ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 1 - 2026

The optimization accuracy of power system bus load time series prediction model based on deep learning is improved

Lei Shi[1,*], *Na Ji*[1], *Hang Zhao*[1], *Jin Ma*[2] *and Ziqi Wei*[2]

[1] Power Dispatching Control Center, State Grid Changzhi Power Supply Company, Changzhi, 046000, Shanxi, China
[2] Technology and Digitalization Department, State Grid Changzhi Power Supply Company, Changzhi, 046011, Shanxi, China

SUMMARY: *This paper proposes a deep learning framework for bus load time series forecasting. The data comes from the hourly operation records of 48 bus nodes in the regional power system, which form a total of 132,480 valid observations. The input is composed of active load, reactive load, time index, node number and rolling statistical characteristics. The model combines local fluctuation convolution extraction, long-term dependence modeling, attention enhancement in key periods and residual correction output to compress the peak section deviation and stabilize the single-step prediction. The framework is trained under unified hyperparameter search and compared with SVR, random forest, GRU, and standard LSTM. Experimental results show that the proposed model has R^2 of 0.9969, MAE of 18.43 MW, and MAPE of 0.86%, which is superior to the comparison models in terms of overall error compression and peak tracking in critical periods. Visual analysis further confirms that the proposed method maintains good response consistency in the pre-peak lifting, peak maintenance and post-peak falling stages, indicating its applicability in scheduling support and bus load management.*

KEYWORDS: *Bus load forecasting; Deep learning; Temporal modeling; Accuracy optimization*

1 Introduction

With the continuous access of scheduling automation, edge acquisition terminals and online monitoring platforms, the bus load data of power system has shown the characteristics of high-frequency, continuous and multi-source parallel calculation. As the key node of power flow convergence and power exchange, the bus load sequence not only reflects the local supply and demand changes, but also carries the coupling information brought by the fluctuation of tie lines, regional switching and operation mode switching. Focusing on the time series forecasting of bus load, the goal is no longer staying at the empirical estimation level, but turning to the data-driven model to complete the state representation, time- dependent learning and prediction error convergence control, which makes this task naturally have strong computer research attributes.

Alrasheedi et al. studied the short-term load forecasting of Saudi Arabia's smart grid and constructed a hybrid deep learning framework to improve the time series fitting ability [1]. Ullah et al. proposed a combination model of deep convolutional LSTM and stacked GRU for short-term power load forecasting [2]. Abumohsen et al. compared the performance differences

*15303452798@163.com
https://doi.org/10.65102/is2026086
Published: 30 April, 2026

between LSTM, GRU and RNN in power load forecasting [3]. Gonzalez et al. studied the deep network structure of ultra-short-term residential load demand forecasting [4]. Chen et al. proposed GCN-LSTM model considering multi-factor input, and used graph structure to deal with new power system load correlation [5]. Mounir et al. studied the short-term load forecasting effect of EMD-BI-LSTM in smart grid energy management [6]. Li et al. proposed a comprehensive energy load forecasting method based on feature separation and fusion and improved CNN [7]. Ran et al. studied a short-term load forecasting model combining CEEMDAN and Transformer [8]. Sun et al. proposed a graph neural network model combined with temporal convolution to describe the spatio-temporal relationship in the load sequence [9]. Waheed and Xu studied the deep neural network modeling path in data-driven short-term load forecasting [10].

Smyl et al. proposed ES-dRNN structure with dynamic attention to enhance the context-driven prediction ability [11]. Zhenglei et al. studied the adaptation mode of improved LSTM in urban power load forecasting [12]. Wen et al. proposed a deep learning hybrid model for smart grid information management [13]. Chen et al. studied the influence of improved decomposition strategy and hybrid deep network on short-term load forecasting accuracy [14]. Su et al. proposed spatio-temporal convolutional graph neural network for load forecasting of regional integrated energy system [15]. Wang et al. studied the performance of the SSA-CNN-LSTM method in short-term power load forecasting [16]. Jiang et al. proposed spatio-temporal fusion graph convolutional network to enhance the modeling ability of load sequence association [17]. Nabavi et al. studied the improvement effect of the combination of DWT and LSTM on prediction accuracy [18]. Zhou et al. proposed a joint method of empirical mode decomposition and SAM-LSTM for short-term power load forecasting [19]. Asiri et al. studied a hybrid deep learning framework for short-term load forecasting in smart grid scenarios [20].

In order to facilitate sorting out the calculation path differences in existing studies in load forecasting modeling, this paper arranges the model types, data objects and reference contents of representative literature as shown in Table 1.

Table 1: Comparison of computational characteristics of representative studies

Reference	Method Characteristics	Data Object	Reusable Insight
[1]	Hybrid Deep Learning	Smart Grid Load	Suitable for building a basic forecasting framework
[5]	GCN-LSTM	Multi-Factor Power Load	Suitable for introducing node-level relational representation
[8]	CEEMDAN + Transformer	Short-Term Load Sequences	Suitable for handling fluctuation decomposition and long-range dependency
[11]	ES-dRNN + Dynamic Attention	Multi-Benchmark Load Data	Suitable for enhancing response to critical time periods
[15]	Spatiotemporal Convolutional Graph Neural Network	Regional Integrated Energy Load	Suitable for modeling coupled spatiotemporal relationships
[18]	DWT + LSTM	Power Load Sequences	Suitable for reducing the impact of high-frequency disturbances

Existing research shows that the deep model has strong nonlinear representation ability in

load forecasting, but for bus-level time series forecasting, it is still necessary to incorporate node operation data, key period fluctuation characteristics and error constraint mechanism into the unified learning link. Based on this idea, this paper constructs a deep learning framework for bus load time series prediction, and verifies its prediction accuracy and peak tracking ability through model comparison, ablation experiments and key period analysis. The framework completes the reorganization of time series samples at the input end, highlights the extraction of local fluctuations at the feature end, and strengthens the error convergence control at the output end, so that the process of model training, validation and deployment maintains a consistent data processing logic, and at the same time retains the computing interface for subsequent online update and function expansion.

2 Methods

The core idea of this study is not to simply expand the network scale, but to restructure the input around the structural characteristics of the bus load series, compress the invalid fluctuations and stabilize the forecast output. In order to ensure the reproducibility of model training and testing, the method part is carried out from five aspects: data organization, sample representation, computing environment, comparison model and evaluation index. The data used for modeling comes from the multi-bus operation records of the regional power system. After unified time alignment, a continuous hourly level sample is formed, which contains a total of 132,480 valid observations covering 48 bus nodes, 12 month periods, and 24 hour operation sections. The original fields include timestamp, active load, reactive load, node number, day type label, and scheduling period label. The timestamp is further combined to derive the hour index, weekday code, intra-week location, and monthly number, which are used to characterize intraday fluctuations, weekly fluctuations, and seasonal differences. All records are sorted in chronological order and enter the calculation process after removing acquisition gaps and duplicates.

To avoid the interference of different dimensions on model convergence, in this paper, MinMax normalization is used for continuous variables and pre-embedding coding is used for discrete-time variables. The sample construction adopts sliding window mechanism, which takes 168 consecutive hour load segments as input and uses the bus load at the next moment as the supervision target. The design can simultaneously retain day-ahead cycle information, weekly repeat information and short-time mutation information, so that the input not only has time continuity, but also maintains node-level computable expression. The training set, validation set and test set are divided into 70%, 10% and 20% according to the time sequence, which can avoid the future information flowing back to the historical samples and maintain the time progressive relationship in the real scheduling scenario. Table 2 gives summary information about data organization versus input representation.

