



Graph Neural Network-Based Short-Term Load Forecasting with Temporal Convolution

Chenchen Sun¹ · Yan Ning¹ · Derong Shen² · Tiezheng Nie²

Received: 17 March 2023 / Revised: 12 October 2023 / Accepted: 17 October 2023 / Published online: 20 November 2023
© The Author(s) 2023

Abstract

An accurate short-term load forecasting plays an important role in modern power system's operation and economic development. However, short-term load forecasting is affected by multiple factors, and due to the complexity of the relationships between factors, the graph structure in this task is unknown. On the other hand, existing methods do not fully aggregating data information through the inherent relationships between various factors. In this paper, we propose a short-term load forecasting framework based on graph neural networks and dilated 1D-CNN, called GLFN-TC. GLFN-TC uses the graph learning module to automatically learn the relationships between variables to solve problem with unknown graph structure. GLFN-TC effectively handles temporal and spatial dependencies through two modules. In temporal convolution module, GLFN-TC uses dilated 1D-CNN to extract temporal dependencies from historical data of each node. In densely connected residual convolution module, in order to ensure that data information is not lost, GLFN-TC uses the graph convolution of densely connected residual to make full use of the data information of each graph convolution layer. Finally, the predicted values are obtained through the load forecasting module. We conducted five studies to verify the outperformance of GLFN-TC. In short-term load forecasting, using MSE as an example, the experimental results of GLFN-TC decreased by 0.0396, 0.0137, 0.0358, 0.0213 and 0.0337 compared to the optimal baseline method on ISO-NE, AT, AP, SH and NCENT datasets, respectively. Results show that GLFN-TC can achieve higher prediction accuracy than the existing common methods.

Keyword Short-term load forecasting · Graph structure learning · Graph neural networks · Dilated 1D-CNN · Temporal dependencies · Spatial dependencies

1 Introduction

With the development of society, the power demand is constantly expanding [1]. As load forecasting is the basis of power dispatching [2], it has attracted extensive attention in power system [3]. Load forecasting is classified into

short-term, medium-term and long-term forecasting. Short-term load forecasting (STLF) deals with forecasts targeting few hours to few days ahead. Medium-term load forecasting (MTLF) deals with load prediction from few weeks to few months. Long-term load forecasting (LTLF) is used to load forecast from one year to several years. LTLF assists in future planning of power system. MTLF aids in power system to inspect and maintain equipment. STLF plays an important role in load dispatching [4]. Since STLF plays an important role in power distribution and reduction in generation cost [5], STLF plays a crucial role in the operation of power system. Accurate load forecasting helps to maximize resource utilization and save a lot of money. Therefore, it is necessary to develop accurate load forecasting methods [6, 7].

Now, there are many methods applied to load forecasting. Mbamalu et al. and Chen et al. tackled the problem of load forecasting by using the statistical-based methods [8, 9]. However, there are various factors that can affect load

✉ Chenchen Sun
suncc_db@163.com

Yan Ning
ny_student@163.com

Derong Shen
shendr@mail.neu.edu.cn

Tiezheng Nie
nietiezheng@mail.neu.edu.cn

¹ School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China

² School of Computer Science and Engineering, Northeastern University, Shenyang, China

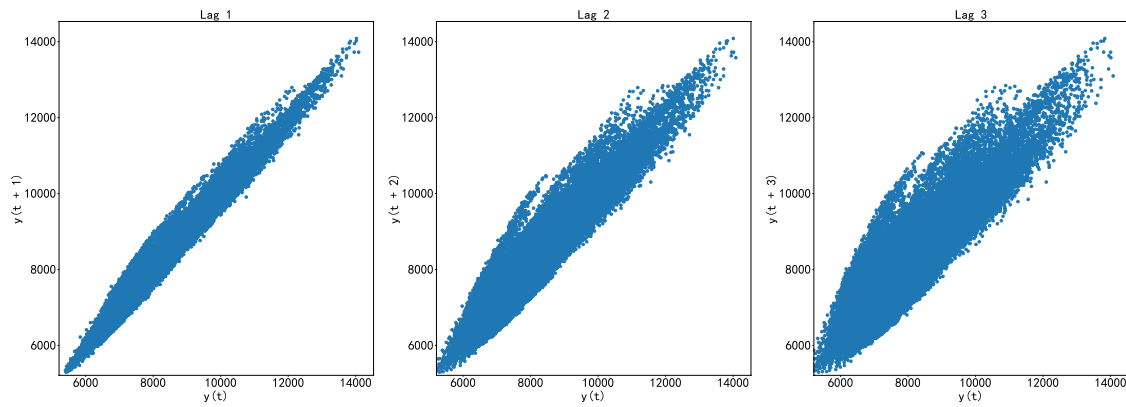


Fig. 1 Lag plot of “load” in AP dataset

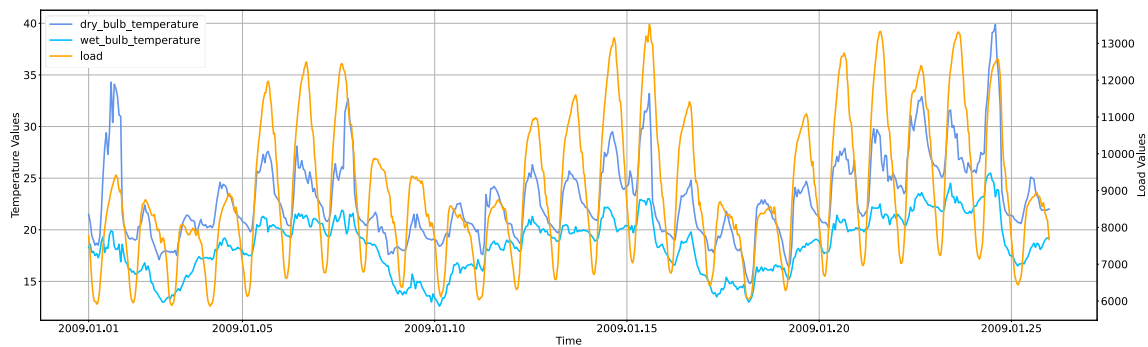


Fig. 2 Sample time series of “load” and other factors in AP dataset

demand. Therefore, the load curves are highly nonlinear and difficult to approximate. In recent years, deep learning methods are widely used in load forecasting due to its property of nonlinearity modeling [10]. For instance, Muzaffar et al. used LSTM network for STLF and its forecasting performance was compared to that of traditional methods [11]. Smyl et al. proposed a new gated recurrent cell that implements an attention mechanism for weighting the input information, and this mechanism can improve the predictive performance of the model [12]. Due to the nonlinearity characteristics of the load curve, the deep learning-based methods are more accurate than the statistical-based methods. Figure 1 shows the scatter plot with Lag 1 ~ 3 of “load” in AP dataset, and it can be seen from the figure that there is autocorrelation in “load,” especially with a strong correlation when the Lag 1. Therefore, the load values are affected by its own historical data, and we need to explore the temporal dependencies behind historical data. Most of the current works use RNN or LSTM to obtain the temporal dependencies of historical data. However, the historical data of each factor is highly related to itself. Therefore, it is necessary to explore the relationships between time steps from the historical data

of each factor in order to better represent the temporal dependencies of this factor.

Figure 2 shows the change trend of “dry_bulb_temperature,” “wet_bulb_temperature” and “load” in AP dataset over a period of time. From this figure, it can be seen that not only does the change of “load” increase with the increase in “dry_bulb_temperature,” and decreases with the decrease in “dry_bulb_temperature,” but also the change trend of “wet_bulb_temperature” is similar to that of “dry_bulb_temperature.” Therefore, the load values are affected not only by its own historical data, but also by other factors, and different factors also interact with each other. In order to improve the prediction accuracy, it is necessary to aggregate data information through the internal relationships between various factors. With the development of deep learning technology, graph neural network (GNN) has been proved to be effective in mining spatial dependencies between variables through graph structure. For example, Yu et al. and Wu et al. have achieved remarkable success in the field of traffic prediction using GNNs [13, 14]. However, deep graph convolutional networks may have the drawback of reducing performance [15], and after stacking multiple layers of graph convolution layers, the features between nodes become too smooth and

indistinguishable, which can also lead to the degradation of network results. The central node may pass the features to most nodes in the entire network through one or two layers of graph convolution, while for edge nodes in the network, multiple propagation is required to affect some nodes in the network [16]. Simple application of residual connection cannot solve this problem, as different nodes propagate features at different speeds. Therefore, in load forecasting tasks, the traditional methods based on graph convolution may cause over-smoothing problem when stacking multiple graph convolution layers. Different nodes require different degrees of neighborhood aggregation information, and the subsequent convolution layers may not be able to effectively utilize the output data of previous convolution layers after passing through multiple convolution layers, resulting in the problem of information loss. Moreover, GNN-based have been rarely studied for STLF problems.

To sum up, the current short-term load forecasting faces the following challenges:

- *Challenge 1 Unknown Graph Structure.* The load values are affected by various factors, and the relationships between these factors are very complex. Therefore, there is no explicit graph structure in most cases. The relationships between variables have to be discovered from data rather than being provided as ground truth knowledge.
- *Challenge 2 Mining temporal dependencies.* Temporal dependencies are one of the important data characteristics of load forecasting task. Therefore, in order to fully explore the relationships between time steps of each factor, it is necessary to obtain temporal dependencies from the historical data of each factor.
- *Challenge 3 GNN Learning.* The current common short-term load forecasting methods (such as LSTM-based methods and CNN-based methods) did not effectively use the relationships between central node and its neighbors for message passing. In addition, the traditional GCN-based models may reduce the utilization of the data information of previous graph convolution layers due to the stacking of graph convolution layers.

Therefore, in this work, we propose a graph neural network-based short-term load forecasting with temporal convolution (GLFN-TC), which learns a graph of relationships between load and weather, temperature and other factors, and use this graph structure to STLF. For challenge 1, the graph learning module uses feature vector to automatically learn the relationships between various factors. This module solves the problem that there is no graph structure in load forecasting. For challenge 2, the temporal dependencies behind the historical data of each factor are obtained through the temporal convolution module. Specifically, we use dilated 1D-CNN to fully explore

the relationship between time steps of each factor. For challenge 3, the densely connected residual convolution module adds a densely connected residual structure to the graph convolution. This module obtains spatial dependencies by stacking multiple graph convolution layers, while ensuring that data information will not be lost.

In summary, the proposed model solves the existing problems of short-term load forecasting, and can also be applied to other non-load fields due to the characteristics of modeling. The main contributions of our work are:

- We propose a new deep learning model for load forecasting. This model works from the data information of each node in historical data. We propose the temporal convolution module that can fully explore the deep data information behind historical data and work in conjunction with the densely connected residual convolution module to effectively handle temporal and spatial dependencies. The module effectiveness assessment further proves the effectiveness of both modules.
- We propose the graph learning module to learn complex relationships between variables. Our method effectively solves the problem of not being able to provide an explicit graph structure in the field of load forecasting. The experiment shows that the graph structure of automatic learning has good interpretability.
- We propose a graph convolution operation with densely connected residual structure, which can fully utilize the data information of each graph convolution layer to handle spatial dependencies and effectively solve the problem of information loss during message passing. The ablation study further proves the effectiveness of this design.
- GLFN-TC supports automatic construction of graph structure between variables with complex relationships. Therefore, the model is general for other fields without predefined graph structure. The model generality assessment further proves the generality of GLFN-TC.
- We conduct experiments on five load datasets and three non-load datasets. Experimental results show that our method outperforms the baseline methods on all datasets.

