applied sciences

MDPI

Article

Multivariate Time Series Deep Spatiotemporal Forecasting with Graph Neural Network

Zichao He, Chunna Zhao * and Yaqun Huang

School of Information Science and Engineering, Yunnan University, Kunming 650504, China; hzc@mail.ynu.edu.cn (Z.H.); huangyq@ynu.edu.cn (Y.H.)
* Correspondence: zhaochunna@ynu.edu.cn

Abstract: Multivariate time series forecasting has long been a subject of great concern. For example, there are many valuable applications in forecasting electricity consumption, solar power generation, traffic congestion, finance, and so on. Accurately forecasting periodic data such as electricity can greatly improve the reliability of forecasting tasks in engineering applications. Time series forecasting problems are often modeled using deep learning methods. However, the deep information of sequences and dependencies among multiple variables are not fully utilized in existing methods. Therefore, a multivariate time series deep spatiotemporal forecasting model with a graph neural network (MDST-GNN) is proposed to solve the existing shortcomings and improve the accuracy of periodic data prediction in this paper. This model integrates a graph neural network and deep spatiotemporal information. It comprises four modules: graph learning, temporal convolution, graph convolution, and down-sampling convolution. The graph learning module extracts dependencies between variables. The temporal convolution module abstracts the time information of each variable sequence. The graph convolution is used for the fusion of the graph structure and the information of the temporal convolution module. An attention mechanism is presented to filter information in the graph convolution module. The down-sampling convolution module extracts deep spatiotemporal information with different sparsities. To verify the effectiveness of the model, experiments are carried out on four datasets. Experimental results show that the proposed model outperforms the current state-of-the-art baseline methods. The effectiveness of the module for solving the problem of dependencies and deep information is verified by ablation experiments.

Keywords: multivariate time series; deep spatiotemporal information; down-sampling convolution; attention; graph neural network

check for updates

Citation: He, Z.; Zhao, C.; Huang, Y. Multivariate Time Series Deep Spatiotemporal Forecasting with Graph Neural Network. *Appl. Sci.* **2022**, *12*, 5731. https://doi.org/10.3390/app12115731

Academic Editors: Dionisis Margaris and Stefanos Ougiaroglou

Received: 16 May 2022
Accepted: 2 June 2022
Published: 5 June 2022

1. Introduction

With the development of the Internet, various sensors and data-storage devices have appeared in modern society. As a result, a large amount of time series data is generated by recording temperature, traffic, power consumption, and financial data. Multivariate time series data consists of time series data generated by multiple sensors or data storage devices, and there are dependencies among multiple time series. For example, a household's daily electricity usage and hourly electricity production from solar panels can be considered time series data. In most cases, the data usually come from the electricity consumed by multiple households or solar data generated at different locations. A multivariate time series is constructed from these data. There may be complex dynamic dependencies between multivariate time series data. Therefore, each time series in a multivariate time series is helpful for forecasting tasks. At the same time, multivariate time series data such as electricity and transportation have periodic characteristics. Multivariate time series forecasting has been studied for a long time in capturing the periodic characteristics and dynamic dependencies of variables [1–4]. For example, studies using the Harmonic regression method enhanced the prediction performance of $PM_{2.5}$ by capturing periodic information [5].

In recent years, some researchers have made efforts in time series analysis and forecasting. For example, traditional forecasting methods represented by statistical methods. Most rely on mathematical equations to describe the evolution of time series, such as the Autoregressive Integrated Moving Average Model (ARIMA) [6]. Therefore, various variants based on the ARIMA model have emerged. Models based on traditional methods are more computationally efficient. However, these models are mostly limited to linear models and univariate predictions. It is difficult to extend to multivariate time series problems.

In the era of big data, with the development of sensor technology and computing power, most of the time series data comes from data collected by various devices. However, the information of large-scale data is difficult to extract by traditional methods. Deep learning is the current popular information extraction method. Features of large-scale data can be obtained through deep learning techniques. In time series forecasting, deep learning techniques are employed to achieve better forecasting accuracy than traditional methods. At the same time, deep learning technology has made great progress in the research and application of image processing, audio processing and natural language processing [7–9]. Although traditional machine learning and statistical methods are often employed in time series forecasting tasks, deep learning techniques are gaining attention from researchers. With further development, there have been studies on applying graph neural networks and attention methods in time series forecasting [10–12]. Capturing spatial information by building a graph structure of multivariate time series. These methods have achieved certain results. A graph neural network allows each node in the graph to acquire information about the surrounding nodes. Multivariate time series can be considered from the perspective of graph nodes. Spatial information can be obtained between multivariate time series by constructing graphs. For periodic data, the information of adjacent nodes is more informative. Compared with vanilla neural networks, graph neural networks can capture the dependency information between different sequences, which is more suitable for the prediction of multivariate time series. Therefore, graph neural networks can achieve more accurate prediction results than vanilla neural networks by aggregating information from multiple sequences.

In this paper, a multivariate time series deep spatiotemporal forecasting model with a graph neural network (MDST-GNN) is proposed. The model consists of four core components: graph learning, temporal convolution, graph convolution, and down-sampling convolution. The local information and deep spatiotemporal information of multivariate time series data can be learned by the model. Dependencies between sequences can be captured by a graph learning module. The temporal information is captured by the 1D convolution of the temporal convolution module. In the graph convolution module, the spatial dependencies between variables are extracted through relational graphs. The spatiotemporal information in sequences with different sparsity is captured by down-sampling convolution modules. The experiments in this paper show that the model has good prediction performance and generalization ability.

The main contributions of this paper are as follows:

1. A more general time series forecasting model based on graph convolutional networks and deep spatiotemporal features is proposed in this paper.
2. The attention mechanism is added to the graph convolution module to realize the filtering of spatiotemporal information.
3. A down-sampling convolution module is proposed to extract deep spatiotemporal information of time series to improve the performance of the model.

The remainder of this paper is organized as follows: Section 2 presents the related work and research status. Section 3 introduces the definition of the problem and presents the overall framework and algorithmic flow of the model. Section 4 describes the experimental part and gives the experimental results of different methods in the dataset, analysis of experimental results, and ablation experiments. Finally, Section 5 is the conclusion of this paper.

2. Related Works

2.1. Time Series Forecasting with Traditional Methods

Time series forecasting has been studied for a long time. Most of the existing work can be divided into statistical methods and machine learning and deep learning methods. Such as Autoregressive Integrated Moving Average Model (ARIMA) [6] and a hybrid ARIMA and multilayer perception model (VARMLP) [13]. ARIMA obtained future forecast values by constructing polynomials of historical information and adding noise. VARMLP modeled linear and nonlinear data combining ARIMA with an artificial neural network model (ANN) [14]. Neural networks are often used to capture nonlinear relationships in time series forecasting [15,16]. However, the main limitations of ARIMA models are the pre-assumed linear patterns and the requirement for high stationarity of the data. Therefore, ARIMA is not suitable for modeling problems of multivariate time series. Furthermore, Linear Support Vector Regression (SVR) [17] treats the prediction problem as a typical regression problem with parameters changing over time. The vector autoregressive model (VAR) [18] extends the AR model and was a commonly used econometric model. However, these models are difficult to extend to multivariate time series forecasting problems.

2.2. Time Series Forecasting with Deep Learning

In recent years, with the improvement of data availability and computing power, deep learning techniques have been adopted to obtain better prediction accuracy than traditional methods. Nonlinear patterns of data can be captured by deep learning models and outperform traditional methods. The time series forecasting models LSTNet [19] and TPA-LSTM [20] excelled in capturing nonlinear information. LSTNet used Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to extract short-term local dependence patterns and long-term patterns of time series. The former encoded the short-term local information of multivariate sequences into low-dimensional vectors. The latter decoded the vectors to capture long-term dependency information. Long Short-Term Memory (LSTM) [21] is a variant of RNN that is often used for time series forecasting tasks. LSTM also shows excellent performance in natural language processing tasks. TPA-LSTM extracted temporal patterns through the LSTM network, and a convolutional network and attention mechanism were used to calculate the attention score. However, LSTNet and TPA-LSTM cannot model the dependencies of multiple variables. RNN and LSTM cannot memorize the dependencies of multiple variables for a long time. Sample Convolution and Interaction Networks (SCINet) [22] use sequence sampling and convolutional neural network methods to capture temporal information. However, SCINet does not consider the dependencies among multiple variables, and the stacking of multiple layer structures results in huge time and space costs. Recently, Temporal Convolutional Networks (TCN) [23] have been applied to time series forecasting problems. In the TCN architecture, dilated convolutions can be used to flexibly adjust the receptive field, and its space complexity is lower than that of RNN. However, the causal convolution of TCN is not suitable for sliding window inputs. With the development of attention methods, the Transformer [24] model has replaced the RNN model in many sequence modeling tasks. Therefore, various time series forecast methods based on Transformers have emerged [25], which are quite effective in predicting long series. However, although time series models for reducing the spatial cost of Transformer have been proposed, the space cost is still large.

