2025 5th International Conference on Artificial Intelligence and Industrial Technology Applications (AIITA)

Multi-Scale Pyramid Networks and Dynamic Graph Learning for Short-Term Power Load Forecasting

Guangpeng Wang
Dept. of Artificial Intelligence
China University of Petroleum (Beijing)
Beijing, China
2022211269@student.cup.edu.cn

Jingwen Wei
Dept. of Artificial Intelligence
China University of Petroleum (Beijing)
Beijing, China
2023216042@student.cup.edu.cn

Kaiwen Xing
Dept. of Artificial Intelligence
China University of Petroleum (Beijing)
Beijing, China
2023216030@student.cup.edu.cn

Hongjun Sun
Dept. of Artificial Intelligence
China University of Petroleum (Beijing)
Beijing, China
* Corresponding author: Sunhj68@cup.edu.cn

***Abstract*—Short-term power load forecasting is a crucial task in power system planning and operation, which is directly related to the stability and economy of power supply. Although several methods have been applied to this field, accurate forecasting still faces challenges, especially when dealing with complex spatio-temporal relationships. In this paper, we propose a short-term power load forecasting model based on the combination of multi-scale pyramid network and graph learning module. The multi-scale pyramid network can effectively capture multi-scale temporal features in the load data, while the multi-temporal graph generator further improves the prediction accuracy by constructing a map of spatio-temporal relationships among the load data. Experimental results on several public datasets show that the model outperforms both traditional methods and some of the latest deep learning models in terms of prediction accuracy and generalization ability. In particular, the contribution of each module to the overall performance of the model is verified through ablation experiments. The research in this paper not only provides new ideas for short-term power load forecasting, but also provides strong technical support for the optimal operation of smart grids.**

Keywords-Short-term power load forecasting; graph learning module; deep learning; graph convolution; multi-scale modeling.

I. INTRODUCTION

Electricity is a clean and efficient energy source that plays an irreplaceable role in our daily life. In addition, electricity is more suitable and efficient for meeting the requirements of an environmentally friendly society than other traditional energy sources such as natural gas, coal and oil [1]. However, electricity as a product is different from material products because it cannot be stored in bulk but must be generated instantly and supplied on demand [2]. In recent years, as the grid continues to evolve, electricity demand patterns have become complex, which can lead to oversupply or power shortages, resulting in inaccurate forecasts and leading to significant economic losses [3]. In addition, power grids are facing many challenges in terms of rising power demand, integration of renewable energy sources and efficient power system operation. Ensuring the security, stability and cost-effectiveness of the power system has become a top priority for utilities and grid operators worldwide. In this context, improving the accuracy of load forecasting has become a key area of research and development [4]. Accurate load forecasting is essential for achieving effective power dispatch and is necessary to ensure reliable and efficient operation of the power system [5]. Short-term load forecasting has received more attention because of its higher accuracy and reliability compared to medium- and long-term load forecasting [6]. However, electricity demand is affected by factors such as weather conditions as well as daily activities. Factors such as seasonality, trends, noise, and outliers need to be considered when conducting studies on electricity demand to ensure the accuracy of forecasts and to reduce power production losses and costs [7]. Therefore, short-term load forecasting methods considering relevant factors have become popular [8]. U.Saini et al [9] used LSTM-GRU combination on univariate dataset and obtained more accurate prediction results. Musaed Alhussein et al [10] used CNN-LSTM model on individual household public electricity dataset and good results were achieved. Tang et al [11] proposed a short-term load forecasting model using TCN and AM, which fully utilized the nonlinear relationship between meteorological factors and load, and conducted experiments on two different datasets. The results show that their method effectively improves the prediction accuracy. Zhang [12]et al proposed time-enhanced Transformer network for short-term power load forecasting, which aims at learning the representation of temporal relationships between input sequences and evaluates it on a real power load dataset. The results show that the model has higher prediction accuracy. However, none of these models take into account the correlation between various exogenous variables and the time-varying characteristics of electricity data. In this paper, a deep learning approach will be used to forecast short-term electricity loads and a hybrid model combining multi-scale pyramid networks and graph learning modules is proposed, i.e., decomposing time series to capture short-term fluctuations by multi-scale pyramid networks, modeling the dynamic correlations among variables over time by a multi-temporal graph generator tweed, and enhancing the ability to capture the complex dependencies among variables by temporal graph convolution operations.

