Transportation Research Part C 139 (2022) 103659

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

AdapGL: An adaptive graph learning algorithm for traffic prediction based on spatiotemporal neural networks

Wei Zhang [a,b], Fenghua Zhu [a,*], Yisheng Lv [a,*], Chang Tan [c], Wen Liu [d], Xin Zhang [e], Fei-Yue Wang [a,f]

[a] The State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] University of Chinese Academy of Sciences, Beijing 100049, China
[c] iFLYTEK CO. LTD, Hefei 230088, China
[d] Hubei Key Laboratory of Inland Shipping Technology, School of Navigation, Wuhan University of Technology, Wuhan 430063, China
[e] Beijing Municipal Institute of City Planning and Design, 100045, Beijing, China
[f] Institute of Systems Engineering, Macau University of Science and Technology, Macao Special Administrative Region of China

ARTICLE INFO

Keywords:
Adaptive graph learning
Traffic prediction
Graph convolutional network
Expectation maximization
Deep learning

ABSTRACT

With well-defined graphs, graph convolution based spatiotemporal neural networks for traffic prediction have achieved great performance in numerous tasks. Compared to other methods, the networks can exploit the latent spatial dependencies between nodes according to the adjacency relationship. However, as the topological structure of the real road network tends to be intricate, it is difficult to accurately quantify the correlations between nodes in advance. In this paper, we propose a graph convolutional network based adaptive graph learning algorithm (AdapGL) to acquire the complex dependencies. First, by developing a novel graph learning module, more possible correlations between nodes can be adaptively captured during training. Second, inspired by the expectation maximization (EM) algorithm, the parameters of the prediction network module and the graph learning module are optimized by alternate training. An elaborate loss function is leveraged for graph learning to ensure the sparsity of the generated affinity matrix. In this way, the expectation maximization of one part can be realized under the condition that the other part is the best estimate. Finally, the graph structure is updated by a weighted sum approach. The proposed algorithm can be applied to most graph convolution based networks for traffic forecast. Experimental results demonstrated that our method can not only further improve the accuracy of traffic prediction, but also effectively exploit the hidden correlations of the nodes. The source code is available at https://github.com/goaheand/AdapGL-pytorch.

1. Introduction

With the rapid development of city, the contradiction between the growing number of vehicles and the insufficient capacity of urban traffic becomes more and more serious. Intelligent Transportation Systems (ITS) is one of the key techniques to solve the problem (Wang, 2014). By mastering the future traffic conditions with the monitored data in real time, ITS can enable the effective coordination among travelers, vehicles and roads, thereby alleviating traffic congestion. Undoubtedly, the accuracy of traffic prediction is essential for the real-time management and effective decision-making. As an integral part of ITS, traffic prediction

* Corresponding authors.
E-mail addresses: fenghua.zhu@ia.ac.cn (F. Zhu), yisheng.lv@ia.ac.cn (Y. Lv).

https://doi.org/10.1016/j.trc.2022.103659
Received 4 June 2021; Received in revised form 23 January 2022; Accepted 20 March 2022
Available online 20 April 2022

has been studied using a wide range of approaches, such as classical statistical methods, machine learning methods, deep learning methods and so on.

Traffic prediction aims to estimate the target state (e.g., traffic flow and speed) in the near future using the observed data, and can be mainly divided into one-to-one prediction and many-to-many prediction, respectively (Chen et al., 2018; Dai et al., 2019). For one-to-one prediction, many approaches focus on the modeling of temporal dependencies of time series, like autoregressive integrated moving average (ARIMA) model (Ahmed and Cook, 1980) and Bayesian model. For many-to-many prediction, how to effectively extract the complex spatial and temporal correlations among different nodes is the main challenge faced by most approaches. Initially, researchers leveraged grid-based map segmentation to convert the studied area to a regular image. Then, spatial dependencies can be depicted by convolutional neural networks (CNNs) (Zhang et al., 2016; Liu et al., 2019a). The main problem of this approach is that the inherent structure of road network is ignored completely and the correlated grids are usually modeled independently, which conflicts with the reality. To this end, graph convolution based spatial–temporal neural networks, which can handle non-Euclidean structured data by aggregating the feature information from neighbors, have attracted great attention throughout the world. With pre-defined graphs, the latent spatial features of nodes (e.g., stations and sensors) can be extracted more effectively compared to other methods and appreciable performance have been achieved.