Table 2: Dataset versus input representation synopsis

Item	Content
Total Sample Size	132,480 hourly records
Number of Bus Nodes	48
Input Window	168 hours
Forecast Horizon	1 hour
Raw Fields	Timestamp, active power load, reactive power load, node ID, day type, dispatch period
Derived Features	Hour index, intra-week position, month index, rolling mean, rolling variance
Data Split	70% for training, 10% for validation, and 20% for testing

NumPy and Pandas are responsible for array calculation and timing arrangement, PyTorch is used for deep learning model construction, Scikit-learn is responsible for normalization, error statistics and baseline model training, and Matplotlib is used for visualization. The computing platform is configured as a 64-bit Windows environment with Intel i7-level eight-thread CPU as the processor, 32GB of memory and RTX 4060 as the graphics card. The baseline models include SVR, random forest, GRU, and standard LSTM, which are used to compare the performance of the proposed model in error compression and peak tracking. In the evaluation stage, four indicators are used: MAE, RMSE, MAPE and R-², where MAE reflects the absolute deviation level, RMSE emphasizes large error penalty, MAPE is used to describe relative error and R-² is used to measure fitting consistency. Table 3 shows the experimental configuration and evaluation content.

Table 3: Experimental configuration and evaluation metrics

Category	Details
Development Framework	PyTorch, Scikit-learn, NumPy, Pandas
Hardware Environment	Intel i7 eight-thread processor, 32 GB memory, RTX 4060
Baseline Models	SVR, Random Forest, GRU, standard LSTM
Optimization Objectives	Reduce MAE, RMSE, and MAPE, and improve R^2
Validation Strategy	Training, validation, and test sets split in chronological order
Output Contents	Predicted values, error curves, peak tracking results, and comparative statistics

Under the above method design, the bus load sequence is no longer regarded as a single scalar stream, but is organized as a composite input with node attributes, time attributes, and local statistical attributes. This processing method provides a unified data interface for subsequent local fluctuation extraction, long-term dependence modeling and attention enhancement in critical periods, and also enables the prediction model to complete high-precision learning without additional introduction of complex prior rules. The baseline model is not set out to be a formal horizontal comparison, but a representative scheme corresponding to different computational paths. SVR is used to reflect the performance boundary of kernel methods in small and medium-sized time series regression, random forest is used to observe the absorption ability of tree model to discrete fluctuations and nonlinear relationships, GRU is used to measure the time series learning effect of lightweight recurrent structure, and standard LSTM is used as the main depth sequence benchmark to test the accuracy gain source of the proposed model under the same training conditions. All models adopt unified input window,

unified data partition and unified evaluation index to ensure the interpretability and reverification of comparison results. At the same time, the prediction output retains the per-bus error record and the time-level response results, which is convenient for the subsequent quantitative verification in the key time analysis, ablation experiment and discussion section. This setup makes the comparison of results between different models clearer, and also facilitates the subsequent verification and verification.

3 Deep learning model for bus load timing prediction

3.1 Time series sample construction and input representation

The input of bus load time series forecasting is not simply intercepting a historical power series, but organizing the bus operation state, time position and local statistics into a unified sample. The bus load has continuity between adjacent moments, repeatability between periods within the same day, and is also jointly affected by the weekday type, scheduling section, and node attributes in cross-day operation, so the input representation must retain both numerical variations and context labels. Based on this understanding, the structural expression of the original observation at a single time is first carried out in this paper. The electrical quantity, time label and node identification are incorporated into the same input vector, and the construction form is shown in Equation (1).

$$z_t = [p_t, q_t, \bar{p}_t^{(6)}, \sigma_t^{(6)}, h_t, d_t, m_t, b_t]^\top \tag{1}$$

Here, p_t represents the active load at time t, q_t represents the reactive load at time t, $\bar{p}_t^{(6)}$ represents the sliding mean of the active load at the last six sampling times, $\sigma_t^{(6)}$ represents the sliding standard deviation within the same window, h_t represents the hourly index, d_t represents the position within the week, m_t represents the month number, and b_t represents the bus number. The function of Equation (1) is to write the instantaneous load value, local fluctuation intensity and time context into the sample at the same time, so that the subsequent network can identify the amplitude change, fluctuation degree and cycle position of the bus load in the input stage.

There are obvious differences in the numerical scale of different features, and the value range of active load and local statistics is much larger than that of the time index variable. If the concatenation is directly entered into the model, the gradient update will be biased towards the large value channel, resulting in an imbalance of input feature contributions. In order to weaken the interference of dimension differences on parameter learning, this paper performs normalization processing on continuous features, and its calculation form is shown in Equation (2):

$$\tilde{x}_{t,j} = \frac{x_{t,j} - x_j^{\min}}{x_j^{\max} - x_j^{\min} + \varepsilon} \tag{2}$$

where $x_{t,j}$ represents the original value of the j continuous feature at time t, $x_j^{\min}$ and $x_j^{\max}$ represent the minimum and maximum value of the feature in the training set, respectively, and ε represents the stable term, which is used to avoid the denominator being zero. After the scale compression of continuous variables is completed in Equation (2), the active load, reactive load and sliding statistics can enter a consistent numerical interval, thus weakening the influence of dimensional differences on the update direction of network parameters.

Discrete variables cannot be treated directly as continuous values because bus numbers, intra-week locations, and month labels themselves carry categorical attributes. If the original integer form is retained, the model is easy to mistake the category number as a linear size relationship. To this end, this paper encodes the continuous features and discrete indices separately, and then fuses them at the representation layer. The joint embedding process is shown in Equation (3):

$$e_t = \phi\left(W_c\tilde{x}_t + W_b u_{b_t} + W_d u_{d_t} + W_m u_{m_t} + b\right) \tag{3}$$

Here, $\tilde{x}_t$ represents the normalized continuous feature vector, u_{b_t} represents the embedding vector corresponding to the bus number, u_{d_t} represents the position embedding within a week, u_{m_t} represents the month embedding, W_c, W_b, W_d and W_m represent the mapping matrix of different feature branches, b represents the bias term, and $\phi(\cdot)$ represents the nonlinear activation function. The function of Equation (3) is to project the continuous electrical characteristics and discrete spatio-temporal labels into a unified representation space, so that the input can not only express the current load amplitude, but also distinguish the bus to which the sample belongs and its time location.

After the single-moment embedding is formed, the model still cannot directly enter the prediction stage, because the short-term evolution of bus load depends on continuous history rather than isolated time points. In this paper, the sliding window mechanism is used to generate time series sample blocks, and the continuous embeddings in the window are organized as two-dimensional input tensors, which are constructed as shown in Equation (4).

$$X_t = [e_{t-L+1}, e_{t-L+2}, \dots, e_t] \in \mathbb{R}^{L\times r} \tag{4}$$

Here, L represents the input window length, r represents the embedding dimension, and X_t represents the time-series sample matrix corresponding to time t. In Equation (4), the joint embeddings of adjacent moments are stacked as input tensors in time order, so that the model can simultaneously read local fluctuation, intra-day repetition and cross-day continuous information, and provide a stable data interface for subsequent local fluctuation feature extraction and long-term dependence modeling.

Through the above processing, the bus load raw records are transformed into a unified input representation with node attributes, time attributes and local statistical attributes. This process not only completes the mapping from raw observations to trainable samples, but also establishes a clear data entry for subsequent convolutional extraction, sequence modeling, and attention enhancement. In this way, the input is no longer just a simple load sequence, but a structured time series sample that can reflect the operating state and time context of the bus, which provides a stable basis for subsequent prediction accuracy improvement.