With further research, graph neural networks have shown great advantages in processing graph data [26–28]. In the structure of graph neural networks, each node is interconnected with its neighbors, and information is transmitted between nodes. The variables of a multivariate time series can be transformed into a graph-like structure. A variable represents a node in the graph. Nodes are interrelated through their hidden dependencies. Therefore, graphs in multivariate time series data modeling can better represent the interdependencies between time series. Currently, many people have used graph neural networks to deal with time series problems [29,30]. Multivariate Time Series Forecasting with Graph Neural Networks (MTGNN) [31] built a graph with variables as nodes. The spatiotemporal information of the data is captured by dilated convolutional networks [32] and relational graphs. However, the dilated convolutional structure of MT-GNN causes the loss of locally continuous and deep-level information. Although the above methods have been successfully applied to many forecasting problems, most of them have corresponding shortcomings.

By comparing traditional methods and classical deep learning models, a multivariate time series deep spatiotemporal forecasting model with graph neural network (MDST-GNN) is proposed in this paper. The model is able to learn relational graphs from multivariate time series data. Deep spatiotemporal information is extracted and filtered through down-sampling convolutional networks and attention methods. Our model can capture the deep nonlinear spatiotemporal features in the sequence and solve the problem of local and global information loss.

3. Methods

Firstly, this section presents the definition of the prediction problem. The MDST-GNN model consists of a graph-learning module, a temporal convolution module, a graph convolution module and a down-sampling convolution module. In this paper, an attention mechanism is introduced into the graph convolution module to filter spatiotemporal information. A down-sampling convolution module is proposed to extract deep spatiotemporal information to improve model performance. Finally, the overall frame diagram and algorithm steps of the MDST-GNN model are given.

3.1. Problem Definition

Definition 1. *There is a multivariate time series with sequence length m, given as a set* $X = \{s_{m,1}, s_{m,2}, s_{m,3}, \ldots, s_{m,n}\}$ *,where* $s_{m,1} = \{t_{1,1}, t_{2,1}, t_{3,1}, \ldots, t_{m,1}\}$ *is the set of the first variable sequence and* $t_{n,[i]} \in R$ *is the value of the ith variable at time step n.*

Definition 2. *A graph can be defined as* $G = \{V, E\}$*. V represents the set of nodes. E represents the set of edges formed by the dependencies between variables. An adjacency matrix is a mathematical form that facilitates graph computation. It is* $A \in R^{N \times N}$*, assuming* $\{vi, vj\} \in V$*. If* $(vi, vj) \in E$*, then* $A_{ij} = C > 0$*, if* $(vi, vj) \notin E$*, then* $A_{ij} = 0$*.*

Definition 3. *The goal is to learn a model* $F(\cdot)$ *and build the mapping of F from X to Y by minimizing the L2 regularization loss function. Given a prediction step size h, it is possible to predict the future value* $Y = \{s_{m+h,n}\}$ *of X after h steps.*

3.2. Graph Learning Module

During the training of the model, the spatial information representation of the input data and the implicit relationships among multiple sequences are obtained by the graph learning module. In existing research, graphs are mostly constructed by the method of node similarity and distance, and these graphs are usually bidirectional or symmetric. For the prediction task in this paper, other nodes may be affected by the change in one node. For example, if the traffic roads in a certain area are congested, other roads in the area will also change. When solar power generation in a region rises, other sites in the region will also have certain changes. Therefore, a one-way relational graph structure is more suitable to be constructed:

$$A_1 = tanh(\alpha \times E_1 \times W_1) \tag{1}$$

$$A_2 = tanh(\alpha \times E_2 \times W_2) \tag{2}$$

$$B = A_1 \times {A_2}^T \tag{3}$$

$$A = Relu(tanh(\alpha \times (B - B^T))) \tag{4}$$

According to the above formula, the initial adjacency matrix A can be obtained. E_1 and E_2 are learnable nodes encoding information. W_1 and W_2 are model-learnable parameters. The hyperparameter α is used to prevent the vanishing gradient problem caused by the saturation of the activation function. The adjacency matrix is converted to a one-way matrix by matrix subtraction and activation function Relu. When A_{ij} is a positive number, $A_{ij} = 0$:

$$A[i, not\ argtopk(A[i,:], k)] = 0 \tag{5}$$

Argtopk() extracts the indices of the top k maxima of the vector. In order to simplify the calculation amount, the k nodes with the largest value among the nodes are selected as the nodes with high relevant information during graph convolution. Unselected nodes are assigned the value 0. This method is more flexible in constructing graphs, and the internal information of the graph can be adjusted according to the data during the training process.

3.3. Temporal Convolution Module

The temporal convolution module consists of two Dilated Inception layers. Tanh and Sigmoid activation functions are used after the Dilated Inception layer. One dilated inception layer is followed by an activation function, and Tanh is used as a filter. The sigmoid function of the other layer is used as the gate unit. The gate unit controls the amount of information that the filter propagates to the next module.

The advantages of the Inception network and Dilated convolution are incorporated into the Dilated Inception layer. Information at different scales can be captured by the Inception network, while the dilated convolutional network ensures that long-term sequences can be processed. First, the receptive field of traditional convolutional networks is limited by the depth of the network and the size of the convolution kernels. Processing long-term sequences requires increasing convolution kernel size and network depth, which increases the cost of computational resources for the problem. For example, for a network with L one-dimensional convolutions and K convolution kernels, the size of the receptive field is:

$$R = (K - 1) \times L + 1 \tag{6}$$

In WaveNet [33], stacked dilated convolutions were used to make the network have a very large receptive field. Computational efficiency and input data integrity are guaranteed with only a few layers. In this paper, the dilated convolution method is used to reduce the cost of model computing resources. The dilation factor is doubled for each layer. With the increase in depth, the dilation factor can make the receptive field grow exponentially. For long time series, the advantage of dilated convolution is that the internal data structure can be preserved without reducing the length of the data input. However, models designed based on dilated convolutions also have some problems. Due to the discontinuity of the convolution kernel, all elements cannot be covered in the dilated convolution, so the continuity of information will be lost. Therefore, the Inception network is introduced into the model. Multiple convolution kernels of different sizes are used to cover the receptive field to retain more information.

In the Inception network, features of multiple scales are obtained by convolution kernels of different sizes. The sparse matrices output by multiple convolutional networks are aggregated into denser submatrices. For the Inception network, the size of the convolution kernel needs to be selected according to the characteristics of the data. According to the periodicity of the time series (2, 3, 6, 7), four convolution kernels are used by the Inception network in this paper. These convolution kernels can be combined to cover multiple time periods of different lengths. The input data X is processed using four convolution kernels of different scales. The outputs of different scale convolutions are fused to obtain complete temporal information.

$$X = concat(X \times I_{1\times 2}, X \times I_{1\times 3}, X \times I_{1\times 6}, X \times I_{1\times 7}) \tag{7}$$

Finally, the convolution results of different lengths are cropped to the same length. In Formula (7), the convolution results are concatenated by the channel dimension. Features of different scales are learned by each layer in the network. The adaptability of the network to different scales of information is increased by the concatenation of channel dimensions.

3.4. Graph Convolution Module

In the graph convolution module, dependency information is extracted from the input data through the adjacency matrix of the graph learning module. Each node of the adjacency matrix is fused with highly dependent node information to obtain the dependency information of each node and its related nodes.

The graph convolution module consists of two Attention Mixed Propagation layers for processing the inflow and outflow information of each node. The structure of the graph convolution module is shown on the left of Figure 1, and A is the adjacency matrix of the node relationship obtained by the graph learning layer. The inflow information of each node is processed by the Attention Mixed propagation layer. A^T is used to process the outflow information of the node. After the Attention Mixed propagation layer, the final node feature information is obtained by summing the inflow and outflow information of the nodes. The structure of the Attention Mixed Propagation layer is shown on the right in Figure 1, and A is the dependency matrix between nodes, D_{in} is the output information of the temporal convolutional layer, and $D^{(k)}$ represents the information that D_{in} propagates K times in the nodes of A. After the Concat connection, C is the channel dimension of the feature information, N is the number of variables in the feature information, and T is the sequence length of the feature information. The node information is weighted and filtered for different dimensions of the feature information, and the weighted sum is used as the output of the Attention Mixed propagation layer. The graph obtained by the graph learning layer is utilized to process the information flow of the relevant nodes in the Attention Mixed propagation layer. The Attention Mixed propagation layer consists of two parts: information propagation and information selection.