Recently, many networks have met the bottleneck which is caused by the intricate road network and the unreliable correlations between nodes. On the one hand, there is a lack of the details of the structure of the studied area, thus, the pre-defined affinity matrices are mainly generated based on expert experience and are usually inaccurate. This will hinder the extraction of spatial dependencies hidden in the traffic data, and the relationships among nodes have to be discovered from real data instead of being provided as ground truth knowledge (Wu et al., 2020). On the other hand, for some spatiotemporal graph neural networks, the framework is divided into two parts, which are a graph learning module and a prediction network module, respectively, but the result of the end-to-end training approach is unsatisfactory (Li et al., 2019b; Wu et al., 2020). It is difficult to correctly control the training direction of learning parameters, thus the information provided by the generated graph is unclear and the improvement of prediction performance is very limited. Specifically, the adaptive training of graph learning module depends on the effectiveness of the other module, but both the two modules are nonoptimal before training, so it is difficult to control the learning direction of modules for an end-to-end training method. In summary, there is still no effective method to optimize the graph structure, which plays a significant role on traffic prediction based on spatiotemporal neural networks.

To address the above problem, we propose an graph convolutional network based adaptive graph learning algorithm (AdapGL). First, a novel Parameterized Graph Learning module (PGL) is designed, and the whole traffic prediction framework is divided into two parts, which are a PGL module to generate a possible graph and a Prediction Network module (PN) to get the final result, respectively. A graph set is built by some pre-defined graphs (e.g., distance and correlation coefficient). Second, an alternate training approach is leveraged for the optimization process to replace the traditional end-to-end training. Specifically, in the training process for the PN module, a pre-defined graph generated by the graph set is fixed as the expectation of the optimal structure. Then, in the training process of the PGL module, the optimized PN module is fixed to generate a more effective adjacency matrix. The two training processes run one by one in a circle, and a well-designed loss function is utilized to ensure the sparsity of the generated graph. Finally, a weighted average approach is designed to update the current graph structure to control the stability of the training process. As a result, the performance of the prediction network module can be further improved with more accurate spatial dependencies. Furthermore, the updated graph can help exploit the hidden correlations of nodes, which is valuable for transportation management and control applications. The convergence of the EM-based AdapGL algorithm can be guaranteed, and this method can be applied to not only existing graph convolution based spatiotemporal prediction networks, but also other kinds of tasks. To evaluate the effectiveness of AdapGL, experiments on three GCN-based neural networks and four real-world datasets are carried out for the multi-step traffic prediction. The results demonstrate that our algorithm can generate a more reliable graph and improve the performance of the tested networks significantly.

The rest of the paper is organized as follows. Section 2 presents a brief review of the recent work on traffic prediction, which are based on deep neural networks. Section 3 introduces the related information, including the graph definition and the problem description. Section 4 proposes the AdapGL algorithm, which is composed of a PGL module and a GCN-based PN module, and the training process is introduced in details. Section 5 illustrates the deployment of the experiment, and the results are analyzed. Finally, Section 6 summarizes the paper.

2. Literature review

Early Studies. As a problem of time series forecasting, traffic flow prediction has been studied for decades. In the early stage, ARIMA and its variants played a dominant role in this field. Then, many other methods were utilized to address the problem, which can be divided into three categories, including parametric techniques, nonparametric approaches and simulations (Lv et al., 2015). Supporting Vector Regression (SVR) (Jeong et al., 2013) and Random Forest Regression (Leshem and Ritov, 2007) are two typical machine learning based parametric models. K-Nearest Neighbor (KNN) (Davis and Nihan, 1991) and Bayesian networks (Sun et al., 2006) were two common nonparametric models.

Deep Learning based Methods. The non-trivial ability of deep learning models to extract the high-dimensional features of data has led to tremendous resounding successes in various tasks, especially in computer vision, natural language processing, etc (Shen et al., 2020, 2021). Deep neural networks have also been widely applied in traffic forecasting to improve the accuracy. Lv et al. (2015) first used Stacked Autoencoders (SAE) to extract the hidden features of traffic flow data for traffic prediction. Long Short-Term Memory (LSTM) and CNNs based frameworks are two typical approaches to capture the temporal and spatial patterns of the