Organization of the rest. Section 2 defines the short-term load forecasting problem and presents the theory basis and implementation details of GLFN-TC. Section 3 describes the datasets, baseline methods, evaluation metrics and experimental setup. In this section, five studies are conducted to validate the proposed method. Finally, the advantages and disadvantages of the proposed model are discussed based on the experimental results, and future research directions for the model are provided. Section 4 introduces the basic knowledge of load forecasting and

graph neural networks, and reviews their related work. Finally, conclusions are drawn in Sect. 5.

2 Short-Term Load Forecasting Based on Graph Neural Network with Temporal Convolution

2.1 Overview

The aim of STLFL is to learn a model Q that maps X to \hat{Y} , that is

$$X \xrightarrow{Q} \hat{Y} \quad (1)$$

where $X = \{\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \dots, \mathbf{x}_{t_e}\}$ denotes a historical data with time step of e . Let $\mathbf{x}_{t_i} \in \mathbb{R}^N$ denotes the value at time step t_i , and N represents the number of variables. The variables can be temperature, humidity, load, etc. $\hat{Y} \in \mathbb{R}^p$ denotes the predict future load values with time step of p , that is $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_p\}$. The known values of input characteristics and corresponding load values are called training data, and model constantly updates parameters through training data. The model can be used subsequently to predict future load values. The new input feature values are also called test data. The prediction process is as follows:

$$\hat{Y}' = Q(X') \quad (2)$$

$$\Delta d = \arg \min |\hat{Y}' - Y'| \quad (3)$$

where $X' = \{\mathbf{x}_{t_{e+1}}, \mathbf{x}_{t_{e+2}}, \dots, \mathbf{x}_{t_{2e}}\}$ denotes test data. Our aim is to make the predicted load values $\hat{Y}' = \{\hat{y}_{p+1}, \hat{y}_{p+2}, \dots, \hat{y}_{2p}\}$ of the test data closer to the real values $Y' = \{y_{p+1}, y_{p+2}, \dots, y_{2p}\}$ through the trained model, that is, Δd is minimized by Eq. (3). When $\Delta d = 0$, it means perfect prediction.

As mentioned in introduction, load forecasting is not only influenced by data between different time steps, but also by other factors. In order to improve prediction accuracy, our method explores the relationships between historical data at different time steps of each node through temporal convolution module and aggregates data information between different nodes through densely connected residual convolution module. Therefore, our method effectively handles temporal and spatial dependencies to accurately predict future load values. Figure 3 provides an overview of GLFN-TC, and it consists of four main components:

- *Graph Learning Module* This module creates a feature vector for each node, and uses the feature vector to learn

a graph structure to express the relationships between nodes.

- *Temporal Convolution Module* This module uses dilated 1D-CNN to obtain the temporal dependencies of each node's historical data.
- *Densely Connected Residual Convolution Module* This module uses graph convolution operation with densely connected residual structure to aggregate the data information of the central node and its neighbors.
- *Load Forecasting Module* This module fuses data information with feature vector to predict future load values.

When the input data is $X \in \mathbb{R}^{N \times T_{in}}$, GLFN-TC first creates N feature vectors through the graph learning module, and automatically learns the graph structure $A \in \mathbb{R}^{N \times N}$ between variables through feature vectors. Then, a dilated 1D-CNN is created for each node through the temporal convolution module to obtain temporal dependencies behind the historical data of each node. The output of the temporal convolution module and graph structure are used as the input of the densely connected residual convolution module, and the data information of nodes are aggregated through the graph convolution operation of densely connected residual structure in this module. Finally, the load forecasting module integrates the data information and predict future load values $\hat{Y} \in \mathbb{R}^{T_{out}}$. Our method not only focuses on message passing, but also solves the problem of no predefined graph structure through graph learning module, and optimizes the graph structure in training stage. All core modules will be described in more detail in the following sections.

2.2 Graph Learning Module

In STLFL, different factors can have very different characteristics. Figure 4 shows the change trend of "temperature" and "hour" over a period of time in ISO-NE dataset, which visually shows that the change trend of different factors is significantly different. But each factor has complex correlation with other factors. For example, to some extent, weekends can reflect the user's behavior (less power is used by families on weekdays and more on weekends). Therefore, we need to capture the complex relationships behind different factors to construct a graph structure.

We create a feature vector for each factor:

$$\mathbf{v}_i \in \mathbb{R}^d, \text{ for } i = 1, 2, \dots, N \quad (4)$$

where N represents the number of factors. d is a hyperparameter and represents the dimension of feature vector. These feature vectors are randomly initialized and trained together with the model. We assign a feature vector to each factor, and use this feature vector to represent the characteristics of the factor. We use a directed graph to represent the

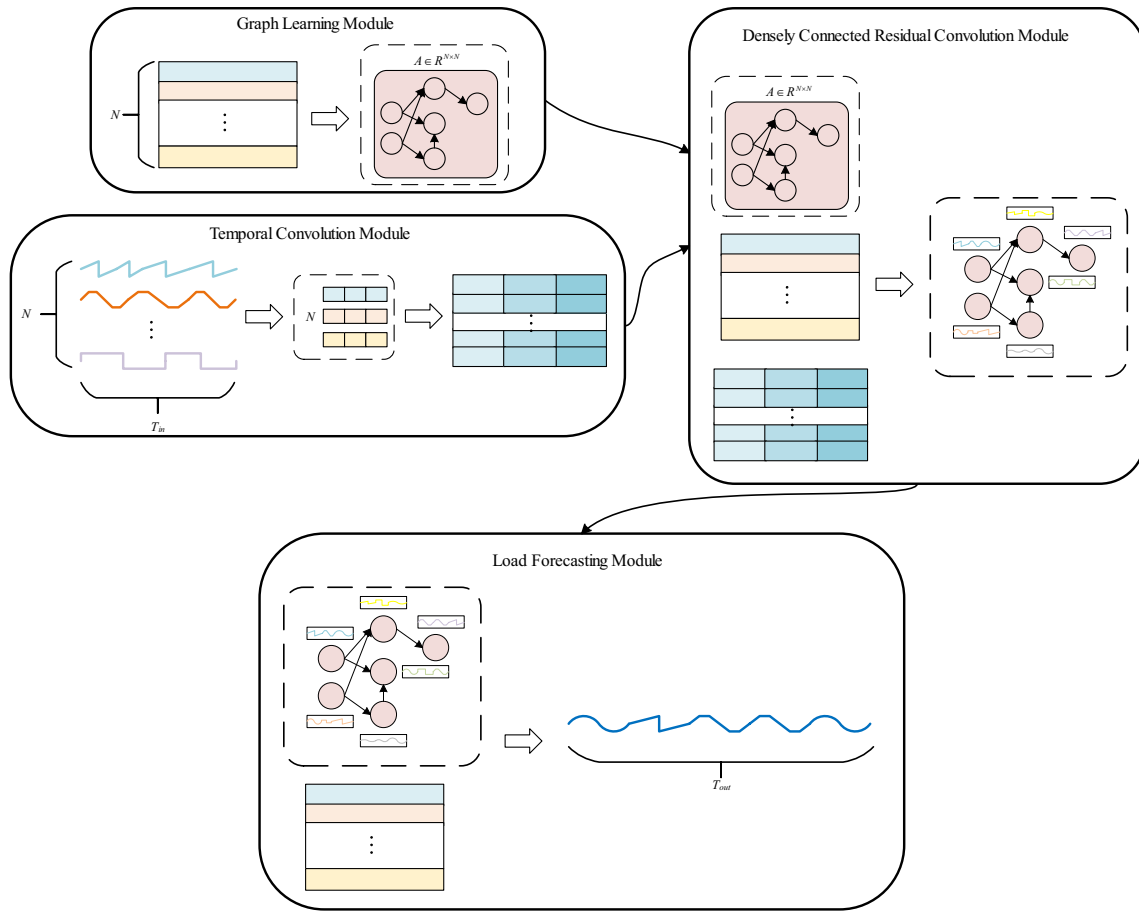


Fig. 3 Framework of graph neural network-based short-term load forecasting with temporal convolution

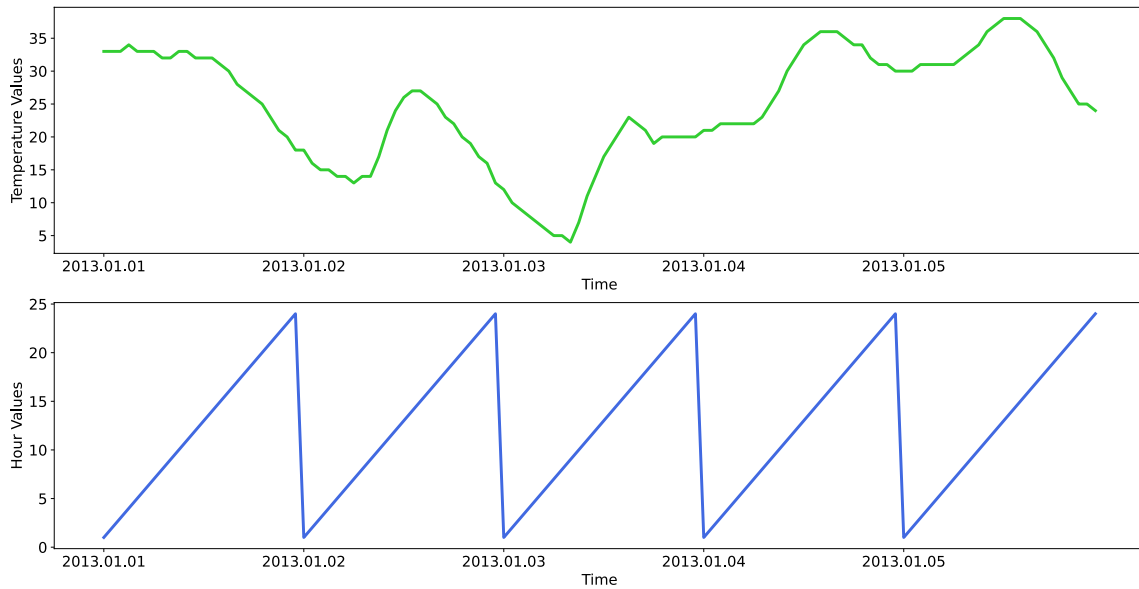


Fig. 4 Sample time series of “temperature” and “hour” in ISO-NE dataset

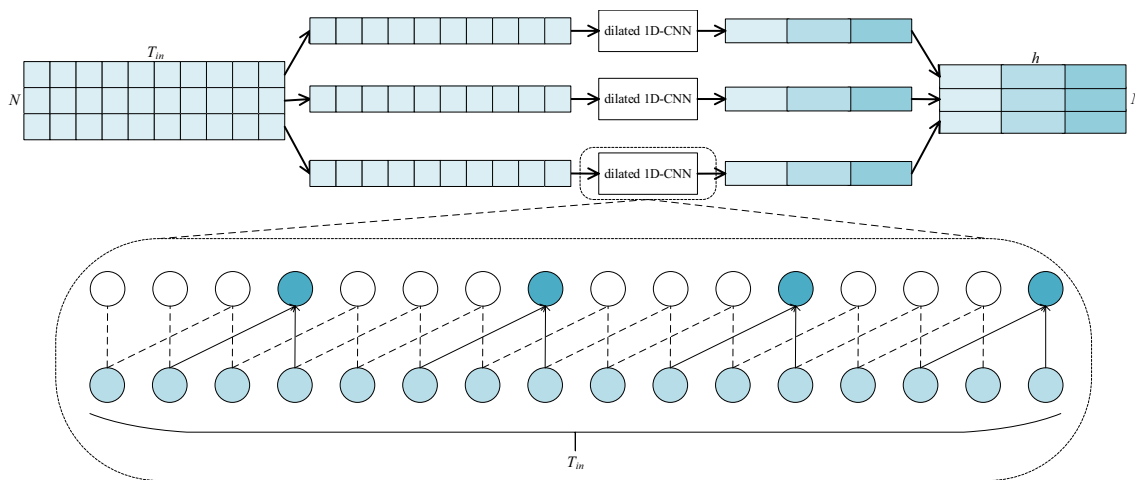


Fig. 5 Architecture of temporal convolution module

relationships between various factors. The nodes of directed graph represent factors, and the edges of directed graph represent the relationships between various nodes. We use a directed graph because the relationships between factors may be asymmetric. For example, when the temperature is high, it may be sunny weather, but in winter, the temperature may not be high on sunny days.

