Original Research Article

The application of a hybrid model based on LSTM and Transformer for electricity consumption prediction in Romania

Abstract

With the growing energy demand and the increasing intelligence of power systems, accurate electricity load forecasting is crucial for grid scheduling and energy management. In recent years, deep learning methods have made significant progress in time series forecasting, with Transformer and LSTM structures gaining attention for their powerful sequence modeling capabilities. Therefore, this study proposes three different combinations of LSTM and Transformer: (1) Parallel Fusion Model ; (2) LSTM-Transformer; and (3) Transformer-LSTM. Additionally, the Particle Swarm Optimization (PSO) algorithm is employed to optimize the model hyperparameters. The experimental dataset consists of hourly electricity consumption data from Romania, and multi-step forecasting is performed based on the Nonlinear AutoRegressive with Exogenous inputs (NARX) model. The predictive performance of these three hybrid models is compared with traditional machine learning models. The experimental results show that different combinations exhibit varying performances in short-term, mid-term, and long-term forecasting tasks. Specifically, the LSTM-Transformer model performs better in mid-term forecasting, while the Transformer-LSTM model excels in long-term forecasting, providing new insights for electricity load prediction.

Keywords: Transformer; LSTM;Particle Swarm Optimization; Electricity load forecasting

1 Introduction

With the rapid development of the social economy, the power system plays a vital role in energy supply. Accurate electricity consumption forecasting is of great significance for the stable operation of power systems, optimal resource allocation, and energy conservation and emission reduction. However, electricity consumption is affected by various complex factors, such as weather conditions and social activities. Its time series exhibits characteristics of nonlinearity, strong correlation, and dynamic variation [1], which poses great challenges to accurate forecasting.

Electricity consumption forecasting methods can be mainly categorized into four types: traditional statistical methods, machine learning methods, deep learning methods, and hybrid models. Traditional statistical methods, such as time series analysis (ARIMA, SARIMA) and regression analysis, were among the earliest approaches applied to electricity consumption forecasting. The ARIMA model was proposed by Box and Jenkins [2] in 1970 and is suitable for handling linear and stationary or difference-stationary time series. However, it shows limited performance when dealing with long-term dependencies and nonlinear fluctuations. The SARIMA model further introduces seasonal differencing and is suitable for data with prominent seasonal patterns [3]. Ranjan and Jain [4] developed a multiple linear regression model for seasonal energy consumption and analyzed the electricity consumption patterns in Delhi during the period from 1984 to 1993. Traditional statistical models are simple and easy to implement, and they require relatively small amounts of data, making them suitable for linear and static datasets [5]. However, they struggle to capture complex features when dealing with nonlinear relationships and long-sequence data.

Machine learning models have been widely applied to electricity consumption forecasting due to their ability to capture nonlinear relationships and handle large, complex datasets. Drucker et al. [6] proposed Support Vector Regression (SVR), which is suitable for nonlinear regression problems but is limited by the choice of kernel functions. Breiman et al. [7] introduced Random Forest, and Chen et al. [8] proposed XGBoost, both of which leverage ensemble learning techniques to enhance model generalization and robustness. Compared to traditional statistical methods, machine learning models are better at handling nonlinear relationships; however, they still show limitations in long-term forecasting. Grandón et al. [9] proposed a novel hybrid method that combines ARIMA and LSTM to forecast hourly electricity consumption in Ukraine, achieving higher predictive accuracy than models based solely on machine learning techniques.

The LSTM network proposed by Hochreiter and Schmidhuber [10] addresses the vanishing gradient problem in RNNs through a gating mechanism, making it suitable for capturing long-term dependencies. Cho et al. [11] introduced the GRU, which simplifies the parameter structure and accelerates model training. Cai et al. [12] applied LSTM to electricity load forecasting tasks and achieved promising results. Convolutional Neural Networks (CNNs) have also been used to capture spatial and temporal features in electricity consumption data, and their hybrid models

with LSTM have further improved prediction performance [13]. The Transformer model, initially proposed by Vaswani et al. [14], employs a self-attention mechanism that effectively handles dependencies across long sequences. Researchers such as Li et al. [15], Zhou et al. [16], and Wu et al. [17] have applied Transformer models to various time series forecasting tasks, achieving remarkable results. Methods like Wen et al.'s [18] Seq2Seq and Smyl et al.'s [19] RNN combined with smoothing techniques have further expanded the representational capacity of deep models. Although deep learning models effectively handle long-term dependencies in sequential data, they require large amounts of training data, involve high model complexity, and have relatively long training times.

As the performance improvements from single models have plateaued, researchers have begun exploring ways to organize and combine models. Borovykh et al. [20] constructed a CNN-LSTM architecture to capture both local and global features; Lai et al. [21] combined CNN and RNN to capture features at different temporal scales; Lim et al. [22] proposed the Temporal Fusion Transformer (TFT) model, which integrates LSTM and Transformer architectures to enhance interpretability and predictive performance. Another study introduced an optimized LSTM-Transformer hybrid model for electricity load forecasting, which leverages LSTM's ability to handle sequential data along with the Transformer's self-attention mechanism, resulting in more accurate multi-step ahead electricity consumption predictions [23]. Khashei et al. [24] explored the combination of ARIMA and neural networks, validating the effectiveness of hybrid models across various tasks. Hybrid models are capable of integrating the strengths of different model types, offering stronger generalization and accuracy in multi-pattern time series forecasting.

Although both LSTM and Transformer models have been widely applied in time series modeling, there is still a lack of systematic comparative studies on their combinations in the field of electricity load forecasting. Currently, no comprehensive research has clearly revealed the performance differences and adaptability issues of different hybrid structures in electricity forecasting scenarios. Therefore, this paper builds hybrid models based on LSTM and Transformer using three different combination strategies and conducts a detailed comparison of their predictive performance against models such as Random Forest, XGBoost, and MLP across forecasting tasks at different time scales. The goal is to provide effective solutions for electricity consumption forecasting in power systems and to strongly support the intelligent development of the power industry.

The remainder of this paper is organized as follows: Section 2 introduces the main model structures.Section 3 describes the use of the Particle Swarm Optimization (PSO) algorithm for model hyperparameter optimization.Section 4 presents a case study on electricity consumption forecasting.Section 5 provides the conclusion.

2 Research methods

2.1 LSTM model

LSTM(Figure 1) is a deep learning model commonly used for processing sequential data. Building upon the foundation of RNN, it introduces memory cells to mitigate the vanishing gradient problem. Through memory cells and gating mechanisms, LSTM can better address long-term dependency issues, demonstrating outstanding performance in various time series prediction tasks. The gating mechanism of LSTM primarily consists of three parts: the forget gate, the input gate, and the output gate.

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f\right), \tag{2.1}$$

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i\right), \tag{2.2}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \tag{2.3}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o\right), \tag{2.4}$$

$$h_t = o_t \circ \tanh\left(c_t\right), \tag{2.5}$$

where f_t represents the activation value of the forget gate, i_t represents the activation value of the input gate, c_t denotes the cell state, o_t indicates the activation value of the output gate,h_t represents the hidden layer state.

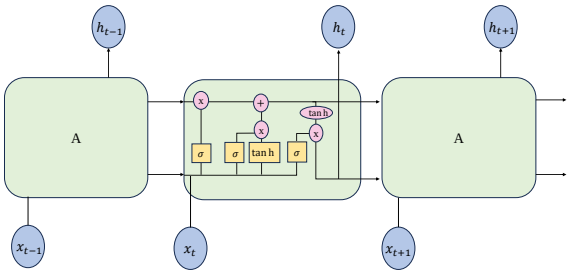


Figure 1. Structure of the LSTM

The forget gate maps values to a range between 0 and 1 through the σfunction, thereby determining whether to forget or retain information from the current time step and the previous hidden state. The input gate decides which new information is added to the cell state using the $tanh$ activation function. Finally, the output gate generates the new hidden state.
Compared to traditional RNNs, LSTM offers several notable advantages:

- Mitigation of the vanishing gradient problem: By maintaining a more stable flow of gradients during backpropagation through time, LSTM is capable of learning long-term dependencies

that standard RNNs struggle with.