3.2 Local fluctuation feature extraction

The local fluctuation in the bus load sequence is not isolated noise, but the result of short-time regulation, period switching and power variation between nodes. If the model only learns along the main trend, the fine-grained changes at the pre-peak rise, post-peak fall, and switching boundaries are easily averaged. Therefore, the goal of local fluctuation feature extraction is not to simply amplify the difference, but to establish a local representation that can reflect the change intensity, change span and change location in a short time window. To this end, this paper first constructs a multi-scale fluctuation enhancement term on the input embedding sequence, which is defined as Equation (5):

$$d_t = [e_t - e_{t-1}\,;\; e_t - e_{t-\tau}\,;\; \mathrm{Var}(E_{t-w:t})] \tag{5}$$

Here, e_t represents the input embedding vector at time t, τ represents the stride difference interval, w represents the local statistical window length, $E_{t-w:t}$ represents the embedding subsequence from $t-w$ to t, $\mathrm{Var}(\cdot)$ represents the variance operator, and d_t represents the multi-scale fluctuation enhancement vector. The function of Equation (5) is to preserve the adjacent variation, stride variation and local dispersion degree at the same time, so that the local fluctuation no longer only depends on a single difference term.

After the formation of the fluctuation enhancement term, this paper uses causal convolution with expansion rate to extract local patterns in a short window, and the calculation process is shown in Equation (6).

$$h_t^{(k)} = \sigma\left(\sum_{i=0}^{s_k-1} K_i^{(k)}\, d_{t-i\delta_k} + b^{(k)}\right) \tag{6}$$

Here, $h_t^{(k)}$ represents the local response of the k convolution branch at time t, s_k represents the convolution kernel length of the branch, δ_k represents the dilation rate, $k_i^{(k)}$ represents the convolution kernel matrix of the k branch at position i, $b^{(k)}$ represents the bias vector, and $\sigma(\cdot)$ represents the nonlinear activation function. The function of Equation (6) is to use different expansion distances to extract near-neighbor fluctuations and inter-interval fluctuations, so that the model can perceive small fluctuations and short-range jumps at the same time.

A single convolution branch can only correspond to a fixed receptive field, so this paper further performs adaptive fusion of multi-scale local responses, whose expression is shown in Equation (7):

$$\beta_t^{(k)} = \frac{\exp\left(v_k^\top h_t^{(k)}\right)}{\sum_{j=1}^{K}\exp\left(v_j^\top h_t^{(j)}\right)}, \quad c_t = \sum_{k=1}^{K}\beta_t^{(k)}\, h_t^{(k)} \tag{7}$$

Here, $\beta_t^{(k)}$ represents the fusion weight of the k scale branch at time t, v_k represents the score vector of the corresponding branch, v_j represents the total number of convolution branches, and c_t represents the local representation after multi-scale fusion. The function of Equation (7) is to automatically adjust the contribution of each scale branch according to the local response at the current time, so that short-time spikes and slowly varying fluctuations can obtain different weights in the unified representation.

Whether local fluctuations need to be reinforced also depends on how significant the current moment is within the window. To this end, local saliency gating is constructed in this paper, and its calculation is shown in Equation (8).

$$\gamma_t = \mathrm{sigmoid}\left(w_g^\top [c_t\,;\; |d_t|\,;\; \mathrm{AvgPool}(E_{t-w:t})] + b_g\right) \tag{8}$$

Here, γ_t represents the local significance coefficient at time t, w_g represents the gating score vector, b_g represents the bias scalar, $|d_t|$ represents the absolute value term of the fluctuation enhancement vector, and $\mathrm{AvgPool}(\cdot)$ represents the average pooling operation. The function of Equation (8) is to incorporate the local fusion feature, fluctuation intensity and

window mean information into the scoring process, so as to determine whether the current local change is worth further strengthening.

After the significance coefficient is obtained, the local fluctuation output is no longer directly equal to the convolution result, but is renormalized by residual gating, and the output form is shown in Equation (9):

$$f_t = \gamma_t \odot c_t + (1 - \gamma_t) \odot Pe_t + Rd_t \tag{9}$$

Here, f_t represents the output vector of the local fluctuation feature extraction module, $\odot$ represents element-wise multiplication, P represents the original embedding projection matrix, and R represents the fluctuation enhancement term mapping matrix. The function of Equation (9) is to recombine the significant local fluctuations, the multi-scale convolution results and the original input representation, so that the strong change region is highlighted, and the weak change region retains the original stable information, while avoiding the local features being overly dominated by a single path.

After the above processing, the short-time variations of bus load in pre-peak, post-peak and switching sections no longer exist in the form of scattered numerical values, but are converted into local structural features with the ability to distinguish amplitude, span and saliency. The output obtained in this way can provide a clearer short-term basis for subsequent long-term dependency modeling, so that the next layer of the network does not have to repeatedly search for local fluctuation cues from the original embedding.

3.3 Modeling long-term dependencies

After the local fluctuation feature extraction, the input sequence already has the short-term change information, but the bus load prediction does not only rely on the local fluctuation of adjacent moments. The bus load has obvious long-term correlation in intra-day period repetition, inter-day operation continuity and weekly load rhythm. Therefore, the task of long-term dependence modeling is to further organize the short-term structure into a memory state that can reflect the continuous evolution trend. As shown in Fig. 1, this paper divides this process into three levels: gated state update, periodic hop fusion, and long-term representation output, so that the model can not only retain local fluctuations, but also accumulate effective memory along the time axis.