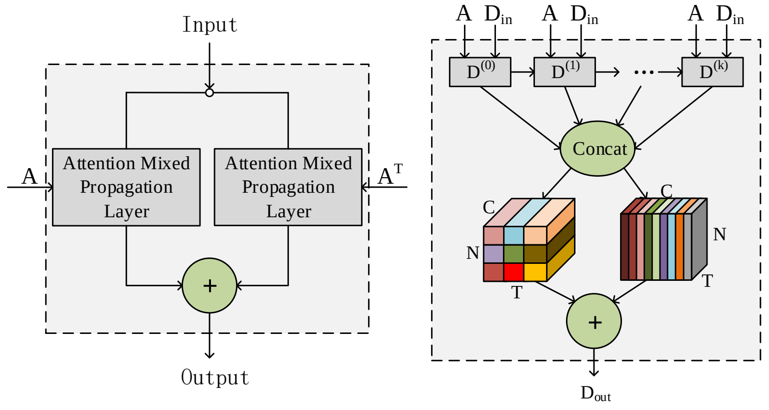


Figure 1. Graph Convolution Module and Attention Mixed Propagation.

The information propagation steps are as follows:

$$D^{(k)} = \beta D_{in} + (1-\beta)\frac{(A+I)}{\sum_{j=0}^{n} A_{ij}} D^{(k-1)} \tag{8}$$

$$D = Concat(D^{(0)}, D^{(1)}, ..., D^{(k)}) \times W^{(k)} \tag{9}$$

Among them, β in the Formula (8) is a hyperparameter used to control the retention rate of initial information. A is the graph adjacency matrix. K is the depth of information propagation. D_{in} is the output of the previous module. It can be seen from Formula (9) that $D^{(k)}$ is concatenated through the channel dimension after the information propagation process. Afterwards, the channel dimension is compressed to the dimension before concatenation by a convolutional network. D contains the features of spatial dependence after each node information is propagated.

However, initial information and new information are continuously fused in graph convolution. The initial information of the node will be continuously lost when reaching a certain depth, which will lead to unreliable data. Therefore, the β parameter is introduced to preserve part of the initial information. Make sure that local and deep-level information of nodes is preserved. At the same time, the cases where the spatial dependence is not obvious or there is no spatial dependence also need to be considered between the data. In this case, the important information of each node will be disturbed by the graph convolution operation instead.

Therefore, Spatial and Channel Squeeze and Excitation (scSE) [34] is used as an information-selection method to filter out some unnecessary information in this paper. The scSE attention method consists of two parts, namely, Channel Squeeze and Excitation (cSE) and Spatial Squeeze and Excitation (sSE). The attention mechanism is used to adjust the weights of features in the network. By weighting important feature maps or feature channels, the influence of unimportant features was reduced, thereby improving prediction results. The core idea of Squeeze and Excitation (SE) was to dynamically learn feature weights through network training. The effective feature weight was amplified and the invalid or small effect feature weights were reduced so that the model can achieve better results.

Among them, cSE refers to the compressed spatial dimension information and the adjusted channel dimension weights. After the data passed through the fully connected layer, the weight of each channel was multiplied with the original input in the channel dimension. Finally, the adjusted feature map was obtained.

Channel attention is defined as follows:

$$z_k = \frac{1}{H \times W} \sum_{i}^{H} \sum_{j}^{W} d_k(i, j) \tag{10}$$

$$\hat{z} = sigmoid(w_1 z), w_1 \in R^{c \times c} \tag{11}$$

$$\hat{D}_{cSE} = [\sigma(\hat{z_1})d_1, \cdots, \sigma(\hat{z_c})d_c] \tag{12}$$

The input feature map is $D = [d_1, d_2, \ldots, d_c]$, where each channel is $d_i \in R^{H \times W}$. D is converted to $z \in R^{1 \times 1 \times C}$ after going through a global pooling layer (GAP). In Formula (10), z_k is the global spatial information in each channel obtained through the global pooling layer. In Formula (11), global spatial information is transformed into channel weight values $\hat{z}$ (between 0 and 1) by sigmoid and fully connected layers. As the network continues to train, the input feature map is adaptively adjusted to emphasize important channels.

The sSE in the method referred to the compression of channel information and the adjustment of the weight of the spatial dimension. After the data passed through the fully connected layer, the weight of each point in the spatial dimension was multiplied with the original input in the spatial dimension. Finally, the adjusted feature map was obtained.

Spatial attention is defined as follows:

$$q = sigmoid(D \times w_1), q \in R^{H \times W} \tag{13}$$

$$\hat{D}_{sSE} = [\sigma(q_{1,1})d_{1,1}, \cdots, \sigma(q_{h,w})d_{h,w}] \tag{14}$$

$$D_{out} = \hat{D}_{cSE} + \hat{D}_{sSE} \tag{15}$$

The input feature maps are $D = [d_{1,1}, d_{1,2}, \ldots, d_{h,w}]$, where each $d_{i,j} \in R^{1\times1\times C}$. In Formula (13), D is converted into a spatial weight value q (between 0 and 1) after passing through the convolutional network and the activation function Sigmoid. Finally, the channel attention feature map and the spatial attention feature map are added to obtain the weighted feature map D_{out}. In addition, D_{out} is the output after information filtering.

3.5. Down-Sampling Convolution Module

The down-sampling convolution module in this paper is a multilayer module. Multiple resolution temporal features are captured by down-sampling convolutions to improve model performance. The down-sampling convolution module consists of basic down-sampling blocks, each D-s Block contains a structure on the right of the figure. The last layer is connected via Concat. FC is a fully connected layer, which maps the output to the specified dimension, as shown on the left of Figure 2. The output of the previous module is used as the input of the binary tree structure, and multiple spatiotemporal short-sequence information is obtained after passing through L layers. The short-sequence information is rearranged into new deep spatiotemporal sequence information. Through residual connection, the original input is added to the newly extracted deep spatiotemporal information sequence to ensure the integrity of the information. A basic down-sampling block of a binary tree structure consists of sequence segmentation, convolution filtering, and information crossing. The output data of the previous module is divided into two subsequences in the down-sampling component as shown in Figure 2 right. The output D_{out} of the graph convolution module is used as input, and then each layer takes the output of the upper layer as input. D_{even} and D_{odd} represent sequences with odd and even subscripts, respectively. Conv1, Conv2, Conv3, and Conv4 are four identical 1D convolutional networks, respectively. The exp function is used to highlight peaks of information. Hadamard product is represented by $\odot$, which is the multiplication between the elements. Finally, the output of the current layer is obtained by subtracting the two sequences. Different convolutional filters are used to extract new sequence information from each sequence. In order to avoid the loss of information caused by dividing the sequence multiple times, the important information of multiple subsequences is preserved through information crossing.

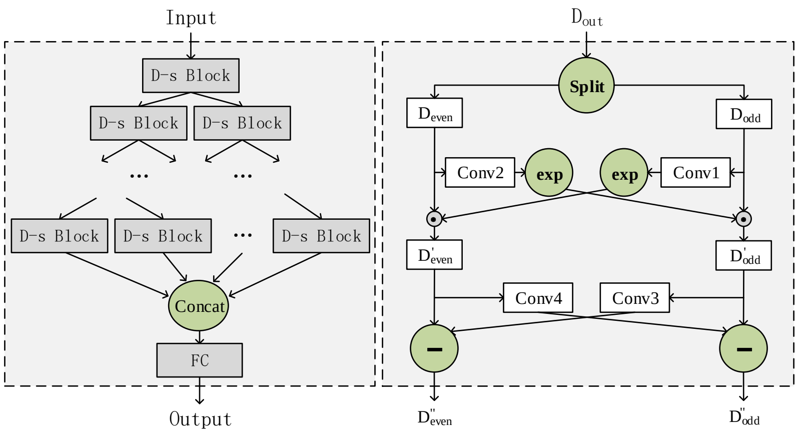


Figure 2. Down-sampling Convolution Module and Down-sampling block (D-s Block).

The input sequence is decomposed into odd and even sequences by the sequence decomposition operation in the down-sampling block of Figure 2. The odd sequences take data at positions 1, 3, 5, etc., in the input sequence. The even sequences take data at positions 0, 2, 4, etc., in the input sequence. The original sequence D_{out} is decomposed into two subsequences D_{even} and D_{odd}. Odd and even sequences reduce the amount of data while retaining most of the information of the original sequence. After that, different convolutional networks are employed to extract the feature information of the input sequence from the two sequences. The feature information obtained by different convolutional networks has stronger representation ability after fusion:

$$D_{even}, D_{odd} = split(D_{out}) \tag{16}$$

$$D'_{even} = D_{even} \odot \exp(conv_1(D_{odd})), \quad D'_{odd} = D_{odd} \odot \exp(conv_2(D_{even})) \tag{17}$$

$$D''_{even} = D'_{even} - conv_3(D'_{odd}), \quad D''_{odd} = D'_{odd} - conv_4(D'_{even}) \tag{18}$$

In Formula (16), D_{out} is split into D_{even} and D_{odd}. In Formula (17), D_{even} and D_{odd} are mapped to the hidden state through two convolutional networks and converted to exp format. In Formula (18), two other convolutional networks map D'_{even} and D'_{odd} to new hidden states, which are subtracted to obtain the final result. The mapping process of the hidden state can be viewed as a scaling transformation of the two subsequences. The scaling factor is learned during network training.