- Selective memory: The gating mechanisms allow LSTM to selectively remember or forget information, making it well-suited for sequences where important information may occur after many time steps.
- Robust performance in practice: LSTM has been successfully applied to a wide range of tasks, including speech recognition, language modeling, financial forecasting, and other time series prediction problems.

Due to these strengths, LSTM is often the model of choice when dealing with complex sequential data, particularly where capturing long-term temporal patterns is critical.

2.2 Transformer Model

The Transformer model is a deep learning architecture based on the self-attention mechanism, proposed by Vaswani [14] et al. in 2017, initially designed for text translation tasks. It consists mainly of two components: the Encoder and the Decoder. The Encoder is composed of multiple encoder layers, each including Multi-Head Self-Attention, Feedforward Neural Network (FNN), Residual Connection, and Layer Normalization, which are used to extract input features. The Decoder, built upon the Encoder, additionally introduces Masked Multi-Head Self-Attention to prevent information leakage from future positions and employs Encoder-Decoder Attention to generate the target sequence.

The computation of the self-attention mechanism relies on the Query (Q), Key (K), and Value (V) vectors, which are obtained through linear transformations of the input and are used to calculate attention weights. The Q vector represents the query at the current input position and matches with the K vectors to measure the similarity between different positions. The K vector encodes the information at each position, while the V vector contains the actual content, which is used to generate the model's output through a weighted sum.The computation formula for the self-attention mechanism is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.6}$$

here, d_k represents the dimension of the K vector, and $\sqrt{d_k}$ is used to prevent the dot product results from becoming too large. Then, a Softmax function is applied to normalize the dot product results, producing attention weights for each position. These weights are then used to compute a weighted sum of the V vectors, resulting in a new representation for each input position.

To capture more features, Transformer commonly employs a multi-head attention mechanism, where each head independently computes its own Q, K, and V projections. The outputs from all heads are then concatenated and passed through a final linear transformation to produce the

final output. The corresponding expression is as follows:

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)W, \tag{2.7}$$

where W is the trainable parameter.The feedforward network performs a nonlinear mapping on the output of the attention mechanism:

$$\text{FFN}(\mathbf{x}) = \max(0, \mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2 + \mathbf{b}_2. \tag{2.8}$$

Compared with traditional recurrent neural networks (such as RNNs and LSTMs), Transformer models offer several significant advantages:

- Strong parallel computing capability: Transformer allows simultaneous computation across all positions, greatly improving training efficiency, especially suitable for large-scale data training.
- Better long-term dependency modeling: The self-attention mechanism can directly capture dependencies between any two positions, unlike RNNs that rely on sequential processing, thus avoiding long-term dependency issues.
- Dynamic attention mechanism: The model can dynamically allocate attention weights based on the input sequence, enabling it to focus on the most relevant information.

Thanks to these advantages, Transformer and its variants have become important architectures in deep learning.

2.3 NARX model

The NARX model, proposed by Billings et al. [25], is a structure suitable for nonlinear time series modeling.Its core idea is that the output at the current time step is determined by a series of past outputs (the autoregressive part) and external inputs (exogenous inputs). In time series forecasting tasks, the NARX model incorporates lag terms to capture the dynamic features and long-term dependencies of the data, thereby improving prediction accuracy. There are two common structures of the NARX model: the series-parallel structure and the parallel structure. The series-parallel structure uses actual historical values as input, while the parallel structure uses the model's own predicted values from the previous time step as part of the input for the next step. This creates a feedback loop between inputs and outputs, enabling multi-step forecasting. In this study, the NARX model adopts the parallel structure.Its mathematical formulation is as follows:

$$\hat{y}(t) = f\left(\hat{y}(t-1), \hat{y}(t-2), \ldots, \hat{y}(t-n),\ x(t-1), x(t-2), \ldots, x(t-m)\right), \tag{2.9}$$

where $\hat{y}(t)$ denotes the predicted output at time t,$\hat{y}(t-1), \hat{y}(t-2), \ldots, \hat{y}(t-n)$ represent the predicted outputs from the previous n time steps,$x(t-1), x(t-2), \ldots, x(t-m)$ represent the

exogenous inputs from the previous m time steps,$f(\cdot)$ is a nonlinear mapping function learned by the model,n and m represent the number of lags for the output and input.

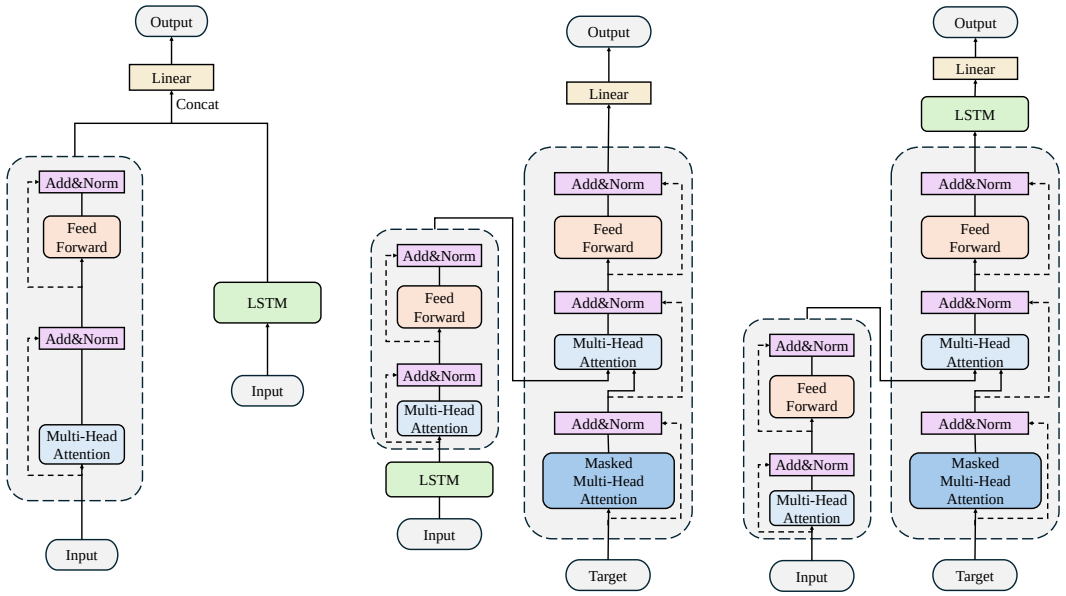
2.4 Combination Models of LSTM and Transformer

LSTM, due to its unique gating mechanism, can capture information embedded in long time series and is therefore widely used in time series forecasting tasks. Transformer, thanks to its self-attention mechanism, can simultaneously focus on different positions in the input sequence, allowing it to better capture long-term dependencies in time series. Moreover, unlike LSTM, it is not affected by the sequential processing of different time steps. In this paper, we combine the two models to explore the performance of the hybrid model across different tasks.

- Parallel Fusion Model
  The data is simultaneously fed into both the LSTM and Transformer models, allowing them to extract features in parallel. Through feature fusion, the features from the last time step of each model are concatenated and then passed through a fully connected layer to obtain the final prediction result.The model structure is shown in Figure 2(a).
- LSTM-Transformer
  First, the LSTM processes the input sequence by updating different hidden states to effectively capture long-term dependencies. The output of the LSTM is then used as the input to the Transformer encoder, which helps alleviate the issue of redundant input in the Transformer and further enhances the ability to extract data features.The model structure is shown in Figure 2(b).
- Transformer-LSTM
  On the basis of the Transformer model, the output of the Transformer decoder is fed into the LSTM model. Then, the gating mechanism of the LSTM is used to selectively retain relevant information. This helps the model better understand and remember long-term dependencies in the sequence, thereby achieving more accurate predictions.The model structure is shown in Figure 2(c).