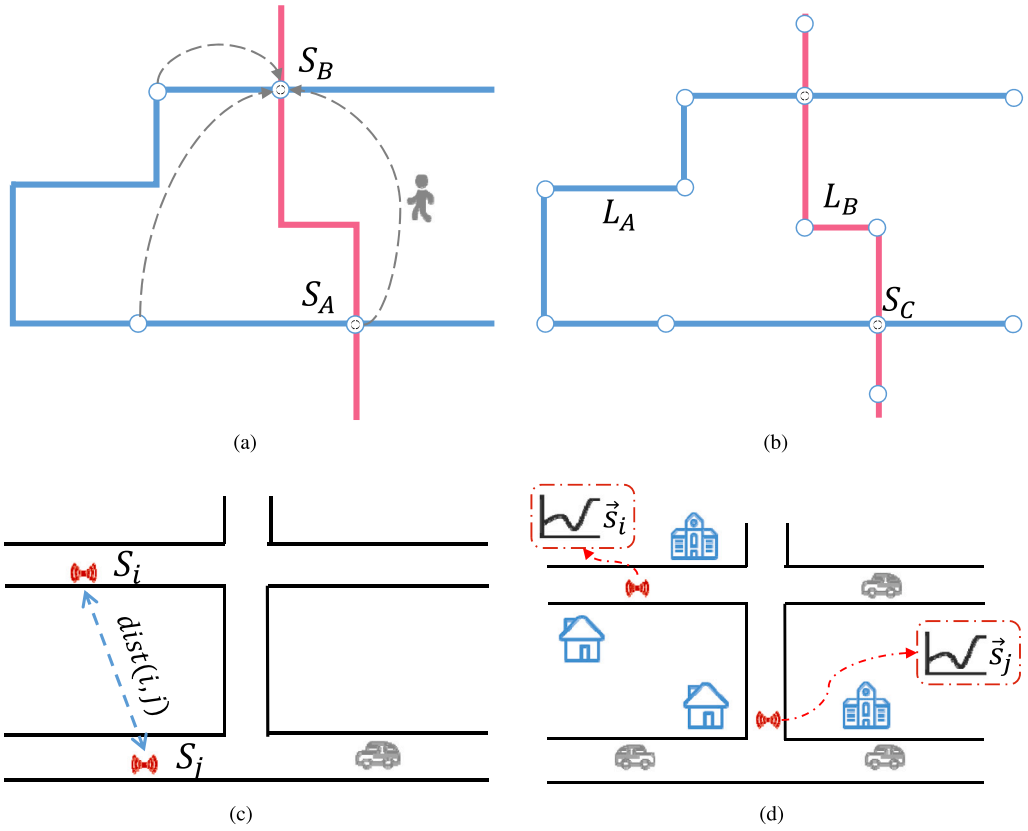


Fig. 1. Different types of relationships between nodes. (a) OD relationship; (b) Transfer relationship; (c) Distance relationship; (d) Correlation coefficient relationship.

traffic data, which attracted the interest of numerous researchers (Dai et al., 2019; Duan et al., 2016). Liu et al. (2017) proposed a hybrid model called Conv-LSTM, which combines LSTM and CNNs together to extract the spatiotemporal dependencies between nodes. Generative adversarial networks (GANs) is another widely applied model for traffic prediction (Chen et al., 2018; Lv et al., 2018). The networks consist of a generator and a discriminator, which accomplish the forecasting task and the evaluation task for the prediction respectively. Regarding that the dynamics of the traffic flow are influenced by multiple factors, Liu et al. (2019b) proposed a hybrid model for passenger flow prediction based on multi-source information fusion. Recently, attention mechanism was extensively used for multi-step traffic prediction to estimate the importance of different features (Cui et al., 2019).

GCN-based Frameworks. The graph convolutional neural networks (Kipf and Welling, 2017) can capture the spatial dependencies that lie on irregular non-Euclidean spaces, and has achieved great successes in various domains, like protein structure, social networks and so on. In recent years, GCN-based spatiotemporal neural networks have become one of the foci in traffic prediction. Based on the diffusion theory, Li et al. (2018) proposed a Diffusion Convolutional Recurrent Neural Network (DCRNN) for average speed prediction, which integrates graph convolution into the recurrent models. With the spatial and temporal attention modules, Attention Based Spatial–Temporal Graph Convolutional Network (ASTGCN) was proposed to exploit the deep spatiotemporal correlations between nodes. Li and Zhanxing (2021) introduced a special temporal graph to improve the performance of the frameworks. Obstructed by the uncertainty of nodes' relationship, there is still no effective approach to extract the spatial dependencies sufficiently. Recently, some researchers used a graph learning module (Li et al., 2019b) to acquire the graph structure. By developing an adaptive dependency matrix, Graph WaveNet (Wu et al., 2019) was proposed to acquire more reliable bi-directional correlations between nodes, but the main problem is that the sparsity of the affinity matrix cannot be guaranteed. Wu et al. (2020) built another graph learning layer to extract the uni-directional dependencies, in which a strategy was used to ensure the sparsity of the generated graph structure, but the results are not ideal to be used in an end-to-end training approach. Guo et al. (2020) utilized a novel Laplace matrix learning layer to dynamically construct the graph matrix according to the input data. The inaccuracy of the pre-defined graph is ignored in this method and the potential correlations between nodes cannot be effectively exploited.