The down-sampling convolution module enlarges the receptive field compared to the dilated convolution used in the WaveNet architecture. More importantly, after the input is divided into two subsequences, the temporal information in D_{even} and D_{odd} are fused by different one-dimensional convolutions. In this process, the integrity of time information is ensured, and the ability to represent information is enhanced. The down-sampling convolution module consists of several basic modules. Among them, the basic module as a whole presents a tree structure. The information in the basic module is accumulated layer by layer. The deep feature information contains the small-scale temporal information transmitted by the shallow layer. In this way, both short-term and long-term dependencies of time series can be captured. In the last layer of the module, all the subsequences are recombined by inverse odd sequence and even sequence segmentation to obtain a new sequence representation. The new sequence information is fused with the original sequence through residual connection to ensure the integrity of the sequence information. Map to the specified output dimension using a fully connected layer.

The output part consists of skip connections and two 1×1 convolutional networks in this paper. Skip connections normalize the output of the down-sampling convolution module to have the desired predicted sequence length. A 1×1 standard convolutional network is used to convert the channel dimension to the desired dimension.

3.6. Experiment Model

The model structure of MDST-GNN is shown in Figure 3. It consists of four parts: a graph learning module, K temporal convolution modules, K graph convolution modules, and a down-sampling convolution module. The graph learning module is used to construct the spatial dependency graph of the data. The temporal convolution module captures the temporal information of the data. The spatiotemporal information is obtained by fusing the dependency graph and temporal information in the graph convolution module. The down-sampling convolution module extracts deep spatiotemporal information. Add residual connections to the model to avoid vanishing gradients during training. A skip connection is added after each temporal convolution module. To obtain the final output, the hidden features are projected onto the desired output dimension by convolution. Algorithm 1 shows the algorithm steps of the model.

Algorithm 1: Algorithm steps of the model.

```
   Input: Dataset X, initialize model parameters, batch size B, the dimension of the
          variable N, time series length T, channel dimension C.
   Output: Ŷ. Predicted result of train/test data X after h steps.
   // Preprocessing of data:
1   (a) Normalize each sequence of a multivariate time series X.
2   (b) Divide X into training set (60%),validation set (20%) and test set (20%).
   // Fitting of Model along with estimation:
3  while not at end of epoch do
4      constructure learning graph A;
5      for $x \in R^{B \times C \times N \times T}$ in X $\in$ (train, valid) do
6          Get high-dimensional features of Xtrain from channel dimension;
7          for L in Layers do
8              add skip connection;
9              The temporal features of x are extracted by temporal convolution;
10             The spatiotemporal features are obtained by fusing the temporal
                 features with A through graph convolution;
11             Layer Norm normalized spatiotemporal features;
12         end
13         A downsampling convolution operation is performed to extract deep
             spatiotemporal information;
14         Concatenate deep spatiotemporal information and skip connections
             information;
15         Forecast the future time series $\hat{y}$;
16         Reverse normalize $\hat{y}$;
17         Compute loss and gradient;
18         Update model parameters by back propagation;
19     end
20 end
   // Prediction:
21 Ŷ = predict(x sample a batch from X), $X \in$(test dataset)
```

- Input: The raw data X of the multivariate time series.
- Output: The prediction result of data X after h (Horizon) steps. The value of h can take 3, 6, 12, and 24.
- Line 1–2: Preprocess the input raw data X. Normalize each sequence of a multivariate time series X. Divide X into training set (60%), validation set (20%), and test set (20%). The Xtrain and Ytrain are obtained by splitting the sequence X with a fixed-length sliding window. Xtrain represents the input data for training, and Ytrain represents the labels of the training data.
- Line 4: Construct the graph structure A of the multivariate time series. A represents the dependencies between different sequences in a multivariate time series by an adjacency matrix.
- Line 6: A 1×1 convolutional network is used to obtain high-dimensional data in the channel dimension of Xtrain.
- Line 8: Add skip connection method to preserve part of the original information of each layer.
- Line 9: Perform a temporal convolution operation on the input data to obtain the temporal information of the data.
- Line 10: In the graph convolution, spatially weighted spatiotemporal information is obtained by multiplying the temporal information with an adjacency matrix A.
- Line 11: The output data are normalized using Layer Norm. Repeat Line 8 to Step 11 for L times. L represents the number of layers of the module.

- Line 13: Perform down-sampling convolution operation on the obtained spatiotemporal information to extract deep spatiotemporal information.
- Line 14–15: Concatenate deep spatiotemporal information and skip connections information. The final output result is obtained through 1 × 1 convolutional dimensionality reduction information.
- Line 16–18: Reverse normalize the output of the model. The mean value (MAE) of the difference between the output and the true value is used as the loss function. The model parameters are updated according to the gradient, and the steps from Step 3 to Step 10 are repeated until the model finally converges to the minimum error.
- Line 21: Finally, input the test set into the trained model to obtain the prediction result.

The above process can be divided into 4 parts. Line 1 to 2 is the data preprocessing part. Lines 4 to 15 are the MDST-GNN model training and fitting process. Line 16 to 18 is the loss function calculation part. Line 21 gives the final prediction result. The source code of the algorithm is available at https://github.com/yiminghzc/MDST-GNN (accessed on 28 May 2022).

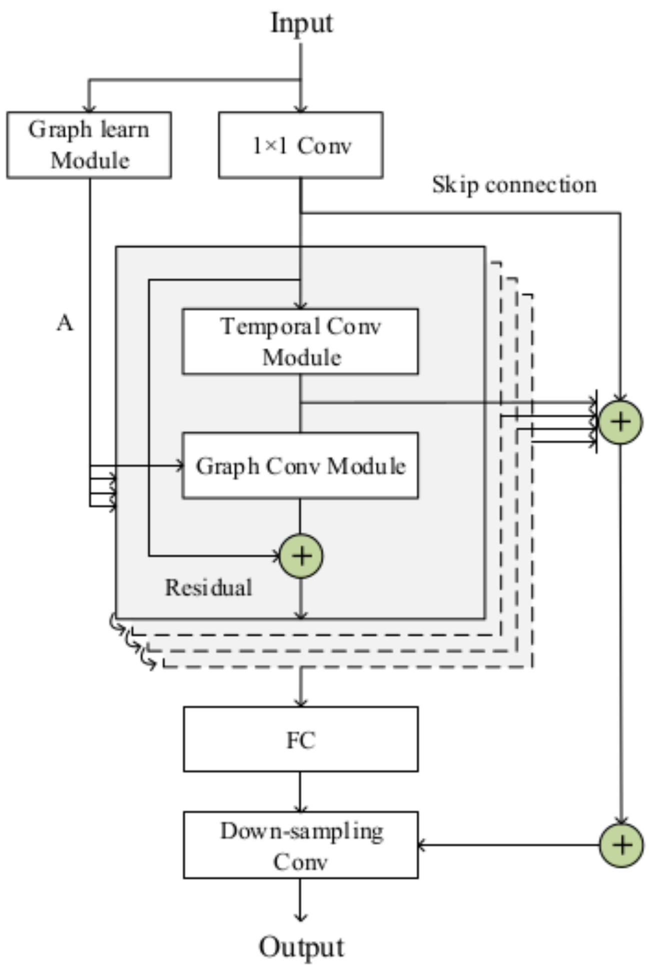


Figure 3. Model structure diagram of MDST-GNN.

4. Experiment Analysis

4.1. Experiment Dataset Analysis

In order to evaluate the effectiveness and generalization ability of the model, a classical multivariate time series dataset is adopted in this paper. The dataset comes from the experimental dataset given by LSTNet [19]. It contains multivariate time series datasets in four different domains:

- Solar Energy: Solar power generation data from Alabama State PV plants in 2006, sampled every 10 min from 137 PV plants;
- Traffic: Hourly road occupancy (between 0 and 1) on San Francisco Bay Area highways for 48 months (2015–2016) recorded by the California Department of Transportation, including data from 862 sensor measurements;
- Electricity: The hourly electricity consumption (kWh) of 321 users from 2012 to 2014;
- Exchange Rate: Daily exchange rate records for 8 countries (Australia, British, Canada, China, Japan, New Zealand, Singapore, and Switzerland) from 1990 to 2016.

Table 1 presents the statistical information of the experimental dataset in this paper. It includes four data sets. Time length represents the time length of a sequence. Variable represents the number of sequences in a multivariate sequence, and Sample rate represents the time interval of data recording. For example, there are 862 sequences with a length of 17,544 in the Traffic dataset, and the sequence data are recorded at an hourly interval.

Table 1. Dataset Statistics.

Dataset	Time Length	Variable	Sample Rate
Solar Energy	52,560	137	10 min
Traffic	17,544	862	1 h
Electricity	26,304	321	1 h
Exchange Rate	7588	8	1 day

The dataset contains linear and nonlinear interdependencies. The characteristics of the datasets are shown by selecting two variables from each dataset. The power consumption values of the two users at different times is shown in Figure 4. The traffic road occupancy values of the two roads at different times are shown in Figure 5. The horizontal axis represents the different time periods of each day. The vertical axis represents electricity consumption and road occupancy during this period.