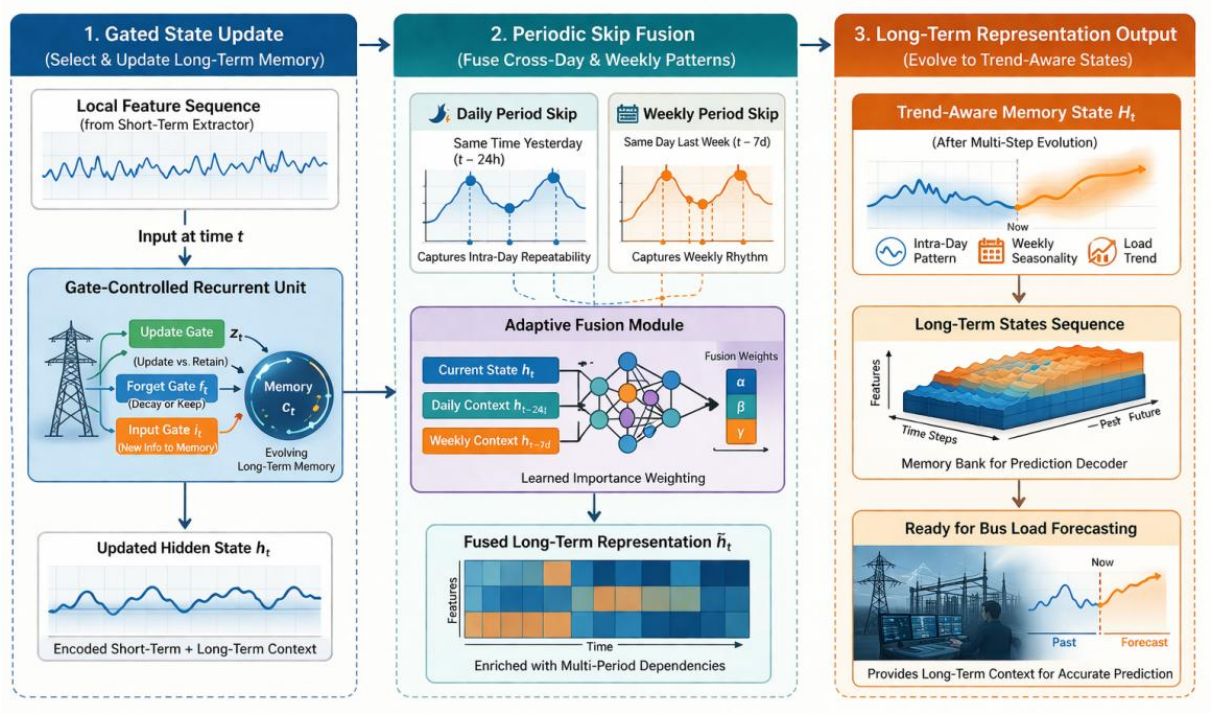

Figure 1: Process for modeling long-term dependencies

In the sequence recursion process, the gating unit first selectively writes the current input and the history state to avoid the short-time noise directly covering the existing memory. The gating variable is calculated as shown in Equation (10).

$$r_t = \sigma(W_r f_t + U_r h_{t-1} + b_r), \quad z_t = \sigma(W_z f_t + U_z h_{t-1} + b_z) \tag{10}$$

where f_t represents the local fluctuation feature vector at time t, h_{t-1} represents the hidden state at the previous time, r_t represents the reset gate, z_t represents the update gate, W_r, W_z represents the input mapping matrix, U_r, U_z represents the state mapping matrix, b_r, b_z represents the bias term, and $\sigma(\cdot)$ represents the compression function. The function of Equation (10) is to control how much new information should be written into the current input and how much old information should be kept in the historical state, so that the sequence modeling has the selective memory ability.

After the gating variable is obtained, the candidate state and the hidden state at the current time are updated in a recursive manner, and the calculation form is shown in Equation (11):

$$\tilde{h}_t = \tanh(W_h f_t + U_h(r_t \odot h_{t-1}) + b_h), \quad h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{11}$$

Here, $\tilde{h}_t$ represents the candidate hidden state, h_t represents the current hidden state, W_h and U_h represent the candidate state mapping matrix, b_h represents the bias term, $\odot$ represents element-wise multiplication, and $\tanh(\cdot)$ represents the hyperbolic tangent activation function. The function of Equation (11) is to gradually absorb the local fluctuation characteristics into the time series state, so that short-term changes can be converted into accumulative long-term memory in the recursive process.

Only relying on sequential recurrence is still not enough to fully express the periodic continuation in the bus load. Therefore, this paper further introduces the daily cycle and weekly cycle states into the current representation to form a cross-scale long-term association, and its fusion form is shown in Equation (12).

$$s_t = \phi(W_s[h_t; h_{t-24}; h_{t-168}] + b_s) \tag{12}$$

where s_t represents the long-term dependence representation at time t, h_{t-24} represents the hidden state at the same time of the previous day, h_{t-168} represents the hidden state at the same time of the previous week, $[\cdot;\cdot;\cdot]$ represents the vector splicing operation, W_s represents the periodic fusion matrix, b_s represents the bias term, and $\phi(\cdot)$ represents the nonlinear transformation function. The function of Equation (12) is to put the current state, daily cycle memory and weekly cycle memory into the same expression space, so that the model can not only see the recurrence trajectory of the current sequence, but also read the repeated patterns across days and weeks.

After the above modeling, the local fluctuation in the bus load sequence is no longer just a short-term fluctuation response, but is further organized as a long-term representation with continuity, periodicity and state inheritance characteristics. The st obtained in this way can provide a stable foundation for the attention enhancement in the subsequent critical period, so that the model has a clear temporal memory structure when it enters the next layer, without having to recover long-range dependent cues from the original local features.

3.4 Attention enhancement in key periods

Attention enhancement in critical periods does not assign weights to all time positions on average, but gives higher weights to moments that are more noteworthy according to the response differences of pre-peak, peak, post-peak and valley sections. Long-term dependency modeling has formed a continuous state representation, but these states are still occupied by ordinary stationary sections. If they are not screened, high-value information in critical periods is easy to be homogenized. Therefore, this paper introduces the key period attention enhancement mechanism on the long-term state, so that the model can actively focus on the time position with more obvious impact on the prediction error, and retain more discriminative temporal features in the subsequent output stage.

In the specific calculation, this paper first generates the attention score based on the current long-term state, the daily cycle state and the local fluctuation response, and its expression is shown in Equation (13).

$$\eta_t = v_a^{\top} \tanh(W_h s_t + W_p s_{t-24} + W_f f_t + b_a) \tag{13}$$

Here, η_t represents the attention score at time t, s_t represents the current long-term dependence state, s_{t-24} represents the periodic state at the same time of the previous day, f_t represents the local fluctuation feature, W_h, W_p, W_f represent the mapping matrix, b_a represents the bias term, v_a represents the score vector. Equation (13) is used to incorporate trend information, cycle information and local disturbance into the scoring process at the same time, so that the identification of critical periods no longer depends on a single state source, but is determined by multi-source time series information.

After the score is obtained, this paper uses the normalized weight to enhance the critical period, and the calculation form is shown in Equation (14):

$$\alpha_t = \frac{\exp(\eta_t)}{\sum_{j=t-u}^{t+u} \exp(\eta_j)}, \quad g_t = \alpha_t s_t + (1 - \alpha_t) f_t \tag{14}$$

Here, α_t represents the normalized attention weight, u represents the local attention window radius, and g_t represents the enhanced output representation of critical time periods.

Equation (14) is used to map scores into comparable weights, and then adjust the contribution ratio of long-term states and local fluctuations accordingly, so that the peak segment can be more fully expressed, and the stable segment can be moderately compressed, so as to reduce the dilution of information in critical periods from ordinary periods.

In order to illustrate the main scope of attention in key periods, the temporal sections that need to be focused on enhancement are sorted out in Table 4 in this paper.

Table 4: Action segments of attention enhancement during key periods

Segment Type	Temporal Characteristics	Enhancement Purpose	Effect on Forecasting
Pre-Peak Rising Segment	Load continues to rise and the gradient increases	Amplify pre-turning-point signals	Improve forecasting sensitivity before peak arrival
Peak Plateau Segment	High load persists with dense fluctuations	Preserve details of the high-load state	Reduce deviations near the peak
Post-Peak Falling Segment	Load declines and the rate of change accelerates	Highlight the falling boundary	Reduce lagging errors during the decline phase
Valley Stable Segment	Load remains low with slow fluctuations	Compress redundant responses	Prevent stable segments from occupying excessive weights

Instead of reading all the states evenly across the timeline, the model is able to allocate more computing resources to critical locations such as the pre-peak, peak, and post-peak. The g_t thus formed not only retains the continuous memory in the long-term dependence, but also strengthens the high-value segments in the local fluctuations, which provides a more targeted time series representation for the subsequent prediction optimization output. On the whole, this layer is not an additional independent module, but to reallocate and focus on the existing timing state, so that the model output is closer to the real change trajectory of the bus load in the critical period.