3 Hyperparameter optimization using Particle Swarm Optimization

To enhance the accuracy of the predictive model, optimizing the model's hyperparameters is crucial. Among various error metrics, the Mean Absolute Percentage Error (MAPE) is chosen as the standard for evaluating the model's predictive performance in this study due to its excellent interpretability. MAPE can intuitively reflect the deviation between predicted and actual values in



(a) Parallel Fusion Model (b) LSTM-Transformer (c) Transformer-LSTM

Figure 2. Model Architecture Diagram

percentage form, which facilitates understanding and comparing the predictive effects of different models. In this study, we divide the dataset into a training set, a validation set, and a test set, and calculate the MAPE for each to assess the model's predictive accuracy at different stages. The specific calculation method is as follows:

$$MAPE_{\text{fit}} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%, \tag{3.1}$$

$$MAPE_{\text{valid}} = \frac{1}{s}\sum_{t=n}^{n+s}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%, \tag{3.2}$$

$$MAPE_{\text{pred}} = \frac{1}{p}\sum_{t=n+s}^{n+s+p}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%. \tag{3.3}$$

To minimize the MAPE, this study employs the Particle Swarm Optimization (PSO) algorithm to adjust the model parameters, thereby enhancing the accuracy of predictions. PSO is a heuristic global optimization method and a population-based intelligence algorithm. It was proposed in 1995 by Eberhart and Kennedy [26] while studying the movement behavior of birds and fish schools. PSO simulates birds in a flock using massless particles, each characterized by two attributes: velocity and position. Velocity represents the speed of movement, while position

indicates the direction of movement. The core idea is that during the search process, each particle keeps track of its own best-found solution, referred to as the personal best. The best among all personal bests is treated as the global best for the entire swarm. Then, each particle updates its velocity and position based on its personal best (pbest) and the global best(gbest). The update formulas for velocity and position are as follows:

$$v_i(t+1) = w \cdot v_i(t) + c_1 \cdot r_1 \cdot (pbest_i - x_i(t)) + c_2 \cdot r_2 \cdot (gbest_i - x_i(t)), \tag{3.4}$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{3.5}$$

here, $v_i(t)$ and $x_i(t)$ represent the velocity and position of the particle at time t, respectively. r_1 and r_2 are random numbers uniformly distributed in the range [0, 1]. c_1 and c_2 are learning factors. $pbest_i$ denotes the personal best position of the particle, while $gbest_i$ represents the global best position found by the entire swarm.PSO is widely used in optimization problems due to its simplicity, ease of implementation, and strong global search capabilities.

Algorithm 1: Hyperparameter Optimization using PSO for MAPE Minimization

Input: Objective function f (MAPE_valid), parameter dimension n, bounds $[lb, ub]$, parameter names P

Output: Best loss f_{best}, best parameters p_{best}, best model $model_{\text{best}}$

Set maximum iterations $T \leftarrow 100$;
Set number of particles $N \leftarrow 40$;
Initialize positions $X_i \sim U(lb, ub)$, $i = 1, \ldots, N$;
Initialize velocities $V_i \sim U(-1, 1)$;
`// Assuming velocity is scaled within [-1, 1]`
Set personal best $pbest_i \leftarrow X_i$ and evaluate $f_{pbest_i} \leftarrow f(pbest_i)$;
Set global best $gbest \leftarrow \arg\min_i f_{pbest_i}$;
for $t = 1$ **to** T **do**
    for $i = 1$ **to** N **do**
        Sample $r_1, r_2 \sim U(0, 1)$;
        Update velocity: $V_i \leftarrow \omega V_i + c_1 r_1 (pbest_i - X_i) + c_2 r_2 (gbest - X_i)$;
        Update position: $X_i \leftarrow X_i + V_i$;
        Clip position: $X_i \leftarrow \text{clip}(X_i, lb, ub)$;
        foreach *integer parameter* p_j *in* P **do**
            $X_{i,j} \leftarrow \text{round}(X_{i,j})$;
        end
        Evaluate fitness: $f_i \leftarrow f(X_i)$;
        if $f_i < f_{pbest_i}$ **then**
            $pbest_i \leftarrow X_i$;
            $f_{pbest_i} \leftarrow f_i$;
        end
    end
    Update global best: $gbest \leftarrow \arg\min_i f_{pbest_i}$;
    $f_{\text{best}} \leftarrow \min_i f_{pbest_i}$;
end
Retrieve best model $model_{\text{best}}$ corresponding to $gbest$;
return $f_{best}, gbest, model_{best}$;

Algorithm 1 outlines the process of optimizing model parameters using PSO, and this algorithm is implemented in Python.

4 Applications in Electricity Consumption Forecasting

Accurately forecasting electricity consumption in Romania is of great significance for ensuring the stability of the country's energy market, optimizing the allocation of power resources, and supporting sustainable economic development. With the continuous growth of Romania's economy and the advancement of its energy transition, changes in electricity demand directly affect power system planning and operation, energy policy formulation, and the integration of renewable energy sources. Moreover, as a member state of the European Union, Romania's electricity market plays a vital role not only domestically but also in regional energy cooperation. Therefore, in-depth analysis of Romania's electricity consumption trends can provide valuable decision-making support for policymakers, energy companies, and investors, contributing to Romania's goals of energy security and sustainable development.

4.1 Data Preprocessing

To achieve accurate forecasting of electricity consumption in Romania, this study collected and organized hourly electricity consumption data from February to March 2024. The dataset is sourced from an open dataset on Kaggle, which can be accessed at: .In the experiment, considering that different input features may vary in scale, we applied Min-Max normalization to scale the data into the range [0, 1] before feeding it into the models. The normalization formula is as follows:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{4.1}$$

where x is the original value, $x_{\min}$ and $x_{\max}$ are the minimum and maximum values in the dataset, and x_{norm} is the normalized value.

4.2 Evaluation Metrics

In this study, we selected the MAPE as the primary evaluation metric. MAPE effectively measures the relative error between predicted and actual values and offers strong interpretability, as its percentage-based form facilitates the comparison of predictive performance across different models. In tasks involving multi-model comparisons and multi-time-scale forecasting, MAPE clearly reveals the stability and generalization ability of models under various scenarios. Therefore, this study adopts MAPE as the core evaluation criterion to assess the predictive performance of each model on both the training and testing sets.Its calculation formula is as follows:

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{y_t - \hat{y}_t}{y_t} \right| \times 100\%, \tag{4.2}$$

where y_t denotes the true value, $\hat{y}_t$ denotes the predicted value, and n represents the number of samples.

4.3 Experimental Settings

We applied PSO to optimize the hyperparameters of ten models mentioned above: Parallel Fusion Model, LSTM-Transformer, Transformer-LSTM, Random Forest (RF), LightGBM (LGB), XGBoost (XGB), MLP, BiLSTM, and LSTM. During the optimization process, we recorded the changes in MAPE values on both the training and validation sets across iterations. The MAPE is used to measure the deviation between the predicted and actual values, helping to evaluate whether the model successfully learns the data features. Furthermore, we conducted three-step-ahead forecasting using the NARX model with lag operators set to 6, 8, 10, 12, 14, and 16, respectively. This analysis allows us to explore the model's forecasting performance across different time scales—short-term, mid-term, and long-term.