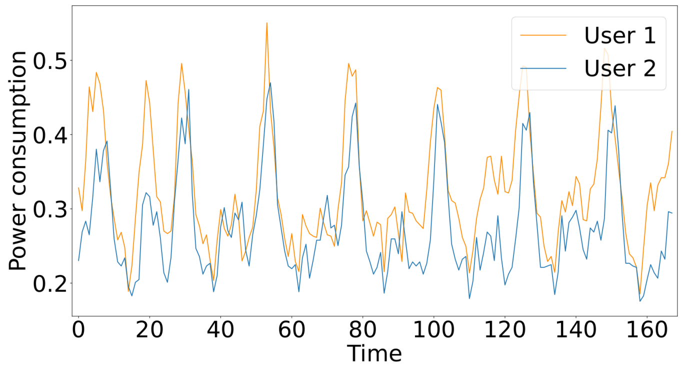


Figure 4. Consumer electricity consumption per hour.

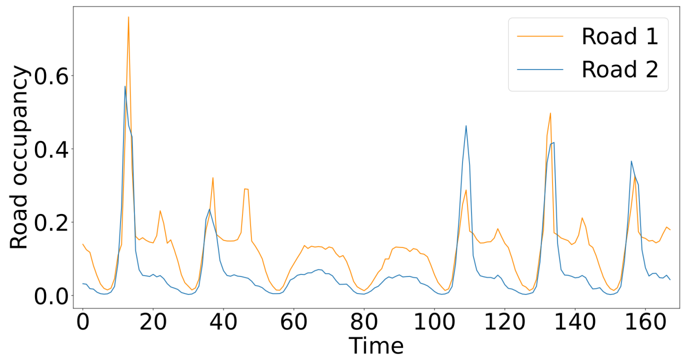


Figure 5. Road occupancy rate per hour.

The power generation values recorded every 10 min at both power stations are shown in Figure 6. Daily exchange rate values for the two countries are shown in Figure 7. The horizontal axis represents the time span in different units. The vertical axis represents power generation and exchange rate values during this period.

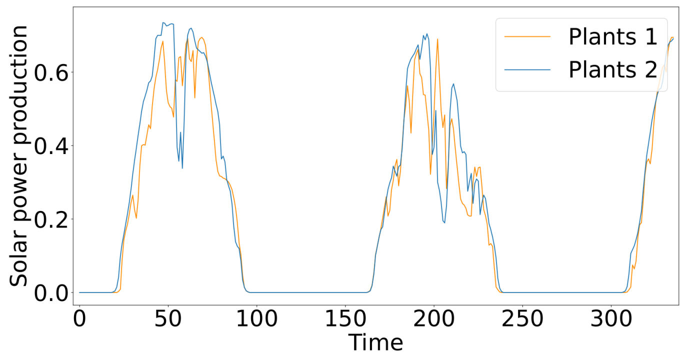


Figure 6. Solar power generation per 10 min.

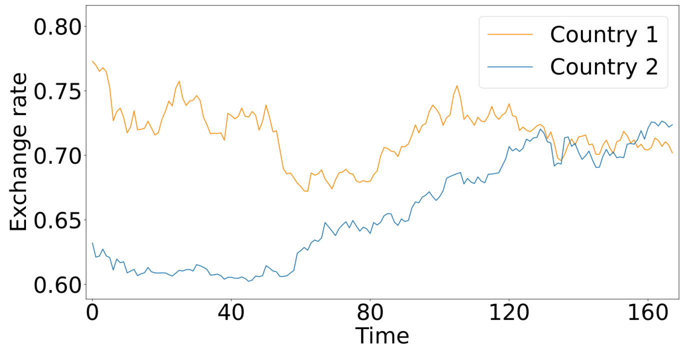


Figure 7. Daily exchange rates for two countries.

From the above data analysis, the data characteristics of the four data sets can be found. As can be seen from the figure, the Traffic, Electricity, and Solar Energy datasets have strong periodic patterns. The time series periods of multiple variables are not exactly the same, but the values of multiple variables at different periods are very similar. The above data analysis results are instructive to this paper. A multivariate time series forecasting model MDST-GNN is proposed in this paper, which can fully utilize the similarity information in the data. Information filtering is enhanced by incorporating attention methods, and the down-sampling convolution module is added to improve the deep information extraction ability. The relationship between multiple variables is modeled through a graph neural network to capture the dependency information between variables. The dependency information obtained by the graph neural network can enhance the forecasting ability of periodic time series.

4.2. Methods for Comparison

In order to effectively evaluate the experimental model and show the performance difference between different methods, the current state-of-the-art methods are compared with the model in this paper. The performance of the model on different dataset is shown by the diversity comparison of different methods.

The methods in our comparative evaluation are the follows:

- VAR-MLP: Hybrid model of multilayer perceptron (MLP) and autoregressive model (VAR) [13];
- GRU: Variant GRU model based on long short-term memory network;
- LSTNet: A hybrid model of deep neural network composed of convolutional neural network and recurrent neural network [19];

- TPA-LSTM: Recurrent Neural Network Model Based on Attention Mechanism [20];
- MTGNN: Hybrid model based on graph neural network and convolutional neural network [31];
- SCINet: Convolutional neural network model based on parity sequence segmentation and intersection [22].

4.3. Metrics

To evaluate the performance of our proposed MDST-GNN model, the same evaluation metrics are used for the comparison methods in this paper. The pros and cons of different methods can be clearly displayed by the same evaluation metrics. The evaluation metrics are the relative root mean square error (RRMSE) and the empirical correlation coefficient (CORR):

$$\mathrm{RRMSE} = \frac{\sqrt{\sum_t \frac{1}{n} \sum_{i=0}^{n} \left(Y_{ti} - \hat{Y}_{ti}\right)^2}}{\sqrt{\sum_t \frac{1}{n-1} \sum_{i=0}^{n} \left(Y_{ti} - (\overline{Y})\right)^2}}, t \in \Omega_{Test} \tag{19}$$

$$\mathrm{CORR} = \frac{1}{n} \sum_{i=1}^{n} \frac{\sum_t \left(Y_{ti} - (\overline{Y_i})\right)\left(\hat{Y}_{ti} - \left(\overline{\hat{Y}_i}\right)\right)}{\sqrt{\sum_t \left(Y_{ti} - (\overline{Y_i})\right)^2 \left(\hat{Y}_{ti} - \left(\overline{\hat{Y}_i}\right)\right)^2}}, t \in \Omega_{Test} \tag{20}$$

In Formulae (19) and (20), Y and $\hat{Y}$ are the truth value and the predicted value, respectively. $\overline{Y}$ and $\overline{\hat{Y}}$ represent the mean of the truth and predicted values. Ω_{Test} represents numerical computation using the test set. For RRMSE, lower values are better, and for CORR, higher values are better.

4.4. Experimental Setup

This section describes the hardware and software environment of the experiment and the configuration of related parameters. Our models are built with the Pytorch open source deep learning library. The device configuration used in the experiment is Intel core i5 10400F 2.9 GHz, the GPU is NVIDIA GeForce RTX 3060 12 G, and the memory is 16 GB.

The four datasets are divided by time into training set (60%), validation set (20%), and test set (20%) in this paper. According to the performance difference of the model among different hyperparameters, the optimal hyperparameters are selected to validate the performance of the model on the test set.

The key hyperparameter settings of MDST-GNN are shown in Table 2. In this paper, the model adopts four graph convolution modules, four temporal convolution modules, and one down-sampling convolution module. Adam is used as the optimizer, and the gradient clipping is 5. The network model is converged to the optimal solution by the learning rate decay technique. The input sequence window length is 168, and the output sequence length is 1. Train the model to predict future target intervals (Horizon) 3, 6, 12, and 24. The starting 1×1 convolution has 1 input channel and 16 output channels. The activation function saturation rate of the graph learning module is 3, and the node-embedding dimension is 40. In the dataset Electricity, Traffic, and Solar Energy, the Number of neighbors (K of Formula (5)) of each node is set to 20. Neighboring node information is not considered for the Exchange Rate. The dilation factor of the convolutional network in the temporal convolution module is 2. A dropout of 0.3 is employed after each temporal convolution module. Layer Norm is used after each graph convolution module. The propagation depth of the Attention Mixed propagation layer is 2. The information retention rate of the propagation layer is 0.05. Both the graph convolution module and the temporal convolution module have 16 output channels. The number of layers (Num layers) of the down-sampling convolution module is shown in Table 2. The input channel of the down-sampling convolution module is 16, and the dropout is 0.3. Two skip connection layers have 32 output channels. In the output part of the model, two standard 1 × 1 convolutional networks with 64 output channels and 1 output channel are used. The training Epoch is 30. More detailed hyperparameters can be found in Table 2.

Table 2. Hyperparameter Settings.

