



# Spatial-temporal multi-factor fusion graph neural network for traffic prediction

Hui Jia<sup>1,2,3</sup> · Zixuan Yu<sup>1</sup> · Yanping Chen<sup>1,2,3</sup>  · Hong Xia<sup>1,2,3</sup>

Accepted: 25 June 2024 / Published online: 16 July 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

## Abstract

Traffic flow prediction, which provides dynamic data support for intelligent transportation systems, has always been a topic of great interest. Accurate prediction of traffic flow is vital for optimizing traffic management and planning and improving traffic efficiency. Existing traffic flow prediction models usually consider only a single influencing factor when modeling spatial-temporal correlation and fail to comprehensively consider the multiscale factors corresponding to each module, resulting in incomplete spatial-temporal modeling and thus affecting prediction accuracy. For this reason, this paper proposes a spatial-temporal multifactor fusion graph neural network (STMFGNN), which aims to characterize spatial-temporal features more comprehensively. First, in the spatial feature extraction module, we parallelly utilize dynamic similarity graphs and static adjacency graphs to perform graph convolution and introduce the gated fusion module to self-learn the dynamic influence weight so that our model can integrate the information of global hidden knowledge and local prior knowledge and capture the multiscale spatial dependencies between nodes. Second, the model combines gated tanh unit convolution with a multireceptive field and gated recurrent units in the temporal feature extraction module. Similarly, it utilizes the gated fusion module to adaptively and dynamically adjust the importance weights of the two components. This enables the acquisition of multiscale temporal dependency information at both short-term and long-term levels. In addition, we employ an improved generative adversarial imputation network for incomplete traffic flow data. The experimental results on four real-world datasets show that our proposed method consistently outperforms other baseline models, achieving state-of-the-art performance. The key source code and data are available at <https://github.com/Keaii3/STMFGNN>.

**Keywords** Traffic flow prediction · Spatial-temporal features · Graph convolution · Gated tanh unit · Gated recurrent units · Generative adversarial imputation network

## 1 Introduction

With the continuous expansion of urban transportation networks, increasing traffic flow, and diversifying travel modes,

accurately predicting traffic flow has become crucial for optimizing traffic management and planning. As an effective tool for alleviating traffic congestion, intelligent transportation systems (ITSs) are drawing global attention [1]. Traffic flow prediction is an important part of ITSs [2]. Given the sequence of traffic flow and roadway topology networks [3], the purpose of traffic flow prediction is to use historical data to predict future traffic flow [4, 5]. However, the road sensor nodes in road networks are spatially connected and interact with each other; traffic flow data from individual nodes form a long time series with dynamically changing time patterns. Therefore, traffic prediction has always been a challenging task due to its complex spatial-temporal features [6]. In recent years, with the continuous advancement of deep learning technologies, deep learning has been increasingly applied in emerging industrial fields, such as intelligent transporta-

Hui Jia, Zixuan Yu, Yanping Chen, and Hong Xia contributed equally to this work.

✉ Yanping Chen  
chenyp@xupt.edu.cn

<sup>1</sup> School of Computer Science & Technology, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

<sup>2</sup> Shannxi Key Laboratory of Network Data Analysis and Intelligent Processing, Xi'an 710121, China

<sup>3</sup> Xi'an Key Laboratory of Big Data and Intelligent Computing, Xi'an 710121, China

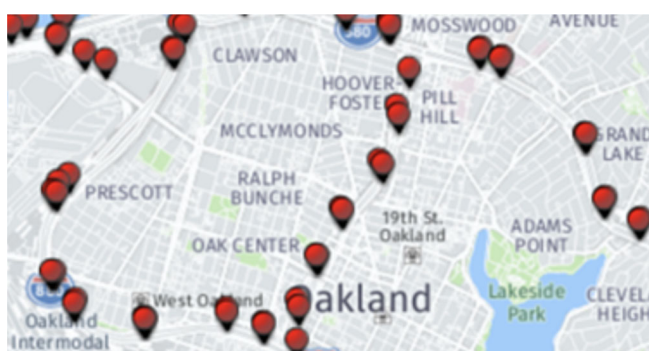
tion [7, 8], intelligent manufacturing [9–11], and medical diagnosis [12, 13]. These applications showcase the robust capabilities and broad prospects of deep learning in industrial domains.

With the continuous development of graph neural networks [14–16], determining how to effectively model graph modeling on spatial-temporal data has become a focus of attention. We believe that, as Fig. 1 shows, nodes on a road network mainly exhibit two types of spatial dependencies: one is the propagation of traffic flow due to the physical proximity between nodes; the other is the similarity of changes due to nodes with similar functions, even though they are not directly physically connected, or are even far apart. For the traffic flow on each node as time series data, there is a specific time dependence between the time slices, whether at a similar moment or at a long time interval.

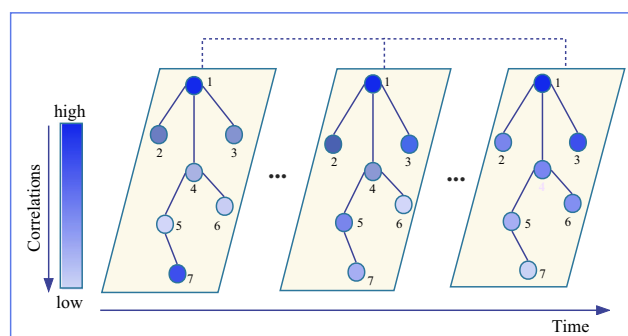
First, an obvious drawback of many previous methods is that they fix spatial relationships, such as the DCRNN [17] and ASTGCN [18], usually use static adjacent matrices (either predefined or self-learned) to represent spatial dependencies between nodes. However, there are similarities between nodes from a global perspective, and the spatial dependencies may be dynamically change over time. The fixed adjacencies cannot model the global similarities between nodes that are dynamically change over time. Recently, methods such as STFGCN [19] and DSTAGNN [20] have learned dynamic graphs through a data-driven approach to better reflect the dynamically changing spatial dependencies between nodes. However, these methods also have some drawbacks. They may overemphasize the influence of globally similar nodes on a node and ignore the traffic fluctuations caused by the changes in the traffic of actual adjacent nodes at a single time step. These methods often

use a static adjacent matrix or dynamic similarity matrix alone to model the spatial dependency between nodes but do not fully consider the impact of the joint effect of multiple factors on traffic flow prediction. Due to the mutual mapping between sensor nodes, traffic flows are complex and cyclical in the long term but sudden and volatile in the short term [21]. Based on the long-term and global scales, there are similarities between nodes, and the data-driven dynamic adjacency matrix may select the most similar node in the time unit of a day as the neighbor; in the short-term and local scales, for a single time step of a single node, the flow change of adjacent nodes in the real world at the previous time still plays a vital role in its flow fluctuation, which cannot be ignored. Neglecting the influence of any of these factors will lead to incomplete modeling of spatial dependence between nodes. Comprehensive consideration is essential to concurrently address the impact of both actual adjacent nodes and globally similar nodes on traffic flow prediction.

Second, for the data of each node in the road network, due to its nature as time-series data, we need to pay attention to its temporal dependence on both short-term and long-term scales. However, current spatial-temporal prediction studies have certain shortcomings in temporal dependence learning. STGCN [22] uses convolutional neural networks (CNNs) to capture short-term temporal dependencies, while DCRNN [17] uses recurrent neural networks (RNNs), and GraphWaveNet [23] uses diffusion convolution to attempt to capture long-term temporal dependencies. However, these methods often struggle to comprehensively consider short-term and long-term temporal correlations, which may lead to the omission of critical temporal features in temporal dependency capture.



(a)



(b)

**Fig. 1** Complex spatial-temporal correlations. The red points distributed in the road network on the map in (a) represent individual sensors, and the connectivity relationship between sensors represents the connectivity relationship between roads. The nodes in (b) represent individual sensors, and the solid line represents the real spatial neighbor relationship. Among them, node No. 1 is the primary node, and the fill

colors of the other nodes indicate their spatial correlation with node No. 1. Not only do the nodes that are adjacent to node 1 have a spatial correlation with it, but the nodes that are not directly adjacent to it may also have spatial correlation with it. The dashed line represents the temporal correlation between individual moments due to the nonlinear relationship of traffic flow data

In addition, due to sensor failure, communication errors, storage loss, and other reasons, data collected by sensors inevitably contain missing data [24]. However, complete and correct data are crucial for subsequent data analysis tasks.

In summary, existing methods often only consider the influence of a single factor when modeling spatio-temporal correlations, resulting in incomplete capture of the spatial dependence between nodes and the temporal dependence of node data. Moreover, the existing methods use only simple statistics-based methods to impute missing data in traffic datasets. The prediction accuracy of the traffic flow prediction model is not accurate enough due to the influence of these factors.

Complete traffic flow data are obtained to indirectly improve the model prediction accuracy. Moreover, to comprehensively consider the static local and dynamic global multifactor influences on spatial dependencies and capture multiscale temporal dependencies, we propose a new neural network framework based on graph convolution: the spatial-temporal multifactor fusion graph neural network (STMFGNN). In the data preprocessing stage, we improved the masking mechanism and training process of the generative adversarial imputation network (GAIN) [25] and applied it to traffic data to impute the missing traffic data. In the process of graph convolution, we used the predefined static reachability matrix and the dynamic similarity matrix generated based on the Wasserstein distance [26] to perform graph convolution and then used the dynamic gated fusion module to self-learn to fuse the two results to better capture the spatial dependence of static and dynamic interwoven traffic flow data. Similarly, to overcome the trade-off between the temporal dependence of the short-term scale and the long-term scale in time series prediction, we synergistically employed the multifield gated tanh unit (GTU) and gated recurrent units (GRUs), adjusting the influence weights of the two with a dynamic gate fusion module adaptively, aiming to comprehensively capture the short- and long-term temporal dependencies of traffic flow data. The main contributions of our work are as follows:

- In the data preprocessing stage, a modified GAIN with an improved masking mechanism and training process is proposed and applied to traffic data. The imputation of commonly missing traffic data indirectly enhances the model performance.
- A graph convolution method incorporating multiple factors is designed. The dynamic similarity graph is used while the static adjacency graph is retained, and the real prior knowledge and self-learning hidden knowledge are combined through the dynamic gated fusion module to jointly model the complex spatial dependence.
- A method combining multiscale GTU convolution and GRU is proposed. One-dimensional convolutions with

multiple fields capture varying degrees of short-term temporal dependencies, while the GRU captures long-term temporal dependencies. A dynamic gate fusion module is used to autonomously learn integrated multiscale temporal features.

- Extensive experiments on real-world traffic datasets demonstrate that our method outperforms numerous baseline models, including state-of-the-art methods, and achieves superior performance.

The rest of the paper is organized as follows: Section 2 reviews and summarizes previous works. The traffic flow prediction problem is described in Section 3, and our proposed model is introduced in detail. Section 4 describes our overall experiment, compares it with other models, and analyzes the results. An ablation study, result visualization, and parameter study follow this approach. Finally, we conclude the entire paper in Section 5.

## 2 Related work

### 2.1 Data imputation

In data completion, the imputation of time series is a critical component of solving the problem of missing traffic flow data, and there are a variety of methods, including statistical, machine learning, and deep learning methods. Statistics-based imputation methods typically use mean values, with the simplest method using historical averages, which impute missing values based on the average of the same time interval in the past. Many previous studies have shown that machine learning-based methods are very effective for data imputation. For example, a Bayesian network learns probability distributions from observed data and imputes missing data using the best fit method [27]. K-nearest neighbors (KNNs) [28] use the average of the K nearest neighbors to the missing values to repair the data. Other methods include the probabilistic principal component analysis (PPCA) [29] technique, which utilizes the daily periodicity and interval variability of traffic data. With the development of deep learning, many neural network models have also been used to solve the data imputation problem. The missing data imputation method of such models usually starts from the data distribution and fills the missing data by fitting the proper distribution. For example, literature [30] used a bidirectional RNN as a generative model to fill in missing data. A convolutional recursive autoencoder was proposed in the literature [31]. Generative adversarial networks (GANs) [32] are generative models in which two networks are trained against each other to generate new sample datasets that follow the training distribution. The GAIN [25] is a GAN-based method that uses parallel data and a GAN to enhance traffic data and uses a hint matrix

based on real observations for unsupervised data generation to fill in missing data.

## 2.2 Graph convolution

In graph convolution, graph convolutional networks generalize convolutional neural networks from standard grid-like to graph-structured data to efficiently extract local patterns in the data. Graph convolutional networks fall into two main categories: spatial-based and spectral-based [33]. Spectral-based graph convolution mainly utilizes the graph Fourier transform (GFT) to achieve convolution. [34] extended convolution to graphs in the spectral domain. ChebNet [35] uses Chebyshev polynomials instead of original polynomials, which avoids eigenvalue decomposition and arithmetic, reduces computational complexity and cost, and takes advantage of the fact that polynomial fitting can optionally use lower order numbers, which transforms computations from global to local. The graph convolution network (GCN) [36] reduces Chebyshev networks to a more straightforward form by invoking a first-order Chebyshev polynomial approximation. Spatial-based graph convolution is applied similarly to convolution in deep learning and centers on updating information by designing different strategies to aggregate neighboring features. Typical works include graph attention networks (GATs) [37], which are the first use of the attention mechanism in the graph domain to learn the weights between two nodes, and graph SAGE [38], which generates node embeddings by sampling a fixed number of neighbors and aggregating their features.

## 2.3 Traffic prediction

In traffic prediction, researchers have been developing various models and methods to predict traffic flow, congestion, travel time, and other relevant factors to improve traffic management and planning efficiency. After years of research and development, researchers have made many achievements. Classical statistical models such as ARIMA [39] and SVM [40] usually consider only temporal information and require the data to satisfy certain assumptions. However, due to the complexity of the traffic flow data, these methods do not perform well in practical applications. The ConvLSTM [41] model combines CNNs and RNNs to capture spatial and temporal correlations. To better capture the spatial correlation of data, recent works have used graph convolution to learn the spatial correlation of data. DCRNN [17] proposes diffusion graph convolution to describe the information diffusion process in spatial networks and uses an RNN to model the temporal correlation. In [42], GCN and long short-term memory (LSTM) were combined to improve the prediction accuracy. ASTGCN [18] introduces

the attention mechanism [43] before spatiotemporal convolution; it uses two attention layers to capture the dynamic spatiotemporal dependencies of neighboring nodes. Graph WaveNet [23] discovers hidden spatial dependencies through learnable node embeddings, which focus spatial dependencies on potential dynamic dependencies of information collocation. Its temporal convolutional layer captures the temporal trend of nodes through dilated causal convolution [44]. Although the transformer algorithm [45] uses a self-attention mechanism to model spatial-temporal correlations, it is prone to the problem of error accumulation. Several studies have focused on designing new graph structures. STFGCN [19] constructs a new spatial-temporal fusion graph through a data-driven approach that can provide correlations that may not be present in predefined spatial graphs and introduces a dilated convolution with a dilatation rate in temporal convolutional layers intending to capture long-range temporal dependencies. DSTAGNN [20] also uses the data-driven construction of a dynamic correlation matrix to generate a new graph structure and a temporal convolutional layer using a multi-scale GTU to capture temporal correlations.