4.4 Forecasting Results and Analysis

In short-term time series forecasting tasks, Figure 3 shows the prediction results of various models on the test set when the lag order is set to 6, and Table 1 provides the corresponding MAPE. In Figure 9(a), the fitting performance of the models on the training set is compared, where LGB and Transformer-LSTM show good fitting performance. However, during the prediction phase, the LSTM-Transformer achieves the highest prediction accuracy in the first step with a MAPE of 2.2408%. In the second step of prediction, the Transformer and LSTM-Transformer reach errors of 4.2545% and 4.4285%, respectively, demonstrating excellent performance, indicating strong generalization capability of the models. Traditional models such as LGB and RF have relatively higher errors at this stage. After entering the third step of prediction, the Transformer still slightly outperforms other models with a MAPE of 6.1512%, showing its stability and robustness in multi-step forecasting tasks.

Table 1. Short-term forecasting MAPE of various models.

Lags	Models	Step 1	Step 2	Step 3
lag=6	RF	3.0761	5.5221	7.9054
	LGB	2.5018	4.5198	6.6792
	XGB	2.5320	4.4464	6.8570
	MLP	2.6394	4.6911	6.6004
	BiLSTM	4.4534	7.3827	9.7710
	LSTM	2.6306	4.8557	6.8998
	Parallel Fusion Model	2.7264	5.6800	8.6260
	Transformer	2.2740	**4.2545**	**6.1512**
	LSTM-Transformer	**2.2408**	4.4285	6.6102
	Transformer-LSTM	2.7554	5.1864	7.9264
lag=8	RF	2.9839	5.4482	7.9853
	LGB	2.5264	4.6314	6.9653
	XGB	2.4975	**4.3581**	6.6772
	MLP	2.6611	4.7047	**6.6225**
	BiLSTM	2.5411	5.3625	8.6902
	LSTM	**2.2569**	4.5628	6.7896
	Parallel Fusion Model	4.2159	7.6155	10.1653
	Transformer	3.4657	6.0521	7.8840
	LSTM-Transformer	2.7601	6.0412	9.3617
	Transformer-LSTM	2.2681	4.7120	6.9104

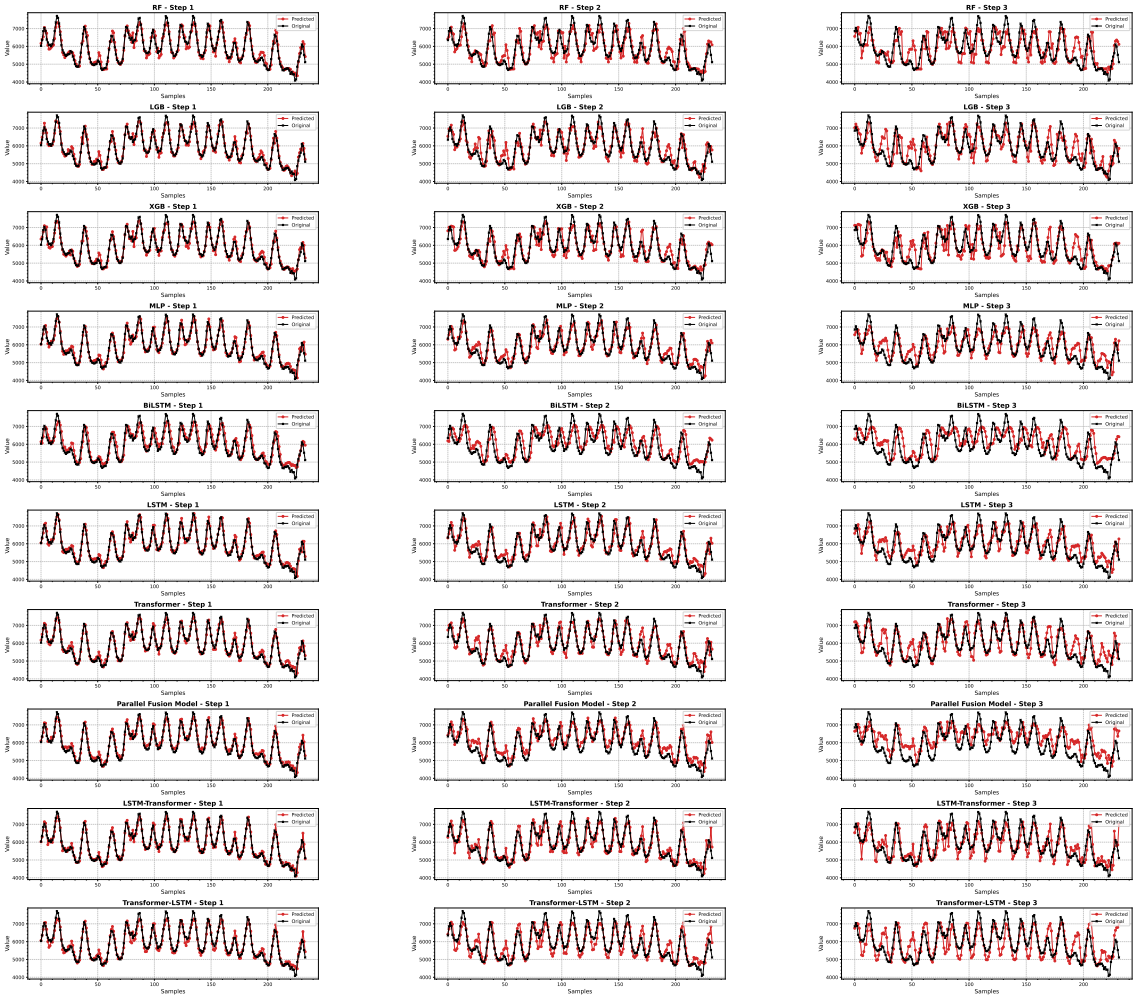


Figure 3. Model prediction comparison with lag=6

When the lag is set to 8, it can be observed from Figure 9(b) that the MAPE of various models has reached its minimum value through iterations. As shown in Table 1, the MAPE of XGB and MLP is around 2.50%, the error of LSTM is reduced to the lowest at 2.2569%, while the error of the Parallel Fusion model is relatively high at 4.2159%. A comparison of the predictive performance of the various models is shown in Figure 4.