979-8-3315-0976-7/25/$31.00 ©2025 IEEE

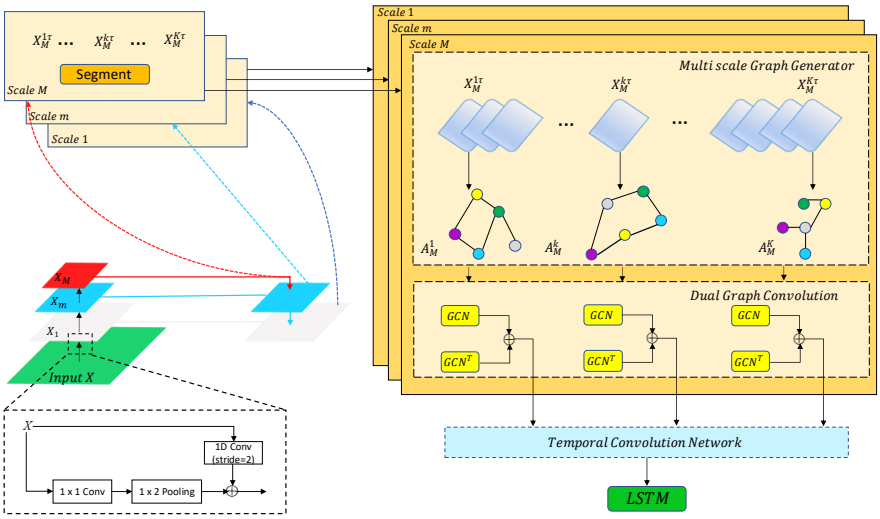

Figure 1. Overall framework diagram.

II. Network Architecture

A. Overall framework

As shown in Figure 1, the model consists of three main parts: the temporal feature pyramid module. It is used to extract the information of different time scales, the multi-temporal graph generator, which sets a fixed time step and learns to generate the corresponding graph expressing the relationship between nodes according to the time step setting, and the temporal graph convolution module which performs a double graph convolution operation on the information of each node and establishes temporal connections for each part of the features.

B. Temporal Feature Pyramid Network

In this network, the input time series are progressively transformed into feature representations at different scales. Each layer of the pyramid structure extracts information from the output of the previous layer and generates new feature representations that capture information at different time scales. Specifically, given an input time series $X \in R^{N\times T}$, the network generates feature representations at multiple scales through multilayer operations, with the convolution operation $\otimes$ at the m layer as follows:

$$X_m[c_{out}, t] = \sum_{c_{in}=0}^{C_{m-1}-1} \sum_{i=0}^{k-1} X_{m-1}[c_{\text{in}}, t\cdot s + i - p] \cdot W_m[c_{\text{out}}, c_{\text{in}}, i] + b_m[c_{\text{out}}] \quad (1)$$

where X_m is the feature representation of the m scale, with the shape $X_m \in R^{c_{\text{out}}\times\frac{T}{2^{m-1}}}$, c_{out} is the number of output channels, $\frac{T}{2^{m-1}}$ denotes the length of the time series after processing in the m layer, t denotes the time step index, k denotes the size of the convolution kernel in the m layer, and $s$denotes the step size of the convolution kernel.