3.5 Prediction optimization output for accuracy improvement

The prediction optimization output for accuracy improvement is not to directly send the features obtained by the previous layers into the linear regression head, but to refuse the long-term dependence state, the enhancement results of the key period and the local fluctuation response, so that the output layer can not only maintain the main trend judgment, but also correct the deviation of the peak section. The bus load forecast is most prone to two types of deviations in the final output stage, one is from the average effect of the stationary section on the strong fluctuation section, and the other is from the phase misalignment between the periodic reference information and the current state. Therefore, this paper adds a two-stage processing of feature fusion and residual correction at the output, so that the predicted value not only comes from the deep representation, but also absorbs the compensation information in local changes and period differences.

In the output mapping stage, this paper firstly concatenates the key period enhancement representation with the preorder state feature and sends it to the optimized output layer, whose calculation form is shown in Equation (15):

$$o_t = \rho(W_o[g_t; s_t; f_t] + b_o) \tag{15}$$

where o_t represents the optimized output representation at time t, g_t represents the enhanced state vector at the critical period, s_t represents the time series state after long-term dependence modeling, f_t represents the local fluctuation characteristics, W_o represents the output mapping matrix, b_o represents the bias term, $[;]$ represents the vector splicing operation, and $\rho(\cdot)$ represents the nonlinear mapping function. Equation (15) is used to compress the time series information from different levels into a unified output space, so that the prediction head no longer relies on a single path feature, but reads the joint information of trend, key section and local change at the same time.

After obtaining the optimized output representation, this paper further introduces the residual correction term to compensate and correct the final prediction result, and its calculation form is shown in Equation (16).

$$\hat{y}_{t+1} = w_y^{\top} o_t + b_y + \lambda\, w_r^{\top}(g_t - s_{t-24}) \tag{16}$$

Here, $\hat{y}_{t+1}$ represents the bus load forecast value at the next time, w_y represents the main output weight vector, b_y represents the output bias term, λ represents the residual correction coefficient, w_r represents the compensation mapping vector, and s_{t-24} represents the periodic reference state at the same time of the previous day. Equation (16) is used to introduce periodic difference correction on the basis of the main forecast value, so that the offset between the current key state and the historical periodic state can participate in the final output, thereby weakening the forecast deviation caused by peak lag and slow decline.

As shown in Fig. 2, the prediction optimization output for accuracy improvement consists of four steps: state stitching - output mapping - residual correction - final prediction value generation. The first two steps are responsible for forming a unified output representation, and the last two steps are responsible for transforming cycle differences and key section information into the final prediction results.

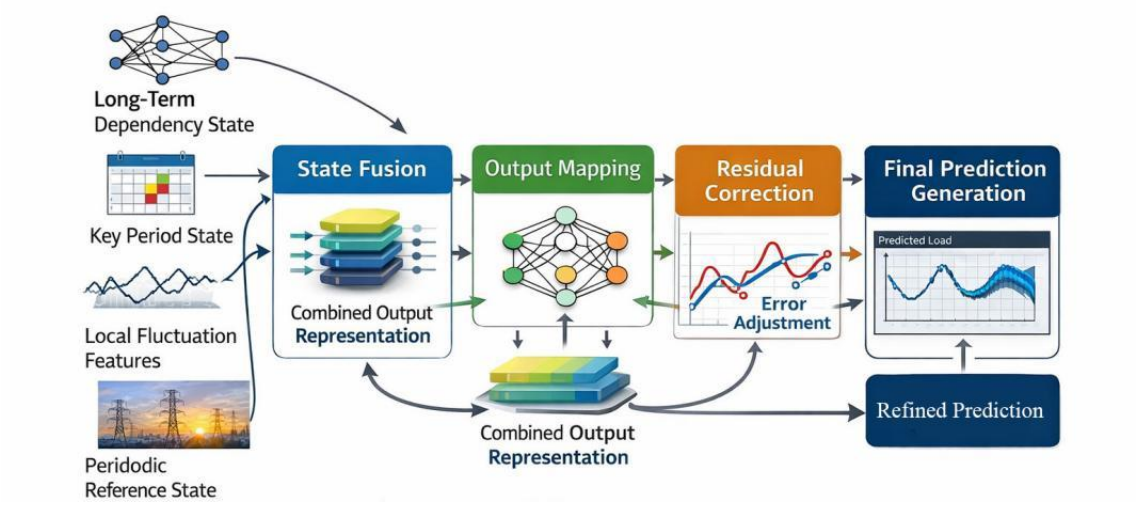

Figure 2: Output flow of prediction optimization for accuracy improvement

After this process, the output layer no longer simply gives a regression result, but integrates the enhanced state information in key periods, the continuous memory in long-term dependencies, and the periodic reference offset into the prediction generation process. The final predicted value formed in this way can be more accurately close to the real change trajectory of the bus load in the pre-peak, peak and decline stages, and also provides a clear target object and adjustment entrance for the hyperparameter optimization in the next chapter, so that the

subsequent tuning of the model no longer stays at the general network configuration level, but can directly focus on the improvement of prediction accuracy.

4 Hyperparameter optimization of prediction model

In order to avoid instability of model response or insufficient fitting of local section caused by parameter setting, this paper focuses on the input window, convolution scale, hidden dimension, attention mapping dimension and training control variable. The window length will affect the retention degree of periodic information, the convolution scale will change the extraction range of local fluctuations, and the hidden dimension and attention mapping dimension will determine the state expression ability and the ability to distinguish key periods. Based on this feature, this paper adopts a hierarchical search strategy driven by validation set, which first determines the input window and local convolution scale, and then jointly searches the parameters related to long-term dependence modeling and critical period attention. Finally, the output layer and training control parameters are adjusted to make the model configuration consistent with the periodic structure and fluctuation intensity of the bus load sequence.

In the specific search, the input window candidate values are set as 96, 120, 168 and 192, the local convolution kernel length candidate values are set as 3, 5, 7 and 9, the hidden dimension candidate values are set as 64, 96 and 128, the attention mapping dimension candidate values are set as 32, 64 and 96, and the dropout candidate values are set as 0.1, 0.2 and 0.3. In the training phase, the validation set MAPE and peak segment MAE are used as the joint criterion, and the parameter combination with more stable error control in the critical period is preferentially retained. The results of multiple rounds of comparison show that when the window length is 168, the convolution kernel length is 3/5/7, the hidden dimension is 96, the attention mapping dimension is 64, and the dropout is 0.2, the model achieves a more balanced performance between the overall error, peak response and convergence stability. Table 5 summarizes the final retained hyperparameter combinations.

Table 5: Optimal hyperparameter combinations

Module	Final Parameters
Input Window	168 time steps
Local Convolution	Kernel sizes of 3/5/7, with 3 branches
Long-Term Dependency Modeling	Hidden dimension of 96, periodic skip connections of 24 and 168
Critical Period Attention	Projection dimension of 64, local window radius of 3
Output Layer	Fully connected dimension of 32, learnable residual correction coefficient
Training Control	AdamW, learning rate 0.001, batch size 32, dropout 0.2, epoch 80

These parameters are obtained by stepwise screening under uniform validation criteria. The window length 168 can completely cover the intra-day repetition and cross-day continuation information of the bus load, the 3/5/7 convolution kernel combination can take into account the local feature extraction of the mutation segment, the transition segment and the slow change segment, and the hidden dimension 96 and the attention mapping dimension 64 make the model maintain a moderate balance between the state expression ability and the computational cost. The training control part adopts a small learning rate and a medium dropout ratio, which is beneficial to maintain the continuity of the convergence process and weaken the error

fluctuation near the critical period.