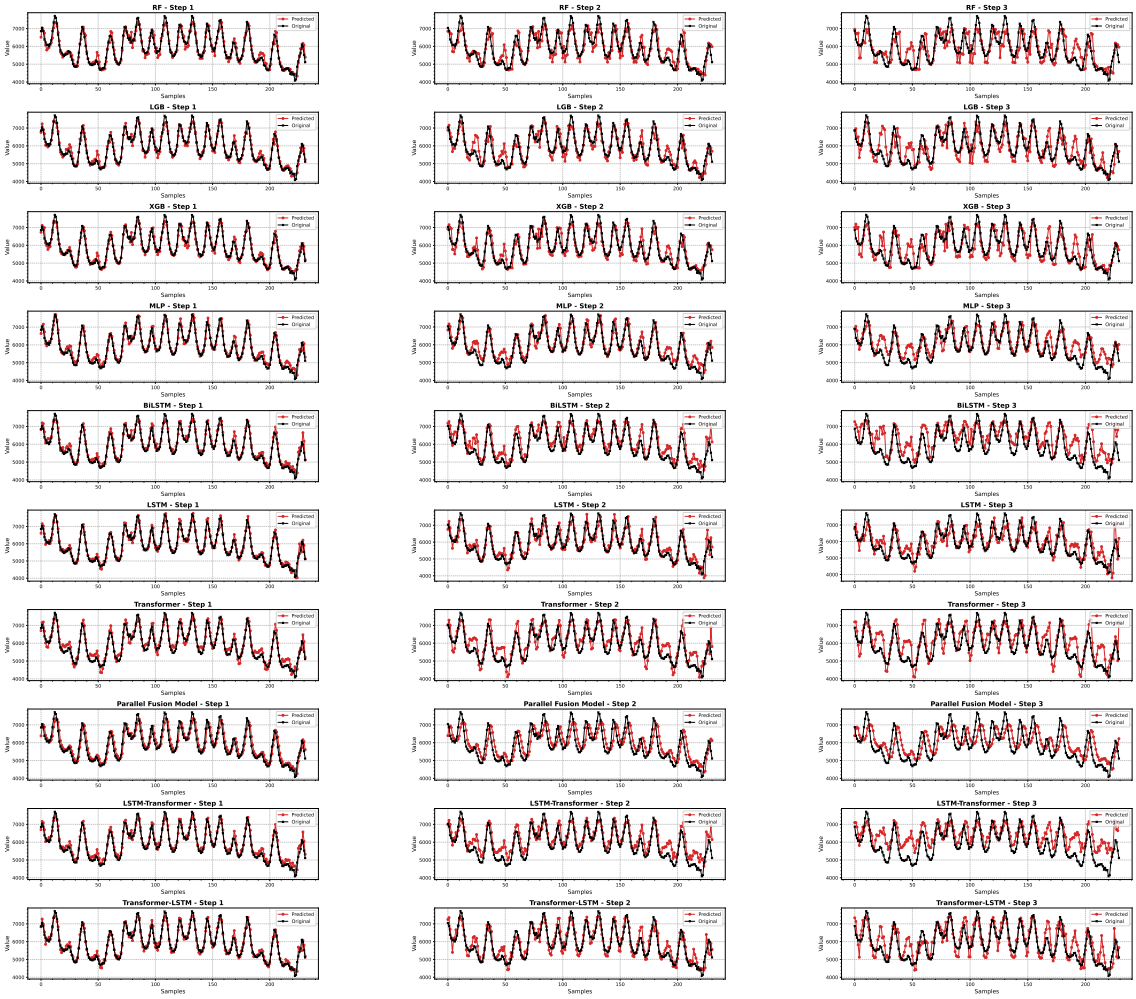


Figure 4. Model prediction comparison with lag=8

In medium-term time series forecasting, Figures 9(c) and 9(d) show that during the training phase, LGB's fitting performance remains superior to other models. When the lag length is increased to lag=10 (Figure 5, Table 2), the LSTM-Transformer achieves the best performance in three-step forecasting, with MAPE values of 2.1311%, 4.0963%, and 6.0373%, significantly outperforming other models. This indicates that under a moderate historical information window, it can effectively integrate LSTM's modeling capability for time dependencies with the Transformer's global attention mechanism, demonstrating exceptional medium-term forecasting ability. The results for lag=12 (Figure 6, Table 2) further support this conclusion, with the LSTM-Transformer maintaining a low overall error level and balanced and accurate performance in three-step forecasting.

Table 2. Medium-term forecasting MAPE of various models.

Lags	Models	Step 1	Step 2	Step 3
lag=10	RF	2.9750	5.2040	7.5556
	LGB	2.5037	4.5863	6.9653
	XGB	2.7817	4.8009	7.2045
	MLP	2.2422	4.3059	6.2042
	BiLSTM	2.6232	4.7086	6.8658
	LSTM	2.4950	5.0044	7.4900
	Parallel Fusion Model	2.3767	4.7176	6.6617
	Transformer	2.5461	4.7184	6.5758
	LSTM-Transformer	**2.1311**	**4.0963**	**6.0373**
	Transformer-LSTM	2.3094	4.4604	6.3156
lag=12	RF	2.9696	5.2324	7.8663
	LGB	2.7584	5.1467	7.7899
	XGB	2.8611	5.1964	7.8059
	MLP	2.4208	4.4991	**6.4970**
	BiLSTM	3.0204	5.1769	7.2548
	LSTM	4.1515	7.4555	10.0590
	Parallel Fusion Model	2.9526	5.6503	8.1212
	Transformer	4.0974	7.4232	10.1914
	LSTM-Transformer	**2.1869**	**4.4371**	6.8918
	Transformer-LSTM	3.9506	7.0818	9.7576