We use an adjacency matrix A to represent this directed graph, each node in A corresponds to each factor. A is calculated as follows:

$$A_{ij} = \begin{cases} 1, & \text{if } i = j \\ \sigma(\mathbf{v}_i^T \mathbf{W} \mathbf{v}_j), & \text{if } i \neq j \end{cases} \quad (5)$$

where $\sigma(\cdot)$ is the activation function and $\mathbf{W} \in \mathbb{R}^{d \times d}$ is a trainable weight matrix. We use Eq. (5) to calculate the correlation, and use activation function to control the correlation in the range [0,1]. The correlation between nodes is calculated by corresponding feature vector and trainable weight matrix. Therefore, in training phase, our adjacency matrix is adaptable to change as training data updates the model parameters, so that the graph structure can reach the optimal state.

2.3 Temporal Convolution Module

LSTM can handle the temporal dependencies of historical data through gating mechanism, but it cannot handle very long sequences well. However, STL task requires the model to be able to explore the relationships between the time steps of each node, in order to obtain data that better represents temporal dependencies. Reference [17] used convolutional networks in sequence modeling and achieved good results, and believed that convolutional networks should be emphasized in sequence modeling.

Therefore, in order to effectively handle the temporal dependencies behind the historical data of each node, and cooperate with the densely connected residual convolution module to improve prediction accuracy, we propose a temporal convolution module.

Temporal convolution module aims to obtain the temporal dependencies of each node's input data. Specifically, we use a dilated 1D-CNN for each node to extract the temporal dependencies behind the historical data of corresponding node. Capture the relationships between different time steps through 1D-CNN and use dilation strategy to increase the receptive field to obtain long-term dependencies. The architecture of temporal convolution module is shown in Fig. 5.

Assuming that the input data have N nodes and time step is T_{in} , we use dilated 1D-CNN with the number of N to perform the convolution operation on each node. Firstly, the temporal convolution module divides the input data $\mathbf{X} \in \mathbb{R}^{N \times T_{in}}$ into N one-dimensional data vectors:

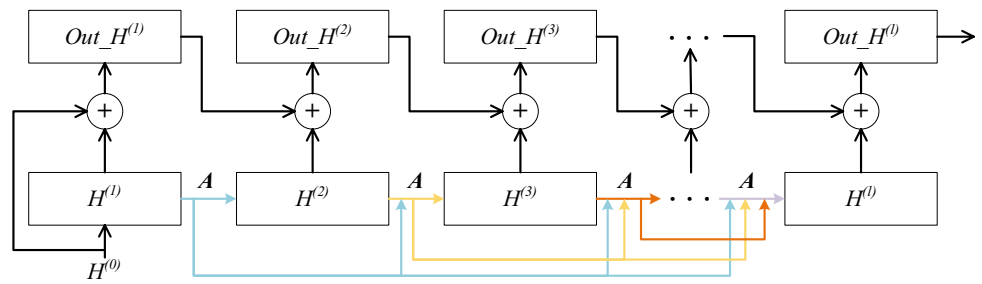
$$\mathbf{z}_i \in \mathbb{R}^{T_{in}}, \text{ for } i = 1, 2, \dots, N \quad (6)$$

where \mathbf{z}_i represents the input data of node i . The convolution layer of dilated 1D-CNN is used to extract the temporal dependencies of each node's data information. The dilation factor can enable the convolution operation to obtain a larger receptive field and let each convolution output contain a wide range of information. The dilated 1D-CNN operation of node i denoted by $\mathbf{z}_i \star \mathbf{f}^{(i)}$ is defined as:

$$\mathbf{z}_i \star \mathbf{f}^{(i)}(t) = \sum_{s=0}^{k-1} \mathbf{f}^{(i)}(s) \mathbf{z}_i(t - df \times s) \quad (7)$$

where $\mathbf{f}^{(i)} \in \mathbb{R}^k$ represents the one-dimensional kernel of node i . df is the dilation factor. It is very important to choose the proper kernel size for convolutional networks. The kernel size is too large to effectively obtain short-term data

Fig. 6 Architecture of densely connected residual convolution module



information. The kernel size is too small to effectively obtain long-term data information. Therefore, we choose a kernel size is 7 [18], i.e., $k = 7$. By fine-tuning the dilation factor, we choose the dilation factor is 3, i.e., $df = 3$. The specific analysis is introduced in Sect. 3.7.

Extract the data features of the one-dimensional historical data vector corresponding to each node by dilated 1D-CNN to effectively handle the temporal dependencies behind the input data of the node. Then, we concatenate the data information of all nodes to obtain the output data of the temporal convolution module. The specific operations are as follows:

$$\mathbf{o} = \text{concat}(\mathbf{o}_1, \mathbf{o}_2, \dots, \mathbf{o}_N) \tag{8}$$

where $\text{concat}(\cdot)$ denote splicing the data information of all nodes, and $\mathbf{o}_i \in \mathbb{R}^h$ represents the data features extracted from the historical data of node i . $\mathbf{o} \in \mathbb{R}^{N \times h}$ represents the final output data of the module and h represents the hidden layer dimension. Through the temporal convolution module, we effectively deal with the temporal dependencies behind the historical data of each node.

2.4 Densely Connected Residual Convolution Module

As mentioned in introduction, graph neural networks can handle spatial dependencies. However, when stacking multiple graph convolution layers, the utilization of data information in previous graph convolution layers will be reduced, leading to the problem of information loss in message passing. Therefore, in order to solve the above problems, we propose a densely connected residual convolution module.

The purpose of densely connected residual convolution module is to make full use of the output information of each graph convolution layer. We add the densely connected structure to the graph convolution operation, and add the residual structure to the output of each graph convolution layer. By graph convolution operation with densely connected residual structure, we fully utilize the data information of each graph convolution layer. Figure 6 shows the architecture of densely connected residual convolution module.

In densely connected residual convolution module, we use the adjacency matrix \mathbf{A} learned in graph learning module to fuse a central node’s information with its neighbors’ information to handle spatial dependencies. We use densely connected structure in graph convolution operation and add residual structure to the output of graph convolution layer. We first provide the input data of the module and the mathematical forms of two structures, and then illustrate the module’s operation process. The specific definition is as follows:

$$\mathbf{g}_i = \mathbf{o}_i \oplus \mathbf{v}_i, \text{ for } i = 1, 2, \dots, N \tag{9}$$

$$\mathbf{H}^{(l)} = \tilde{\mathbf{A}} \sigma \left(\sum_{i=1}^{l-1} \mathbf{H}^{(i)} \right), \text{ for } l = 2, 3, \dots \tag{10}$$

$$\text{Out_H}^{(l)} = \sigma(\text{Out_H}^{(l-1)} + \mathbf{H}^{(l)}), \text{ for } l = 1, 2, \dots \tag{11}$$

where \oplus denotes connection operation, $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1} \mathbf{A}$, $\tilde{\mathbf{D}}_{ii} = \sum_j \mathbf{A}_{ij}$, l is the layer of graph convolution and $\sigma(\cdot)$ represents the activation function. In Eq. (9), $\mathbf{o}_i \in \mathbb{R}^h$ denotes the data representation of node i , and it is obtained from \mathbf{o} . N represents the number of nodes. We combine the data output by temporal convolution module with \mathbf{v}_i according to the corresponding nodes, and use it as the input data of densely connected residual convolution module. The purpose of combine operation to make the input data contain the data information of feature vector. Get the input data $\mathbf{g}_i \in \mathbb{R}^{(h+d)}$ of node i of densely connected residual convolution module from Eq. (9), then the input data of this module can be represented as $\mathbf{g} \in \mathbb{R}^{N \times (h+d)}$. Equation (10) represents the densely connected structure. The graph convolution uses densely connected structure, and the input of the l -th layer is the addition of outputs from the 1-th layer to the $l - 1$ -th layer of graph convolution step. This operation enables the input of the l -th layer to include the output information of all previous layers. Because we use the densely connected structure in graph convolution operation, the original input data does not participate in subsequent operations, that is, l starts from 2. When $l = 1$, $\mathbf{H}^{(1)} = \mathbf{A}\mathbf{H}^{(0)}$, and $\mathbf{H}^{(0)} = \mathbf{g}$. This structure can effectively use the output information of previous graph convolution layers during message passing. Therefore, it can strengthen the propagation of node information.

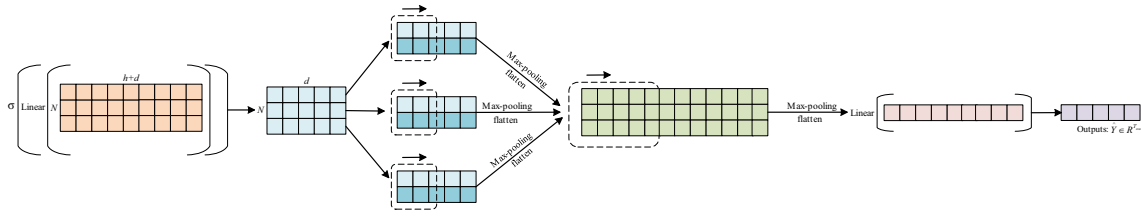


Fig. 7 Architecture of load forecasting module

Equation (11) represents the residual structure, where $\mathbf{H}^{(l)}$ is the output of graph convolution operation at l -th layer and $\text{Out_H}^{(0)} = \mathbf{H}^{(0)}$. The purpose of this structure is to retain the data information of the previous layer when outputting data. Therefore, when $l = 1$, we also add the residual structure to retain the information of original input data, that is, l starts from 1. This structure retains the data information of previous layer, which can enhance the nonlinear approximation ability of model, and avoid the information loss problem caused by the densely connected residual convolution module stacking graph convolution layers.

2.5 Load Forecasting Module

From the above temporal convolution module and densely connected residual convolution module, we obtain the data representation of N nodes. Now, the data representation of each node has fully explored the temporal and spatial dependencies of historical data. At this time, we need to predict future load values through the data information of nodes together with the corresponding feature vector. Figure 7 shows the architecture of load forecasting module.