5 Experimental Results

5.1 Comparison and analysis of prediction accuracy of different models

In order to verify the effectiveness of the proposed model in bus load time series prediction, SVR, random forest, GRU and standard LSTM are selected as comparison models, and a unified evaluation is completed under the same data partition conditions. The evaluation indicators are R^2, MAE, RMSE and MAPE, where R^2 is used to reflect the overall fitting consistency, MAE and RMSE are used to measure the absolute error, and MAPE is used to characterize the relative error. The test results show that the traditional machine learning model can complete the basic trend fitting, but the error control is still weak in the high fluctuation section. The sequence model expresses the temporal dependence more fully, and the overall accuracy is significantly improved. On this basis, the proposed model continues to compress the critical section deviation, so that the overall fitting accuracy and the local response ability are improved simultaneously.

As shown in Fig. 3, the distribution difference of different models on the test set error index is clear. The MAE, RMSE and MAPE of SVR are 61.38 MW, 88.42 MW and 3.27%, respectively. The corresponding index of random forest decreases to 49.76 MW, 71.05 MW and 2.68%, indicating that the tree model is slightly better than the kernel method in overall error control, but the expression of complex time series structure is still limited. After GRU and standard LSTM enter the sequence modeling, the error shrinks significantly, and the MAE and RMSE of the standard LSTM are reduced to 22.63 MW and 33.47 MW, respectively, and the MAPE is reduced to 1.04%. In contrast, the MAE of the proposed model is further reduced to 18.43 MW, the RMSE is reduced to 27.16 MW, and the MAPE is compressed to 0.86%, while the R^{-2} reaches 0.9969. The core feature reflected in Fig. 3 is not that there is a simple ordering among models, but that the traditional model, the common sequence model and the proposed model form a clear stratification in the error level, indicating that the local fluctuation extraction and key period enhancement indeed change the accuracy structure of the prediction output.

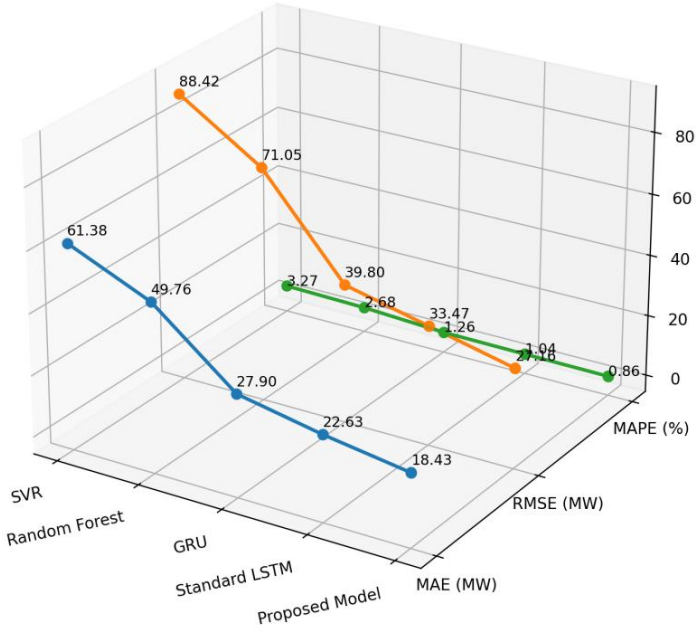

Figure 3: Comparison plots of error metrics on test sets for different models

Fig. 4 further shows the comparison of the prediction curves of typical buses in critical periods. During this period, the real load continued to rise from 742.5MW to 812.6MW at the peak and then began to decline. SVR is predicted to be 764.9 MW at the peak position, and random forest is predicted to be 776.4 MW, both of which show significant underestimation. GRU prediction reaches 795.1 MW, and the standard LSTM is increased to 801.3 MW, which can track the overall trend well, but there is still a certain amplitude deviation in the peak maintenance section. The peak prediction of the proposed model reaches 808.7 MW in the same time period, and the gap between the model and the true value is only 3.9 MW. After entering the fallback section, the real load drops to 746.8MW, the proposed model predicts 748.9MW, the deviation is 2.1MW, and the deviation of the standard LSTM at the same point is 7.4MW. The differences presented in Fig. 4 show that the proposed model not only has lower overall error, but also has stronger response consistency in the three key stages of pre-peak lifting, peak maintenance and post-peak falling, and the phase lag and amplitude deviation of the curves are controlled.

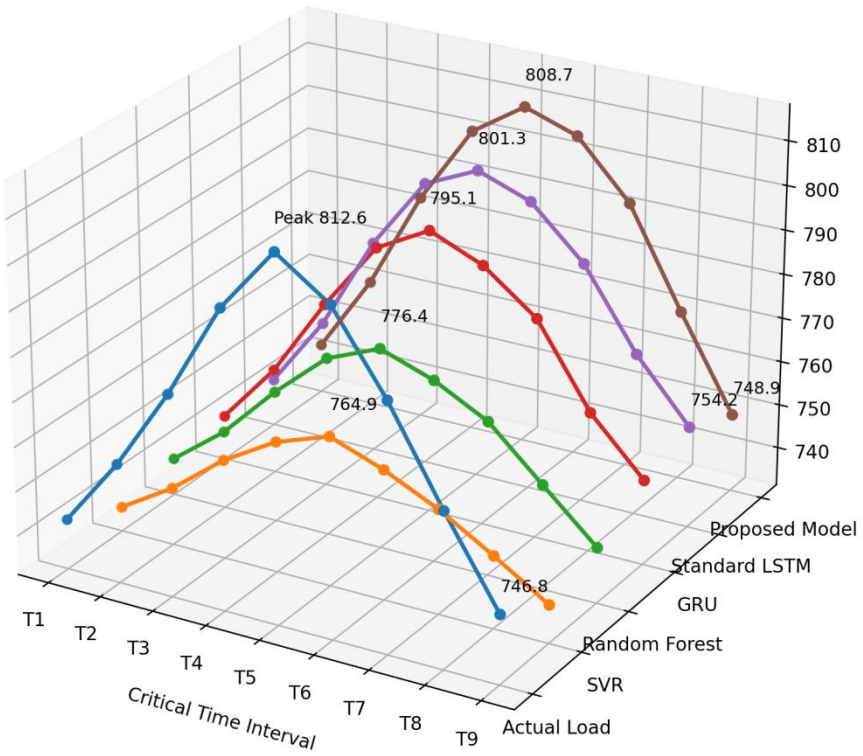

Figure 4: Comparison of forecast curves for key periods of typical buses

Taken together, the comparison results in this section form a mutually confirming relationship with Fig.s 3 and 4. Fig. 3 proves that the model in this paper is in the optimal position in terms of the overall error index, and Fig. 4 further shows that this advantage is not only reflected in the average error level, but implemented in the actual fitting ability of the critical period. GRU and standard LSTM have been able to track the main trend of bus load well. However, the model in this paper maintains more stable prediction accuracy in both critical sections and continuous sections, so it shows stronger adaptability in both overall performance and local details.

5.2 Analysis of bus load forecasting results in critical periods

The overall error index can reflect the average performance of the model, but the actual effectiveness of the bus load forecast depends more on the fitting quality of the critical period. For the power system operation, the pre-peak lifting section is related to the load warning and reserve arrangement, the peak maintenance section is related to the dispatch safety under high load state, the post-peak falling section is related to the output back and power flow recovery, and the stable section is related to the stability of the model under weak fluctuation scenarios. Therefore, instead of repeating the overall indicators, this section focuses the analysis on the local prediction results of typical critical time periods to examine the model response differences in different dynamic segments.