As the layers deepen, different sizes of convolution kernels are used to gradually extract a wider range of temporal features. In order to find a balance between large-scale and small-scale temporal features, larger convolutional kernels are used in the initial layer of the network, and the kernel size is gradually reduced in subsequent layers. This design allows the model to flexibly adjust the sense field so as to capture both fine-grained short-term fluctuations and identify long-term trend changes, for example, by setting the size of each convolution kernel in each pyramid to 1×7,1×5,1×3, and by setting the step size of the convolution to 2:

$$X_{m,trans} = \text{ReLU}\left(W_{m,trans} \otimes X_{m-1} + b_{m,trans}\right) \quad (2)$$

where $W_{*,trans}$ and $b_{*,trans}$ denote the convolution kernel and bias vector.

To further enhance the model's performance on different scales, we introduce a parallel convolutional structure as in [14]. This structure uses a 1×1 convolutional kernel in combination with a pooling operation (e.g., 1×2 pooling) to capture different temporal features. The process is formalized as:

$$X_{m,pool} = \text{MaxPool}\left(\text{ReLU}\left(W_{m,norm} \otimes X_{m-1} + b_{m,norm}\right)\right) \quad (3)$$

This multi-scale feature fusion strategy allows the network to retain fine-grained information in the time series while capturing global trend changes, thus improving the model's overall understanding and prediction of time series data.

C. Multi-Segment Graph Generator

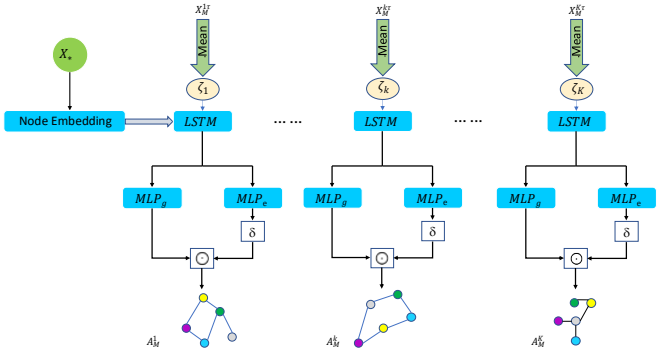

Figure 2. Multi-segment graph generator

Most of the time it is possible that the correlations of variables within adjacent time stamps may be similar or even the same, but the dynamic correlations between variables are not static in the long run. Therefore, we assume that the correlations of variables remain constant within a time interval and have an evolutionary relationship between adjacent time intervals. To capture this dynamic relationship, we use the idea of LSTM, which not only takes into account the dependencies of the current input values, but also incorporates the graph structure of the previous time step, which is updated in a recursive manner. Figure 2 illustrates a dynamic recursive processing architecture where the node embedding module generates initial node features, which serve as the input for the LSTM module at the first time step. The LSTM module processes the sequential data, capturing temporal dependencies, and outputs hidden states at each time step. These hidden states are then passed to the corresponding MLP module. The MLP module transforms the LSTM outputs into representations suitable for generating graph structures. The graph structure generation module utilizes these representations to create dynamic graph structures at each time step. The generated graph structures can influence the input to the LSTM module in subsequent time steps, forming a feedback loop that enables the model to learn and adapt to dynamic correlations over time.

$$A^{(t)} = \mathrm{G}\left(A^{(t-1)}, \theta^{(t)}\right) \tag{4}$$

where $A^{(t)} \in R^{N\times N}$ is the adjacency matrix of time step t, which describes the dynamic correlation of the current time step, $\theta^{(t)}$ denotes the node features, and G denotes the function that extracts the dynamic correlation. Taking the m scale convolutional layer output $\mathrm{X_m} \in R^{N\times T}$ as an example, it is divided into a number of segments, and then a mean aggregation operation is applied to the features of each segment to obtain the input sequence of the $LSTM$:

$$\zeta_{\text{mean}}^{(k)} = \frac{1}{\tau}\sum_{t=(k-1)\tau+1}^{k\tau} X(:,t) \in R^{N} \tag{5}$$

where τ is the time interval, K is the total number of segments, and $\mathrm{X}(:,\mathrm{t})$ represents the features of all nodes at time step t.