The specific operation process of this module is as follows:

$$\mathbf{r} = \sigma(\text{Linear}(\text{Out_H}^{(l)})) \quad (12)$$

$$\mathbf{E}_1 = \text{concat}(\varphi(r_1 \otimes v_1), \varphi(r_2 \otimes v_2), \dots, \varphi(r_N \otimes v_N)) \quad (13)$$

$$\mathbf{E}_2 = \varphi(\mathbf{E}_1) \quad (14)$$

$$\varphi(\cdot) = \text{flatten}(\text{Maxpooling}(\sigma(\text{Conv}(\cdot)))) \quad (15)$$

$$\hat{\mathbf{Y}} = \text{Linear}(\mathbf{E}_2) \quad (16)$$

In Eq. (12), $\text{Linear}(\cdot)$ represents a linear layer, $\text{Out_H}^{(l)}$ represents the output data of densely connected residual convolution module, and $\sigma(\cdot)$ is the activation function. Firstly, according to Eq. (12), we use a linear layer and activation function to change the dimension of the output data of the densely connected residual convolution module to d . The purpose of

this operation is to maintain consistency with the dimension of the feature vector to aggregate the data information and feature vector of each node. At this time, $\mathbf{r} \in \mathbb{R}^{N \times d}$ represents the changed data information. In Eq. (13), \otimes refers to the operation of splicing two one-dimensional data vectors into a two-dimensional data vector. According to Eq. (13), we splice the data representation $r_i \in \mathbb{R}^d$ of node i with its corresponding feature vector v_i . Then, the two-dimensional data vector is used as the input data for convolution operation and max-pooling operation. $\text{concat}(\cdot)$ denote splicing the data information of all nodes to obtain an N -dimensional data vector. In Eq. (15), $\text{Conv}(\cdot)$ represents convolution operation, and the purpose of this operation is to fuse data information. $\text{Maxpooling}(\cdot)$ represents max-pooling operation, and the purpose of this operation is to reduce the number of parameters to retain important features. $\sigma(\cdot)$ is the activation function, and its function is to introduce nonlinear characteristics into the operation. $\text{flatten}(\cdot)$ represents flatten operation. Through Eqs. (13) and (15), we aggregate the data representation and feature vector of each node, which is represented as $\mathbf{E}_1 \in \mathbb{R}^{N \times h_1}$. N represents the number of nodes, and h_1 represents the hidden layer dimension. Through Eqs. (14) and (15), we fuse the data information of all nodes. We obtain $\mathbf{E}_2 \in \mathbb{R}^{h_2}$ and use it as input data for subsequent operations. h_2 represents the hidden layer dimension. According to Eq. (16), we use a linear layer to predict T_{out} steps load values, i.e., the predict load values is $\hat{\mathbf{Y}} \in \mathbb{R}^{T_{\text{out}}}$.

The loss function defines the optimization objective of training. Specific definitions are as follows:

$$L_{MSE} = \frac{1}{T_{\text{train}}} \sum_{i=1}^{T_{\text{train}}} (Y_i - \hat{Y}_i)^2 \quad (17)$$

where T_{train} represents the number of training data. We use the Mean Square Error between the predicted output $\hat{\mathbf{Y}}$ and the real data \mathbf{Y} as the loss function for minimization.

3 Experiments

In this section, we evaluate the performance of the proposed GLFN-TC for STLF on five load datasets and the generality of GLFN-TC on three other field datasets. In particular, we aim to answer the following research questions:

- *RQ 1 (Comparison With Baseline Methods)* How does the proposed method perform compared with baseline methods for short-term load forecasting?
- *RQ 2 (Model Generality Assessment)* Is the proposed method also effective when applied for other fields?
- *RQ 3 (Ablation Study)* How do the various components of the proposed model affect the overall performance of the model?
- *RQ 4 (Module Effectiveness Assessment)* Do modules achieve the expected effect in our method?
- *RQ 5 (Repeatability Assessment And Parameter Sensitivity)* Is the result of multiple runs of our method stable? Is the dilation factor affect the results?

We first introduce the datasets, evaluation metrics, experimental setup and baseline methods. Then answer the above research questions through experiments. To answer RQ 1, we compare GLFN-TC with baseline methods in five load datasets. Specifically, in order to verify the nonlinear characteristics of GLFN-TC, we compare it with the conventional methods. In order to verify the prediction performance of GLFN-TC in deep learning methods, we compare it with other deep learning methods. To answer RQ 2, we validate the generality of GLFN-TC using three datasets from other fields. To answer RQ 3, in five load datasets, we eliminate some structures in GLFN-TC to verify the contribution of key components to GLFN-TC. To answer RQ 4, in five load datasets, we use the traditional method to replace the corresponding module to verify whether the temporal convolution module and the densely connected residual convolution module designed in GLFN-TC achieves our expected effect. At the same time, the effectiveness of the graph learning module is further verified through specific example. To answer RQ 5, we verify whether the experimental results of GLFN-TC running for many times are similar, that is, verify the stability of GLFN-TC. At the same time, we verify the influence of value of dilation factor on the experimental results. Finally, through the discussion section, we will discuss the advantages and disadvantages of the proposed model, and provide future research directions for the model to further improve prediction accuracy.

3.1 Datasets and Evaluation Metrics

We use ISO-NE dataset,¹ AT dataset,² AP dataset,³ SH dataset⁴ and NCENT dataset⁵ for short-term load forecasting. The details of five datasets are as follows:

¹ Available at <https://github.com/ningningLiningning/iso-ne>.

² Available at <https://github.com/Lizhuoling/DCN>.

³ Available at <https://zhuanlan.zhihu.com/p/150954853>.

⁴ Available at <https://github.com/Mark-THU/load-point-forecast>.

⁵ Available at <https://github.com/kmcclwee/load-forecasting>.

- *ISO-NE* ISO-NE dataset covers the data from March 2003 to December 2014, and the sample rate is 1 h. The number of nodes in this dataset is 7, including load, temperature, etc.
- *AT* AT dataset covers the data from January 2011 to December 2016, and the sample rate is 1 h. The number of nodes in this dataset is 6, including load, temperature, wind speed, wind direction, etc.
- *AP* AP dataset covers the data from January 2006 to December 2010, and the sample rate is 0.5 h. The number of nodes in this dataset is 7, including load, electricity price, humidity, etc.
- *SH* SH dataset covers the data from January 2017 to August 2020, and the sample rate is 1 h. The number of nodes in this dataset is 16, including load, week of year, day of week, etc.
- *NCENT* NCENT dataset covers the data from January 2002 to December 2018, and the sample rate is 1 h. The number of nodes in this dataset is 6, including load, year, etc.

Our subsequent experiments will be conducted on the above five datasets. The sample load data are shown in Fig. 8.

We use mean square error (MSE) and mean absolute error (MAE) to evaluate the error between real load values and predicted load values:

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

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (19)$$

where y_i and \hat{y}_i denotes real values and predicted values. We calculate MSE according to Eq. (18) and calculate MAE according to Eq. (19). For MSE and MAE, lower values are better.

The upper limit of MSE and MAE is $+\infty$, unless the maximum MSE and MAE values are provided, or unless the distribution of all the ground truth values are known, we cannot effectively evaluate the overall quality of model. We focus on one rate that actually generate a high score only if the majority of the elements of a ground truth group has been correctly predicted: R^2 . R^2 can have negative values, which mean that the model performed poorly. When $R^2 = 1$, it means perfect prediction. Therefore, high R^2 value can clearly indicate a good model performance, regardless of the ranges of the ground truth values and their distributions. Specifically, we use R^2 in Sect. 3.3 to further evaluate the prediction accuracy of model. R^2 is calculated as follows:

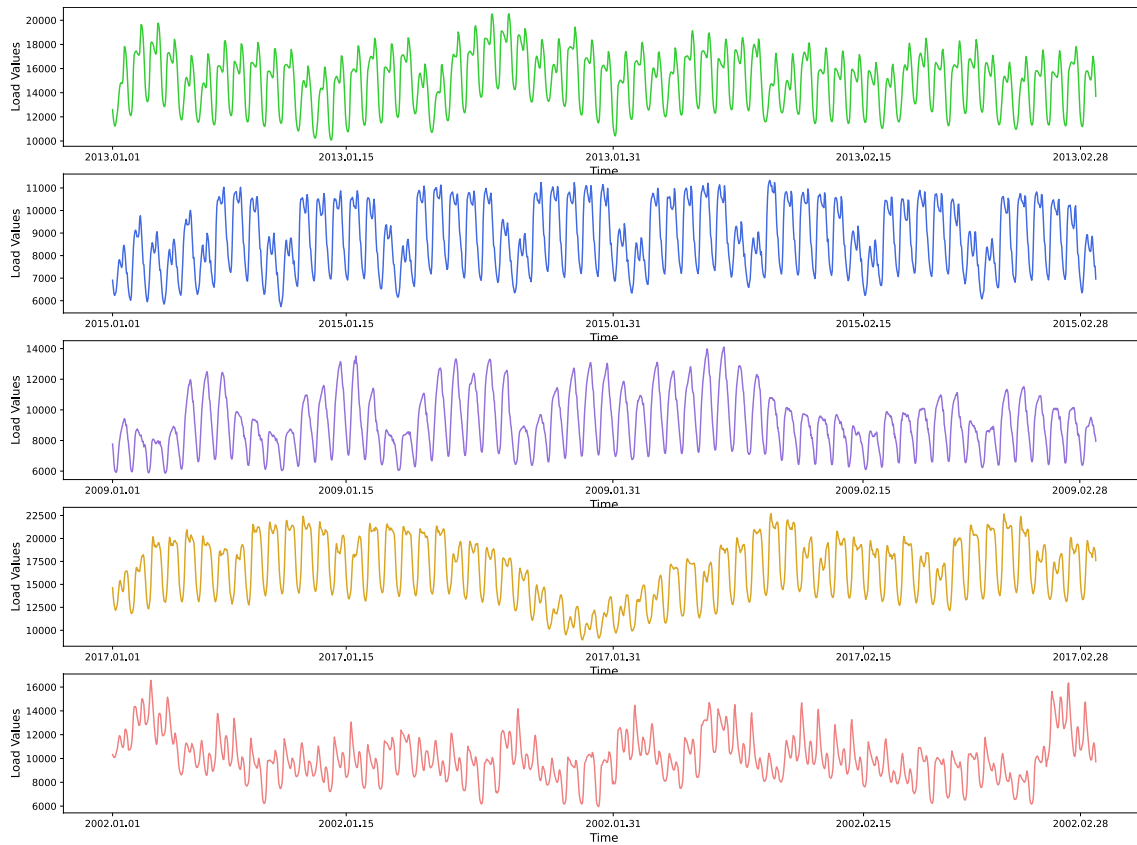


Fig. 8 Sample load data for ISO-NE dataset (green), AT dataset (blue), AP dataset (purple), SH dataset (yellow) and NCENT dataset (red)

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (20)$$

where y_i denotes real values, \hat{y}_i denotes predicted values, and \bar{y} is the average value of real values.

3.2 Experimental Setup

In ISO-NE dataset, we take the data from January 1, 2013 to December 31, 2014. In AT dataset, we take the data from January 1, 2015 to December 31, 2016. In AP dataset, we take the data from January 1, 2009 to December 31, 2010. In SH dataset, we take the data from January 1, 2017 to December 30, 2018. In NCENT dataset, we take the data from January 1, 2002 to December 31, 2003. We divided the extracted data into training set, validation set and test set according to 6:2:2. For five datasets, we set $T_{in} = 72$ and $T_{out} = 240$, that is, input 72 historical data to predict 240 future load values. The data normalization method is as follows:

$$X_y = \frac{X_x - \mu_x}{\sigma_x} \quad (21)$$

where X_x is the original data, μ_x is the average of the original data, σ_x is the standard deviation of the original data, and X_y is the normalized data.