Model Setting	**Solar Energy**				**Traffic**				**Electricity**				**Exchange Rate**			
Horizon	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
Batch size	8				16				32				4			
Learning rate	1×10^{-4}				1×10^{-4}				1×10^{-3}				5×10^{-4}			
Window length	168				168				168				168			
Layers	4				4				4				4			
Num nodes	137				862				321				8			
Weight decay	1×10^{-4}				1×10^{-5}				1×10^{-4}				1×10^{-4}			
Num layer	4				3				3				3			

Hyperparameters are obtained by changing the research parameters and fixing other parameters to obtain the optimal hyperparameters in this paper. Each experiment was repeated 5 times with 30 Epochs per run, and the hyperparameter that minimized the RRMSE among the 5 experiments was chosen. We investigate the hyperparameters that affect the results of the MDST-GNN model. The number of neighbors and layers were selected from the parameters. The parameters are analyzed by the results obtained from 5 experiments on the dataset Exchange Rate. Figure 8 shows the experimental results of the parameters. It can be seen from Figure 8a that fewer related nodes have better results, which indicates that the spatial dependence among multiple exchange rates is weak, and increasing the information of adjacent nodes will only increase the noise. Figure 8b shows that increasing the number of layers can achieve good results, but the change in the RRMSE tends to be flat when the number of layers is greater than 4, which is because the overfitting phenomenon occurs when the number of layers is too large.

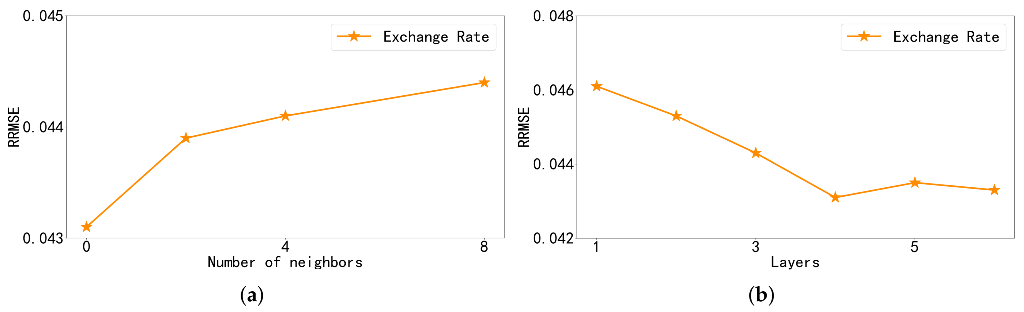


Figure 8. Number of neighbors and Layers parameter analysis in Exchange Rate. Left (**a**) is the RRMSE result of the parameter Number of neighbors taking 0, 2, 4, and 8 respectively, and right (**b**) is the RRMSE result of the parameter Layers from 0 to 6.

4.5. Results

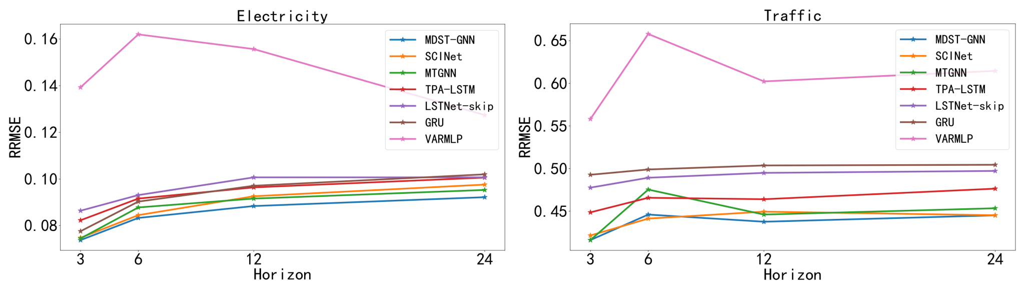
The experiments are carried based on four test sets for the validation of the MDST-GNN model. And the evaluation results of all methods are summarized in Table 3. The horizon is set to 3, 6, 12, and 24 to predict the value after a specified time period in the future. For example, it represents forecasting 3 to 24 h into the future for electricity and traffic values in the Traffic and Electricity datasets. In Solar Energy, it means forecasting solar energy values 30 to 240 min into the future. In Exchange Rate, it represents the forecasted exchange rate value for the next 3 to 24 days. When the value of Horizon is larger, the prediction task is more difficult. The optimal results for forecast are highlighted in the table in bold black, and the suboptimal results are shown in red. Finally, the comparison results between the MDST-GNN model and other related models are shown in Table 3.

Table 3. Comparison of forecasting methods for multivariate time series methods.

Dataset		Solar Energy				Traffic				Electricity				Exchange Rate			
		Horizon				Horizon				Horizon				Horizon			
Methods	Metrics	3	6	12	24	3	6	12	24	3	6	12	24	3	6	12	24
VARMLP	RRMSE	0.1922	0.2679	0.4244	0.6841	0.5582	0.6579	0.6023	0.6146	0.1393	0.162	0.1557	0.1274	0.0265	0.0394	0.0407	0.0578
	CORR	0.9829	0.9655	0.9058	0.7149	0.8245	0.7695	0.7929	0.7891	0.8708	0.8389	0.8192	0.8679	0.8609	0.8725	0.828	0.7675
GRU	RRMSE	0.2058	0.2832	0.3726	0.464	0.4928	0.499	0.5037	0.5045	0.0776	0.0903	0.0971	0.102	0.0195	0.0258	0.0347	0.0447
	CORR	0.98	0.9601	0.9289	0.8857	0.851	0.8465	0.8431	0.8428	0.9412	0.9222	0.9088	0.9079	0.9768	0.9686	0.9534	0.9355
LSTNet-skip	RRMSE	0.1843	0.2559	0.3254	0.4643	0.4777	0.4893	0.495	0.4973	0.0864	0.0931	0.1007	0.1007	0.0226	0.028	0.0356	0.0449
	CORR	0.9843	0.969	0.9467	0.887	0.8721	0.869	0.8614	0.8588	0.9283	0.9135	0.9077	0.9119	0.9735	0.9658	0.9511	0.9354
TPA-LSTM	RRMSE	0.1803	0.2347	0.3234	0.4389	0.4487	0.4658	0.4641	0.4765	0.0823	0.0916	0.0964	0.1006	0.0174	**0.0241**	0.0341	0.0444
	CORR	0.985	**0.9742**	0.9487	0.9081	0.8812	0.8717	0.8717	0.8629	0.9439	0.9337	0.925	0.9133	0.979	0.9709	0.9564	0.9381
MTGNN	RRMSE	0.1778	0.2348	0.3109	0.427	0.4162	0.4754	0.4461	0.4535	0.0745	0.0878	0.0916	0.0953	0.0194	0.0259	0.0349	0.0456
	CORR	0.9852	0.9726	0.9509	0.9031	**0.8963**	0.8667	0.8794	0.881	0.9474	0.9316	0.9278	0.9234	0.9786	0.9708	0.9551	0.9372
SCINet	RRMSE	0.1775	**0.2301**	**0.2997**	**0.4081**	0.4216	**0.4414**	0.4495	0.4453	0.0748	0.0845	0.0926	0.0976	0.018	0.0247	0.034	0.0442
	CORR	0.9853	0.9739	**0.955**	**0.9112**	0.892	**0.8809**	0.8772	**0.8825**	**0.9492**	**0.9386**	**0.9304**	**0.9274**	0.9739	0.9662	0.9487	0.9255
MDST-GNN	RRMSE	**0.1764**	0.2321	0.3082	0.4119	**0.4162**	0.4461	**0.4377**	**0.4452**	**0.0738**	**0.0833**	**0.0884**	**0.0922**	**0.0172**	0.0245	**0.0337**	**0.0431**
	CORR	**0.9855**	0.9735	0.9519	0.9103	0.8958	0.8803	**0.8841**	0.8792	0.9454	0.9346	0.9264	0.9222	**0.9811**	**0.9727**	**0.9578**	**0.9392**

Note: The predicted optimal results are in bold black, and the suboptimal results are shown in red.

We plot the RRMSE results for all methods of Electricity and Traffic, as shown in Figure 9. From the figure, it can be found that the method using neural network significantly outperforms the VARMLP model in the dataset. Obviously, neural networks have more advantages in time series forecasting. Subsequent models using neural networks have improved their prediction accuracy. Among them, MTGNN combined with spatial information outperforms LSTNet and TAP-LSTM models in results. The SCINet model also has good results in some datasets. Figure 9a shows that the RRMSE of multiple models is relatively stable, and the prediction accuracy of each model is improved. At the same time, Figure 9b can see that the MTGNN model fluctuates when the horizon is 6, which indicates that the spatial dependence of the dataset is weak when the horizon is 6, which interferes with the model. Compared with MTGNN, the MDST-GNN model has smaller fluctuations in this part, which shows that the attention method plays a role in filtering information and reduces the interference of irrelevant information. When the horizon is other values, the RRMSE tends to be stable.