As shown in Fig. 5, a continuous window in the test set containing "pre-peak rise -- peak maintenance -- post-peak fall" was selected for comparison. The real bus load continued to rise from 736.4MW, reached 811.8MW at 08:00, and fell back to 752.6MW after maintaining the high level for about 3 sampling points. The standard LSTM can track the overall trend well in this section, but there is still a slight lag in the pre-peak lifting section. The peak forecast is 803.5MW, which is 8.3MW lower than the real value. After entering the peak maintenance section, the prediction curve began to slide in advance, showing a certain lack of high load maintenance. In contrast, the peak predicted by the proposed model at the same location is 809.6 MW, which is only 2.2 MW away from the true value, and it can still maintain good amplitude consistency within the maintenance segment. The differences presented in Fig. 5 indicate that after the enhanced attention in the critical period, the model's response to the peak segment is no longer just following the upward trend, but is able to maintain higher local sensitivity in the pre-peak and peak stages.

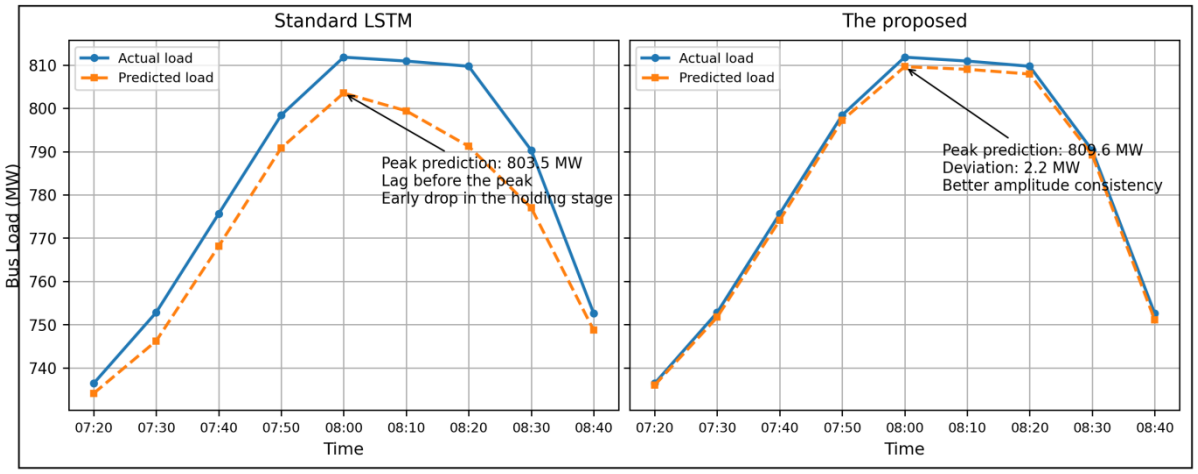

Figure 5: Comparison of bus load prediction curves between pre-peak lifting and peak maintenance stages

By further observing the post-peak decline and the stable stage of the trough, as shown in Fig. 6, the traditional model is prone to two types of deviations in the decline stage, one is the lag of decline, and the other is the decline too fast. The standard LSTM can basically maintain the same decline direction in this section, but when the real load drops from 784.2MW to 708.5MW, its prediction curve is still about 6.8MW higher than the real value in the middle section. The average deviation of the proposed model in the same section is compressed to 2.4 MW, and it can enter the smooth trough state faster after the fall down. In the valley section, the real load fluctuates slightly around 694 MW, the fluctuation range of the standard LSTM is about 7.1 MW, and the model in this paper shrinks to 4.3 MW, indicating that the residual

correction of the output layer not only improves the fitting accuracy of the peak section, but also makes the prediction fluctuation of the valley stage more stable. Fig. 6 further proves that the advantage of the proposed model is not only reflected in the vicinity of the peak, but also maintains good response continuity in both key sections of the fall transition and the smooth valley.

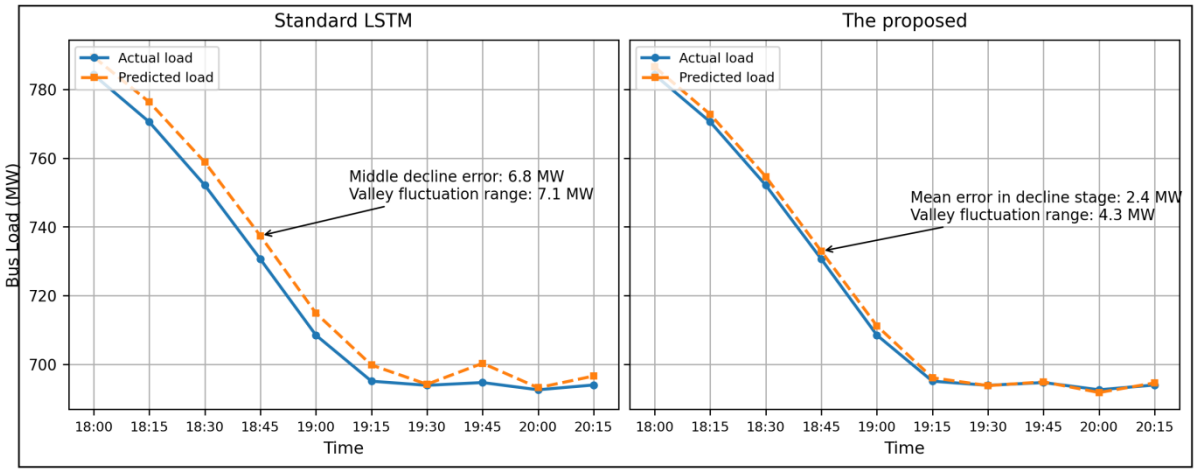

Figure 6: Comparison of the bus load forecasting curves in the post-peak decline and the plateau stage of the trough

From a comprehensive point of view, the key period analysis and the comparison of the overall indicators in the previous paragraphs form a mutual corroboration relationship. The overall results show that the proposed model is better in terms of average error, and the results in this section further show that this advantage is specifically implemented in the local fitting ability of key sections such as pre-peak, peak, post-peak and valley. The standard LSTM has been able to express the main trend of bus load, and the proposed model continues to compress the deviation in turning point location, peak maintenance and falling transition. Therefore, its accuracy improvement has a clearer time structure basis, rather than the surface improvement obtained by pulling down the average error of a single stationary section.

5.3 Ablation experiment

In order to test the actual contribution of each core module to the accuracy of bus load time series prediction, this paper sets up four sets of ablation experiments based on the complete model, removing the local fluctuation feature extraction, long-term dependence modeling, attention enhancement in key periods and residual correction output modules respectively, and keeping the other training conditions consistent. The ablation results focus on comparing the effect of different structures removed on error compression, peak response, and overall fitting ability.

The results show that after removing the local fluctuation feature extraction, the response of the model in the pre-peak rise and post-peak fall sections is significantly dull, and the MAPE increases from 0.86% to 1.14%. After removing the long-term dependence modeling, the model can track the local changes, but the stability of the cross-day repeating section is weakened, and the R^2 drops to 0.9928. After removing the attention enhancement in the critical period, the error of the peak maintenance segment expands, and the MAE rises to 24.67 MW. After removing the residual correction output, the overall trend can still be maintained, but the amplitude deviation at the key turning point position increases, and the RMSE rises to 35.42 MW. Table 6 presents the main indicators of each ablation model as follows.