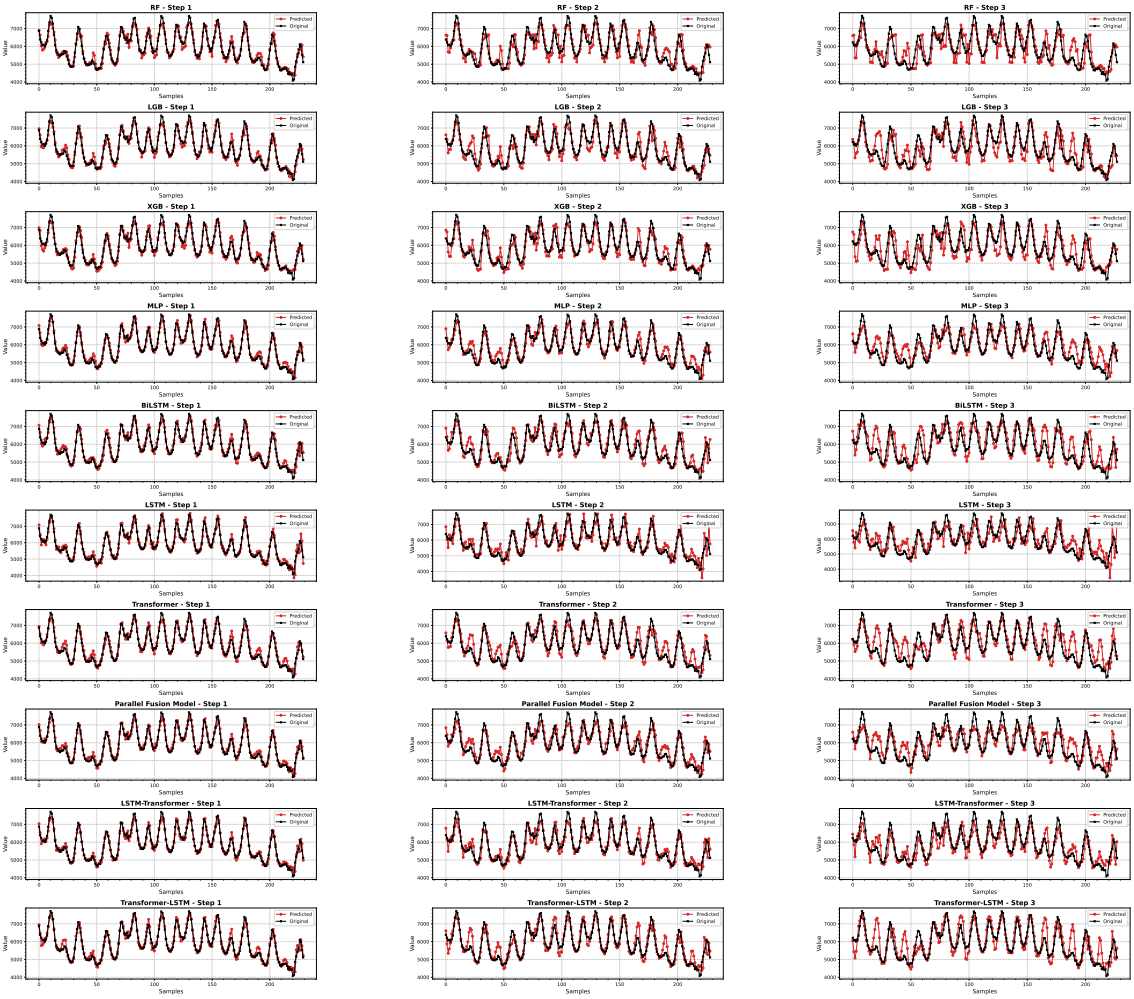


Figure 5. Model prediction comparison with lag=10

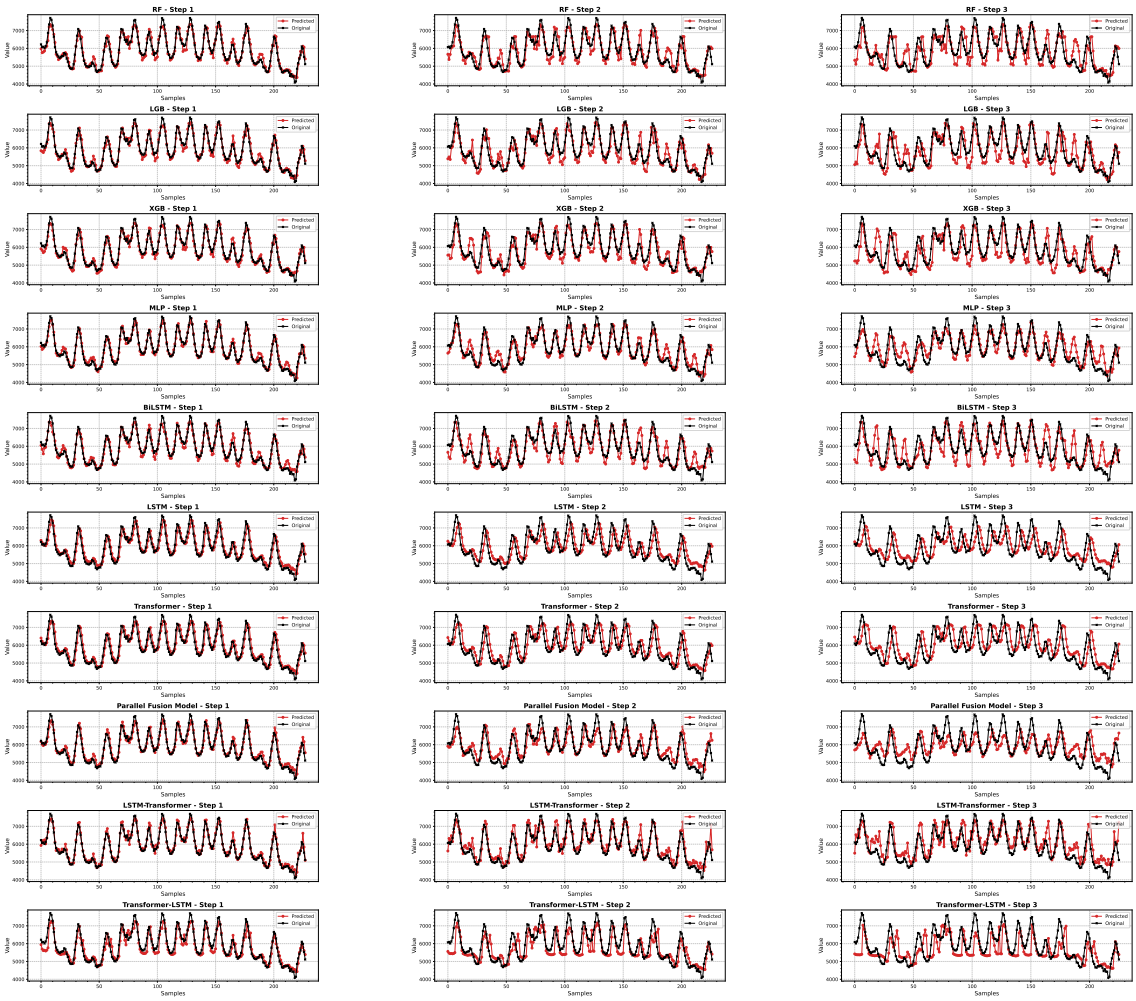


Figure 6. Model prediction comparison with lag=12

As the lag length increases to 14 and 16, it can be observed from Figures 9(e) and 9(f) that the Transformer-LSTM and LGB models show a significantly better fitting performance compared to other models. During the prediction phase, as indicated in Table 3, the Transformer-LSTM model maintains stable performance in the second and third steps of prediction, with the MAPE controlled at around 6.4%, which is significantly better than other models. This suggests that under conditions of longer historical input, the Transformer-LSTM can more effectively capture long-term dependencies in time series data.Figures 7 and 8 illustrate the predictive performance of each model.

Table 3. Long-term forecasting MAPE of various models.

Lags	Models	Step 1	Step 2	Step 3
lag=14	RF	2.9180	5.0977	7.5127
	LGB	2.7331	5.0789	7.8893
	XGB	2.9387	5.1762	7.6264
	MLP	2.8589	5.3554	7.5793
	BiLSTM	3.0173	5.8383	9.2642
	LSTM	3.7925	6.8869	9.4080
	Parallel Fusion Model	2.9323	5.8709	8.4079
	Transformer	**2.3494**	4.7558	7.3286
	LSTM-Transformer	11.7446	11.7858	11.8306
	Transformer-LSTM	2.3787	**4.2982**	**6.0029**
lag=16	RF	3.1401	5.5891	8.3751
	LGB	2.8326	5.2225	7.9747
	XGB	2.8630	4.9372	7.2709
	MLP	2.6736	4.8134	7.0434
	BiLSTM	2.3480	4.6128	6.8414
	LSTM	2.3240	4.6849	6.9712
	Parallel Fusion Model	3.0860	6.0740	8.9079
	Transformer	4.0148	7.3458	10.1984
	LSTM-Transformer	3.5385	6.9828	10.1333
	Transformer-LSTM	**2.1926**	**4.4083**	**6.8181**