We compare three conventional methods and six deep learning methods. The details are as follows:

- *NF* Naive Forecast, this method uses the load value of the last time step of the training data as the load values for future prediction.
- *SA* Simple Average, this method takes the average value of all load values in training data as the load values for future prediction.
- *MA* Moving Average, in this method, the load value of the current time step is the average of the load values of the previous n time steps. In the experiment, $n = 4$.
- *RNN* Recurrent neural network is a kind of neural network with short-term memory ability. In other words, the output of the network is not only related to the current input data, but also related to the previous input data.

Table 1 Comparison of MSE and MAE under short-term load forecasting for proposed method and conventional methods

Models	ISO-NE		AT		AP		SH		NCENT	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
NF	1.0025	0.8271	2.2295	1.2169	1.0993	0.8204	5.9534	2.2432	2.1699	1.3288
SA	0.8642	0.7304	1.0576	0.8735	0.8035	0.7119	0.9736	0.7803	0.8812	0.7300
MA	0.8139	0.7225	2.3775	1.2613	1.3199	0.9163	7.2194	2.5027	3.6731	1.7699
GLFN-TC	0.2167	0.3494	0.1187	0.2406	0.1631	0.3036	0.1852	0.2990	0.2432	0.3902

Bold represents the best result

- *CNN* Convolutional neural network can extract the characteristics of input data in load forecasting tasks. In experiments, we use 1D-CNN for STLFL.
- *LSTM* Long short-term memory network is a variant of RNN. Compared with RNN, it can effectively capture association between long sequences and alleviate the phenomenon of gradient vanishing or gradient exploding.
- *CNN_LSTM* It is the combination of 1D-CNN and LSTM, so that the model can have the characteristics of CNN and LSTM at the same time.
- *Informer* Informer is an improvement based on transformer. Informer uses ProbSparse self-attention, self-attention distilling and generative decoder to solve some problems when transformer is applied to LSTFL, such as high memory usage [19].
- *T-GCN* Temporal graph convolutional network model combines graph convolution network and gated recurrent unit to capture spatial and temporal dependencies simultaneously. Specifically, the former captures spatial dependencies, while the latter captures temporal dependencies [20].

To eliminate the randomness of model, we repeat all experiments 3 times and report the average as the final result. Due to the experimental setup of Informer and T-GCN, they perform multiple predictions on one data during the testing phase, but other models only perform one prediction operation. Therefore, in order to maintain consistency, we use the first prediction result of the data in Informer and T-GCN as the final prediction result of the model to evaluate its prediction performance. Due to the assumption in this paper that there is a correlation between each node, in T-GCN, we set the correlation between nodes to 1. The proposed model is trained by the Adam optimizer. We use the mean squared error as the loss function. The learning rate is 0.0001, and the Dropout is 0.2. The number of layers of graph convolution is 5. We train models for up to 50 epochs, and stop training if the loss of validation set does not decline for 10 consecutive times during the training period.

3.3 RQ 1: Comparison with Baseline Methods

GLFN-TC is essentially a deep learning model. In order to verify its nonlinear characteristics, in this subsection, we first compare our model with three conventional methods.

Table 1 shows the experimental results of the proposed method and conventional method. Taking MSE as an example, the MSE of NF on five datasets is 1.0025, 2.2295, 1.0993, 5.9534 and 2.1699, respectively. The MSE of SA on five datasets is 0.8642, 1.0576, 0.8035, 0.9736 and 0.8812, respectively. The MSE of MA on five datasets is 0.8139, 2.3775, 1.3199, 7.2194 and 3.6731, respectively. NF, SA and MA cannot effectively mine the nonlinear characteristics behind historical data. Therefore, the three methods cannot accurately predict the future load values. The MSE of GLFN-TC on five datasets is 0.2167, 0.1187, 0.1631, 0.1852 and 0.2432, respectively. The results show that GLFN-TC outperforms the NF, SA and MA. Therefore, GLFN-TC has a better nonlinear approximation capability than NF, SA and MA.

In Fig. 9, we show the R^2 of our proposed method and conventional methods on ISO-NE, AT, AP, SH and NCENT datasets. Because the conventional methods have no good nonlinear approximation ability, the result of R^2 is negative.

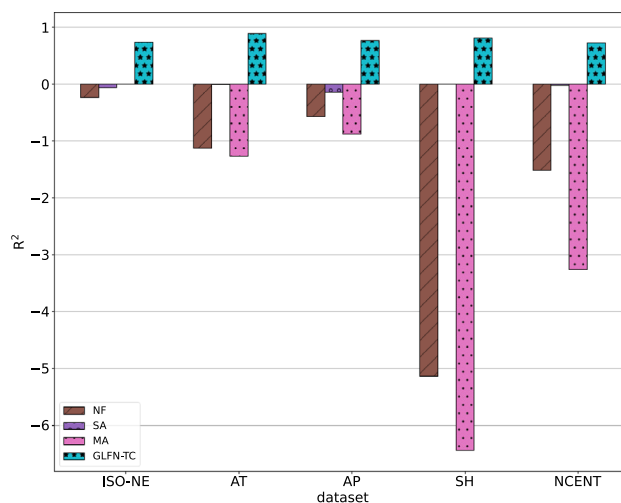


Fig. 9 Comparison of R^2 for proposed method and conventional methods

Table 2 Comparison of MSE and MAE under short-term load forecasting for proposed method and other deep learning methods

Models	ISO-NE		AT		AP		SH		NCENT	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
RNN	0.3643	0.4623	0.1541	0.2685	0.2280	0.3532	0.2065	0.3296	0.3926	0.5141
CNN	0.3195	0.4340	0.1377	0.2625	0.2562	0.3879	0.2539	0.3606	0.2907	0.4390
LSTM	0.3761	0.4990	0.1549	0.2793	0.2412	0.3682	0.2583	0.3732	0.7627	0.7169
CNN_LSTM	0.3793	0.5158	0.1324	0.2507	0.2262	0.3554	0.2161	0.3435	0.3285	0.4653
Informer	0.2563	0.3825	0.1715	0.2866	0.1989	0.3118	0.2213	0.3577	0.2769	0.4121
T-GCN	0.4053	0.5195	0.3902	0.4689	0.5270	0.6207	0.7063	0.6928	1.2220	0.9521
GLFN-TC	0.2167	0.3494	0.1187	0.2406	0.1631	0.3036	0.1852	0.2990	0.2432	0.3902

Bold represents the best result

Due to the high R^2 of GLFN-TC in five datasets, it further shows that the prediction accuracy of GLFN-TC is better than that of the conventional methods.

In order to further verify the effectiveness of GLFN-TC, we compare it with other deep learning methods.

Table 2 shows the experimental results of our proposed method and other deep learning methods in short-term load forecasting task. In general, our GLFN-TC achieves state-of-the-art results in five datasets. In ISO-NE dataset, the MSE and MAE of GLFN-TC decreased by 0.0396 and 0.0331, respectively, compared with the best method in baseline methods. In AT dataset, the MSE and MAE of GLFN-TC decreased by 0.0137 and 0.0101, respectively, compared with the best method in baseline methods. In AP dataset, the MSE and MAE of GLFN-TC decreased by 0.0358 and 0.0082, respectively, compared with the best method in baseline methods. In SH dataset, the MSE and MAE of GLFN-TC decreased by 0.0213 and 0.0306, respectively, compared with the best method in baseline methods. In NCENT dataset, the MSE and MAE of GLFN-TC decreased by 0.0337 and 0.0219, respectively, compared with the best method in baseline methods. GLFN-TC performs better than RNN, CNN, LSTM, CNN_LSTM and Informer on five datasets, mainly because it not only obtains temporal dependencies of historical data, but also uses densely connected residual convolution modules to aggregate the data information of the central node and its neighbors. GLFN-TC can handle temporal and spatial dependencies more effectively, which is more conducive to short-term load forecasting task. The reason why GLFN-TC performs better than T-GCN is that the graph learning module can automatically learn the relationships between variables and reuse node information through the densely connected residual convolution module to enhance the propagation of node information. Therefore, GLFN-TC can better handle spatial dependencies for short-term load forecasting task.

In Fig. 10, we show the R^2 of our proposed method and other deep learning methods on ISO-NE, AT, AP, SH and NCENT datasets. The results show that the R^2 of GLFN-TC

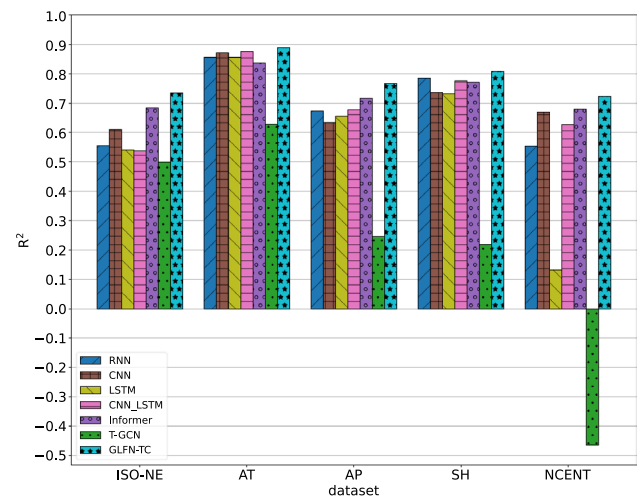


Fig. 10 Comparison of R^2 for proposed method and other deep learning methods

in five datasets is higher than that of baseline methods. Combined with the experimental results in Table 2, it is further shown that the prediction accuracy of GLFN-TC in deep learning methods are higher than that of the baseline methods.

3.4 RQ 2: Model Generality Assessment

Because GLFN-TC can effectively handle temporal and spatial dependencies and can automatically learn graph structure to solve problems where there is no graph structure in the applied field, GLFN-TC is also suitable for other non-load fields. To verify the generality of GLFN-TC, we use TN,⁶ PL⁷ and TC⁸ datasets from different fields for the

⁶ Available at https://github.com/juiomshanti/wind_speed_forecasting.

⁷ Available at <https://github.com/piekarsky/Short-Term-Electricity-Price-Forecasting-at-the-Polish-Day-Ahead-Market>.

⁸ Available at <https://github.com/Eudyptla/Time-Series-PV-Forecast>.

prediction task. The specific information of datasets are as follows:

- *TN* The field of TN dataset is wind speed prediction; it covers the data from January 2000 to December 2014, and the sample rate is 1 h. The number of nodes in this dataset is 6, including wind speed, year, etc.
- *PL* The field of PL dataset is price prediction, it covers the data from November 2017 to December 2020, and the sample rate is 1 h. The number of nodes in this dataset is 4, including electricity price, energy from wind sources, etc.
- *TC* The field of TC dataset is photovoltaic power prediction, it covers the data from January 2016 to December 2017, and the sample rate is 1 h. The number of nodes in this dataset is 9, including power, temperature, sunshine, etc.

Our subsequent experiments will be conducted on the above three datasets. The sample wind speed data, electricity price data and power data are shown in Fig. 11.

In TN dataset, we take the data from January 1, 2000 to September 30, 2001. In PL dataset, we take the data from

January 1, 2018 to December 31, 2019. In TC dataset, we take the data from January 1, 2016 to December 31, 2016. Other experimental settings follow the instructions in Sect. 3.2. We use MSE and MAE as evaluation metrics for models.