Table 6: Results of ablation experiments

Model Variant	R^2	MAE (MW)	RMSE (MW)	MAPE (%)
Full Model	0.9969	18.43	27.16	0.86
Without Local Fluctuation Extraction	0.9946	23.91	34.05	1.14
Without Long-Term Dependency Modeling	0.9928	27.36	39.84	1.27
Without Critical Period Attention	0.9941	24.67	35.98	1.09
Without Residual Correction Output	0.9937	25.42	35.42	1.12

The comprehensive comparison shows that all four modules form a direct contribution to the prediction accuracy, but the direction of action is not the same. The local fluctuation extraction improves the short-term response, the long-term dependence modeling maintains the cycle consistency, the attention in the critical period enhances the compression of the peak section error, and the residual correction output further tights the final deviation.

5.4 Discussion

The advantage of the proposed model in bus load timing prediction is not only reflected in the overall error reduction, but also reflected in the collaborative expression of different time structures in the same calculation link. The local fluctuation extraction strengthens the recognition of short-term changes in pre-peak rise and post-peak fall, the long-term dependence modeling ensures the continuous expression of cross-day and cross-week repetitive patterns, the attention enhancement in critical periods improves the state discrimination ability of high-value segments, and the residual correction output further reduces the amplitude deviation near the peak. Compared with the traditional models, the proposed method does not rely on fixed feature maps and piecewise rule fitting, but completes the continuous learning from sample representation to prediction output through a unified deep temporal modeling process. The experimental results show that this structural combination not only improves the overall fitting accuracy, but also enhances the response stability of key sections, which makes the model show better adaptability and generalization ability in bus-level load forecasting tasks. Further analysis shows that the accuracy improvement is not the result of simply increasing the depth of the network, but the result of input reorganization, local extraction, long-term memory and key section reinforcement. This structure can not only maintain the prediction stability of the stationary section, but also maintain sufficient response sensitivity in the turning section, so that the overall error compression and local detail fidelity can be achieved simultaneously. This also shows that the deep time series structure for bus load forecasting not only has good engineering applicability, but also has a clear structural interpretation basis.

6 Conclusions

Focusing on the accuracy improvement of bus load time series prediction in power system, this paper constructs a deep learning prediction framework consisting of time series sample recombination, local fluctuation extraction, long-term dependence modeling, attention enhancement in key periods, and residual correction output. The experimental results show that the proposed model is superior to SVR, random forest, GRU and standard LSTM in terms of R^2, MAE, RMSE and MAPE. It also maintains higher fitting consistency for the key sections such as pre-peak rise, peak maintenance and post-peak fall. This shows that the deep timing structure for bus-level load forecasting can not only compress the overall error, but also enhance the ability to preserve local details. It should be pointed out that the model in this paper still has

a relatively clear applicable boundary. The training samples are mainly from the bus load data of a single area, and the difference of load structure between regions has not been included in the validation. The input variables are still dominated by electricity quantity and time characteristics, and the exogenous factors such as meteorological disturbance, price fluctuation and operation mode switching have not been fully integrated. The prediction task is currently limited to hourly short-term scenarios, and the stability under multi-step rolling prediction and more high-frequency sampling conditions still lacks further testing. Further research can be carried out in the directions of cross-regional sample expansion, exogenous variable fusion, lightweight deployment, uncertainty estimation, and multi-step prediction mechanism, so as to enhance the adaptation ability of the model in complex scheduling environments.

Funding

Project ID: 5205D0250002 | Project Title: Intelligent Bus Load Prediction Technology and Application Development for New Power Systems | Note: Supported by the National Grid Shanxi Electric Power Co., Ltd. Science and Technology Project

References

[1] Alrasheedi A, Almalaq A. Hybrid deep learning applied on Saudi smart grids for short-term load forecasting[J]. Mathematics, 2022, 10(15): 2666.

[2] Ullah F U M, Ullah A, Khan N, et al. Deep Learning-Assisted Short-Term Power Load Forecasting Using Deep Convolutional LSTM and Stacked GRU[J]. Complexity, 2022, 2022(1): 2993184.

[3] Abumohsen M, Owda A Y, Owda M. Electrical load forecasting using LSTM, GRU, and RNN algorithms[J]. Energies, 2023, 16(5): 2283.

[4] Gonzalez R, Ahmed S, Alamaniotis M. Implementing very-short-term forecasting of residential load demand using a deep neural network architecture[J]. Energies, 2023, 16(9): 3636.

[5] Chen H, Zhu M, Hu X, et al. Research on short-term load forecasting of new-type power system based on GCN-LSTM considering multiple influencing factors[J]. Energy Reports, 2023, 9: 1022-1031.

[6] Mounir N, Ouadi H, Jrhilifa I. Short-term electric load forecasting using an EMD-BI-LSTM approach for smart grid energy management system[J]. Energy and Buildings, 2023, 288: 113022.

[7] Li K, Mu Y, Yang F, et al. A novel short-term multi-energy load forecasting method for integrated energy system based on feature separation-fusion technology and improved CNN[J]. Applied Energy, 2023, 351: 121823.

[8] Ran P, Dong K, Liu X, et al. Short-term load forecasting based on CEEMDAN and Transformer[J]. Electric Power Systems Research, 2023, 214: 108885.

[9] Sun C, Ning Y, Shen D, et al. Graph Neural Network-Based Short-Term Load Forecasting

with Temporal Convolution[J]. Data Science and Engineering, 2024, 9(2): 113-132.

[10] Waheed W, Xu Q. Data-driven short term load forecasting with deep neural networks: Unlocking insights for sustainable energy management[J]. Electric Power Systems Research, 2024, 232: 110376.

[11] Smyl S, Dudek G, Pełka P. Contextually enhanced ES-dRNN with dynamic attention for short-term load forecasting[J]. Neural Networks, 2024, 169: 660-672.

[12] Zhenglei Z, Jun C, Zhou Y, et al. Research on urban power load forecasting based on improved LSTM[J]. Frontiers in Energy Research, 2024, 12: 1443814.

[13] Wen X, Liao J, Niu Q, et al. Deep learning-driven hybrid model for short-term load forecasting and smart grid information management[J]. Scientific reports, 2024, 14(1): 13720.

[14] Chen J, Liu L, Guo K, et al. Short-term electricity load forecasting based on improved data decomposition and hybrid deep-learning models[J]. Applied Sciences, 2024, 14(14): 5966.

[15] Su Z, Zheng G, Hu M, et al. Short-term load forecasting of regional integrated energy system based on spatio-temporal convolutional graph neural network[J]. Electric Power Systems Research, 2024, 232: 110427.

[16] Wang Y, Hao Y, Zhang B, et al. Short-term power load forecasting using SSA-CNN-LSTM method[J]. Systems Science & Control Engineering, 2024, 12(1): 2343297.

[17] Jiang H, Dong Y, Dong Y, et al. Power load forecasting based on spatial–temporal fusion graph convolution network[J]. Technological Forecasting and Social Change, 2024, 204: 123435.

[18] Nabavi S A, Mohammadi S, Motlagh N H, et al. Deep learning modeling in electricity load forecasting: Improved accuracy by combining DWT and LSTM[J]. Energy Reports, 2024, 12: 2873-2900.

[19] Zhou Y, Su Z, Gao K, et al. A short-term electricity load forecasting method integrating empirical modal decomposition with SAM-LSTM[J]. Frontiers in Energy Research, 2024, 12: 1423692.

[20] Asiri M M, Aldehim G, Alotaibi F A, et al. Short-term load forecasting in smart grids using hybrid deep learning[J]. IEEE access, 2024, 12: 23504-23513.