(**a**) (**b**)

Figure 9. RRMSE comparison of all methods in Electricity and Traffic. Left (**a**) is the RRMSE result of all methods in Electricity, right (**b**) is the RRMSE result of all methods in Traffic, and horizon takes 3, 6, 12, and 24 respectively.

The RRMSE metric of our model is improved by 1%, 1.4%, 3.5%, and 3.3% on the Electricity dataset compared to the state-of-the-art method. Furthermore, on the Traffic dataset, when the horizon is 12, the RRMSE metric is 1.9% higher than the best baseline method, which proves the effectiveness of the model. More importantly, forecasts have also improved for noncyclical exchange rate data. This is because the change trend of the exchange rate is relatively stable. In this paper, the temporal information of the exchange rate is well captured by the model's deep information extraction. It is demonstrated that the framework can capture deep spatiotemporal information well, even in data with weak periodicity.

To illustrate the effectiveness of MDST-GNN in modeling spatiotemporal features in time series data, Figures 10–12 show the performance of the MDST-GNN on a specific time series (one of the output variables). Our model obtains results that are consistent with the truth value in the periodic pattern of the data. The comparative prediction results and truth values on the test set Electricity are shown in Figure 10. The comparison of the prediction results on the test set Traffic is shown in Figure 11. The comparison of the prediction results on the test set Solar Energy is shown in Figure 12. Our model achieves excellent performance on prediction tasks. The error is small between the predicted result and the truth value.

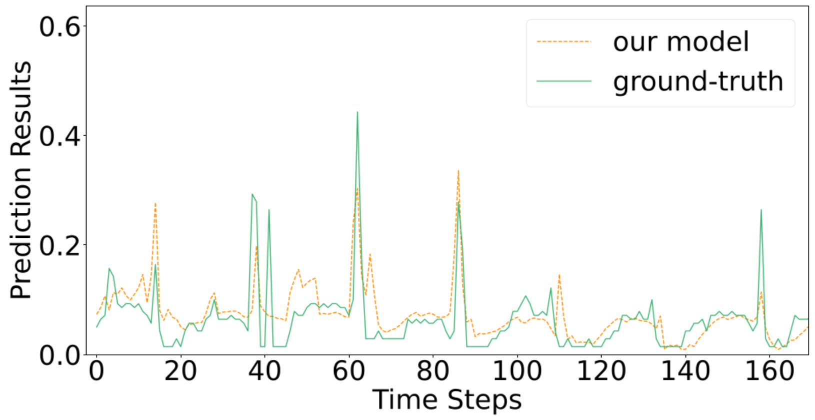


Figure 10. Electricity (Horizon = 24) forecast results.

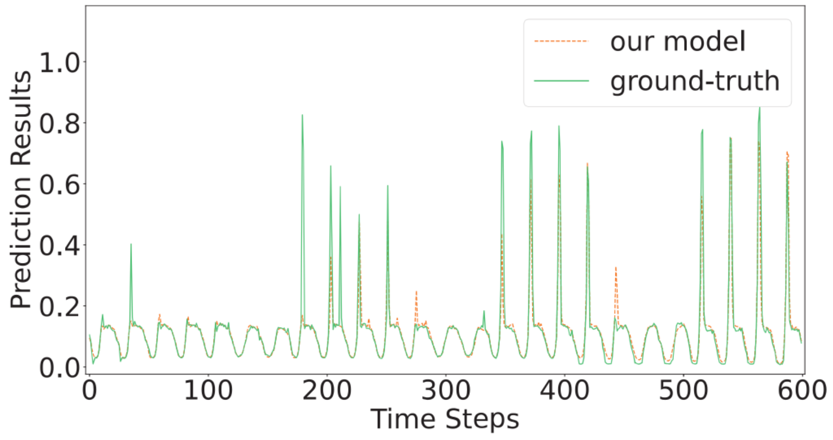


Figure 11. Traffic (Horizon = 24) forecast results.

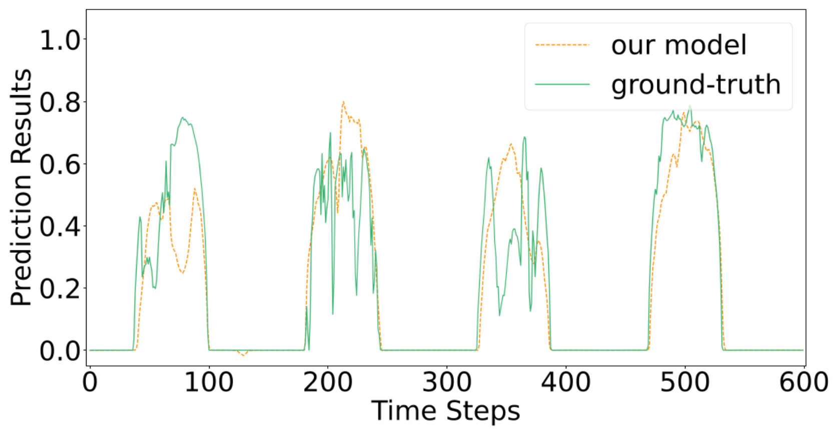


Figure 12. Solar Energy (Horizon = 24) forecast results.

However, the model performed slightly worse on the prediction task of Solar Energy data. As can be seen from Figure 12, the periodic pattern is captured by the model well, but there is an error between the results and the truth value within the periodic time. This is because the fluctuation in the data at a certain moment causes the periodic information to be disturbed. In Figure 6, it can be seen that the data remain periodic, but the data fluctuate greatly in a certain period of time. At the same time, the power generation of solar power plants is also affected by many other factors. For example, the local weather and temperature.

Overall, the model is more dominant on most tasks, achieving excellent performance on the exchange rate, traffic, and electricity datasets.

4.6. Ablation Study

A model combining an attention mechanism and time series segmentation is proposed in this paper. In this section, an ablation experiment is performed on this model, and the effectiveness of the above method will be verified. Among them, the models containing different modules are named as follows:

- w/o A: The attention mechanism part is removed from the information selection process of the graph convolution module. The output of the information selection step is passed directly to the next section.
- w/o S: Remove the down-sampling convolution module from the model and replace that part with a fully connected layer.

The effectiveness of the module is verified by removing the attention method and the down-sampling convolution module. The results of the ablation experiments on the test set are shown in Table 4. Ablation experiments verify the effectiveness of the modules in four datasets. The best results are shown in bold black. The overall model outperforms the ablation model by comparison. In Table 4, the attention method has advantages on Electricity and Traffic, and the down-sampling convolution method is more effective on Exchange Rate and Solar Energy. The effectiveness of integrating these two parts into the model is verified in Table 4. Among them, the data are filtered by the attention method, so that the data retain more important information and enhance the robustness. Down-sampling convolution captures the deep spatiotemporal information of the sequence and enhances the predictive power of the data on cyclical trends. Overall, the accuracy of model predictions is improved. To verify the effectiveness of the module, we plot the one-dimensional time series (one variable) of the Electricity test set in Figure 13, where the green curve is the real data and the yellow dashed line is the predicted value. We can see that the full model captures the spikes in the data better, and the w/o S model does not capture this change.

Table 4. Ablation study (Horizon = 24).

Dataset	Metrics	w/o A	w/o S	MDST-GNN
Solar Energy	RRMSE	0.4279	0.429	**0.4119**
	CORR	0.903	0.9042	**0.9103**
Traffic	RRMSE	0.4584	0.4577	**0.4452**
	CORR	0.8732	0.87	**0.8792**
Electricity	RRMSE	0.0952	0.095	**0.0922**
	CORR	0.9212	0.9213	**0.9222**
Exchange Rate	RRMSE	0.0444	0.0459	**0.0431**
	CORR	0.939	0.9344	**0.9392**

Note: The optimal results are in bold black.

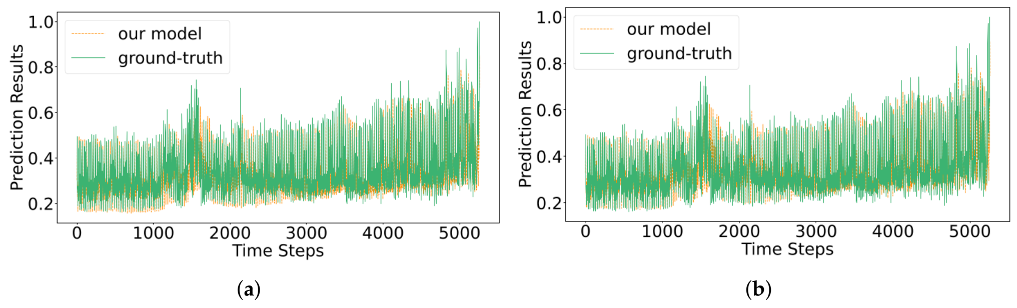


Figure 13. Time series (one variable) predicted and true values for the Electricity test set (Horizon = 24). Left (**a**) is the w/o S model result, right (**b**) is the result of the full model.