The existing traffic prediction models usually use statistics-based methods to fill in missing data, but these methods need to be more accurate. Moreover, the existing missing data imputation networks cannot be directly applied to traffic datasets. Therefore, in this paper, we will improve the masking mechanism and training process of GAIN in the data preprocessing stage so that it can be applied to traffic datasets and fill in missing data more accurately. Second, the research mentioned above must be more comprehensive for modeling spatial and temporal correlations by considering only single-factor effects. Unlike classical methods that use only a single adjacent matrix, dynamically fusing graph convolution results of static and dynamic adjacency matrices can effectively enhance the model's ability to fully utilize spatial features. In capturing temporal dependence, a time series prediction method that combines two different methods of working can focus on both short-term and long-term scales and grasp the nonlinear change pattern of time-series data in greater depth.

## 3 Methodology

In this section, we first present a mathematical formulation of the problem addressed in this paper. We then detail the three critical components of our proposed framework: the data imputation layer, the graph convolution layer, and the time series prediction layer. Collaboratively, these components aim to enhance the model's performance and improve its prediction accuracy.

### 3.1 Problem definition

We represent the road network as a graph  $G = (V, E, A)$ , where  $V$  denotes the set of sensors within the road network  $|V| = N$ , corresponding to the observations of  $N$  sensors;  $E$  denotes the set of edges connected by sensors; and  $A \in \mathbb{R}^{N \times N}$  denotes the adjacency or similarity matrix representing between nodes. The observed graph signal  $X^{(t)} \in \mathbb{R}^{N \times r}$  represents an arbitrary time step of the traffic state, whose elements are the  $r$  features observed by each sensor, such as flow and speed. We aim to predict only the traffic volume. Given the history of the  $S$  time step data, a model  $F$  can be trained to predict the future  $T$  time step graph signals at a time step on the road network as follows:

$$X^{(t+1):(t+T)} = F \left[ X^{(t-S,t)}; G \right] \quad (1)$$

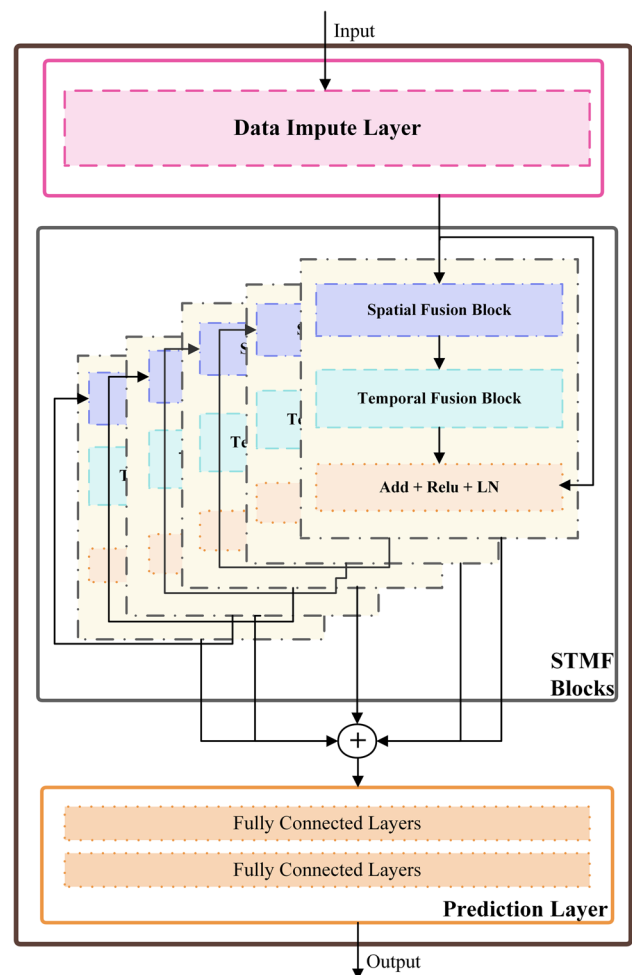
### 3.2 Model architecture

This paper proposes a spatial-temporal multifactor fusion graph neural network model that considers multiple factors when modeling spatial and temporal dependencies and dynamically self-learns parallel data for fusion to capture spatial and temporal dependencies more comprehensively and predict traffic flow more accurately. We illustrate the overall architecture of our proposed STMFGNN model in Fig. 2, which is composed of a data-completion layer, as shown in Fig. 3; stacked spatial-temporal multifactor fusion (STMF) blocks, as shown in Fig. 4; and a prediction layer. The outputs of each STMF block are concatenated and fed into the prediction layer via a residual connection. We provide specific details of the model in the following subsections.

#### 3.2.1 Data imputation layer

Traffic flow data are usually collected by sensors. In practical applications, the data collected by sensors are often subject to damage or loss of content due to equipment changes. Many traffic flow prediction models rely on complete datasets and often require accurate imputation of missing data before statistical analysis. This paper uses the GAIN with an improved masking mechanism and training process to impute the missing data.

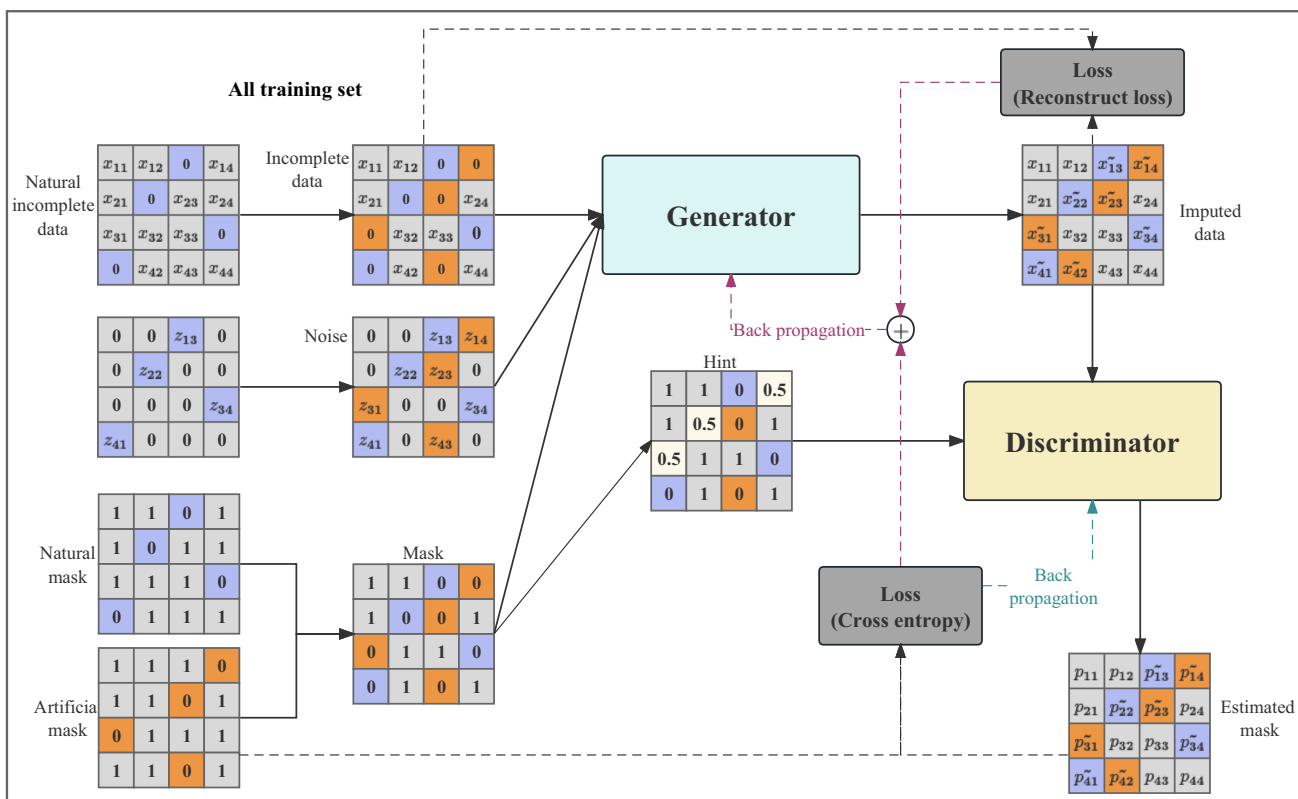
Take the traffic data of  $N$  recording points in a day as an example,  $\mathcal{X} = (x_1, \dots, x_N) \in \mathbb{R}^d$ , where  $d$  is the number of recording points in a day (if it is recorded every five minutes, then  $d = 288$ ). For any data vector  $x_i$  in  $\mathcal{X}$ , there is a corresponding binary mask vector  $m_i = \{0, 1\}^d$  corresponding to it, where if  $m_{ij} = 1$ ,  $x_{ij}$  is observable; if  $m_{ij} = 0$ , then the data point  $x_{ij}$  is missing. Since the



**Fig. 2** Framework of the STMFGNN. The architecture consists of three main components: a data imputation layer, several STMF blocks, and a prediction block

traffic dataset used in this paper is already missing to varying degrees, we mark the location of each naturally missing point in the dataset to form a predefined vector of naturally missing masks  $M_{natural}$ . In addition, to train the model, we also need to introduce an artificial missing mask  $M_{artificial}$ ,  $M = M_{natural} + M_{artificial}$ , and train the model through the imputation loss of artificial missing data and the reconstruction loss of the original observable data.

As shown in Fig. 3, generator  $G$  takes the incomplete data vector  $x \in \mathbb{R}^d$ , mask vector  $m \in \mathbb{R}^d$ , and noise variable vector  $z \in \mathbb{R}^d$  as inputs. The output interpolation vector  $\tilde{x} \in \mathbb{R}^d$  and hint vector  $h = \{0, 0.5, 1\}^d$  are used as inputs to the discriminator  $D$ , which then tries to distinguish which components of the whole vector are observed or estimated, equivalent to predicting the mask vector. The hint vector  $h$  is introduced to provide  $D$  with additional information about the missing data, and by defining  $h$  in different ways, it is possible to control the amount of information about the mask vector  $m$  contained in the hint vector  $h$ .



**Fig. 3** Diagram of the data imputation part of the framework. GAIN with an improved masking mechanism and training process was used. The network consists of a generator and a discriminator

The output of generator  $G$  is the estimation vector  $x_G$ , which is defined as follows:

$$x_G = G(x, m, (1 - m) \odot z) \tag{2}$$

where  $\odot$  denotes the Hadamard product operation. Note that  $G$  generates not only the values of the missing components but also the values of the entire data vector, which includes the values that can be observed. Thus, the imputation vector  $\tilde{x}$  and the reconstruction vector  $\bar{x}$  are defined as:

$$\tilde{x} = m \odot x + (1 - m \odot x_G) \tag{3}$$

$$\bar{x} = m \odot x_G + (1 - m \odot x) \tag{4}$$

Given a random vector  $r = \{0, 1\}^d$ , the hint vector is generated as follows:

$$h = m \odot r + 0.5(1 - r) \tag{5}$$

Subsequently, the discriminator  $D$  takes the hint vector  $h$  and interpolation vector  $\tilde{x}$  as inputs, and the output  $m_D$  is the prediction of the mask vector  $m$ , which is denoted as:

$$m_D = D(\tilde{x}, h) \tag{6}$$

The cross-entropy between the output of the discriminator  $m_D$  and the artificial missing matrix  $m_a$  in the generator is used as the loss function of the discriminator, and is calculated as follows:

$$\mathcal{L}_D(m_a, m_D, r) = m_a \log m_D + (1 - m_a) \log (1 - m_D) \tag{7}$$

The generator uses the reconstruction loss of the original observable data and the imputation loss of artificial missing data. At the same time, for the iterative optimization of the discriminator, the cross-entropy between the discriminator's output and the generator's needs to be calculated. These three parts are added together as the generator's loss function:

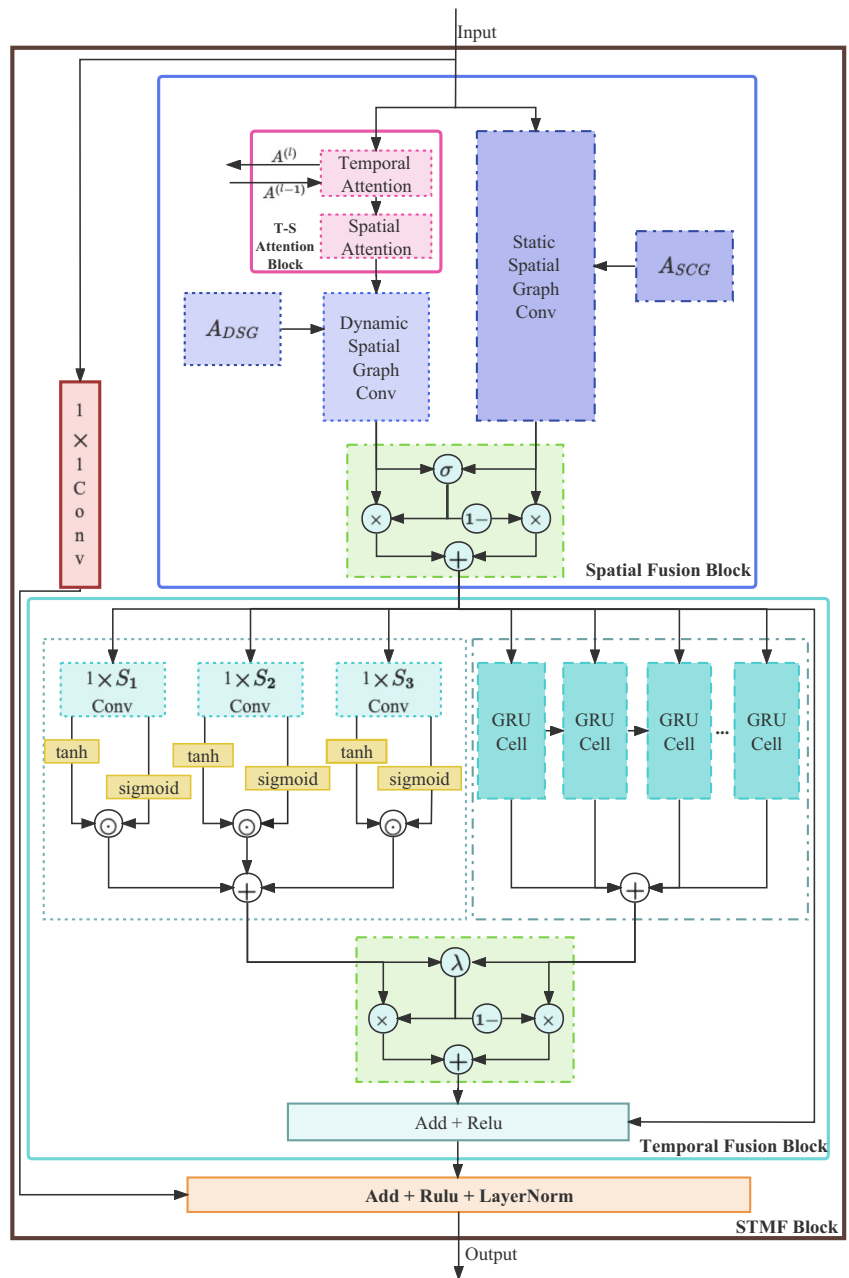
$$\mathcal{L}_{G_1}(\bar{x}, x) = \|\bar{x} - x\|^2 \tag{8}$$

$$\mathcal{L}_{G_2}(\bar{x}, x, m_a) = \|\bar{x} - x\|^2 \odot m_a \tag{9}$$

$$\mathcal{L}_{G_3}(m_a, m_D, r) = -[(1 - m_a) \log m_D + m_a \log (1 - m_D)] \tag{10}$$

$$\mathcal{L}_G = \mathcal{L}_{G_1} + \alpha (\mathcal{L}_{G_2} + \mathcal{L}_{G_3}) \tag{11}$$