Table 3 shows the experimental results of GLFN-TC and baseline methods on TN, PL and TC datasets. The reason for the poor performance of T-GCN on TN and PL datasets is that the change trend of the two datasets is not obviously periodic and the data is not smooth. Therefore, T-GCN cannot effectively handle temporal and spatial dependencies. However, due to the characteristics of the field of TC dataset itself, its data have a certain periodicity, and the prediction difficulty is reduced compared with TN and PL datasets, so it performs better on this dataset. GLFN-TC outperform baseline methods on all metrics across three datasets. In prediction tasks in different fields, GLFN-TC can automatically learn the graph structure through graph learning module to solve the problem that there is no predefined graph structure in the corresponding fields. At the same time, GLFN-TC effectively handles temporal and spatial dependencies through temporal convolution module and densely connected residual

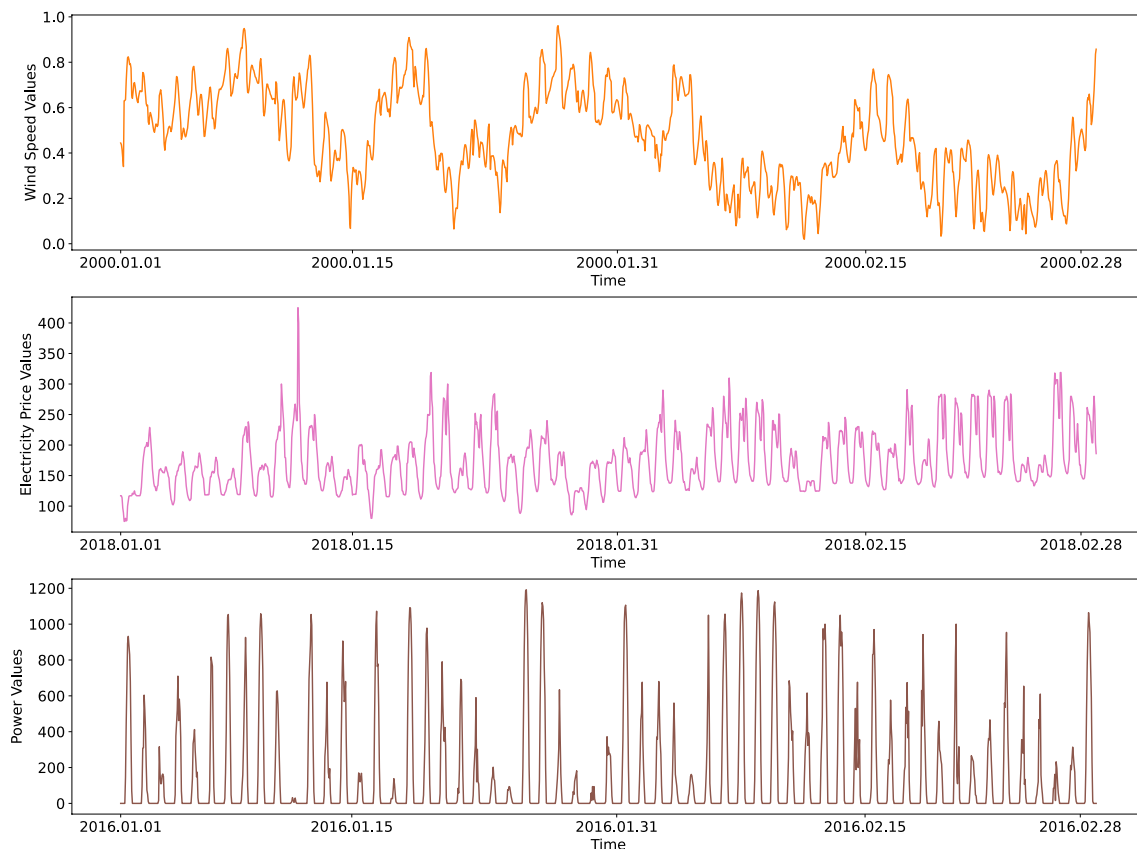


Fig. 11 Sample wind speed data for TN dataset (orange) and sample electricity price data for PL dataset (pink) and sample power data for TC dataset (brown)

Table 3 Comparison of MSE and MAE under prediction tasks in other fields for proposed method and baselines methods

Models	TN		PL		TC	
	MSE	MAE	MSE	MAE	MSE	MAE
NF	0.6569	0.6701	1.5642	1.0392	1.3010	0.6057
SA	1.1904	1.2523	0.5952	0.6075	0.9411	0.8111
MA	0.7292	0.7169	1.6641	1.0793	0.9456	0.7552
RNN	0.2828	0.4306	0.3573	0.4516	0.2413	0.3275
CNN	0.3945	0.5150	0.3559	0.4539	0.2655	0.3525
LSTM	0.3182	0.4559	0.4190	0.5017	0.2406	0.3226
CNN_LSTM	0.5330	0.5969	0.3359	0.4460	0.2392	0.3179
Informer	0.3971	0.5021	0.5598	0.5771	0.8683	0.6099
T-GCN	53.7011	5.1406	71.8001	7.2619	0.2490	0.3206
GLFN-TC	0.2712	0.4136	0.3212	0.4365	0.2326	0.2961

Bold represents the best result

convolution module, respectively. Therefore, the experimental results validate the generality of GLFN-TC.

3.5 RQ 3: Ablation Study

We conduct an ablation study on five load datasets to validate the contribution of key components in GLFN-TC. We eliminate the different components and name them as follows:

- *GLFN-TC-tc* GLFN-TC without the temporal convolution module. We replace the temporal convolution module with a linear layer.
- *GLFN-TC-sc* GLFN-TC without the densely connected residual convolution module. We replace the densely connected residual convolution module with a linear layer.
- *GLFN-TC-dcrc* GLFN-TC eliminates the densely connected residual convolution structure so that the output of the previous layer in the graph convolution step is the input of the next layer and takes the output of the last layer as the output of densely connected residual convolution module.

Table 4 provide the experimental results of ablation study. The introduction of temporal convolution module significantly improves the results on five datasets, because this module uses dilated 1D-CNN to enable the model to mine the relationships between time steps from the historical data of each node, enabling the output data to better represent temporal dependencies. The introduction of densely connected residual convolution module also significantly improves the prediction results, because it makes full use of the output information of previous graph convolution layers, and makes the information flow between the interdependent nodes. Therefore, the data information between nodes

is effectively aggregated. The densely connected residual convolution structure improves the prediction results of ISO-NE dataset, AP dataset, SH dataset and NCENT dataset, especially in ISO-NE dataset and SH dataset. Because this structure makes full use of the output information of each graph convolution layer, avoid the problem that the information utilization rate of previous graph convolution layers will decrease with the increase in graph convolution layers. However, it increases the prediction error of AT dataset. From Fig. 8, we can see that the load data change trend of AT dataset has obvious periodicity, and the load data change range is smaller. Therefore, the prediction of this dataset is less challenging. Adding densely connected residual convolution structure increases the complexity and parameters of the model, which leads to over-fitting and increases prediction error. However, the prediction results are obviously improved on the ISO-NE dataset and SH dataset with greater difficulty.

3.6 RQ 4: Module Effectiveness Assessment

In this subsection, we will evaluate the effectiveness of GLFN-TC modules. First, we verify the effectiveness of the temporal convolution module and the densely connected residual convolution module through the method of module replacement. Secondly, we extract the learned relationship graph to verify whether the relationships learned by GLFN-TC in the graph learning module is valid.

GLFN-TC obtains temporal and spatial dependencies of historical data through temporal convolution module and densely connected residual convolution module. However, traditional methods can also achieve the above objectives. Therefore, we use traditional methods to replace the above two modules to verify the effectiveness of the designed modules. We name the variants of GLFN-TC as follows:

Table 4 Ablation study

Models	ISO-NE		AT		AP		SH		NCENT	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
GLFN-TC	0.2167	0.3494	0.1187	0.2406	0.1631	0.3036	0.1852	0.2990	0.2432	0.3902
GLFN-TC-tc	0.2764	0.3863	0.1379	0.2675	0.1838	0.3154	0.1999	0.3295	0.2521	0.3956
GLFN-TC-sc	0.2704	0.3976	0.1451	0.2786	0.2260	0.3546	0.2609	0.3951	0.3805	0.4901
GLFN-TC-dcrc	0.2894	0.3961	0.1107	0.2344	0.1720	0.3102	0.2690	0.3782	0.2605	0.3992

Bold represents the best result

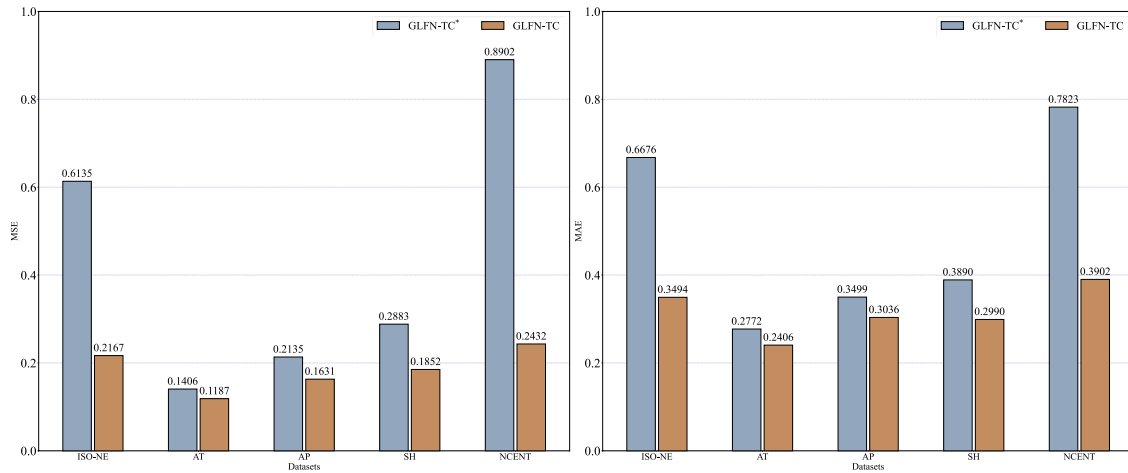


Fig. 12 Comparison of MSE (left) and MAE (right) in GLFN-TC* and GLFN-TC

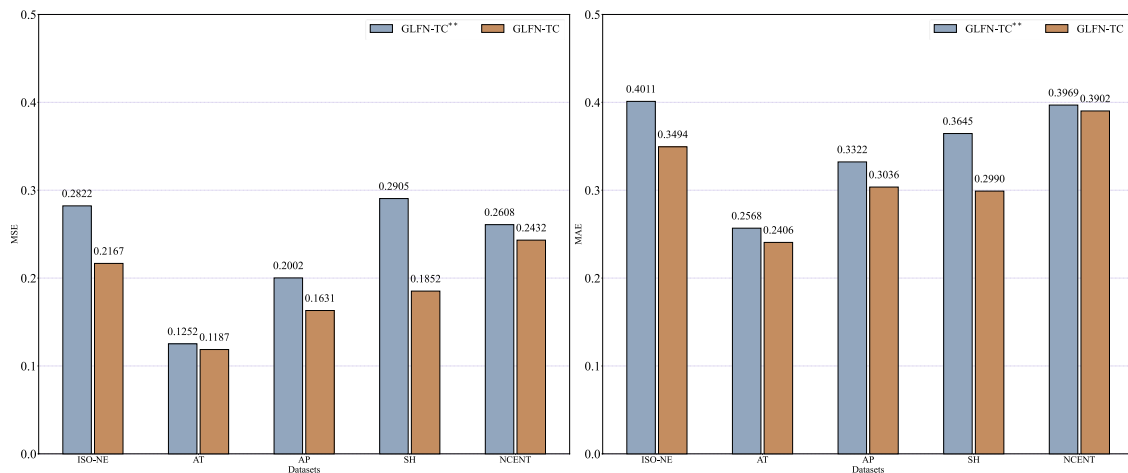


Fig. 13 Comparison of MSE (left) and MAE (right) in GLFN-TC** and GLFN-TC

- **GLFN-TC*** This variant replaces the temporal convolution module in GLCN-TC with LSTM.
- **GLFN-TC**** This variant replaces the densely connected residual convolution module in GLCN-TC with two-layer GCN.