In summary, time series forecasting involves temporal, spatial, and periodic information. Of course, introducing the modules proposed in this paper will incur additional computational costs. However, it is very helpful to consider the above information. The improvement of this information on the prediction task is demonstrated by the results of the ablation experiments.

5. Conclusions

Multivariate time series have complex spatiotemporal information, and there are dependencies among multiple series. In addition, the time series has a periodic pattern. According to the characteristics of multivariate time series, a multivariate time series deep spatiotemporal forecasting model with a graph neural network (MDST-GNN) is proposed in this paper. It comprises four modules: graph learning, temporal convolution, graph convolution, and down-sampling convolution. In order to deal with complex information better, the model adopts a graph neural network to transform complex information into processable graph information. Furthermore, the model utilizes a temporal convolutional network to extract temporal information in the predicted sequence. Graph convolutional networks are used to extract spatiotemporal information of predicted sequences. The attention method is added to improve the information filtering ability. Down-sampling convolutional networks are used to extract deep spatiotemporal information. The MDST-GNN model can filter out irrelevant dependency information between graph nodes and is more suitable for multivariate time series forecasting. The prediction accuracy of data peaks can be enhanced by capturing deep spatiotemporal information. In experiments, the proposed model predicts future values at different intervals for four datasets. Based on the above discussion and experimental results, the advantages of MDST-GNN are verified in the experimental results on four sets of public datasets, and the attention method and down-sampling convolutional network proposed in this paper improve the prediction accuracy of the model.

The MDST-GNN model has achieved significant improvements on multiple datasets, but there are still certain problems in predicting data with large short-term fluctuations such as Solar Energy. This is mainly caused by the insensitivity of convolutional networks to local fluctuations and the lack of long-term memory information. At the same time, the model has many parameters, and how to reduce the model parameters while maintaining good prediction performance is a problem worth pondering. Our next research will revolve around the problem of model lightweight and insensitivity to local fluctuations.

Author Contributions: Conceptualization, Z.H., C.Z. and Y.H.; data curation, Z.H.; methodology, Z.H. and C.Z.; software, Z.H.; supervision, C.Z. and Y.H.; validation, Z.H.; writing—original draft, Z.H. and C.Z.; writing—review and editing, Z.H., C.Z. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (Nos. 61862062, 61104035).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are openly available in Github (https://github.com/laiguokun/multivariate-time-series-data, accessed on 2 February 2022).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Taylor, J.W.; McSharry, P.E. Short-Term Load Forecasting Methods: An Evaluation Based on European Data. *IEEE Trans. Power Syst.* **2007**, *22*, 2213–2219. [CrossRef]
2. Du, S.; Li, T.; Yang, Y.; Horng, S.J. Deep Air Quality Forecasting Using Hybrid Deep Learning Framework. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 2412–2424. [CrossRef]
3. Chen, J.L.; Li, G.; Wu, D.C.; Shen, S. Forecasting Seasonal Tourism Demand Using a Multiseries Structural Time Series Method. *J. Travel Res.* **2019**, *58*, 92–103. [CrossRef]
4. Zhuang, D.E.H.; Li, G.C.L.; Wong, A.K.C. Discovery of Temporal Associations in Multivariate Time Series. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 2969–2982. [CrossRef]
5. Akdi, Y.; Glveren, E.; Nlü, K.; Yücel, M. Modeling and forecasting of monthly PM 2.5 emission of Paris by periodogram-based time series methodology. *Environ. Monit. Assess.* **2021**, *193*, 622. [CrossRef] [PubMed]
6. Box, G.E.P.; Jenkins, G.M.; MacGregor, J.F. Some Recent Advances in Forecasting and Control. *J. R. Stat. Soc. Ser. Appl. Stat.* **1974**, *23*, 158–179. [CrossRef]
7. Chouhan, V.; Singh, S.K.; Khamparia, A.; Gupta, D.; Tiwari, P.; Moreira, C.; Damaševičius, R.; de Albuquerque, V.H.C. A Novel Transfer Learning Based Approach for Pneumonia Detection in Chest X-ray Images. *Appl. Sci.* **2020**, *10*, 559. [CrossRef]
8. Lin, T.H.; Akamatsu, T.; Tsao, Y. Sensing ecosystem dynamics via audio source separation: A case study of marine soundscapes off northeastern Taiwan. *PLoS Comput. Biol.* **2021**, *17*, e1008698. [CrossRef] [PubMed]
9. Li, Q.; Li, S.; Zhang, S.; Hu, J.; Hu, J. A Review of Text Corpus-Based Tourism Big Data Mining. *Appl. Sci.* **2019**, *9*, 3300. [CrossRef]
10. Li, G.; Nguyen, T.H.; Jung, J.J. Traffic Incident Detection Based on Dynamic Graph Embedding in Vehicular Edge Computing. *Appl. Sci.* **2021**, *11*, 5861. [CrossRef]
11. Simeunovic, J.; Schubnel, B.; Alet, P.J.; Carrillo, R.E. Spatio-Temporal Graph Neural Networks for Multi-Site PV Power Forecasting. *IEEE Trans. Sustain. Energy* **2022**, *13*, 1210–1220. [CrossRef]
12. Abbasimehr, H.; Paki, R. Improving time series forecasting using LSTM and attention models. *J. Ambient Intell. Humaniz. Comput.* **2022**, *13*, 673–691. [CrossRef]
13. Zhang, G. Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing* **2003**, *50*, 159–175. [CrossRef]
14. Zhang, G.; Patuwo, B.E.; Hu, M.Y. Forecasting with artificial neural networks: The state of the art. *Int. J. Forecast.* **1998**, *14*, 35–62. [CrossRef]
15. Kaur, T.; Kumar, S.; Segal, R. Application of artificial neural network for short term wind speed forecasting. In Proceedings of the 2016 Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE), Bengaluru, India, 21–23 January 2016; pp. 1–5. [CrossRef]
16. Bukhari, A.H.; Raja, M.A.Z.; Sulaiman, M.; Islam, S.; Shoaib, M.; Kumam, P. Fractional Neuro-Sequential ARFIMA-LSTM for Financial Market Forecasting. *IEEE Access* **2020**, *8*, 71326–71338. [CrossRef]
17. Cao, L.J.; Tay, F.E.H. Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Trans. Neural Netw.* **2003**, *14*, 1506–1518. [CrossRef]
18. Qin, D. Rise of VAR modelling approach. *J. Econ. Surv.* **2011**, *25*, 156–174. [CrossRef]

19. Lai, G.K.; Chang, W.C.; Yang, Y.M.; Liu, H.X. Modeling Long–Short-Term Temporal Patterns with Deep Neural Networks. In Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 95–104. [CrossRef]
20. Shih, S.Y.; Sun, F.K.; Lee, H.Y. Temporal pattern attention for multivariate time series forecasting. *Mach. Learn.* **2019**, *108*, 1421–1441. [CrossRef]
21. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
22. Liu, M.; Zeng, A.; Xu, Z.; Lai, Q.; Xu, Q. Time Series is a Special Sequence: Forecasting with Sample Convolution and Interaction. *arXiv* **2021**. [CrossRef]
23. Bai, S.; Kolter, J.Z.; Koltun, V. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv* **2018**. [CrossRef]
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Proces. Syst.* **2017**, *30*, 5999–6009. [CrossRef]
25. Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *arXiv*. [CrossRef]
26. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. [CrossRef]
27. Mai, W.; Chen, J.; Chen, X. Time-Evolving Graph Convolutional Recurrent Network for Traffic Prediction. *Appl. Sci.* **2022**, *12*, 2824. [CrossRef]
28. Cui, Z.; Henrickson, K.; Ke, R.; Wang, Y. Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4883–4894. [CrossRef]
29. Khodayar, M.; Wang, J. Spatio-Temporal Graph Deep Neural Network for Short-Term Wind Speed Forecasting. *IEEE Trans. Sustain. Energy* **2019**, *10*, 670–681. [CrossRef]
30. Qi, Y.; Li, Q.; Karimian, H.; Liu, D. A hybrid model for spatiotemporal forecasting of PM2.5 based on graph convolutional neural network and long short-term memory. *Sci. Total Environ.* **2019**, *664*, 1–10. [CrossRef]
31. Wu, Z.; Pan, S.; Long, G.; Jiang, J.; Chang, X.; Zhang, C. Connecting the dots: Multivariate time series forecasting with graph neural networks. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Virtual Event, 6–10 July 2020; pp. 753–763. [CrossRef]
32. Yu, F.; Koltun, V. Multi-scale context aggregation by dilated convolutions. *arXiv* **2016**. [CrossRef]
33. Oord, A.V.D.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. WaveNet: A generative model for raw audio. *arXiv* **2016**. [CrossRef]
34. Roy, A.G.; Navab, N.; Wachinger, C. Concurrent Spatial and Channel 'Squeeze & Excitation' in Fully Convolutional Networks. *Lect. Notes Comput. Sci.* **2018**, *11070*, 421–429. [CrossRef]