**Fig. 4** An STMF block. This figure shows the details of the STMF block. The STMF block includes a static-dynamic spatial fusion module and a long-short-term temporal fusion module. The spatial fusion module is composed of a static graph convolution module and a dynamic graph convolution module processed by a spatial-temporal attention block working in parallel, followed by a dynamic gated fusion module. The temporal fusion module consists of a multireceptive field GTU responsible for short-range scale feature capture, a GRU responsible for long-range scale feature capture, and a dynamic gated fusion module



where  $\alpha$  is a hyperparameter. The training objectives of D and G are shown below:

$$\min \frac{1}{N} \sum_{k=1}^N \mathcal{L}_D(m_a, m_D, r)$$

$$\min_G \frac{1}{N} \sum_{k=1}^N \mathcal{L}_{G_1}(\bar{x}, x) + \alpha (\mathcal{L}_{G_2}(\bar{x}, x, m_a) + \mathcal{L}_{G_3}(m_a, m_D, r)) \tag{12}$$

### 3.2.2 Spatial fusion block

**Adjacency matrix generation** Given the graph structure, convolution operations are needed to extract node features,

while the adjacency matrix is crucial because it determines how nodes aggregate information about themselves and their neighbors. To fully utilize the topology of the network as well as the information of the data itself, this paper uses ChebNet-based graph convolution combined with a spatial-temporal attention mechanism to learn the network structure and node information. Unlike previous studies that only used static predefined graphs or dynamic graphs in the graph convolution part, the graphs used in graph convolution in this paper are based on two aspects. On the one hand, the predefined static adjacent matrix  $A_{SCG}$  based on the actual sensor connectivity network is used. On the other hand, the dynamic similarity matrix  $A_{DSG}$  is generated by data-driven generation, a bina-

rization of the dynamic correlation matrix  $A_{DRG}$  based on the Wasserstein distance. Through parallel graph convolution from the static localized graph and the dynamic similarity graph, the missing partial spatial dependencies caused by relying only on a single factor are avoided to mine the complex spatial dependencies between nodes more effectively and comprehensively.

There is a specific correlation between the traffic flows of different road nodes, which changes with time, so it is essential to effectively extract the dynamic correlation features between nodes. We regard the traffic data collected at each node as discrete data and split the historical data into a single vector at a specific time step, such as one day; then, the historical traffic data at each node are represented as a sequence of vectors. Then, we transform the data from these nodes into probability distributions and use the Wasserstein distance to calculate the probability distribution similarity between nodes to capture the spatial correlation between nodes. The Wasserstein distance is a method used to measure the difference between two probability distributions. Given two distributions  $p$  and  $q$ , the minimum cost required to convert from distribution  $p$  to  $q$  distribution is evaluated. Taking the traffic data of  $N$  recording points on  $D$  days as an example,  $X^f \in \mathbb{R}^{D \times d_t \times N}$ , where  $d_t$  is the number of recording points in a day (if it is recorded every five minutes, then  $d_t = 288$ ). The sequence of vectors at the  $n$  ( $n \in N$ ) node is denoted as  $X_n^f = (w_{n1}, w_{n2}, \dots, w_{nd})$ ,  $w_{nd} \in \mathbb{R}^{d_t}$ , where  $d \in [1, D]$ . Using the cosine distance as the cost function, the conversion cost of the traffic flow vector  $w_{n1i}$  at point  $n_1$  on day  $i$  to the traffic flow  $w_{n2j}$  at point  $n_2$  on day  $j$  is as follows:

$$cost(w_{n1i}, w_{n2j}) = 1 - \frac{w_{n1i}^T \cdot w_{n2j}}{\|w_{n1i}\|_2 \times \|w_{n2j}\|_2} \tag{13}$$

where  $T$  represents the transpose of the matrix, so the dynamic correlation distance can be expressed as:

$$d_{DRD}(n_1, n_2) \triangleq DRD(X_{n_1}, X_{n_2}) = \inf_{\gamma \in \Pi[P_{n_1}, P_{n_2}]} \int_x \int_y \gamma(x, y) \left( 1 - \frac{w_{n_1x}^T \cdot w_{n_2y}}{\sqrt{w_{n_1x}^T w_{n_1x}} \times \sqrt{w_{n_2y}^T w_{n_2y}}} \right) dx dy$$

$$s.t. \int \gamma(x, y) dy = \frac{\|w_{n_1x}\|_2}{\sum_{x=1}^D \|w_{n_1x}\|_2}, \int \gamma(x, y) dx = \frac{\|w_{n_2y}\|_2}{\sum_{y=1}^D \|w_{n_2y}\|_2} \tag{14}$$

We obtain a matrix  $A_{DRD} \in \mathbb{R}^{N \times N}$  representing the degree of relevance between the recorded points, where  $A_{DRD}[i, j] = 1 - d_{DRD}(i, j) \in [0, 1]$ . Under the premise of maintaining a certain sparsity level  $P_{sp}$  ( $P_{sp}$  as a hyperparameter), for each node of the road network, the  $N_r = N \times P_{sp}$  elements with the largest value in row  $i$  are retained (the remaining elements are set to 0). As a result, the dynamic

relevance graph  $A_{DRG} \in \mathbb{R}^{N \times N}$  is obtained. The dynamic similarity graph  $A_{DSG} \in \mathbb{R}^{N \times N}$  is obtained as a graph structure by binarizing the  $A_{DRG}$ . That is, if the value of these elements is nonzero, these elements are set to 1. This means that the most similar  $N_r$  nodes of each given node will be aggregated in the graph convolution. A dynamic similarity matrix  $A_{DSG}$  is available for each time step according to the particular time step of the segmented historical data.

Short-term local traffic flow changes are sudden, fluctuating, and transitive. The matrix based on the real road network in the dataset is used as the static predefined adjacency matrix  $A_{SCG} \in \mathbb{R}^{N \times N}$ . For the static adjacency matrix, in the actual road network, if two nodes are connected, the value is set to 1; otherwise, it is set to 0. This matrix is represented as follows:

$$A_{SCG}_{ij} = \begin{cases} 1, & \text{if } v_i \text{ adjacent } v_j \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

**Graph convolution layer** After obtaining the graph structure based on different factors, a convolution operation is needed to aggregate and update the node features. To fully utilize the topology of the network and the information of the data itself, this paper uses a parallel static-dynamic graph convolution based on ChebNet to learn the network structure and node information. The scaled Laplacian matrices used in the static and dynamic graph convolution parts are defined separately in the graph convolution. The scaled Laplacian matrix for the static Chebyshev polynomial is defined as  $\tilde{L}^s = \frac{2}{\lambda_{max}^s} (D^s - A^s) - I_N$ , where  $A^s = A_{SCG}$ ,  $I_N$  is the unit matrix,  $D^s \in \mathbb{R}^{N \times N}$  is the degree matrix, and the elements  $D_{ii}^s = \sum_j A_{ij}^s$  and  $\lambda_{max}^s$  are the largest eigenvalues in the Laplace matrix  $L^s = (D^s - A^s)$ . The scaled Laplace matrix of the dynamic Chebyshev polynomial is defined as  $L^d = \frac{2}{\lambda_{max}^d} (D^d - A^d) - I_N$ , where  $A^d = A_{DSG}$ ,  $I_N$  is the unit matrix,  $D^d \in \mathbb{R}^{N \times N}$  is the degree matrix, and the elements  $D_{ii}^d = \sum_j A_{ij}^d$  and  $\lambda_{max}^d$  are the largest eigenvalues in the Laplace matrix  $L^d = (D^d - A^d)$ .

Both the static and dynamic graph convolution parts use  $K$ th-order Chebyshev polynomials to aggregate the graph signals; that is, they perform graph convolution. The difference is that the static part directly processes the graph signals as follows:

$$g_\theta * Gx = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}^s)x \tag{16}$$

In the dynamic part, to fully use the dynamics between nodes and the nodes themselves, the graph signal is processed by the widely used spatial-temporal attention module to obtain the attention matrix. The attention matrix adjusts



each item of the Chebyshev polynomial and then multiplies it with the original graph signal, which is expressed as follows:

$$\mathbf{g}_\theta * \mathbf{G}x = \mathbf{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k \left( T_k(\tilde{L}^s) \odot P^{(k)} \right) x \quad (17)$$

where  $\mathbf{g}_\theta$  denotes the approximate convolution kernel, which extracts the information of the  $\mathbf{0}$  to  $\mathbf{K} - 1$ -order surrounding neighbors centered at each node;  $*\mathbf{G}$  represents the graph convolution operation;  $\theta \in \mathbb{R}^k$  is a learnable vector of polynomial coefficients;  $\odot$  represents the Hadamard product; and  $P^{(k)} \in \mathbb{R}^{N \times N}$  is the attention matrix. This definition can be extended to graph signals with multichannel inputs, such as inputs  $\mathcal{X}^r = (X_1, X_2, \dots, X_M) \in \mathbb{R}^{N \times C_{r-1} \times M}$ . Each input feature of each node has  $C_{r-1}$  feature channels, and the convolution kernel parameter is  $\mathbf{g}_\theta \in \mathbb{R}^{K \times C_{r-1} \times C_r}$ . Eventually, each node can aggregate  $\mathbf{0} \sim \mathbf{K} - 1$  order neighbor node information to update its node information.

Therefore, the static graph convolution part aggregates information about local neighboring nodes by employing a predefined static adjacent matrix based on the actual sensor connectivity network. The dynamic graph convolution part aggregates information about the most globally similar nodes using a dynamic similarity matrix. The result of static graph convolution is  $Z^s \in \mathbb{R}^{N \times M \times C_r}$ , and the dynamic graph convolution result is  $Z^d \in \mathbb{R}^{N \times M \times C_r}$ . Static and dynamic two-part graph convolutions consider interactions between nodes from local and global perspectives. As a result, they should exert distinct influences on the final prediction. The specific influence should be learned from historical data. To realize this, we introduce a dynamic gating fusion module to rationally assign the respective importance weights and adaptively fuse the static and dynamic convolution results:

$$Z^r = \delta \cdot Z_s^{(r)} + (1 - \delta) \cdot Z_d^{(r)} \quad (18)$$

where  $\delta \in [0, 1]$  is the self-learning fusion parameter, which reflects the contribution degree of the static graph convolution and dynamic graph convolution parts to the result.  $Z^{(r)} \in \mathbb{R}^{N \times M \times C_r}$  is the final output of the graph convolution module, which is used as input to the temporal fusion layer of the next module. It is worth noting that as mentioned above, the dynamic graphs are calculated by dividing the data according to a specific time step. Each specific time step has a different dynamic graph, so the fusion parameters are also different at each time step according to this time step.

### 3.2.3 Temporal fusion layer

Traffic flow data are typical spatial-temporal data. After the extraction and processing of spatial dependencies between road network nodes via graph convolution, the temporal

dependencies of data on each node must be further captured. Most traffic prediction models use convolution alone to extract temporal features, which results in the loss of dynamic long-term data features. With the development of deep learning methods, a large number of researchers have found that RNNs can learn time series features better than back propagation (BP) neural networks [46]. As a variant of RNN, GRU [47] is more capable of extracting dynamic long-term features in the time dimension, has fewer parameters and is easier to train than LSTM [48], compensating for the deficiency of convolutional methods. Therefore, to utilize multi-scale temporal features, this paper introduces GRU while using M-GTU, so as to capture the temporal dependencies more comprehensively with two time-series data prediction methods that work in different ways. M-GTU uses gated convolution with three different receptive fields to enhance the model's ability to perceive different degrees of short-term temporal dependencies. GRU uses a gating mechanism to control the flow of information in the sequence, which can better handle the flow of information between each stage in a long-term sequence and provide more robust long-term time-dependent modeling capabilities. By fusing multiscale temporal features, the patterns of data changes can be better understood, further enhancing the model's ability to perceive time dependence.

In the GTU section, we use a convolutional kernel  $\Gamma \in \mathbb{R}^{1 \times S \times C_r \times 2C_r}$  to double the number of channels for the input  $Z^{(r)} \in \mathbb{R}^{N \times M \times C_r}$ , where  $\mathbf{1} \times S$  represents the kernel size, and the input becomes  $Z'^{(r)} \in \mathbb{R}^{N \times (M - (S - 1)) \times 2C_r}$  after the convolution. The GTU of traditional convolution is defined as follows:

$$\Gamma_{*_\tau} Z^{(r)} = \phi(E) \odot \sigma(F) \in \mathbb{R}^{N \times (M - (S - 1)) \times 2C_r} \quad (19)$$

where,  $*_\tau$  is the gated convolution operation,  $\phi(\cdot)$  and  $\sigma(\cdot)$  are the tanh function and sigmoid function, respectively, and  $E$  and  $F$  correspond to the pre- $r$  and post- $r$  parts of the channel of input  $Z'$ , respectively. To enhance the ability of the GTU part to perceive different degrees of time dimensions, gated convolutional modules with different receptive fields are used respectively, and then their outputs are fused and introduced into the residual structure as the output of the M-GTU part, which is defined as follows:

$$Z_{mout}^{(r)} = M - GTU(Z^{(r)}) = Relu\left(FC\left(Concat\left(\Gamma_{1*_\tau} Z^{(r)}, \Gamma_{2*_\tau} Z^{(r)}, \Gamma_{3*_\tau} Z^{(r)}\right)\right) + Z^r\right) \quad (20)$$

where  $\Gamma_1, \Gamma_2$  and  $\Gamma_3$  ls of  $\mathbf{1} \times S_1, \mathbf{1} \times S_2$ , and  $\mathbf{1} \times S_3$ , respectively.  $FC(\cdot)$  denotes the fully connected operation to adjust the different sizes of the same feature dimension caused by the operation  $Concat(\cdot)$  to ensure that it matches the original input size of the M-GTU, and the residual structure is

realized by skip connection. Finally, the M-GTU partial output  $Z_{mout}^{(r)} \in \mathbb{R}^{N \times M \times C_r}$  is obtained by the ReLU activation function.

The input of the GRU part is the same as that of the M-GTU part, but since the GRU-Cell is a single-step prediction, the input for each step is . The GRU part is defined as:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t]) \tag{21}$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t]) \tag{22}$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t]) \tag{23}$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t \tag{24}$$

$$Z_{gout}^{(r)} = GRU(Z^{(r)}) = Stack(h_{t_1}, h_{t_2}, \dots, h_{t_M}) \tag{25}$$

The input of each moment  $t$  includes the hidden state of the previous moment  $h_{t-1}$  and the input of the current moment  $x_t$ , and the output of the update gate  $z_t$  is used to control the updated degree of the current state. The output of the reset gate  $r_t$  is used to control the influence of the past state on the current state to filter unnecessary information; the candidate hidden state  $\tilde{h}_t$  is formed by the superposition of the current input and the past state. The hidden state at the current moment  $h_t$  is updated by the update gate, the weighted average of the past state, and the weighted average of the candidate's hidden state. After accumulating predictions for  $M$  time steps, the output  $Z_{gout}^{(r)} \in \mathbb{R}^{N \times M \times C_r}$  is obtained as the result of the GRU part.