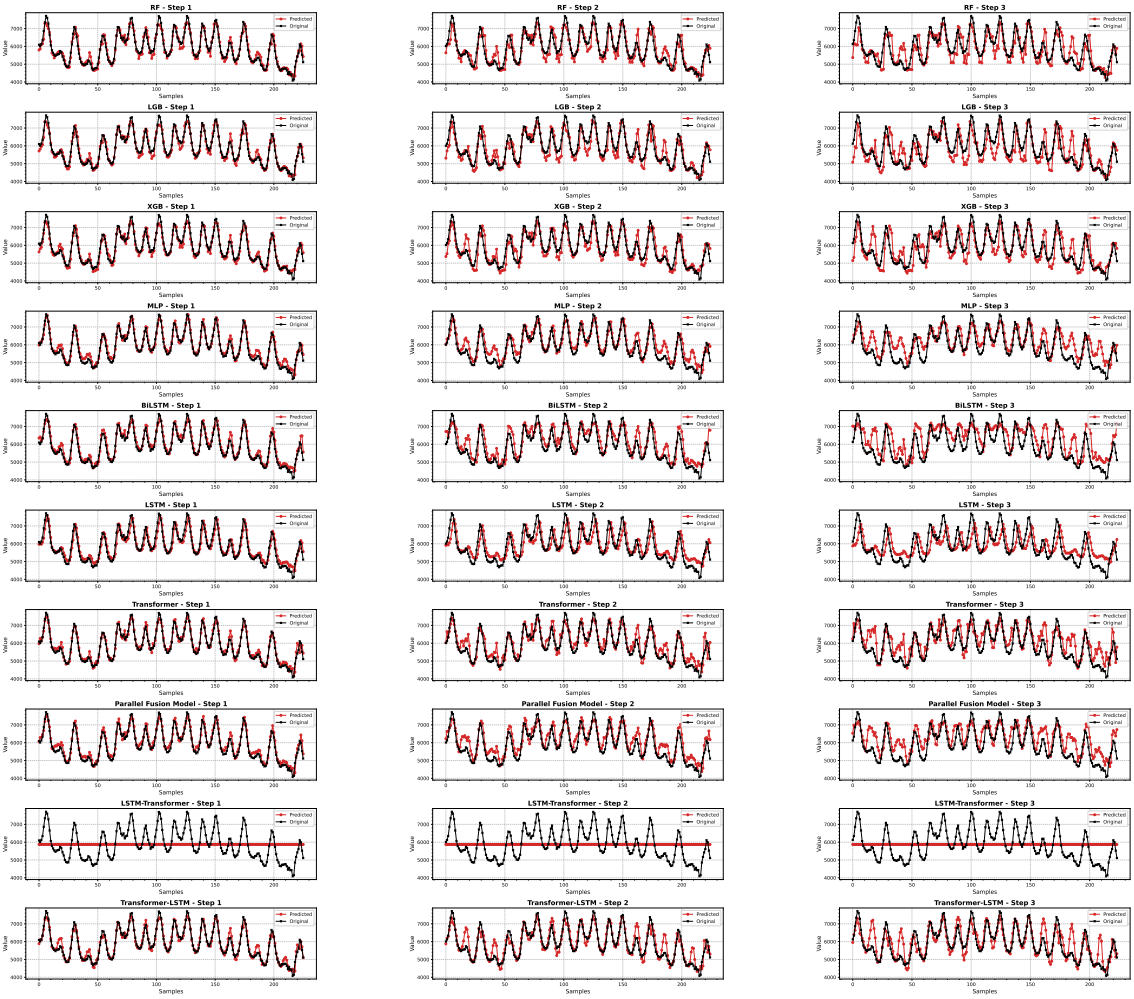


Figure 7. Model prediction comparison with lag=14

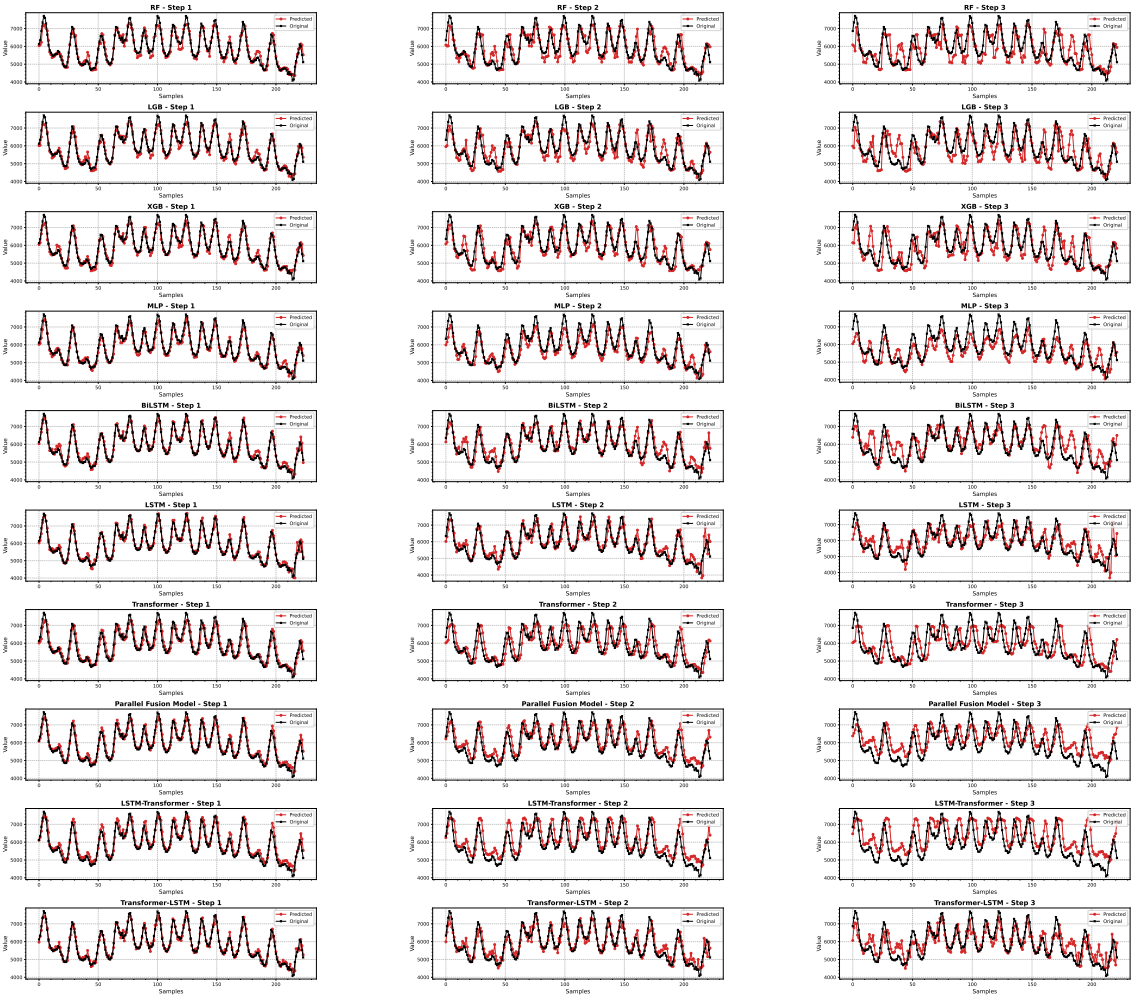
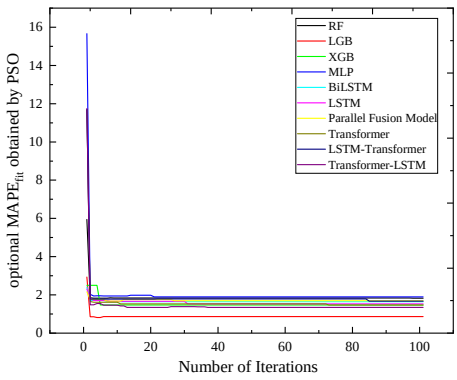
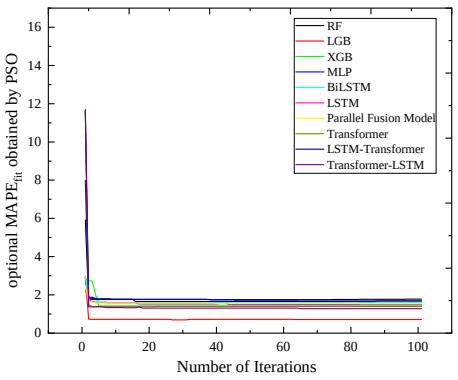


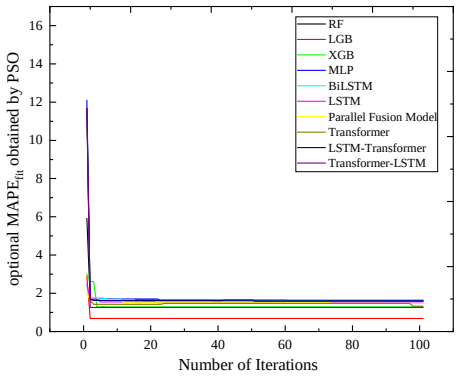
Figure 8. Model prediction comparison with lag=16



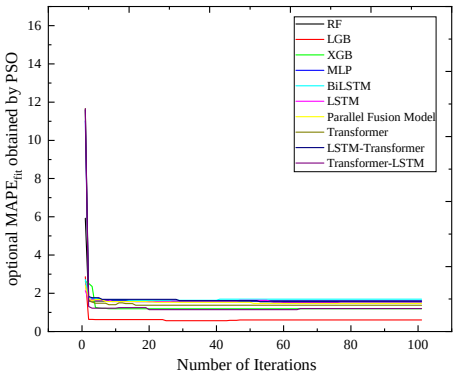
(a) $MAPE_{\text{fit}}$ convergence at lag 6.



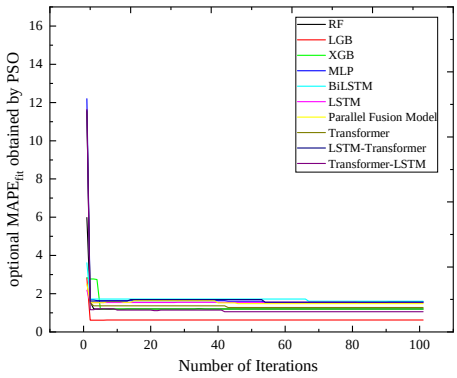
(b) $MAPE_{\text{fit}}$ convergence at lag 8.