Then, the result after aggregating all scales is: $\left[\zeta^{(1)}, \zeta^{(2)}, \ldots, \zeta^{(k)}, \ldots, \zeta^{(K)}\right]$Directly parameterizing the adjacency matrix would make the model overly complex to compute. Therefore, we $LSTM$ the hidden state h to represent the changing nodes, which will keep changing over time, and to deal with the cold-start problem, we use the output of the convolutional layer at the current scale to initialize the static table of nodes:

$$\beta_{s,i}\mathrm{X} = X_{m,i} \tag{6}$$

where $\beta_{s,i}$ is the static representation of node i, $X_{m,i}$ is the data of the i node, L is the feature extractor which can be a multilayer perceptron, recurrent neural network, etc., and then use β_s through MLP_s as the initialization of the hidden state $h^{(0)}$.

$$h^{(0)} = MLP_s\left(\beta_{s,i}\right) \tag{7}$$

The modeling of the dynamics is represented by continuous iteration of the $LSTM$, which is updated as:

$$i^{(k)} = \sigma\left(W_i\zeta^{(k)} + U_i h^{(k-1)} + b_i\right) \tag{8}$$

$$f^{(k)} = \sigma\left(W_f\zeta^{(k)} + U_f h^{(k-1)} + b_f\right) \tag{9}$$

$$o^{(k)} = \sigma\left(W_o\zeta^{(k)} + U_o h^{(k-1)} + b_o\right) \tag{10}$$

$$\hat{c}^{(k)} = \tanh\left(W_c\zeta^{(k)} + U_c h^{(k-1)} + b_c\right) \tag{11}$$

$$\mathrm{c^{(k)}} = \mathrm{f^{(k)}} \odot \mathrm{c^{(k-1)}} + \mathrm{i^{(k)}} \odot \hat{c}^{(k)} \tag{12}$$

$$\mathrm{h^{(k)}} = \mathrm{o^{(k)}} \odot \tanh\left(\mathrm{c^{(k)}}\right) \tag{13}$$

where $i^{(k)}$, $f^{(k)}$, $o^{(k)}$ are the input, oblivion, and output gates, respectively, $c^{(k)}$ is the candidate cell state, W_i, W_f, W_o, W_c are the weight matrices of the inputs to the gate, U_i, U_f, U_o, U_c are the weight matrices of the last hidden state to the gate, b_i, b_f, b_o, b_c are the bias terms, σ is the $sigmoid$ activation function, $tanh$ is the hyperbolic tangent activation function, and $\odot$ denotes the $Hadamard$ product. Get $h^{(k)}$ as a new node feature, pass $h^{(k)}$ through MLP to derive the adjacency matrix $\hat{A}_{ij}^{(k)}$, learn a mask $M_{ij}^{(k)}$ to control the information output proportions while deriving the graph structure:

$$\hat{A}_{ij}^{(k)} = \mathrm{MLP}_g\left(\beta_i^{(k)}, \beta_j^{(k)}\right) \tag{14}$$

$$M_{ij}^{(k)} = \mathrm{MLP}_e\left(\beta_i^{(k)}, \beta_j^{(k)}\right) \tag{15}$$

$$A^{(k)} = \hat{A}_{ij}^{(k)} \odot \sigma\left(M^{(k)}\right) \tag{16}$$

where $\hat{A}_{ij}^{(k)}, M_{ij}^{(k)}$ denote the value of the learned graph structure and the i row$j$ column k-mask matrix, respectively, and $A^{(k)}$ denotes the final graph structure of the k time period.