As a result, the M-GTU mainly captures short-term temporal dependencies, and the output is  $Z_{Mout}^{(r)} \in \mathbb{R}^{N \times M \times C_r}$ . In contrast, the GRU part improves the capture of long-term temporal dependencies, and the output is  $Z_{gout}^{(r)} \in \mathbb{R}^{N \times M \times C_r}$  for this part. Similar to the graph convolution static-dynamic parallel module, both the M-GTU and GRU parts also use the dynamic gated fusion module to adaptively fuse the results of these two parts to self-learn the influence of short-term and long-term dependencies from historical data. It is expressed as follows:

$$Z_{out}^{(r)} = \lambda \cdot Z_{mout}^{(r)} + (1 - \lambda) \cdot Z_{gout}^{(r)} \tag{26}$$

where  $Z_{out}^{(r)} = \lambda \cdot Z_{mout}^{(r)} + (1 - \lambda) \cdot Z_{gout}^{(r)}$  is the adaptive learning parameter that also changes dynamically with the specific time step mentioned in the previous section, reflecting the extent to which the short-term versus the long-term influence of the time series prediction component impacts final prediction result, and  $Z_{out}^{(r)} \in \mathbb{R}^{N \times M \times C_r}$  is the final output of the temporal fusion module.

## 4 Experimentation

In this section, we present experiments conducted on real datasets to verify the effectiveness of the STMFGNN. First, we introduce the datasets used for our experiments. Second, we introduce some baseline methods, conduct comparative experiments with overall and different prediction time targets, and conduct ablation experiments and statistical analysis to showcase the performance of our model compared to the baselines. Finally, we conduct parameter research experiments and time cost experiments, as well as some intermediate component and result visualization experiments.

### 4.1 Datasets

The experiments were conducted on four actual roadway datasets (PEMS03, PEMS04, PEMS07, and PEMS08) from California. These datasets were collected in real time by the PEMS system every 30 seconds, aggregating the raw data into 5-minute intervals. In addition, the datasets contain spatially adjacent matrices constructed based on actual road networks. Table 1 shows more details about the datasets.

### 4.2 Baseline method

We compare our STMFGNN with the following baselines:

1. **DCRNN** [17]: Diffusion Convolutional Recurrent Neural Network. It captures spatial correlation by modeling traffic flow changes as a one-dimensional convolutional diffusion process.
2. **STGCN** [22]: Spatio-Temporal Graph Convolutional Network. It integrates graph convolution into 1D convolutional units and incorporates causal convolution to process temporal information.
3. **GraphWaveNet** [23]: GraphWaveNet combines adaptive graph convolution and extended causal convolution.
4. **ASTGCN** [18]: Attention Based Spatial-Temporal Graph Convolutional Networks. It introduces an attention mechanism before spatio-temporal convolution.
5. **STSGCN** [19]: Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting. It designs spatio-temporal subgraphs to capture heterogeneity in local spatio-temporal graphs.
6. **AGCRN** [49]: Adaptive Graph Convolutional Recurrent Network. It proposes an adaptive graph convolutional recurrent network to automatically unify information through node embedding to capture fine-grained spatio-temporal traffic sequence correlations.
7. **STFGNN** [50]: Spatial-Temporal Fusion Graph Neural Networks for Traffic Flow Forecasting. It proposes the "temporal graph" and compensates for correlations that

**Table 1** Statistics and description of the datasets

Datasets	Sensors	Edges	Time step	Time scale	Missing ratio(%)
PEMS03	358	547	26208	2018/9/1-2018/11/30	0.672
PEMS04	307	340	16992	2018/1/1-2018/2/29	3.182
PEMS07	883	866	28224	2017/5/1-2017/8/31	0.452
PEMS08	170	295	17856	7/1/2016 - 8/31/2016	0.696

may not be reflected in the spatial graph through the spatio-temporal fusion graph.

- Z-GCNETS** [51]: Time Zigzags at Graph Convolutional Networks for Time Series Forecasting. It introduces the concept of time-aware zigzag persistence into learning temporal conditional graph structures and develops a zigzag topology layer (Z-GCNET) for time-aware graph convolutional networks (GCNs).
- DSTAGNN** [20]: Dynamic Spatial-Temporal Aware Graph Neural Network. It uses dynamic spatial-temporal aware graphs to model spatial-temporal interactions in road networks.

### 4.3 Experiment settings

For a fair comparison with the previous baselines, we divided the data from PEMS03, PEMS04, PEMS07, and PEMS08 into a training set, a validation set, and a test set at the ratio of 6:2:2. We use the historical traffic flow data for the past 1 hour (that is, 12 consecutive time steps) to predict the traffic flow data for the next 1 hour (that is, 12 consecutive time steps). The experiments were conducted in an AMD EPYC 7T83 CPU and an NVIDIA RTX 4090 24 GB GPU environment. We set the following hyperparameters: all experiments stacked 5 layers of STMF blocks. The number of Chebyshev polynomial terms  $K$  is 3. The number of heads of the spatial-temporal multihead attention mechanisms paired with the dynamic graph convolution is 3. The sparsity  $P_{sp}$  of the dynamic graph is 0.01. The convolution kernel of the MGTU along the time dimension is  $\{S_1, S_2, S_3\} = \{2, 5, 8\}$ . To train the model, we employed the Huber loss function, and the threshold parameter of the loss function was set to 1. We used the Adam optimizer with a learning rate of 0.0001 for 100 epochs. The settings for the batch size varied from dataset to dataset and were 32, 32, 12 and 64.

### 4.4 Evaluation indicators

The mean absolute error (MAE), the mean absolute percentage error (MAPE), and the root mean squared error (RMSE) are used to calculate the model's performance in terms of error [52]. Lower values of MAE and RMSE indicate lower prediction errors of the model, while lower values of MAPE indicate lower relative errors of the model. In addition, some

of the baseline method results (marked with \* in Tables 2 and 3) were obtained by running the corresponding open-source code. In contrast, the other part was extracted directly from published papers. The above mathematical formulas are as follows:

$$RMSE(y, \hat{y}) = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (27)$$

$$MAE(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (28)$$

$$MAPE(y, \hat{y}) = \frac{100\%}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{y_i} \quad (29)$$

## 4.5 Experimental results and analysis

### 4.5.1 Overall comparison

Table 2 shows the average results of our STMF-GNN and ten baseline methods for 60-minute predictions on the PEMS03, PEMS04, PEMS07, and PEMS08 datasets. The results show that our STMF-GNN outperforms the baseline model in terms of all the metrics on all four datasets, achieving the best results.

The relatively poor performance of GraphWaveNet may be due to its inability to superimpose its spatial-temporal layers, resulting in a relatively minor feature receptive range. Moreover, methods such as the DCRNN, STGCN and ASTGCN highly rely on predefined graph structures which is helpful for short-term prediction. However, the dynamic relationship between nodes will be ignored over time, so the effect of long-term prediction is significantly reduced; thus, these methods perform poorly. STSGCN extracts only local spatial dependencies and requires more data to build sub-graphs for training; thus, limited by the size of the dataset, it performs generally and may perform better on large-scale graphs. In contrast, the STFGCN and DSTAGNN, which use a dynamic graph structure to model spatial dependencies, significantly improve the prediction performance, indicating that such methods can capture the underlying spatial dependencies over time. However, they discard the predefined graph structure and use only the dynamic graph structure that

**Table 2** Performance comparison of our STMFGNN and baseline models

Datasets	Models	MAE	MAPE(%)	RMSE
Pems03	Graph WaveNet	19.85	19.31	32.94
	DCRNN	18.18	18.91	30.31
	STGCN	17.49	17.15	30.12
	ASTGCN	17.69	19.40	29.66
	STSGCN	17.48	16.78	29.21
	AGCRN	15.98	15.23	28.25
	STFGNN	16.77	16.30	28.34
	Z-GCNETS	16.64	16.39	28.15
	DSTAGNN*	<u>15.64</u>	<u>14.73</u>	<u>27.33</u>
	STMFGNN(Ours)	<b>15.51</b>	<b>14.43</b>	<b>27.01</b>
Pems04	Graph WaveNet	25.45	17.29	39.70
	DCRNN	24.70	17.12	38.12
	STGCN	22.70	14.59	35.55
	ASTGCN	22.93	16.56	35.22
	STSGCN	21.19	13.90	33.65
	AGCRN	19.83	12.97	32.26
	STFGNN	19.83	13.02	31.88
	Z-GCNETS	19.50	<u>12.78</u>	<u>31.61</u>
	DSTAGNN*	<u>19.45</u>	12.83	31.72
	STMFGNN(Ours)	<b>19.32</b>	<b>12.69</b>	<b>31.42</b>
Pems07	Graph WaveNet	26.85	12.12	42.78
	DCRNN	25.30	11.66	38.58
	STGCN	25.38	11.08	38.78
	ASTGCN	28.05	13.92	42.57
	STSGCN	24.26	10.21	39.03
	AGCRN	22.37	<u>9.12</u>	36.55
	STFGNN	22.07	9.21	35.80
	Z-GCNETS	21.77	9.25	35.17
	DSTAGNN*	<u>21.72</u>	9.25	<u>34.96</u>
	STMFGNN(Ours)	<b>21.54</b>	<b>9.10</b>	<b>34.77</b>
Pems08	Graph WaveNet	19.13	12.68	31.05
	DCRNN	17.86	11.45	27.83
	STGCN	18.02	11.40	27.83
	ASTGCN	18.61	13.08	28.16
	STSGCN	17.13	10.96	26.80
	AGCRN	15.95	10.09	25.22
	STFGNN	16.64	10.60	26.22
	Z-GCNETS	<u>15.76</u>	10.01	25.11
	DSTAGNN*	15.81	<u>9.96</u>	<u>25.06</u>
	STMFGNN(Ours)	<b>15.66</b>	<b>9.86</b>	<b>24.71</b>

Note: \* denotes retraining \_ denotes the best indicator among the baselines

changes in time units of days, resulting in ignoring the mutual transmission of traffic fluctuations between actual adjacent nodes at a single time point. Furthermore, predefined graphs might contain noise, and the generated dynamic graphs are also constrained by these limitations. Thus, they are not as

effective as our STMFGNN. Second, most of the above models still fundamentally rely on variants of 1D convolution for modeling temporal dependence. For time series data, 1D convolution does not fully consider historical data, leading to inaccurate predictions of time series data and affecting overall predictive performance.

Our method comprehensively considers multiple factors, aiming to model spatial and temporal dependencies as thoroughly as possible. In spatial dependency modeling, we utilize both the static adjacency graph as local prior knowledge and the dynamic similarity graph generated based on Wasserstein distance as global hidden knowledge. This design enables our model to comprehensively capture complex spatial dependencies. For temporal dependency, we dynamically fuse short-term temporal features extracted by M-GTU and long-term temporal information captured by GRU, overcoming the trade-off between short-term and long-term temporal dependencies. Therefore, our model avoids the problem of missing spatial-temporal dependency modeling caused by excessive dependence on a single factor. As a result, our model performs better than the baseline methods, especially on the PEMS07 dataset, which has the largest graph size and the longest total time steps.

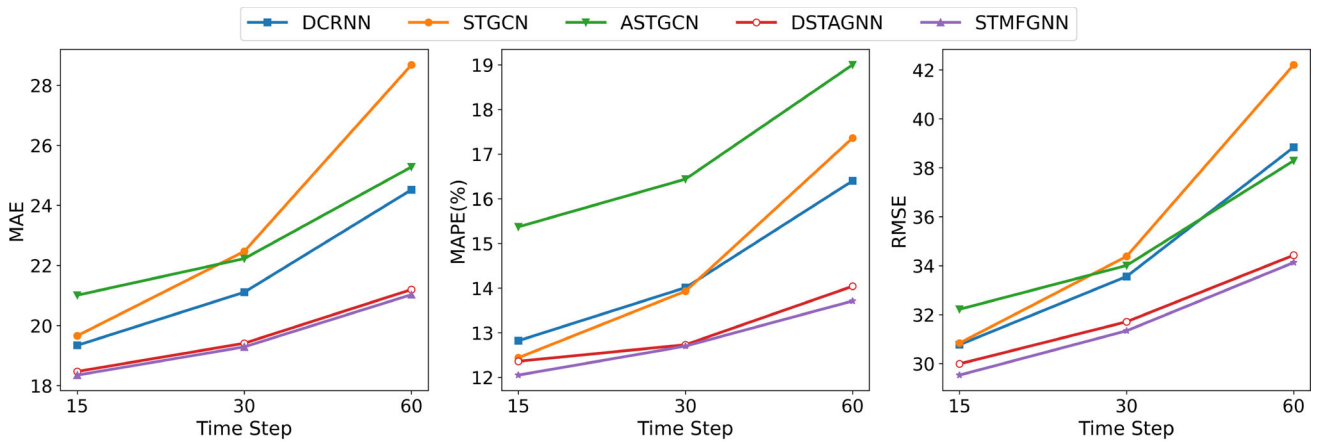
To demonstrate our model's performance in both short-term and long-term forecasting tasks, we also selected some of the baseline models and compared their prediction performances with our proposed STMFGNN under the same conditions for different prediction steps. Specifically, Table 3 shows the performance comparison between the STMFGNN and selected baseline models for 15 minutes, 30 minutes, and 60 minutes on the PEMS04 and PEMS08 datasets. The results show that our STMFGNN outperforms the selected baseline models in terms of all the metrics, both for short-term and long-term predictions. This improvement may be attributed to the complex modules in our model, which are tailored to excel in situations where they are most needed for both short-term and long-term predictions. To more visually show the performance of some of the baseline models and our STMFGNN under different prediction steps, Fig. 5 plots the data from Table 3 as a line graph.