Table 5 shows the experimental results of module effectiveness assessment. Figures 12 and 13 visually show the data in Table 5. The experimental results can directly reflect the effectiveness of the temporal convolution module and the densely connected residual convolution module in

Table 5 Module effectiveness assessment

Models	ISO-NE		AT		AP		SH		NCENT	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
GLFN-TC*	0.6135	0.6676	0.1406	0.2772	0.2135	0.3499	0.2883	0.3890	0.8902	0.7823
GLFN-TC**	0.2822	0.4011	0.1252	0.2568	0.2002	0.3322	0.2905	0.3645	0.2608	0.3969
GLFN-TC	0.2167	0.3494	0.1187	0.2406	0.1631	0.3036	0.1852	0.2990	0.2432	0.3902

Bold represents the best result

GLCN-TC. GLCN-TC* uses LSTM to replace the temporal convolution module in GLCN-TC. It uses LSTM to obtain the temporal dependencies behind historical data. Compared with LSTM, the temporal convolution module is more effective in mining temporal dependencies from behind the historical data of each node by dilated 1D-CNN. The design of temporal convolution module has effectively improved the experimental results, especially on NCENT dataset. GLFN-TC** uses two-layer GCN to replace the densely connected residual convolution module in GLCN-TC. It uses two-layer GCN to aggregate the data information of the central node and its neighbors to handle spatial dependencies. Compared with the traditional GCN, the densely connected residual convolution module can not only effectively use the relationship between the central node and its neighbors for message transmission through the stacking of graph convolution layers, but also ensure the full use of data information. Therefore, this module improves the experimental results of five datasets.

In addition, we extract the learned graph structure to verify the effectiveness of the graph learning module. Take ISO-NE dataset as an example, Figure 14a shows the heat map of the relationships between nodes learned by graph learning module in ISO-NE dataset. Take “demand” as an example in Fig. 14a, it has a large correlation with “month” and a small correlation with “year.” Figure 14b shows the box plot of “demand” in different months in ISO-NE dataset. From Fig. 14b, we can see that the data distribution between different years is similar, while there are significant differences in different months of the same year. The results shown in Fig. 14b is consistent with the conclusions in Fig. 14a. Therefore, the relationship structure learned by graph learning module can approximately represent the strength of relationships between different nodes.

3.7 RQ 5: Repeatability Assessment and Parameter Sensitivity

In the implementation of our method, the parameters of the deep learning model are initialized randomly. Therefore, when the GLFN-TC models are trained independently, there exist some differences between obtained results. Based on the experimental setup in Sect. 3.2, in this subsection, we show the results of three experiments in five datasets to

evaluate the stability of GLFN-TC. In detail, the GLFN-TC model is trained independently for 3 times.

Table 6 shows the experimental results of repeatability assessment. The three experimental results of GLFN-TC model are represented by Trained 1~3, and Avg is the average value of three experimental results. Taking MSE as an example, in ISO-NE dataset, the three results are 0.2141, 0.2210 and 0.2150, respectively. The average value of the three values is 0.2167. The maximum difference between the three values and the average value is 0.0043. In AT dataset, the three results are 0.1102, 0.1198 and 0.1262, respectively. The average value of the three values is 0.1187. The maximum difference between the three values and the average value is 0.0085. In AP dataset, the three results are 0.1518, 0.1754 and 0.1620, respectively. The average value of the three values is 0.1631. The maximum difference between the three values and the average value is 0.0123. In SH dataset, the three results are 0.1837, 0.1686 and 0.2032, respectively. The average value of the three values is 0.1852. The maximum difference between the three values and the average value is 0.018. In NCENT dataset, the three results are 0.2450, 0.2628 and 0.2219, respectively. The average value of the three values is 0.2432. The maximum difference between the three values and the average value is 0.0213. Therefore, we can get that the results of GLFN-TC model are similar, and their differences are small.

We conduct experiments on ISO-NE dataset to explore the impact of the dilation factor in Sect. 2.3 on the model. Specifically, we take the values of dilation factor as 1, 2, 3, 4 and 5. The experimental results are shown in Fig. 15. When the value of dilation factor is 1, the results are poor. The reason is that convolution operations have become ordinary 1D-CNN, and long-term dependencies cannot be obtained through dilation strategy. As the value of dilation factor increases, the results gradually improve. We can see that GLFN-TC’s performance is generally stable for a value of dilation factor in range of [2–4]. When the value of dilation factor is 5, the results are the worst. The reason is that large dilation factor value cannot effectively obtain short-term data information. In the experiment, we used the general prediction level of the model, i.e., value of dilation factor is 3.

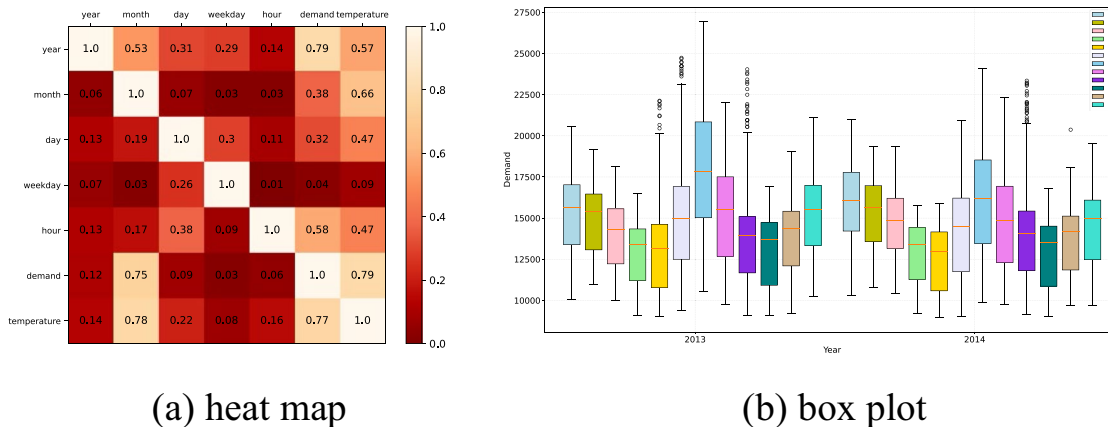


Fig. 14 Heat map and box plot in ISO-NE dataset

Table 6 Repeatability assessment

Trained	ISO-NE		AT		AP		SH		NCENT	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
1	0.2141	0.3516	0.1102	0.2345	0.1518	0.2927	0.1837	0.2915	0.2450	0.3878
2	0.2210	0.3467	0.1198	0.2367	0.1754	0.3132	0.1686	0.2826	0.2628	0.4056
3	0.2150	0.3498	0.1262	0.2505	0.1620	0.3050	0.2032	0.3229	0.2219	0.3771
Avg	0.2167	0.3494	0.1187	0.2406	0.1631	0.3036	0.1852	0.2990	0.2432	0.3902

3.8 Discussion

On the one hand, we analyzed the ability of GLFN-TC to handle temporal dependencies. In ablation study, eliminating temporal convolution module can lead to a decrease in model prediction performance. The results indicate that our proposed temporal convolution module can handle the temporal dependencies behind historical data. In module effectiveness assessment, using LSTM instead of the temporal convolution module reduced the predictive performance of the model, further verifying that the temporal convolution module can better handle temporal dependencies and cooperate with densely connected residual convolution module to improve predictive performance. GLFN-TC outperforms other deep learning methods such as LSTM and CNN in the field of short-term load forecasting. This indicates that GLFN-TC can better handle temporal dependencies through temporal convolution module and achieve more accurate prediction results in conjunction with densely connected residual convolution module.

On the other hand, we analyzed the ability of GLFN-TC to handle spatial dependencies. In the ablation study, the experimental results validated the effectiveness of adding densely connected residual structure to graph convolution. In module effectiveness assessment, replacing the densely connected residual convolution module with GCN reduced the predictive performance of the model, further verifying

that the densely connected residual convolution module can better handle spatial dependencies. GLFN-TC builds on graph neural networks. Compared with previous works, such as T-GCN and GCN, GLFN-TC not only solves the problem of no predefined graph structure in prediction tasks, but also effectively processes the temporal dependencies of input data for each node through dilated 1D-CNN and solves the problem of information loss in message passing through graph convolution operation with densely connected residual structure.

According to its own characteristics, GLFN-TC can also be applied to prediction problems in non-load fields, and it is further verified by model generality assessment. However, treating the output information of each graph convolution layer equally in densely connected residual convolution module can lead to the reuse of some useless information and increase the complexity of the model. As shown in the results of ablation study in AT dataset, eliminating densely connected residual convolution structure reduces prediction errors. Therefore, future research directions can extract useful output information from each graph convolutional layer, avoiding the reuse of useless information and reducing model complexity. At the same time, in temporal convolution module, we can further consider attention mechanism to focus on important time steps in historical data, so as to further improve the prediction accuracy of the approach.

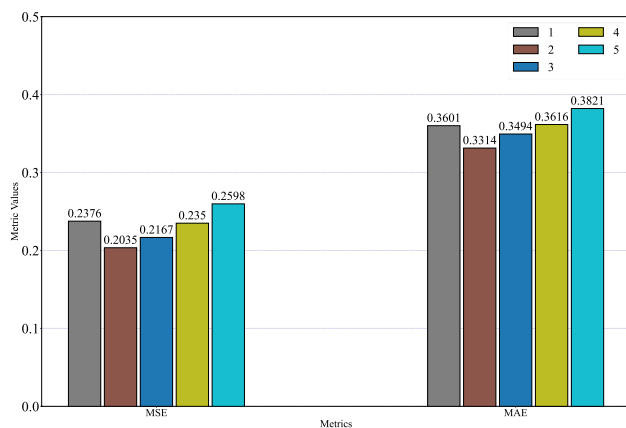


Fig. 15 MSE and MAE for different dilation factor in ISO-NE dataset

4 Related Work

4.1 Load Forecasting

Load forecasting is based on historical data such as load, weather, temperature, etc., and explore the change rule of historical data. By seeking the internal relationships between load data and related factors, so as to forecast the future load values.