D. Dual Graph Convolution Module

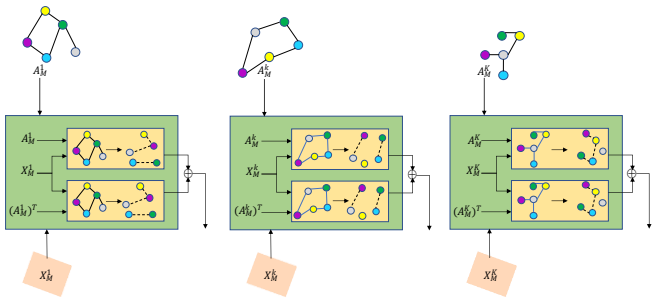

Figure 3. Dual graph convolution model

A series of adjacency matrices$\{A_m^1, A^k, \ldots, A^K\}$ are generated according to each scale via the multi-segment graph generator in Figure 3 Similar to [13], we introduce the transpose of A^k and A^k of $(A^k)^T$ and utilize two GCNs to capture the information and then sum the results of the GCNs:

$$u_{(m)}{}^k = \mathrm{GCN}_1^k\left(\zeta_t^k, A^k, W_1^k\right) + \mathrm{GCN}_2^k\left(\zeta_t^k, (A^k)^T, W_2^k\right) \tag{17}$$

where W_*^k denotes the kth scale trainable parameter. This dual GCN structure is able to capture different types of

information flow in the graph (i.e., from the adjacency matrix and its transpose). Then we get all the outputs $\{u^1, u^k, \dots, u^K\}$ passed into the temporal convolutional network:

$$U_m = \mathrm{TCN}_m(\{u^1, \dots, u^k, \dots, u^K\}) \quad (18)$$

Finally, all the features are passed into the LSTM network for final output prediction.

III. Experiments

In this section, we evaluate the performance of the proposed model on five real datasets and verify its generalization ability by examining its performance on two different domains. In the SH dataset, which includes 16 nodes and records load, week and other information, we selected the data from January 1, 2017 to December 30, 2018, it can be found at this URL: https://github.com/Mark-THU/load-point-forecast. In the AP dataset, which contains seven nodes covering load, electricity price, humidity information, we used the data from January 1, 2009 to December 31, 2010, it can be found at this URL: https://zhuanlan.zhihu.com/p/150954853. In the AT dataset, which contains six nodes and includes information such as load, temperature, wind speed and wind direction, we selected the data from January 1, 2015 to December 31, 2016, it can be found at this URL: https://github.com/Lizhuoling/DCN. In the ISO-NE dataset, which contains seven nodes and covers load, temperature and other information, we used the data from January 1, 2013 to December 31, 2014, it can be found at this URL: https://github.com/ningningLiningning/iso-ne. In the NCENT dataset, there are six nodes involving information such as load and year, and we selected the data from January 1, 2002 to December 31, 2003, it can be found at this URL: https://github.com/kmcelwee/load-forecasting.

We divided the extracted data into training, validation and test sets in the ratio of 6:2:2. For these five datasets, we set T_in = 72 and T_out = 240, i.e., 72 historical data were input to predict 240 future load values, and the mean square error (MSE) and mean absolute error (MAE) were used to evaluate the error between the true and predicted values.

TABLE 1. COMPARISON WITH BASELINES

Dataset / Model	SH		AP		AT		ISO-NE		NCENT	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
SA	0.7803	0.9736	0.7119	0.8035	0.8735	1.0576	0.7304	0.8642	0.7300	0.8812
NF	2.2432	5.9534	0.8204	1.0993	1.2169	2.2295	0.8271	1.0025	1.3288	2.1699
MA	2.5027	7.2194	0.9163	1.3199	1.2613	2.3775	0.7225	0.8139	1.7699	3.6731
RNN	0.3296	0.2065	0.3532	0.2280	0.2685	0.1541	0.4623	0.3643	0.5141	0.3926
CNN	0.3606	0.2539	0.3879	0.2562	0.2625	0.1377	0.4340	0.3195	0.4390	0.2907
LSTM	0.3732	0.2583	0.3682	0.2412	0.2793	0.1549	0.4990	0.3761	0.7169	0.7627
CNN_LSTM	0.3435	0.2161	0.3554	0.2262	0.2507	0.1324	0.5158	0.3793	0.4653	0.3285
Informer	0.3577	0.2213	0.3118	0.1989	0.2866	0.1715	0.3825	0.2563	0.4121	0.2769
T-GCN	0.6928	0.7063	0.6207	0.5270	0.4689	0.3902	0.5195	0.4053	0.9521	1.2220
GLFN-TC	0.2990	0.1852	0.3036	0.1631	0.2406	0.1187	0.3494	0.2167	0.3902	0.2432
Ours	**0.2498**	**0.1503**	**0.2794**	**0.1443**	**0.1318**	**0.0968**	**0.3147**	**0.1972**	**0.3371**	**0.2197**