In general, as the prediction target time interval becomes longer, the corresponding prediction difficulty increases, leading to larger prediction errors. As shown in Table 3 and Fig. 5, models using predefined graph structures, such as DCRNN, STGCN, and ASTGCN, do not show significant differences in performance compared to improved models using dynamic graphs in short-term predictions. However, as the prediction interval increases, the prediction accuracy of these models decreases significantly. In contrast, DSTAGNN, which uses dynamic graph structures, shows a slower decline in performance. This is mainly because the previously mentioned models only consider the influence of neighboring nodes and fail to capture the time-varying

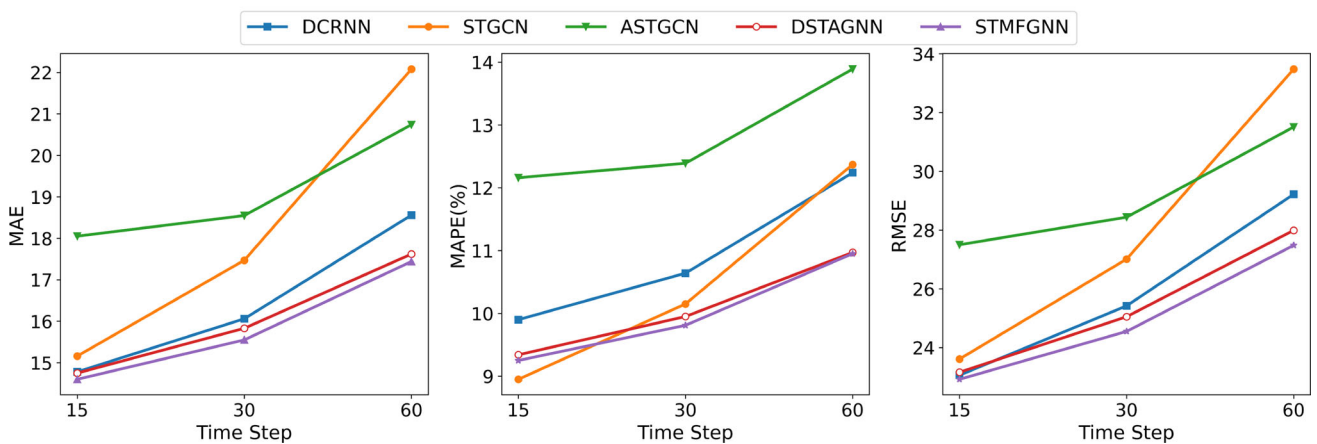
**Table 3** Performance comparison of our STMFGNN and selected baseline models on the PEMS04 and PEMS08 datasets

Models	15min			30min			60min		
	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)	MAE	RMSE	MAPE(%)
PEMS04									
DCRNN	19.34	30.77	12.82	21.11	33.56	14.01	24.52	38.84	16.40
STGCN	19.66	30.84	12.44	22.47	34.38	13.93	28.68	42.21	17.36
ASTGCN	21.01	32.22	15.37	22.23	34.01	16.44	25.28	38.29	19.00
DSTAGNN*	18.47	29.99	12.36	19.41	31.71	12.73	21.20	34.43	14.04
STMFGNN	<b>18.35</b>	<b>29.53</b>	<b>12.05</b>	<b>19.29</b>	<b>31.34</b>	<b>12.70</b>	<b>21.03</b>	<b>34.13</b>	<b>13.71</b>
PEMS08									
DCRNN	14.78	23.06	9.90	16.06	25.42	10.64	18.56	29.22	
STGCN	15.16	23.61	8.95	17.47	27.01	10.15	22.08	33.48	12.37
ASTGCN	18.05	27.50	12.16	18.55	28.44	12.39	20.74	31.51	13.89
DSTAGNN*	14.75	23.16	9.34	15.83	25.05	9.95	17.62	27.99	10.97
STMFGNN	<b>14.60</b>	<b>22.92</b>	<b>9.25</b>	<b>15.55</b>	<b>24.55</b>	<b>9.81</b>	<b>17.44</b>	<b>27.48</b>	<b>10.95</b>

Note: \* denotes retraining \_ denotes the best indicator among the baselines



(a) Performance comparison at different prediction steps on PEMS04 with some baseline models



(b) Performance comparison at different prediction steps on PEMS08 with some baseline models

**Fig. 5** Comparison of the performances of several of the baseline models on the PEMS04 and PEMS08 datasets

dynamic spatio-temporal correlation. DSTAGNN leverages the generated dynamic graph structure to better model the dynamic spatio-temporal dependencies that are more crucial for long-term predictions. However, DSTAGNN considers only global dynamic spatio-temporal dependencies and overlooks the impact of local neighboring node traffic fluctuations on node traffic changes, resulting in less effective performance compared to our model. The results demonstrate that our STMFGNN outperforms the selected baseline models in all metrics, both in short-term and long-term predictions. This superior performance is due to our model simultaneously modeling both static local and dynamic global spatial dependencies and dynamically integrating GTU, responsible for capturing short-term temporal features, with GRU, responsible for long-term temporal dependency modeling.

#### 4.5.2 Ablation experiments

To further evaluate the effectiveness of the individual components in the STMFGNN, we made the following variants of the STMFGNN:

1. RemDI: removes the data preprocessing layer and uses the original traffic flow data directly as input; that is, there is no complementary processing of missing data;
2. RemSSC: in the graph convolution part, the static graph convolution part that uses static graphs is removed, but the dynamic convolution module is retained;
3. RemGRU: in the time series prediction layer, the M-GTU module is retained, but the GRU module is removed.

We selected the PEMS08 dataset, which has the smallest graph size, and the PEMS07 dataset, which has the largest graph size, as representative datasets to conduct ablation experiments and compared the results with DSTAGNN, which is the state-of-the-art model in the baseline. Table 4 shows the measurements of each performance metric, and it

can be clearly observed that our STMFGNN outperforms the other variants and the baseline, confirming the effectiveness of each individual module in our model.

Table 4 shows the average results of the 60-min prediction performance for comparison, while Figs. 6 and 7 compare the prediction performance at different time settings for the 5-min, 20-min, 40-min, and 60-min settings. These different prediction durations represent short-term predictions (5 and 20 minutes) and long-term predictions (40 and 60 minutes), respectively.

The results of the ablation study are shown in Figs. 6 and 7, from which it can be seen that each component of the model contributes positively to the performance improvement of the whole model, and the conclusions obtained are as follows:

- As shown in Fig. 6, for the model without data imputation preprocessing, the effect of the model using the other parts of the critical design also outperforms that of the baseline, indicating the effectiveness of other crucial design aspects. However, as shown in Fig. 7, the variant after data imputation preprocessing consistently outperforms the baseline in various performance metrics, highlighting the vital role of the data imputation module. For the PEMS07 dataset, which is characterized by large-scale and minimal missing data, the impact of data imputation on model performance is not significant since the available data are sufficient to support network training. However, as illustrated in Fig. 6a, for the smaller-scale PEMS08 dataset with existing missing data, the data imputation module significantly enhances model performance.
- As shown in Fig. 7, removing the static spatial convolution part of the graph convolution layer has a significant impact on the short-range prediction performance. However, due to the combination of short-term and long-term time series prediction methods used in the temporal prediction layer, especially GRU, which is responsible for long-term time series prediction, the long-term prediction performance is better than that of the RemGRU, which illustrates the effectiveness of the GRU component for long-term temporal dependency capture. On the large-scale graph-structured PEMS07 dataset, this effect is even more pronounced. As shown in Fig. 7b, starting from the 20-minute prediction target, the GRU component has already demonstrated a significant enhancement in predictive performance.
- As shown in Fig. 7, removing the GRU component of the time series prediction layer has a significant impact on the long-term prediction performance. However, due to the effect of the static local spatial convolution module in the graph convolutional layers, the short-term predictive performance is better than that of the non-static spatial convolution variant. Especially for the 5-minute predic-

**Table 4** Ablation experiment of module effectiveness

Datasets	Module Composition	MAE	MAPE(%)	RMSE
PMES08	DSTAGNN	15.81	9.96	25.06
	RemDI	15.76	9.88	24.90
	RemSSC	15.72	9.93	24.81
	RemGRU	15.73	9.90	24.82
	<b>STMFGNN</b>	<b>15.66</b>	<b>9.86</b>	<b>24.71</b>
PMES07	DSTAGNN	21.72	9.25	34.96
	RemDI	21.55	9.10	34.78
	RemSSC	21.56	9.11	34.80
	RemGRU	21.57	9.13	34.95
	<b>STMFGNN</b>	<b>21.54</b>	<b>9.10</b>	<b>34.77</b>

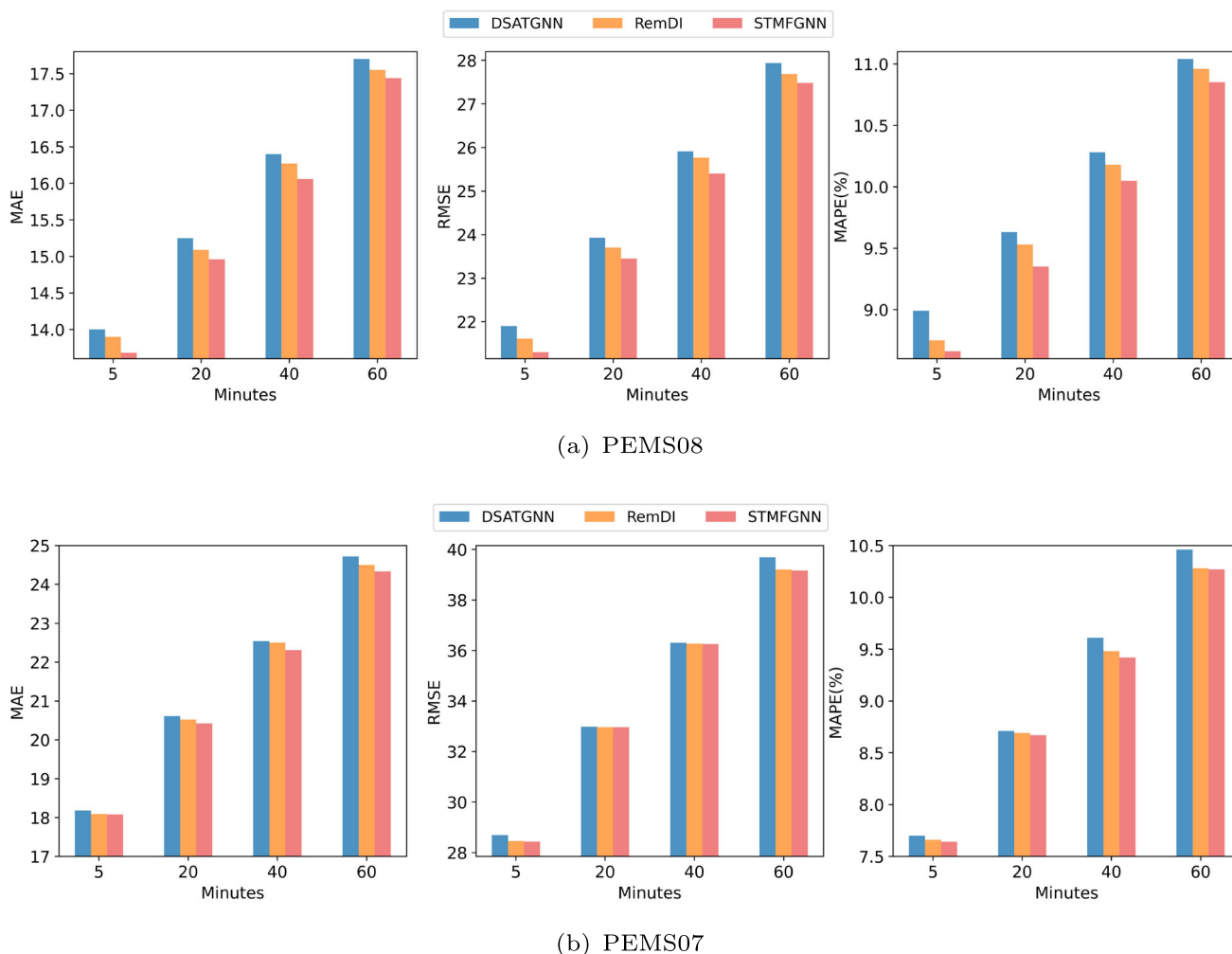


Fig. 6 Ablation experiment of the unimputed variant

tion target, the static local spatial convolution has an obvious effect, indicating that it has a good ability to capture traffic fluctuations. This confirms the effectiveness of the static spatial convolution for extracting local static spatial relational dependencies and verifies its effectiveness in improving the performance of short-term traffic prediction.

- Overall, the results from Table 4 and Figs. 6 and 7 collectively demonstrate that our STMFGNN model outperforms its various variants and baseline models. This confirms the effectiveness of the synergy of the various components in our model.

### 4.5.3 Visualization of data imputation

The improvement effect of missing data point imputation on the overall model prediction performance has been explained in detail in the previous section on ablation experiments.

Since the PEMS04 dataset has a high missing data rate among the four datasets, the nodes with missing data points in this dataset are used as examples to visually compare the effect of imputing to demonstrate the superiority of our method more comprehensively. Each node in the PEMS04 dataset contains 16,992 data points, and Fig. 8 shows the imputation comparison results for node 10, which contains 2,464 missing data points, and node 12, which contains 677 missing data points.

As shown in Fig. 8, our imputation method can accurately learn the distribution trend of the data and reasonably fill in missing data points, whether for continuous or discrete missing points. Specifically, for consecutive missing data points, our method accurately captures the overall trend of the data and performs smooth imputation. For scattered missing data points, our method effectively restores the variability of the data, ensuring that the imputed data aligns with the trend of the surrounding data points. Overall, the data after imputation show a more complete and consistent trend, which is a significant improvement compared to the original data.