Many classical methods have been applied to the field of load forecasting. Wei et al. established ARIMA differential autoregressive moving average model and analyzed the advantages and disadvantages of ARIMA model in load forecasting [21]. Goude et al. proposed a semi-parametric approach based on generalized additive models theory and considers both the short- and middle-term forecasting conditions. These generalized additive models estimate the relationships between load and explanatory variables (such as temperature and calendar variables). This methodology has been well-applied in the French grid [22]. Chen et al. proposed an artificial neural network (ANN) model. This model can differentiate between the weekday loads and the weekend loads and can predict the hourly loads for an entire week. The results show that the model has greater prediction accuracy than the traditional statistical model [23]. Fan et al. proposed a short-term load forecasting method based on an adaptive two-stage hybrid network with self-organized map (SOM) and support vector machine (SVM). Firstly, the SOM network is used to cluster the input dataset into several subsets. Then, SVMs are used to fit the training data of each subset. The proposed structure is robust with different data types and can deal with the non-stationarity of load series well [24]. Shi et al. proposed a novel pooling-based deep recurrent neural network for household load forecasting. In essence, the model could address the over-fitting issue by increasing data diversity and volume [25]. Wilms

et al. proposed a general purpose forecasting method, that is, a sequence-to-sequence machine learning architecture for time series forecasting based on recurrent neural networks. It is evaluated in short-term electric load forecasting, and the results show that it outperforms other machine learning forecasting techniques [26]. Incremona et al. used a Gaussian process (GP) estimator to track the difference between the target Easter Week and an average Easter Week load profile. Differently from usual GP approaches that employ “canonical” kernels, it is a customized kernel that is tailored to the specific statistical properties of the signal to be predicted [27]. Kiruthiga et al. proposed a new optimized deep learning (DL) network design for time series load forecasting. Firstly, the hyper-parameters of DL are optimized by LF-PSO technique. Then, the optimized DL model is used for load prediction [28].

In practice, due to the abnormal operation of the measurement system, the measured load data have abnormal values, which affects the quality of the predicted values obtained [29]. In order to solve the problem of impact on the accuracy of the load forecasting caused by abnormal load data. Ma et al. used iForest to clear abnormal historical load data and used iForest-LSTM for short-term load forecasting. Compared with standard LSTM and iForest-BP methods, iForest-LSTM improved the forecasting accuracy [30]. CNN is also often used for load forecasting in recent years. Khan et al. and Dong et al. used CNN network in load forecasting to obtain more accurate results [31, 32]. Sajjad et al. used a hybrid model of CNN and GRU to forecast electricity consumption and evaluated its performance over several benchmark datasets [33].

4.2 Graph Neural Networks

In recent years, GNNs have achieved great success in graph-structured data. In general, GNNs assume that the state of a node will be affected by the states of its neighbors. For example, Kipf et al. proposed a graph convolution networks (GCNs) model, a node’s feature representation by aggregating the representations of its one-step neighbors. The experimental results on citation networks and on a knowledge graph dataset show that this method outperforms related methods by a significant margin [34]. GNNs can obtain the dependency characteristics between multivariate data, and its related variants are often used in various fields. For example, in the field of traffic prediction, Chen et al. proposed the multi-range attentive bicomponent GCN (MRA-BGCN). Firstly, node-wise graph and edge-wise graph are constructed and then implement the interactions of both nodes and edges using bicomponent graph convolution. Multi-range attention mechanism is introduced to aggregate information in different neighborhood ranges and automatically learn the importance of different ranges [35].

5 Conclusion

In this paper, we propose a novel framework for short-term load forecasting, namely graph neural network-based short-term load forecasting with temporal convolution (GLFN-TC). GLFN-TC creates a feature vector for each variable to represent the characteristics of the variable and automatically learns a graph of relationships between variables through the graph learning module. Its advantage is that it can automatically update in the training stage to make the relationship graph optimal. The temporal convolution module uses dilated 1D CNN to capture the temporal dependencies of the historical data of each node. The densely connected residual convolution module can make full use of the data information of each graph convolution layer and avoid the problem of information loss caused by stacking the graph convolution layers. Finally, the load forecasting module integrates the data representation and feature vector of each node to predict the future load values. The excellent performance of the proposed GLFN-TC is verified on five load datasets. Since GLFN-TC can solve the problem of no predefined graph structure in the application field, and it can effectively deal with temporal and spatial dependencies. Therefore, three non-load datasets are used to further prove the generality of the model.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Grant Nos.62002262, 62172082, 62072086, 62072084).

Author Contributions CS involved in conceptualization, methodology, investigation, data curation, writing (original draft preparation and reviewing) and funding. YN involved in investigation, validation, data curation, software and visualization and writing (original draft preparation and reviewing). DS involved in conceptualization, writing (reviewing and editing) and funding. TN involved in conceptualization, writing (reviewing and editing) and funding.

Data Availability Datasets are available at: (1) <https://github.com/ningningLiningning/iso-ne>; (2) <https://github.com/Lizhuoling/DCN>; (3) <https://zhuanlan.zhihu.com/p/150954853>. Graph Neural Network-based Short-Term Load Forecasting with Temporal Convolution.

Declarations

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of the paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will

need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Huang Y, Zhao R, Zhou Q, Xiang Y (2022) Short-term load forecasting based on a hybrid neural network and phase space reconstruction. *IEEE Access* 10:23272–23283
- Liang Y, Niu D, Hong WC (2019) Short term load forecasting based on feature extraction and improved general regression neural network model. *Energy* 166:653–663
- Jagait RK, Fekri MN, Grolinger K, Mir S (2021) Load forecasting under concept drift: online ensemble learning with recurrent neural network and ARIMA. *IEEE Access* 9:98992–99008
- Din GMU, Marnerides AK (2017) Short term power load forecasting using deep neural networks. In: 2017 International conference on computing, networking and communications (ICNC), IEEE, pp 594–598
- Liu F, Dong T, Hou T, Liu Y (2021) A hybrid short-term load forecasting model based on improved fuzzy c-means clustering, random forest and deep neural networks. *IEEE Access* 9:59754–59765
- Muzumdar AA, Modi CN, Vyjayanthi C (2021) Designing a robust and accurate model for consumer-centric short-term load forecasting in microgrid environment. *IEEE Syst J* 16(2):2448–2459
- Farrag TA, Elattar EE (2021) Optimized deep stacked long short-term memory network for long-term load forecasting. *IEEE Access* 9:68511–68522
- Mbamalu GAN, El-Hawary ME (1993) Load forecasting via suboptimal seasonal autoregressive models and iteratively reweighted least squares estimation. *IEEE Trans Power Syst* 8(1):343–348
- Chen JF, Wang WM, Huang CM (1995) Analysis of an adaptive time-series autoregressive moving-average (ARMA) model for short-term load forecasting. *Electric Power Syst Res* 34(3):187–196
- Ahmad N, Ghadi Y, Adnan M, Ali M (2022) Load forecasting techniques for power system: research challenges and survey. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2022.3187839>
- Muzaffar S, Afshari A (2019) Short-term load forecasts using LSTM networks. *Energy Procedia* 158:2922–2927
- Smyl S, Dudek G, Pelka P (2022) ES-dRNN with dynamic attention for short-term load forecasting. In: 2022 International Joint Conference on Neural Networks (IJCNN), IEEE, pp 1–8
- Yu B, Yin H, Zhu Z (2018) Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. In: Lang J (Ed), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Jul 13–19, 2018, Stockholm, Sweden, pp 3634–3640
- Wu Z, Pan S, Long G, Jiang J, Zhang C (2019) Graph WaveNet for deep spatial-temporal graph modeling. In Kraus S (Ed), Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, Aug 10–16, 2019 (pp 1907–1913)
- Rong Y, Huang W, Xu T, Huang J (2020) DropEdge: towards deep graph convolutional networks on node classification. In: 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, Apr 26–30
- Xu K, Li C, Tian Y, Sonobe T, Kawarabayashi K, Jegelka S (2018) Representation learning on graphs with jumping knowledge networks. In: Dy JG, Krause A (Eds) Proceedings of the 35th International Conference on Machine Learning, ICML 2018,

- Stockholmsmässan, Stockholm, Sweden, Jul 10–15, Vol 80, pp 5449–5458
17. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *ArXiv*, abs/1803.01271
 18. Tudose A M, Sidea DO, Picioroaga II, Boicea VA, Bulac C (2020) A CNN based model for short-term load forecasting: a real case study on the Romanian power system. In: 2020 55th International Universities Power Engineering Conference (UPEC), IEEE, pp 1–6
 19. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI conference on artificial intelligence, Vol 35, No 12, pp 11106–11115
 20. Zhao L, Song Y, Zhang C, Liu Y, Wang P, Lin T, Deng M, Li H (2020) T-GCN: a temporal graph convolutional network for traffic prediction. *IEEE Trans Intell Transp Syst* 21(9):3848–3858
 21. Wei L, Zhen-gang Z (2009) Based on time sequence of ARIMA model in the application of short-term electricity load forecasting. In: 2009 International Conference on Research Challenges in Computer Science. IEEE, pp 11–14
 22. Goude Y, Nedellec R, Kong N (2013) Local short and middle term electricity load forecasting with semi-parametric additive models. *IEEE Trans Smart Grid* 5(1):440–446
 23. Chen ST, Yu DC, Moghaddamjo AR (1992) Weather sensitive short-term load forecasting using nonfully connected artificial neural network. *IEEE Trans Power Syst* 7(3):1098–1105
 24. Fan S, Chen L (2006) Short-term load forecasting based on an adaptive hybrid method. *IEEE Trans Power Syst* 21(1):392–401
 25. Shi H, Xu M, Li R (2017) Deep learning for household load forecasting—A novel pooling deep RNN. *IEEE Trans Smart Grid* 9(5):5271–5280
 26. Wilms H, Cupelli M, Monti A (2018) Combining auto-regression with exogenous variables in sequence-to-sequence recurrent neural networks for short-term load forecasting. In: 2018 IEEE 16th international conference on industrial informatics (INDIN). IEEE, pp 673–679
 27. Incremona A, De Nicolao G (2022) Short-term forecasting of the Italian load demand during the Easter Week. *Neural Comput Appl* 34:1–15
 28. Kiruthiga D, Manikandan V (2023) Levy flight-particle swarm optimization-assisted BiLSTM+ dropout deep learning model for short-term load forecasting. *Neural Comput Appl* 35(3):2679–2700
 29. Almeida VA, Pessanha JF, Caloba LP (2018) Load data cleaning with data mining techniques. In: 2018 Brazilian Symposium on Electrical Systems (SBSE). IEEE, pp 1–6
 30. Ma Y, Zhang Q, Ding J, Wang Q, Ma J (2019) Short term load forecasting based on iForest-LSTM. In: 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA), IEEE, pp 2278–2282
 31. Khan S, Javaid N, Chand A, Khan ABM, Rashid F, Afridi IU (2019) Electricity load forecasting for each day of week using deep CNN. In: Web, artificial intelligence and network applications: proceedings of the workshops of the 33rd international conference on advanced information networking and applications (WAINA-2019) 33, pp 1107–1119, Springer International Publishing
 32. Dong X, Qian L, Huang L (2017) Short-term load forecasting in smart grid: a combined CNN and K-means clustering approach. In: 2017 IEEE international conference on big data and smart computing (BigComp), IEEE, pp 119–125
 33. Sajjad M, Khan ZA, Ullah A, Hussain T, Ullah W, Lee MY, Baik SW (2020) A novel CNN-GRU-based hybrid approach for short-term residential load forecasting. *IEEE Access* 8:143759–143768
 34. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, Apr 24–26, Conference Track Proceedings
 35. Chen W, Chen L, Xie Y, Cao W, Gao Y, Feng X (2020). Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In: Proceedings of the AAAI conference on artificial intelligence, Vol 34, No 04, pp 3529–3536