TABLE 2. GENERALIZATION STUDY

Dataset / Model	TN		PL	
	MAE	MSE	MAE	MSE
SA	0.7803	0.9736	0.7119	0.8035
NF	2.2432	5.9534	0.8204	1.0993
MA	2.5027	7.2194	0.9163	1.3199
RNN	0.3296	0.2065	0.3532	0.2280
CNN	0.3606	0.2539	0.3879	0.2562
LSTM	0.3732	0.2583	0.3682	0.2412
CNN_LSTM	0.3435	0.2161	0.3554	0.2262
Informer	0.3577	0.2213	0.3118	0.1989
T-GCN	0.6928	0.7063	0.6207	0.5270
GLFN-TC	0.2990	0.1852	0.3036	0.1631
Ours	**0.2498**	**0.1503**	**0.2794**	**0.1443**

TABLE 3. ABLATION STUDY

Dataset / Model	SH		AP		AT		ISO-NE		NCENT	
	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE
Ours	0.2498	0.1503	0.2794	0.1443	0.1318	0.0968	0.3147	0.1972	0.3371	0.2197
Ours-merge	0.2691	0.1631	0.2902	0.1525	0.1597	0.1064	0.3365	0.2298	0.3454	0.2356
Ours-sg	0.3041	0.2063	0.3347	0.1940	0.1692	0.1237	0.3789	0.2495	0.4122	0.2694
Ours-tc	0.2936	0.1892	0.3215	0.1764	0.1543	0.1135	0.3598	0.2257	0.3858	0.2509

Table 1 demonstrates the experimental results of our proposed model with the baseline method in the short-term load forecasting task. Overall, our model achieves optimal performance on all five datasets, proving its effectiveness and reliability in short-term load forecasting. To verify the general applicability of our model, we conducted prediction experiments on the following datasets, as shown in Table 2, and our model outperforms the baseline approach on both TN and PL datasets. To validate the contribution of the key design in our model. We eliminated different components and named them:

Ours-merge leaves each layer of the feature pyramid unmerged and passes it into subsequent modules separately.

Ours-sg no longer performs dynamic learning of graph structures, but instead each layer is a post-learned two-layer graph convolution operation with a fixed graph structure.

Ours-tc has no temporal convolution module, the resulting features are directly spliced and then passed into the final output.

Table 3 shows the results of the ablation experiments. The experiments show that the fusion of features at each scale of the feature pyramid improves the results of the five datasets in a small way. The full model (Ours) performs best on all datasets, indicating that the synergy of the modules can significantly improve the predictive performance of the model. Specifically, after removing the merge module, the model performance decreases slightly, especially on the SH, AP, ISO-NE, and NCENT datasets, demonstrating the critical importance of feature fusion in capturing multi-scale features. The removal of the sg module results in a significant decrease in performance, especially on the ISO-NE and NCENT datasets, suggesting that dynamically learned graph structures are critical for accurately capturing complex dependencies between variables. Removal of the tc module results in a significant deterioration of the model's performance, especially on the AP, AT, ISO-NE, and NCENT datasets, suggesting that temporal convolution is critical for capturing temporal dependencies in time series.