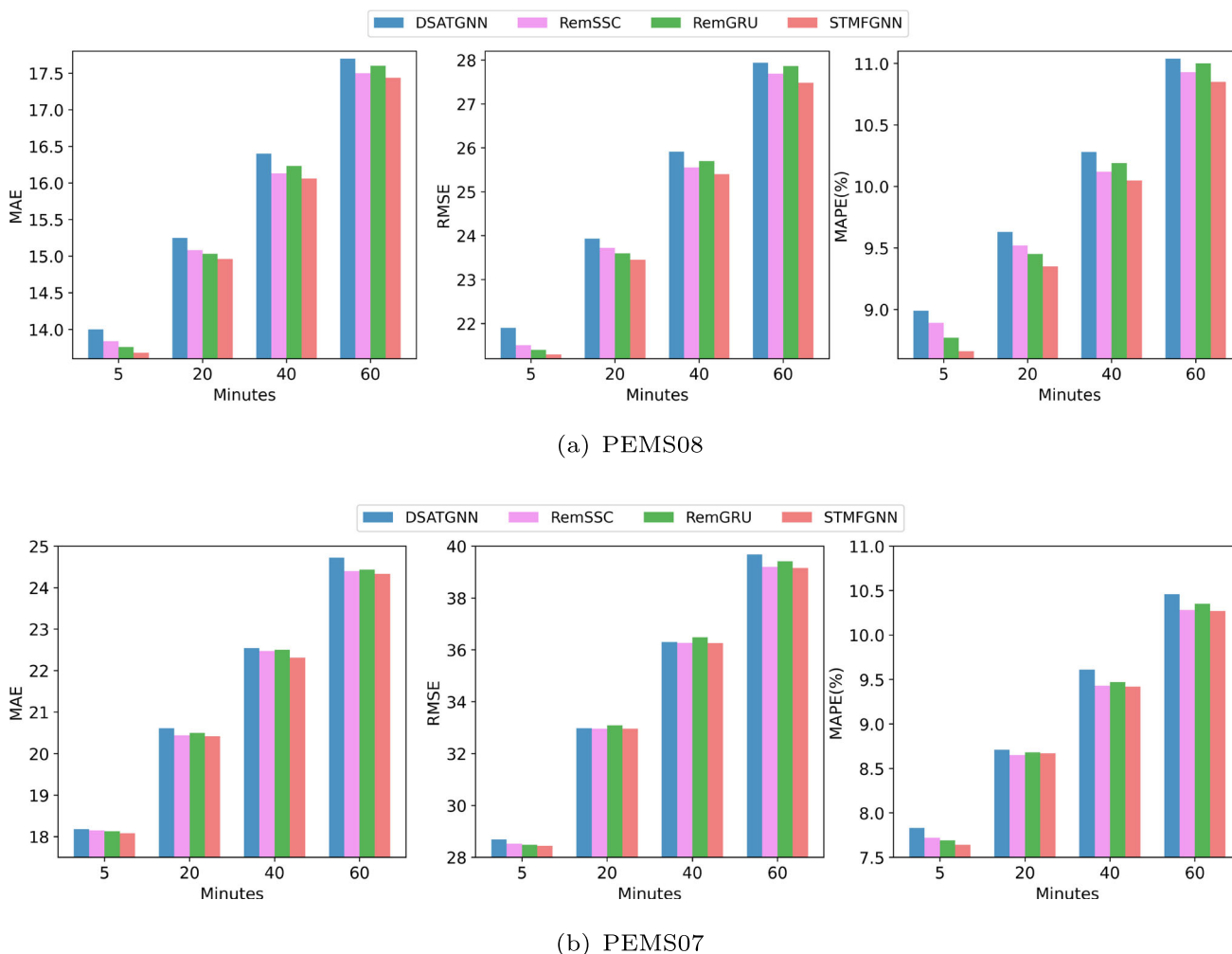


Fig. 7 Ablation experiment of other variant

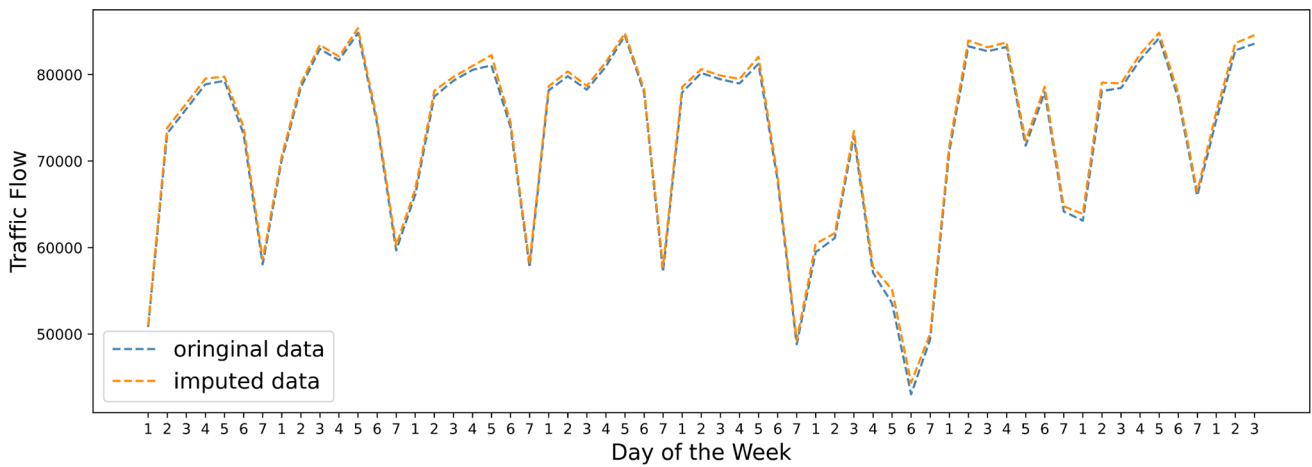
#### 4.5.4 Visualization of spatial dependency

To enhance the interpretability of the proposed model and demonstrate its effectiveness in capturing dynamic spatio-temporal dependencies, Fig. 9 shows the visualization results of the STMFGNN’s predefined static adjacent matrix and the dynamic correlation matrix on the first day of recording for 20 nodes from nodes 0 to 19 in the PEMS08 dataset. As seen from Fig. 9, in terms of days, the two nodes adjacent to each other are not necessarily the most correlated, although they are physically closest. In contrast, even when two nodes are not adjacent or are geographically distant, they may be highly relevant to each other due to their similar patterns of change. Our model is able to capture potential spatial correlations in a global scope beyond just the adjacent nodes, indicating that STMFGNN extracts complex information within the road network and can capture time-varying

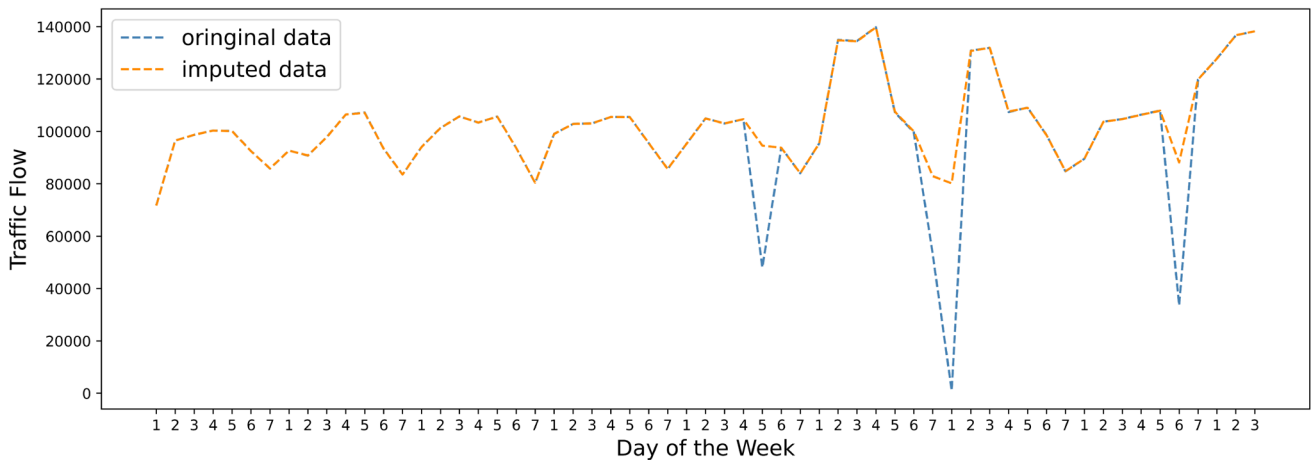
spatial dependencies. This capability allows the model not only to rely on predefined adjacency relationships but also to dynamically adjust the correlations between nodes based on actual observed data, thereby more accurately reflecting the spatio-temporal interactions between nodes in the real traffic network.

The visualized results of these dynamic correlation matrices clearly show how our method surpasses the limitations of traditional static graph structures, revealing deeper associations between nodes. This further demonstrates the effectiveness and superiority of STMFGNN in modeling dynamic spatio-temporal dependencies, providing strong support for understanding and interpreting the model. Additionally, this dynamic modeling capability makes our model more flexible and adaptable in practical applications, allowing it to more accurately tackle complex and variable traffic flow prediction tasks.



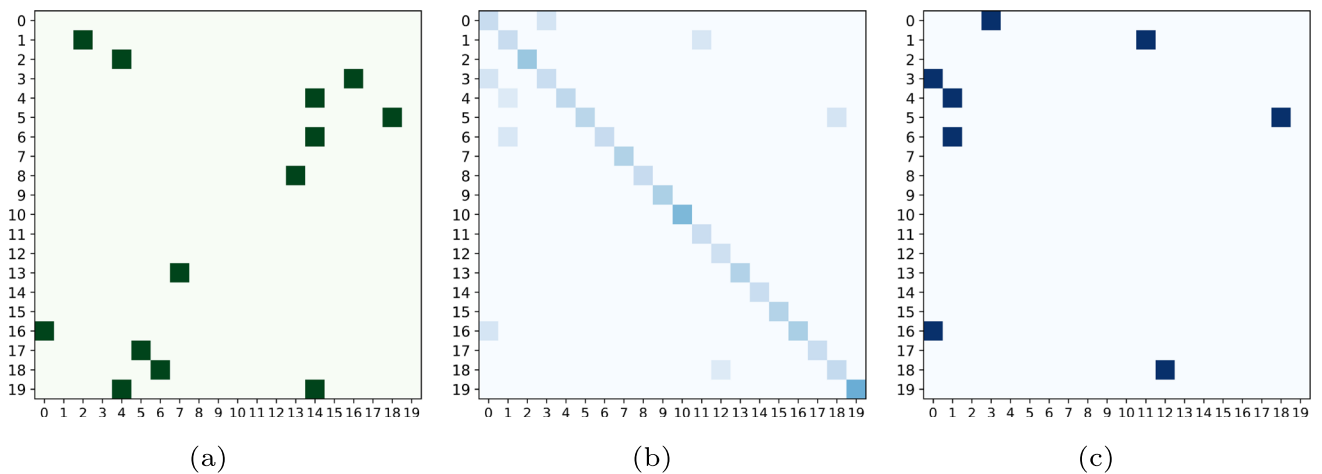


(a) Node #10 in PEMS04



(b) Node #12 in PEMS04

**Fig. 8** Comparison of missing data imputed on the PEMS04 dataset. Figures (a) and (b) show the comparison between the original data and the missing point data after filling on node 10 and node 12 of the PEMS04 dataset, respectively



**Fig. 9** Visualization results of the graphs used for the STMFGNN at nodes 0 to 19 on the PEMS08. Figure (a) shows the predefined static adjacent matrix of the nodes. Figure (b) shows the dynamic correlation matrix. Figure (c) shows the binarization of the dynamic correlation matrix with a specific sparsity setting

**Table 5** Parameter study of STMF block stacking layers

Stacking Layers	MAE	MAPE(%)	RMSE
4	15.82	9.94	25.06
5	15.66	9.86	24.71
6	15.84	10.08	24.95

#### 4.5.5 Parametric studies

To further investigate the effect of hyperparameter setup on model performance, we conducted a parameter study experiment on the PEMS08 dataset for the control variables that have a significant impact on model performance. This includes the stacked layers of the STMF block, the size of the convolutional kernel along the time dimension in the M-GTU, and the batch size. KS represents the kernel size along the time dimension in the M-GTU module. Tables 5, 6, and 7 show the results of the experiments.

We observed that the number of STMF blocks significantly affects the performance of our model. There is an optimal range for the number of STMF blocks that balances the benefits of deeper architectures with the risks of overfitting and gradient issues. Typically, a moderate depth outperforms very shallow or excessively deep models. Deeper models performing better than shallow ones because fewer layers lead to model underfitting, but too deep networks may lead to problems such as vanishing gradients, exploding gradients, or model overfitting. As shown in Table 5, the stacking of STMF blocks can significantly improve the model performance. The reason behind this phenomenon is that adding more STMF blocks enables the model to capture more complex spatial-temporal features, enhancing the model's expressive capability. However, as the number of layers continues to increase, the model performance begins to decline. We attribute this to the issue of vanishing or exploding gradients caused by overly deep network structures, thereby impacting the model's training effectiveness. Moreover, an excessive number of STMF blocks may introduce too much noise, leading to overfitting of the training data. The experimental results demonstrate that the model achieves optimal performance when stacked with 5 layers of STMF blocks, striking a balance between capturing intricate features and avoiding overfitting.

**Table 6** Parameter study of K-S

K-S	MAE	MAPE(%)	RMSE
3,5,7	15.73	9.86	24.84
2,5,8	15.66	9.86	24.71
4,5,6	15.91	9.90	25.01

The kernel size significantly affects temporal feature extraction. In particular, a larger kernel size increases the receptive field, allowing the model to capture longer-term dependencies. However, if the kernel size is too large, it may include irrelevant information and noise, which can degrade performance. Conversely, a smaller kernel size focuses on short-term dependencies and may miss out on important long-term patterns. Table 6 illustrates the impact of different convolution kernel sizes on model performance. It is observed that utilizing a combination of 2, 5, 8 receptive fields significantly enhances the model's predictive performance. This suggests that when simultaneously utilizing short-term and long-term temporal information, the model can better capture patterns and trends within time series data. Smaller convolution kernels (e.g., 2) are adept at capturing fine-grained short-term variations, while larger kernels (e.g., 8) excel at capturing long-term trends. The combination of varying kernel sizes allows the model to exhibit greater flexibility in processing temporal information, thereby improving prediction accuracy.

Batch size significantly affects the stability and convergence of model training. As shown in Table 7, a batch size of 64 yielded the best performance for the model on the PEMS08 dataset. Smaller batch sizes, such as 32, update the model parameters more frequently, which can introduce high noise levels during the update process, causing the model to get stuck in local minima. Additionally, models trained with small batches may experience unstable convergence due to frequent parameter updates. On the other hand, excessively large batch sizes can lead to each update containing too much information, which may cause the model to overfit the training data and perform poorly on the test set. A batch size of 64 strikes a balance between update frequency and stability, allowing the model to avoid local minima and preventing overfitting. This balanced batch size ensures that the model benefits from stable convergence while maintaining the ability to generalize well to unseen data.

These experimental results demonstrate the performance of our model under different hyperparameter settings and help to understand how to optimize model parameters for best performance. By reasonably adjusting the stacked layers of the STMF block, convolution kernel size and data batch size, we can further improve the prediction accuracy and stability of the model.

**Table 7** Parameter study of data batch size

Batch Size	MAE	MAPE(%)	RMSE
32	15.76	9.87	24.95
64	15.66	9.86	24.71
128	15.78	9.92	25.00

**Table 8**  $T_F$  values corresponding to each metric

Metrics	$T_F$
MAE	64.6923
RMSE	52.3846
MAPE	45.5890

**4.5.6 Statistical analysis**

To verify the performance differences between our model and the baseline models listed in Table 2, we conducted the Friedman test and Nemenyi post-hoc test.

The Friedman test assumes that all  $k$  compared methods exhibit the same performance across  $N$  datasets. The first step is calculating  $T_{x^2}$  and  $T_F$  according,

$$T_F = \frac{(N - 1) T_{x^2}}{N(k - 1) - T_{x^2}} \tag{30}$$

where  $T_{x^2} = \frac{12N}{k(k+1)} (\sum_{i=1}^k r_i^2 - \frac{k(k+1)}{4})$  and  $r_i$  represents the average rank value of the  $i$ -th model. In addition,  $T_F$  obeys the F-distribution with degrees of freedom  $k - 1$  and  $(k - 1)(N - 1)$ .

The second step tests whether the assumption is true by comparing  $T_F$  and its corresponding threshold. If the assumption is denied, there are significant differences in the performance of the models being compared. Then, a post hoc test is required to further distinguish the algorithms. The Nemenyi test is a common post-hoc test.

The Nemenyi calculates the critical distance by Formula 31 to reflect the differences between the average rank values of each method:

$$CD = q_\alpha \sqrt{\frac{k + 1}{6N}} \tag{31}$$

where  $q_\alpha$  represents the critical value of the Tukey distribution and  $CD$  is the critical value for the Nemenyi test. If the difference between the average rank values of two methods exceeds the critical value range, it indicates significant performance differences between the two methods; otherwise, no significant differences are observed.

