parallel CNN and GRU with improved ResNet," *Electric Power Systems Research*, vol. 216, p. 109057, 2023, https://doi.org/10.1016/j.epsr.2022.109057.

[38] S. Luo, Z. Ni, X. Zhu, P. Xia, and H. Wu, "A Novel Methanol Futures Price Prediction Method Based on Multicycle CNN-GRU and Attention Mechanism," *Arabian Journal for Science and Engineering*, vol. 48, no. 2, pp. 1487-1501, 2023, https://doi.org/10.1007/s13369-022-06902-6.

[39] X. Li, H. Guo, L. Xu, and Z. Xing, "Bayesian-Based Hyperparameter Optimization of 1D-CNN for Structural Anomaly Detection," *Sensors*, vol. 23, no. 11, p. 5058, 2023, https://doi.org/10.3390/s23115058.

[40] S. M. H. Rizvi, "Time Series Deep learning for Robust Steady-State Load Parameter Estimation using 1D-CNN," *Arabian Journal for Science and Engineering*, vol. 47, pp. 2731-2744, 2021, https://doi.org/10.1007/s13369-021-05782-6.

[41] C. Wang, X. Li, Y. Shi, W. Jiang, Q. Song, and X. Li, "Load forecasting method based on CNN and extended LSTM," *Energy Reports*, vol. 12, pp. 2452-2461, 2024, https://doi.org/10.1016/j.egyr.2024.07.030.

[42] Y. Wang *et al*., "Short-Term Load Forecasting for Industrial Customers Based on TCN-LightGBM," in *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 1984-1997, 2021, https://doi.org/10.1109/TPWRS.2020.3028133.

[43] Z. Xu *et al*., "PhaCIA-TCNs: Short-Term Load Forecasting Using Temporal Convolutional Networks With Parallel Hybrid Activated Convolution and Input Attention," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 1, pp. 427-438, 2024, https://doi.org/10.1109/TNSE.2023.3300744.

[44] Y. Feng, J. Zhu, P. Qiu, X. Zhang, and C. Shuai, "Short-term Power Load Forecasting Based on TCN-BiLSTM-Attention and Multi-feature Fusion," *Arabian Journal for Science and Engineering*, vol. 50, pp. 5475-5486, 2024, https://doi.org/10.1007/s13369-024-09351-5.

[45] J. Wen, Y. Peng, W. Zhang, X. Huang and Z. Wang, "Short-term power load forecasting based on TCN-LSTM model," *2024 IEEE 6th International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pp. 734-738, 2024, https://doi.org/10.1109/ICCASIT62299.2024.10827868.

[46] Y. Shang *et al*., "Loss of Life Estimation of Distribution Transformers Considering Corrupted AMI Data Recovery and Field Verification," *IEEE Transactions on Power Delivery*, vol. 36, no. 1, pp. 180-190, 2021, https://doi.org/10.1109/TPWRD.2020.2978809.

[47] H. Ma, P. Yang, F. Wang, X. Wang, D. Yang, and B. Feng, "Short-Term Heavy Overload Forecasting of Public Transformers Based on Combined LSTM-XGBoost Model," *Energies*, vol. 16, no. 3, p. 1507, 2023, https://doi.org/10.3390/en16031507.

[48] H. Liao and K. K. Radhakrishnan, "Short-Term Load Forecasting with Temporal Fusion Transformers for Power Distribution Networks," *2022 IEEE Sustainable Power and Energy Conference (iSPEC)*, pp. 1-5, 2022, https://doi.org/10.1109/iSPEC54162.2022.10033079.

[49] I. Diahovchenko, A. Chuprun, and Z. Čonka, "Assessment and mitigation of the influence of rising charging demand of electric vehicles on the aging of distribution transformers," *Electric Power Systems Research*, vol. 221, p. 109455, 2023, https://doi.org/10.1016/j.epsr.2023.109455.

[50] J. Lee, "Estimating Near-Surface Air Temperature From Satellite-Derived Land Surface Temperature Using Temporal Deep Learning: A Comparative Analysis," *IEEE Access*, vol. 13, pp. 28935-28945, 2025, https://doi.org/10.1109/ACCESS.2025.3539581.

[51] M. Dai *et al*., "N-BEATS-GAN: A risk-aware financial time series forecasting with generative adversarial networks," *Applied Soft Computing*, vol. 186, p. 114235, 2026, https://doi.org/10.1016/j.asoc.2025.114235.

[52] 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.

[53] A. Binte 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.

[54] A. D. Manchalwar, N. R. Patne, B. V. S. Vardhan, and M. Khedkar, "Peer-to-peer energy trading in a distribution network considering the impact of short-term load forecasting," *Electrical Engineering*, vol. 105, no. 4, pp. 2069-2081, 2023, https://doi.org/10.1007/s00202-023-01793-8.

[55] J. Cardo-Miota, R. Trivedi, S. Patra, S. Khadem, and M. Bahloul, "Data-driven approach for day-ahead System Non-Synchronous Penetration forecasting: A comprehensive framework, model development and analysis," *Applied Energy*, vol. 362, p. 123006, 2024, https://doi.org/10.1016/j.apenergy.2024.123006.

[56] B. Bischl *et al*., "Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges," *WIREs Data Mining and Knowledge Discovery*, vol. 13, no. 2, p. e1484, 2023, https://doi.org/10.1002/widm.1484.

[57] T. N. Tran and T. A. Nguyen, "Hyperparameter Optimization for Deep Learning Modeling in Short-Term Load Forecasting," *International journal of electrical and computer engineering systems*, vol. 16, no. 6, pp. 443-450, 2025, https://doi.org/10.32985/ijeces.16.6.2.

[58] S. Hanifi, A. Cammarano, and H. Zare-Behtash, "Advanced hyperparameter optimization of deep learning models for wind power prediction," *Renewable Energy*, vol. 221, p. 119700, 2024, https://doi.org/10.1016/j.renene.2023.119700.

[59] Y. Da Jhong, C. S. Chen, B. C. Jhong, C. H. Tsai, and S. Y. Yang, "Optimization of LSTM Parameters for Flash Flood Forecasting Using Genetic Algorithm," *Water Resources Management*, vol. 38, no. 3, pp. 1141–1164, 2024, https://doi.org/10.1007/s11269-023-03713-8.

[60] M. Sakib, T. Siddiqui, S. Mustajab, R. M. Alotaibi, N. M. Alshareef, and M. Z. Khan, "An ensemble deep learning framework for energy demand forecasting using genetic algorithm-based feature selection," *PLoS One*, vol. 20, no. 1, p. e0310465, 2025, https://doi.org/10.1371/journal.pone.0310465.

[61] Y. Wang *et al*., "Considering the dual endogenous-exogenous uncertainty integrated energy multiple load short-term forecast," *Energy*, vol. 285, p. 129387, 2023, https://doi.org/10.1016/j.energy.2023.129387.

[62] Z. Mustaffa and M. H. Sulaiman, "Advanced forecasting of building energy loads with XGBoost and metaheuristic algorithms integration," *Energy Storage and Saving*, vol. 4, no. 4, pp. 421-438, 2025, https://doi.org/10.1016/j.enss.2025.03.005.