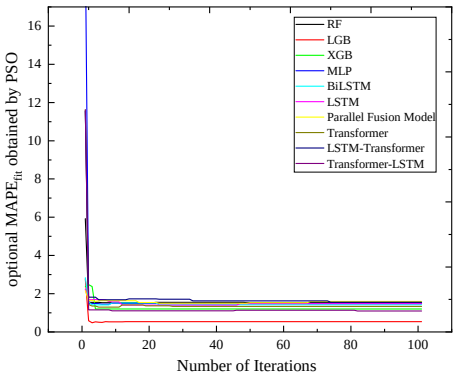
(c) $MAPE_{\text{fit}}$ convergence at lag 10.



(d) $MAPE_{\text{fit}}$ convergence at lag 12.



(e) $MAPE_{\text{fit}}$ convergence at lag 14.



(f) $MAPE_{\text{fit}}$ convergence at lag 16.

Figure 9. Convergence processes of $MAPE_{\text{fit}}$ for different lags.

4.5 Discussion

This study delves into forecasting tasks across different time scales, aiming to reveal the performance of various models in short-term, medium-term, and long-term predictions. Through the analysis of these tasks, we can gain a comprehensive understanding of the models' predictive capabilities and generalization performance, thereby providing valuable insights for the field of time series forecasting.

Short-Term Forecasting (Lag = 6 and 8):The LSTM-Transformer model demonstrated excellent performance in the initial steps of prediction, achieving the lowest MAPE of 2.2408% in the first step. This indicates that the combination of LSTM's modeling capability for time dependencies and Transformer's global attention mechanism provides a powerful framework for short-term forecasting. However, as the number of prediction steps increased, the Transformer model also showed significant stability, maintaining a MAPE of approximately 6.1512% in the third step. This highlights the model's robustness in handling multi-step forecasting tasks. Traditional models such as LightGBM and RF exhibited relatively higher errors, suggesting that they may not capture the complex patterns in the data as effectively as the LSTM-Transformer or Transformer models.

Medium-Term Forecasting (Lag = 10 and 12):The LSTM-Transformer performed best in all three steps of prediction, with the lowest MAPE values. This demonstrates the effectiveness of integrating time dependencies and global attention mechanisms, making it particularly suitable for medium-term forecasting. The results also indicate that the LSTM-Transformer can effectively utilize moderate historical information to improve forecasting accuracy.

Long-Term Forecasting (Lag = 14 and 16): The Transformer-LSTM and LGB models showed better fitting performance compared to other models. Specifically, the Transformer-LSTM maintained stable performance in the second and third steps of prediction, with MAPE controlled around 6.4%. This suggests that the Transformer-LSTM can effectively capture long-term dependencies in time series data, especially when provided with longer historical inputs. This capability is essential for long-term forecasting tasks where understanding long-term trends and dependencies is crucial.

In conclusion, the LSTM-Transformer model stands out in medium-term forecasting tasks due to its ability to integrate the advantages of both LSTM and Transformer architectures. The Transformer-LSTM excels in long-term forecasting tasks. These findings indicate that model selection should be based on the specific requirements of the forecasting task, including the forecasting horizon and the availability of historical data. Future work can further optimize these models or develop new architectures to enhance performance across all forecasting ranges, providing more effective solutions for electricity consumption forecasting in power systems, helping governments and energy companies optimize electricity production and distribution, and avoiding waste of electricity.

5 Conclusions

This study focuses on multi-step forecasting of electric power load, conducting an in-depth investigation of three hybrid models combining LSTM and Transformer architectures: Parallel Fusion Model, LSTM-Transformer, and Transformer-LSTM. The PSO algorithm is employed for hyperparameter tuning, and the performance of each model is systematically evaluated across short-term, medium-term, and long-term forecasting scenarios. Experiments are based on real hourly electricity consumption data from Romania, using a NARX model framework to analyze the multi-step prediction results under different lag settings.

The results demonstrate that each model architecture has its own strengths at different time scales. LSTM-Transformer performs most stably in medium-term forecasting, effectively integrating both local and global features of the time series. Transformer-LSTM achieves higher accuracy in long-term forecasting, showing strong robustness and the ability to capture long-term dependencies. Traditional machine learning models perform relatively well in short-term forecasting but are generally outperformed by the deep learning fusion models. Overall, hybrid architectures outperform single-structure models, and the order of LSTM and Transformer components has a significant impact on predictive performance.

This study not only verifies the effectiveness of hybrid architectures in electric load forecasting but also provides practical guidance for model selection in different forecasting tasks. It offers valuable insights for real-world applications and future research. Subsequent work may incorporate external influencing factors (such as temperature and holidays) to further improve model generalization and explore more efficient fusion strategies and optimization methods.

References

[1] Josivan Rodrigues Dos Reis, Jonathan Muñoz Tabora, Matheus Carvalho de Lima, Flávia Pessoa Monteiro, Suzane Cruz de Aquino Monteiro, Ubiratan Holanda Bezerra, and Maria Emília de Lima Tostes. Medium and long term energy forecasting methods: A literature review. *IEEE Access*, 13:29305–29326, 2025.

[2] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[3] Robert H Shumway, David S Stoffer, and David S Stoffer. *Time series analysis and its applications*, volume 3. Springer, 2000.

[4] Manish Ranjan and VK Jain. Modelling of electrical energy consumption in delhi. *Energy*, 24(4):351–361, 1999.

[5] Sameer Thakare, Neeraj Dhanraj Bokde, Andrés Elías Feijóo Lorenzo, et al. Forecasting different dimensions of liquidity in the intraday electricity markets: A review. *AIMS Energy*, 2023.

[6] Harris Drucker, Christopher J Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in neural information processing systems*, 9, 1996.

[7] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[8] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[9] T González Grandón, Johannes Schwenzer, Thomas Steens, and Julia Breuing. Electricity demand forecasting with hybrid classical statistical and machine learning algorithms: Case study of ukraine. *Applied Energy*, 355:122249, 2024.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[12] Changchun Cai, Yuan Tao, Tianqi Zhu, and Zhixiang Deng. Short-term load forecasting based on deep learning bidirectional lstm neural network. *Applied Sciences*, 11(17):8129, 2021.

[13] Adel Binbusayyis and Mohemmed Sha. Energy consumption prediction using modified deep cnn-bi lstm with attention mechanism. *Heliyon*, 11(1):e41507, 2025.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[15] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32, 2019.

[16] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[17] Sifan Wu, Xi Xiao, Qianggang Ding, Peilin Zhao, Ying Wei, and Junzhou Huang. Adversarial sparse transformer for time series forecasting. *Advances in neural information processing systems*, 33:17105–17115, 2020.

[18] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A multi-horizon quantile recurrent forecaster. *arXiv preprint arXiv:1711.11053*, 2017.

[19] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International journal of forecasting*, 36(1):75–85, 2020.

[20] Anastasia Borovykh, Sander Bohte, and Cornelis W Oosterlee. Conditional time series forecasting with convolutional neural networks. *arXiv preprint arXiv:1703.04691*, 2017.

[21] Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 95–104, 2018.

[22] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021.

[23] Vasileios Pentsos, Spyros Tragoudas, Jason Wibbenmeyer, and Nasser Khdeer. A hybrid lstm-transformer model for power load forecasting. *IEEE Transactions on Smart Grid*, pages 1–1, 2025.

[24] Mehdi Khashei and Mehdi Bijari. A novel hybridization of artificial neural networks and arima models for time series forecasting. *Applied soft computing*, 11(2):2664–2675, 2011.

[25] SA Billings and KM Tsang. Spectral analysis for non-linear systems, part i: Parametric non-linear spectral analysis. *Mechanical Systems and Signal Processing*, 3(4):319–339, 1989.

[26] Qinghai Bai. Analysis of particle swarm optimization algorithm. *Computer and information science*, 3(1):180, 2010.