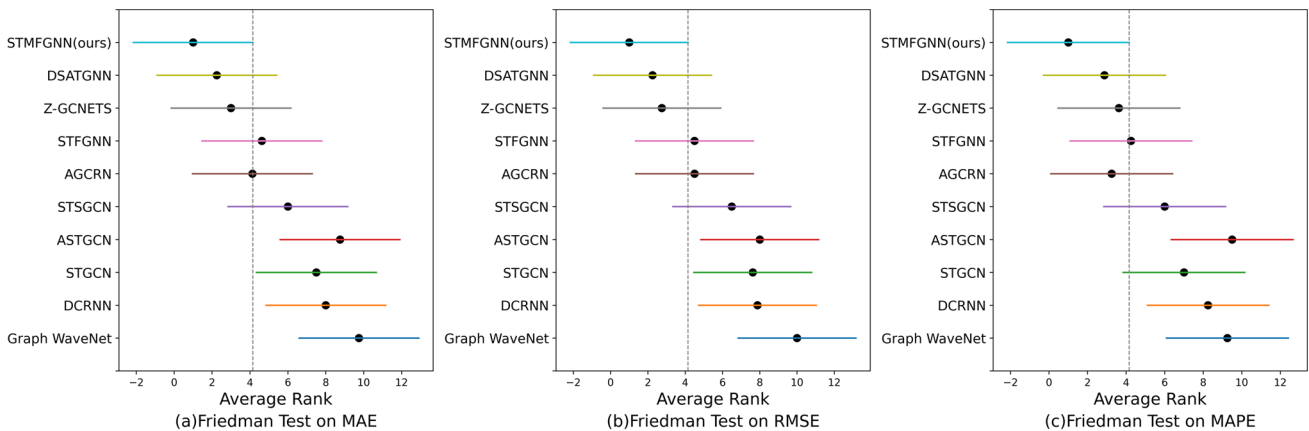
In our experiment,  $N = 4$  and  $k = 0$ . When  $\alpha = 0.05$ , according to (30), Table 8 displays the corresponding values for the MAE, RMSE, and MAPE, all of which surpass the threshold of 2.2501. In other words, the assumption that all algorithms have the same performance is denied; therefore, we need to use the Nemenyi test to continue verification. With  $CD = 6.3113$ , as calculated by (31), we generated the Friedman test graph, as shown in Fig. 10.

Overall, we can conclude that, overall, our proposed STMFGNN exhibits significant differences in MAE, RMSE, and MAPE compared to those of the STFGNN, AGCRN, STSGCN, ASTGCN, STGCN, DCRNN, and Graph WaveNet, with no substantial differences observed compared to those of the top-performing methods, DSTAGNN and Z-GCNETS, in recent years. As shown in Fig. 10, our method achieves the best average ranking in comparison with the other methods regardless of which metric is used. In summary, our proposed method is statistically superior.

**4.5.7 Time cost study**

The time cost of each module in our model is closely related to the size of the dataset. To show the time cost more intuitively, we test the time cost of the proposed model on four datasets under the experimental settings described in Section 4.3, and the results are shown in Table 9.

The data imputation module includes a generative adversarial imputation network, which includes a generator and a discriminator. The time complexity of the generator and discriminator can be expressed as  $O(1)$  or  $O(k)$ , where  $k$  is the



**Fig. 10** Friedman test charts. For each algorithm, the blue dot marks its average rank. The horizontal lines with the dot as the center indicate the critical distance. No overlapping areas of the lines indicate a significant difference

**Table 9** STMFGNN run speed comparison

	PEMS03	PEMS04	PEMS07	PEMS08
Data Imputation Module(s)	29	22	139	11
Precomputing the Dynamic Graph (s)	1604	1374	5400	655
Graph Convolution Process (s/epoch)	240	128	425	70
Time Series Prediction (s/epoch)	324	210	500	112
STMFGNN (s/epoch)	578	460	1224	204

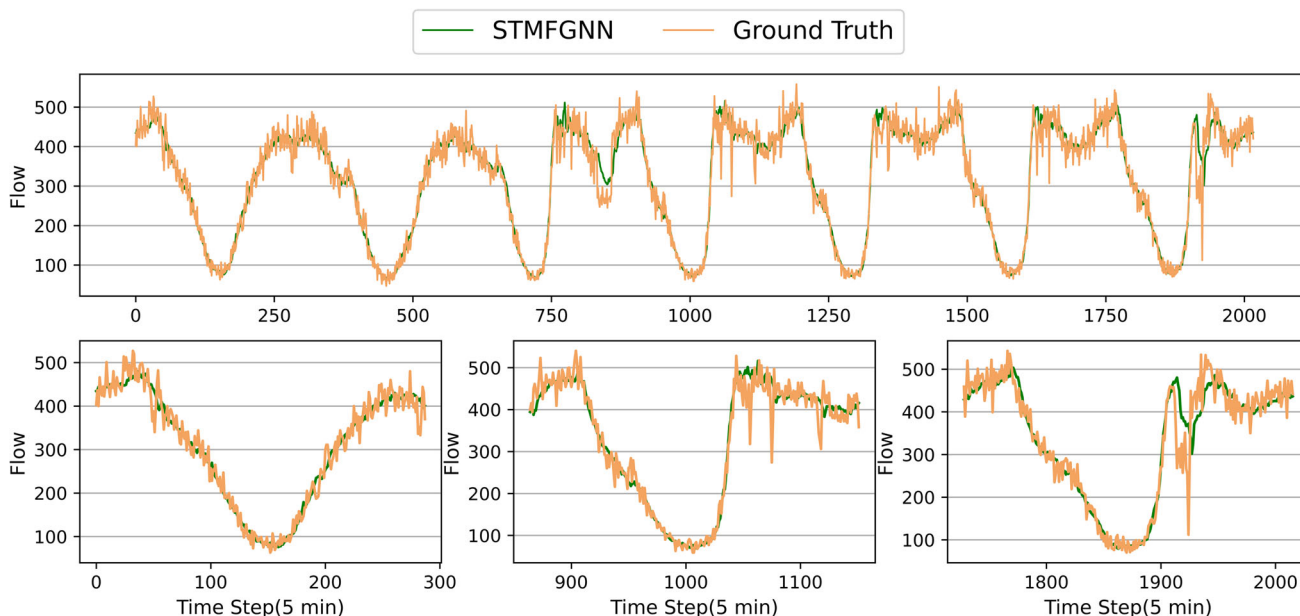
number of network parameters, and usually does not change significantly with the size of the dataset. The time complexity of training a generative adversarial imputation network is related to the dataset size and is usually expressed as  $O(TN)$ , where  $T$  represents the total time step in the dataset and  $N$  represents the number of sensors. Specifically, Table 9 shows that the time cost of the data imputation module on different datasets, and the PEMS07 dataset has the highest time cost of 139 seconds, which correlates with its larger data size.

The time cost of the spatial fusion module mainly stems from the process of precomputing the dynamic graph and graph convolution process. For the precalculated dynamic graphs, the time complexity can be expressed as  $O(DN^2)$ . In this prediction task, sensors collect data every 5 minutes, and the dynamic correlations of nodes change according to the time of day, so the total time steps are divided according to the day as the segmentation step; thus,  $O(kNd)$ . For graph convolution, the  $k$ -order Chebyshev polynomial is used for graph signal aggregation, and the computational time complexity is denoted by  $O(kNd)$ , where  $k$  represents the order of the Chebyshev polynomial,  $N$  represents the number of nodes in the graph, and  $d$  represents the feature dimension

of each node. Table 9 shows that the dynamic graph computation time cost is the highest on PEMS07 dataset, which is 5400 seconds, mainly due to its large number of nodes and long time step.

The time cost of the time series prediction part mainly comes from the GRU. For a sequence length of  $Q$ , the total time complexity can be expressed as  $O(Qd)$ , where  $Q$  is the sequence length and  $d$  is the dimension of the input features or the dimension of the hidden states. In the traffic flow prediction task, the prediction target is usually the traffic condition for the next hour, which means that  $Q$  is 12. Table 9 shows that the time series prediction block has the highest time cost of 500 seconds on the PEMS07 dataset, which is related to the sequence length and feature dimension.

Overall, the time cost of model training increases with the assembly of individual modules and increases with increasing dataset size. However, as seen in the previous ablation experiments, each part is critical to the accuracy of the model's final prediction of the target. The collaborative work of these modules enables STMFGNN to perform excellently in traffic flow prediction tasks, maintaining high accuracy even with large datasets and missing data.



**Fig. 11** Visualization of the prediction results at node 0 for 15 mins

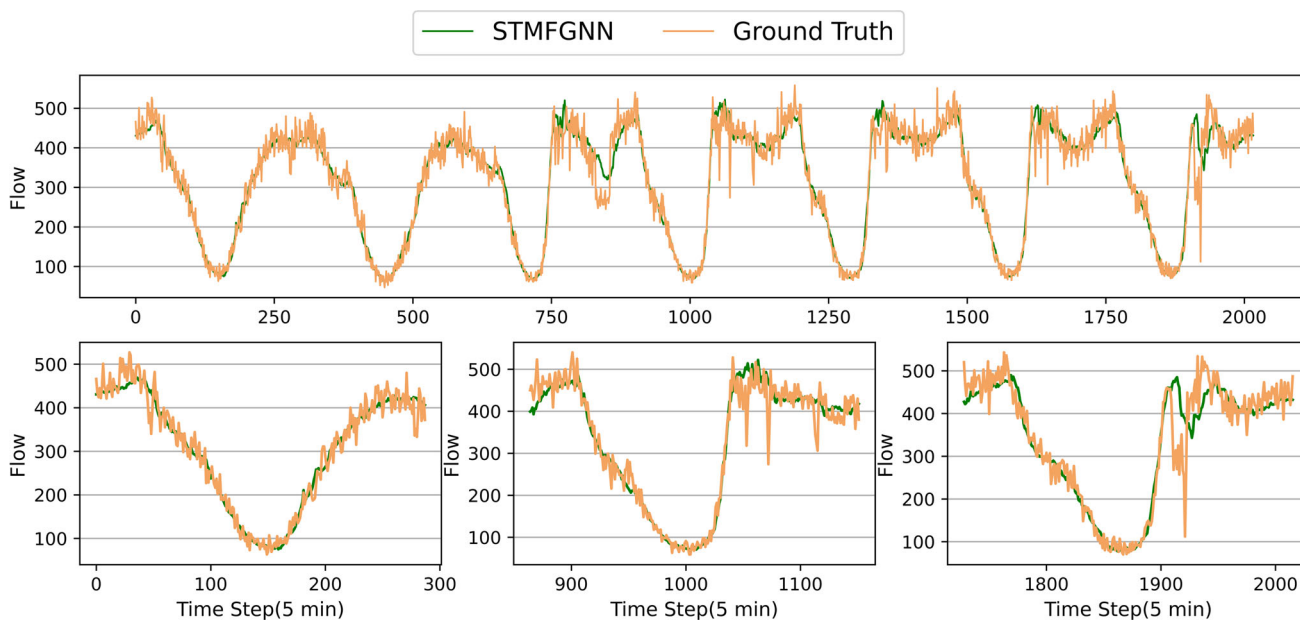


Fig. 12 Visualization of the prediction results at node 0 for 30 mins

#### 4.5.8 Visualization of forecast results

We compared the one-week traffic flow predictions on the PEMS08 dataset with the actual values and zoomed in on the data for the first, fourth, and seventh days. Figures 11, 12, and 13 compare the model’s predictions with the actual values on PEMS08 at node 0 for 15 minutes, 30 minutes, and 60 minutes, respectively. In addition, to show that the model performs consistently across nodes, Fig. 14 com-

pares the actual values and the predictions at node 1 for 60 minutes.

We observe that (1) The general trend of the STMFGNN is consistent with the actual values and it can track the change pattern of traffic flow well. This indicates the model’s excellent performance in capturing spatiotemporal dependencies, accurately reflecting real-world traffic flow changes. (2) In instances where the actual values are significantly lower, this discrepancy may be attributed to the presence of missing

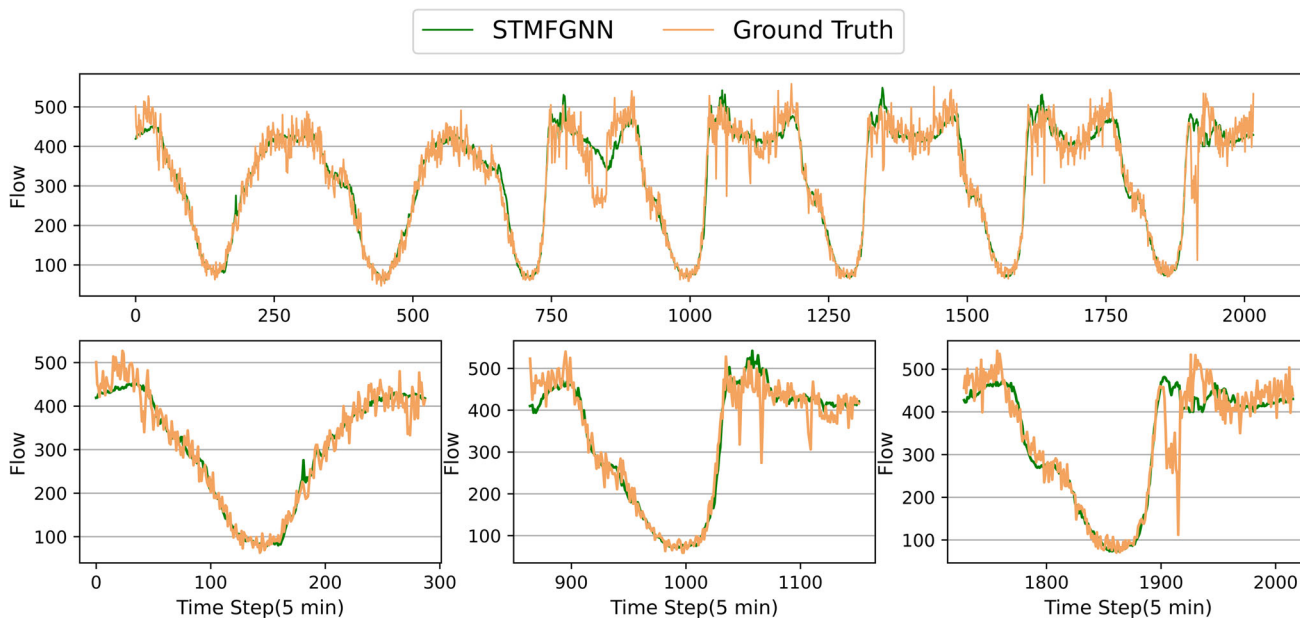
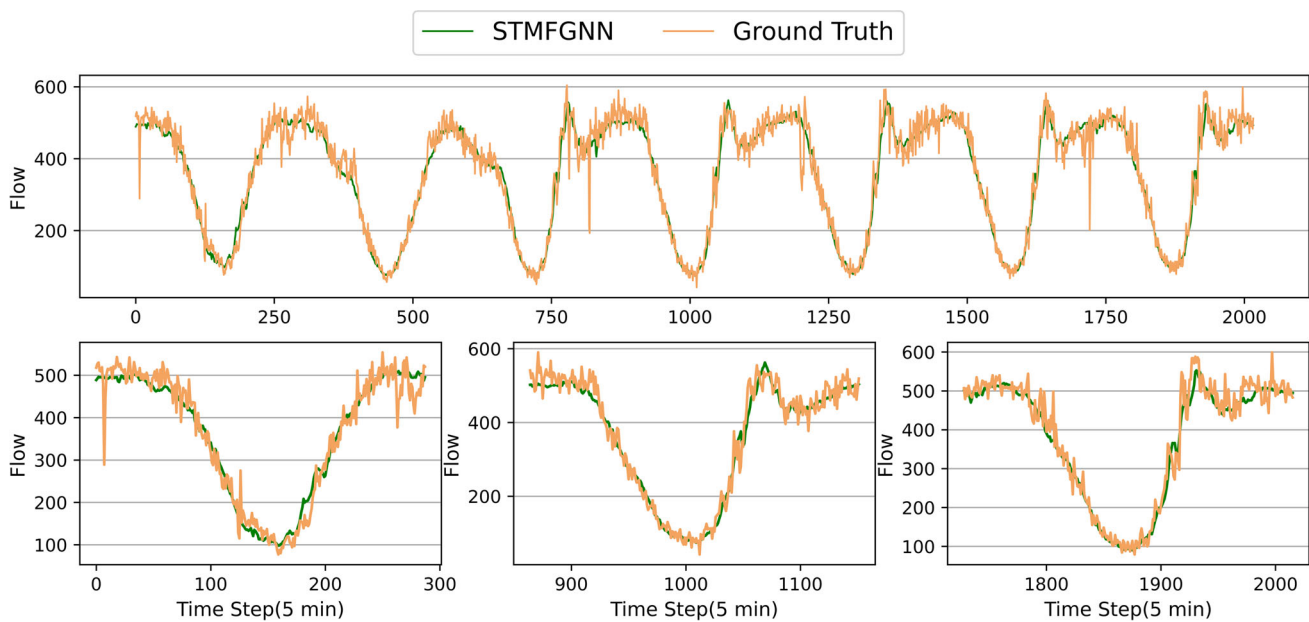


Fig. 13 Visualization of the prediction results at node 0 for 60 mins



**Fig. 14** Visualization of the prediction results at node 1 for 60 mins

data points. Our model does not follow these extreme deviations but instead predicts values that conform to the overall distribution. This showcases STMFGNN's robustness in handling missing data. Through an effective data imputation method, the model maintains high predictive accuracy even with incomplete data. (3) Peak values are further accentuated, illustrating STMFGNN's ability to perform well under challenging conditions. In the face of sharp changes in traffic flow, the model can still accurately predict the peak flow, which proves its modeling ability under complex spatio-temporal dependencies.