IV. CONCLUSIONS

In this paper, we propose a novel prediction model based on the combination of multi-scale pyramid networks and graph learning modules to address the challenges in short-term power load forecasting. Through experimental validation on several public datasets, the model is demonstrated to outperform both traditional methods and some of the latest deep learning models in terms of accuracy and generalization ability. In particular, the multi-scale temporal pyramid network proposed in this paper is able to effectively capture the multi-scale temporal features in the load data, while the graph learning module further improves the prediction accuracy by capturing the spatio-temporal relationships among the load data.

REFERENCES

[1] Lin, Y., Luo, H., Wang, D., Guo, H., & Zhu, K. (2017). An ensemble model based on machine learning methods and data preprocessing for short-term electric load forecasting. Energies, 10(8), 1186.

[2] Reddy, M.D., & Vishali, N. (2017). Load forecasting using linear regression analysis in time series model for RGUKT, R.K. Valley Campus HT feeder. International Journal of Engineering Research and, 6.

[3] Nalcaci, G., Özmen, A., & Weber, G.W. (2019). Long-term load forecasting: Models based on MARS, ANN and LR methods. Central European Journal of Operations Research, 27, 1033-1049.

[4] Huang, R., Zhu, L., Gao, F., & Zhang, W. (2022). Short-term power load forecasting method based on variational modal decomposition for convolutional long-short-term memory network. Modern Electric Power, 10.

[5] Ding, Y., Zhu, Y., Feng, J., Zhang, P., & Cheng, Z. (2020). Interpretable spatio-temporal attention LSTM model for flood forecasting. Neurocomputing, 403, 348-359.

[6] He, F., Zhou, J., Feng, Z.K., Liu, G., & Yang, Y. (2019). A hybrid short-term load forecasting model based on variational mode decomposition and long short-term memory networks considering relevant factors with Bayesian optimization algorithm. Applied Energy, 237, 103-116.

[7] Stefenon, S.F., Seman, L.O., Mariani, V.C., & Coelho, L.D.S. (2023). Aggregating Prophet and seasonal trend decomposition for time series forecasting of Italian electricity spot prices. Energies, 16(3), 1371.

[8] Mamun, A.A., Sohel, M., Mohammad, N., Haque Sunny, M.S., Dipta, D.R., & Hossain, E. (2020). A comprehensive review of the load forecasting techniques using single and hybrid predictive models. IEEE Access, 8, 134911-134939.

[9] Saini, U., Kumar, R., Jain, V., & Krishnajith, M.U. (2020). Univariant time series forecasting of agriculture load by using LSTM and GRU RNNs. In 2020 IEEE Students Conference on Engineering & Systems (SCES), Prayagraj, India, 1-6.

[10] Alhussein, M., Aurangzeb, K., & Haider, S.I. (2020). Hybrid CNN-LSTM model for short-term individual household load forecasting. IEEE Access, 8, 180544-180557.

[11] Tang, X., Chen, H., Xiang, W., Yang, J., & Zou, M. (2022). Short-term load forecasting using channel and temporal attention based temporal convolutional network. Electric Power Systems Research, 205, 107761.

[12] Zhang, G., Wei, C., Jing, C., & Wang, Y. (2022). Short-term electrical load forecasting based on time augmented transformer. International Journal of Computational Intelligence Systems, 15(1), 67.

[13] Chen, L., Chen, D., Shang, Z., Wu, B., Zheng, C., Wen, B., & Zhang, W. (2023). Multi-scale adaptive graph neural network for multivariate time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 35(10), 10748-10761.

[14] Chen, L., Chen, D., Shang, Z., Wu, B., Zheng, C., Wen, B., & Zhang, W. (2023). Multi-scale adaptive graph neural network for multivariate time series forecasting. IEEE Transactions on Knowledge and Data Engineering, 35(10), 10748-10761.