In summary, STMFGNN accurately captures the trends in traffic flow across different nodes and time intervals, exhibiting outstanding performance. These experimental results highlight the model's significant advantage in capturing spatio-temporal dependencies. Particularly in handling missing data, our imputation strategy enables the model to deliver high-quality predictions despite data incompleteness.

## 5 Conclusion

This paper proposes a new GNN framework, the STMFGNN, for traffic flow prediction. This model considers the influence of multiple factors when modeling spatial-temporal dependencies and fuses multiple factors by using a dynamic gated fusion mechanism to self-learn dynamic importance. When modeling spatial dependence, we dynamically fuse the static adjacency graph as local prior knowledge and the dynamic similarity graph as global hidden knowledge to model complex spatial dependence as completely as possible.

Regarding temporal dependence, the short-term characteristics of different receptive field gated convolutions and the long-term information captured by the GRU are dynamically fused, which overcomes the trade-off problem between short-term and long-term temporal dependence. In addition, we imputed the traffic data presenting incomplete observability using a generative adversarial imputation network with an improved masking mechanism and training process. Experimental results on four real datasets (PEMS03, PEMS04, PEMS07, and PEMS08) show that the STMFGNN achieves state-of-the-art results.

However, the model proposed in this paper has several areas of potential improvement, such as insufficient consideration of the influence of external factors and underutilization of data periodicity. In the future, we will consider the influence of external factors (such as weather) on data fluctuations and model the periodicity of data to further optimize the integrity of traffic flow prediction model modeling and improve the accuracy of prediction. In addition, since STMFGNN is a general spatio-temporal prediction framework for graph-structured data, we can also apply it to other applications, such as traffic speed prediction.

**Author Contributions** All authors read and approved the final manuscript. Zixuan Yu: Methodology, Software, Programming, Writing. Hui Jia: Conceptualization, Validation, Supervision, Reviewing. Yanping Chen: Supervision, Reviewing. Hong Xia: Conceptualization, Supervision, Reviewing.

**Funding** This work was partly supported by the Youth Innovation Team Project of Science Research Program Project of Shaanxi Provincial Department of Education (No. 23JP164), the National Natural Science Foundation of China (No. 62301425).

**Data availability and access** The PEMS03, PEMS04, PEMS07 and PEMS08 datasets analysed during the current study are available in the ASTGNN repository, <https://github.com/guoshnBJTU/ASTGNN>.

## Declarations

**Competing interests** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

**Ethical and informed consent for data used** All authors strictly understand and comply with the ethical and informed consent for data used.

## References

- Poonia P, Jain V, Kumar A (2018) Short term traffic flow prediction methodologies: a review. *Mody Univ Int J Comput Eng Res* 37–39
- Lu H, Ge Z, Song Y, Jiang D, Zhou T, Qin J (2021) A temporal-aware lstm enhanced by loss-switch mechanism for traffic flow forecasting. *Neurocomputing* 427:169–178
- Cui Z, Henrickson K, Ke R, Wang Y (2019) Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Trans Intell Transp Syst* 21(11):4883–4894
- Yuan H, Li G (2021) A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Sci Eng* 6(1):63–85. <https://doi.org/10.1007/s41019-020-00151-z>
- Xu M, Liu H (2021) A flexible deep learning-aware framework for travel time prediction considering traffic event. *Eng Appl Artif Intell* 106:104491. <https://doi.org/10.1016/j.engappai.2021.104491>
- Tian R, Wang C, Hu J, Ma Z (2023) MFSTGN: a multi-scale spatial-temporal fusion graph network for traffic prediction. *Appl Intell* 53(19):22582–22601. <https://doi.org/10.1007/s10489-023-04703-4>
- Yin X, Wu G, Wei J, Shen Y, Qi H, Yin B (2022) Deep learning on traffic prediction: Methods, analysis, and future directions. *IEEE Trans Intell Transp Syst* 23(6):4927–4943. <https://doi.org/10.1109/TITS.2021.3054840>
- Jiang W, Luo J (2022) Graph neural network for traffic forecasting: A survey. *Expert Syst Appl* 207:117921. <https://doi.org/10.1016/j.eswa.2022.117921>
- Feng K, Ji J, Zhang Y, Ni Q, Liu Z, Beer M (2023) Digital twin-driven intelligent assessment of gear surface degradation. *Mech Syst Signal Process* 186:109896
- Ni Q, Ji J, Halkon B, Feng K, Nandi AK (2023) Physics-informed residual network (piresnet) for rolling element bearing fault diagnostics. *Mech Syst Signal Process* 200:110544
- Ni Q, Ji J, Feng K, Zhang Y, Lin D, Zheng J (2024) Data-driven bearing health management using a novel multi-scale fused feature and gated recurrent unit. *Reliab Eng Syst Saf* 242:109753
- Suganyadevi S, Seethalakshmi V, Balasamy K (2022) A review on deep learning in medical image analysis. *Int J Multimed Inf Retr* 11(1):19–38. <https://doi.org/10.1007/s13735-021-00218-1>
- Varoquaux G, Cheplygina V (2022) Machine learning for medical imaging: methodological failures and recommendations for the future. *npj Digital Med* 5(1):48. <https://doi.org/10.1038/s41746-022-00592-y>
- Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32(1):4–24
- Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, Wang L, Li C, Sun M (2020) Graph neural networks: A review of methods and applications. *AI open*. 1:57–81
- Duan W, Xuan J, Qiao M, Lu J (2022) Learning from the dark: boosting graph convolutional neural networks with diverse negative samples. *Proc AAAI Conf Artif Intell* 36:6550–6558
- Li Y, Yu R, Shahabi C, Liu Y (2018) Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings. OpenReview.net. <https://openreview.net/forum?id=SJiHXGWAZ>
- Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proc AAAI Conf Artif Intell* 33(01):922–929. <https://doi.org/10.1609/aaai.v33i01.3301922>
- Song C, Lin Y, Guo S, Wan H (2020) Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. *Proc AAAI Conf Artif Intell* 01:914–921. <https://doi.org/10.1609/aaai.v34i01.5438>
- Lan S, Ma Y, Huang W, Wang W, Yang H, Li P (2022) DSTAGNN: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In: Chaudhuri K, Jegelka S, Song L, Szepesvari C, Niu G, Sabato S. (eds.) Proceedings of the 39th International Conference on Machine Learning, vol. 42. PMLR, Baltimore, Maryland, USA, pp 11906–11917
- Song C, Lin Y, Guo S, Wan H (2020) Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. *Proc AAAI Conf Artif Intell* 34(01):914–921. <https://doi.org/10.1609/aaai.v34i01.5438>
- Lau YH, Wong RC-W (2021) Spatio-temporal graph convolutional networks for traffic forecasting: Spatial layers first or temporal layers first? In: Proceedings of the 29th International Conference on Advances in Geographic Information Systems. SIGSPATIAL '21. Association for Computing Machinery, New York, NY, USA, pp 427–430. <https://doi.org/10.1145/3474717.3484207>
- Wu Z, Pan S, Long G, Jiang J, Zhang C (2019) Graph wavenet for deep spatial-temporal graph modeling. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. IJCAI'19. AAAI Press, Washington, DC, USA, pp 1907–1913
- Zhang W, Zhang P, Yu Y, Li X, Biancardo SA, Zhang J (2021) Missing data repairs for traffic flow with self-attention generative adversarial imputation net. *IEEE Trans Intell Transp Syst* 23(7):7919–7930
- Yoon J, Jordon J, Schaar M (2018) GAIN: Missing data imputation using generative adversarial nets. In: Dy J, Krause A (eds) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80. PMLR, Baltimore, Maryland, USA, pp 5689–5698
- Panaretos VM, Zemel Y (2019) Statistical aspects of Wasserstein distances. *Annu Rev Stat Appl* 6:405–431
- Hamzaçebi C (2008) Improving artificial neural networks' performance in seasonal time series forecasting. *Inf Sci* 178(23):4550–4559. <https://doi.org/10.1016/j.ins.2008.07.024>
- Hudak AT, Crookston NL, Evans JS, Hall DE, Falkowski MJ (2008) Nearest neighbor imputation of species-level, plot-scale forest structure attributes from lidar data. *Remote Sens Environ* 112(5):2232–2245. <https://doi.org/10.1016/j.rse.2007.10.009>
- Qu L, Li L, Zhang Y, Hu J (2009) Ppca-based missing data imputation for traffic flow volume: A systematical approach. *IEEE Trans Intell Transp Syst* 10(3):512–522. <https://doi.org/10.1109/TITS.2009.2026312>
- Berglund M, Raiko T, Honkala M, Kärkkäinen L, Vetek A, Karhunen J (2015) Bidirectional recurrent neural networks as generative models. In: Proceedings of the 28th International Con-

- ference on Neural Information Processing Systems - Volume 1. NIPS'15. MIT Press, Cambridge, MA, USA, pp 856–864
31. Benkraouda O, Thodi BT, Yeo H, Menéndez M, Jabari SE (2020) Traffic data imputation using deep convolutional neural networks. *IEEE Access*. 8:104740–104752. <https://doi.org/10.1109/ACCESS.2020.2999662>
  32. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2020) Generative adversarial networks. *Commun ACM* 63(11):139–144. <https://doi.org/10.1145/3422622>
  33. Wu Z, Pan S, Chen F, Long G, Zhang C, Philip SY (2020) A comprehensive survey on graph neural networks. *IEEE Trans Neural Netw Learn Syst* 32(1):4–24
  34. Bruna J, Zaremba W, Szlam A, LeCun Y (2013) Spectral networks and locally connected networks on graphs. *CoRR*. [arXiv:1312.6203](https://arxiv.org/abs/1312.6203)
  35. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: *Proceedings of the 30th international conference on neural information processing systems*. NIPS'16. Curran Associates Inc., Red Hook, NY, USA, pp 3844–3852
  36. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net, Toulon, France. <https://openreview.net/forum?id=SJU4ayYgl>
  37. Liu Z, Zhou J (2020) *Graph Attention Networks*. Springer, Cham, pp 39–41. [https://doi.org/10.1007/978-3-031-01587-8\\_7](https://doi.org/10.1007/978-3-031-01587-8_7)
  38. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) *Advances in neural information processing systems*, vol 30. Curran Associates Inc, Red Hook, NY
  39. Williams BM, Hoel LA (2003) Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *J Transp Eng* 129(6):664–672
  40. Drucker H, Burges CJC, Kaufman L, Smola A, Vapnik V (1996) Support vector regression machines. In: *Proceedings of the 9th International Conference on Neural Information Processing Systems*. NIPS'96. MIT Press, Cambridge, MA, USA, pp 155–161
  41. Shi X, Chen Z, Wang H, Yeung D-Y, Wong W-k, Woo W-c (2015) Convolutional lstm network: a machine learning approach for precipitation nowcasting. In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. MIT Press, Cambridge, MA, USA, pp 802–810
  42. Yao H, Wu F, Ke J, Tang X, Jia Y, Lu S, Gong P, Ye J, Chuxing D, Li Z (2018) Deep multi-view spatial-temporal network for taxi demand prediction. In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. AAAI'18/IAAI'18/EAAI'18. AAAI Press, Washington D.C (USA)
  43. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. *Adv Neural Inf Process Syst* 30
  44. Yu F, Koltun V (2016) Multi-scale context aggregation by dilated convolutions. In: Bengio Y, LeCun Y (eds) *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings*. [arXiv:1511.07122](https://arxiv.org/abs/1511.07122)
  45. Wang X, Ma Y, Wang Y, Jin W, Wang X, Tang J, Jia C, Yu J (2020) Traffic flow prediction via spatial temporal graph neural network. In: *Proceedings of The Web Conference 2020*. Association for Computing Machinery, New York, NY, USA, pp 1082–1092. <https://doi.org/10.1145/3366423.3380186>
  46. Xia D, Yang N, Jian S, Hu Y, Li H (2022) Sw-bilstm: a spark-based weighted bilstm model for traffic flow forecasting. *Multimed Tool Appl* 81:1–26. <https://doi.org/10.1007/s11042-022-12039-3>
  47. Cho K, Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Moschitti A, Pang B, Daelemans W (eds) *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pp 1724–1734 <https://doi.org/10.3115/v1/D14-1179>, <https://aclanthology.org/D14-1179>
  48. Hochreiter S, Schmidhuber J (1997) Long Short-Term Memory. *Neural Comput* 9(8):1735–1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
  49. Bai L, Yao L, Li C, Wang X, Wang C (2020) Adaptive graph convolutional recurrent network for traffic forecasting. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Curran Associates Inc., Red Hook, NY, USA
  50. Li M, Zhu Z (2021) Spatial-temporal fusion graph neural networks for traffic flow forecasting. *Proc AAAI Conf Artif Intell* 35:4189–4196
  51. Chen Y, Segovia-Dominguez I, Gel YR (2021) Z-gcnets: Time zigzags at graph convolutional networks for time series forecasting. In: Meila M, Zhang T (eds) *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18–24 July 2021, Virtual Event*. *Proceedings of Machine Learning Research*, vol. 139. pp 1684–1694. <http://proceedings.mlr.press/v139/chen21o.html>
  52. Stefenon SF, Ribeiro MHD, Nied A, Mariani VC, Coelho LDS, Leithardt VRQ, Silva LA, Seman LO (2021) Hybrid wavelet stacking ensemble model for insulators contamination forecasting. *IEEE Access*. 9:66387–66397

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.