3. Preliminaries

In this section, four different types of relationships between nodes will be briefly introduced, including OD relationship, transfer relationship, distance relationship and correlation coefficient relationship. As shown in Fig. 1, the relationships may be applicable

to distinct objects to be researched (i.e., bus routes and stations), which present valuable prior information for traffic prediction. And the definition of the research problem is formulated.

3.1. OD relationship

Origin–destination (OD) relationship can quantify the correlation between nodes in some cases. Take two metro stations S_A, S_B as an example, there exists an OD relationship between them if there are passengers departing from S_A and ending at S_B. The weight of their correlation can be computed by the average number of passengers in a period, then the OD graph of the studied are A_{OD} can be obtained. Note from that it is not enough to reflect the OD relationship between two nodes when the weight is relatively small, A_{OD} can be modified as

$$A_{OD}^{(i,j)} = \begin{cases} N_{pas}(i,j) & \text{if } N_{pas}(i,j) > TO_i \\ 0 & \text{else} \end{cases} \tag{1}$$

$$A_{OD} \Leftarrow D_{OD}^{-1} A_{OD} \tag{2}$$

where $A_{OD}^{(i,j)}$ is the value corresponding to the ith row and jth column, $N_{pas}(i,j)$ denotes the average number of passengers from station S_j to S_i, TO_i is the threshold of the ith row (i.e., $\frac{\max_j A_{OD}^{(i,j)}}{2}$), D_{OD} is a diagonal matrix and $D_{OD}^{(i,i)} = \sum_j A_{OD}^{(i,j)}$.

3.2. Transfer relationship

Transfer relationship can be utilized to reveal the hidden correlations between lines of public transportation (i.e., bus and subway). For two lines L_A, L_B and a station S_C, if S_C is the stop point of both lines L_A and L_B, there exists a transfer relationship between the two lines. More common stop points of lines means greater transfer probability of passengers, which can be leveraged to quantify the actual correlations. Analogous to the modification process of A_{OD}, a transfer graph A_{Tr} can be obtained as follows.

$$A_{Tr}^{(i,j)} = \begin{cases} N_{sp}(i,j) & \text{if } N_{sp}(i,j) > TR_i \\ 0 & \text{else} \end{cases} \tag{3}$$

$$A_{Tr} \Leftarrow D_{Tr}^{-1} A_{Tr} \tag{4}$$

where $N_{sp}(i,j)$ is the number common stop points between line L_i and L_j, and TR_i is the threshold of the ith row.

3.3. Distance relationship

For the sensors located on the road network, the weight of the edge connecting sensor S_i and sensor S_j can be defined via a threshold Gaussian kernel weighting function (Shuman et al., 2013), as

$$A_D^{(i,j)} = \begin{cases} exp(-\frac{|dist(i,j)|^2}{2\theta^2}) & \text{if } dist(i,j) \le \text{TD} \\ 0 & \text{else} \end{cases} \tag{5}$$

where $A_D^{(i,j)}$ represents the weight of distance graph A_D corresponding to the ith row and jth column, $dist(i,j)$ is the Euclidean distance between S_i and S_j, θ is the standard deviation of distances and TD is the threshold.

3.4. Correlation coefficient relationship

Some researches have shown that source nodes with analogous patterns (i.e., intraday trends) tend to indicate a causal relationship in traffic prediction (Li et al., 2019a). Therefore, the correlation coefficient between the time series of different nodes can reflect the similarities of traffic dynamics, where the most used are Spearman rank correlation coefficient and Pearson correlation coefficient. Let $\vec{s}_i$ be a vector that records the historical data of node S_i, the Pearson correlation coefficient is formulated as

$$r(\vec{s}_i, \vec{s}_j) = \frac{Cov(\vec{s}_i, \vec{s}_j)}{\sqrt{var(\vec{s}_i) var(\vec{s}_j)}} \tag{6}$$

where Cov denotes the covariance between two vectors, and var is the variance of one vector. For the calculation of Spearman rank correlation coefficient, $\vec{s}_i$ should be replaced by the rank of the traffic series. We can define the correlation coefficient matrix A_C as:

$$A_C^{(i,j)} = \begin{cases} |r(\vec{s}_i, \vec{s}_j)| & \text{if } |r(\vec{s}_i, \vec{s}_j)| > TC_i \\ 0 & \text{else} \end{cases} \tag{7}$$

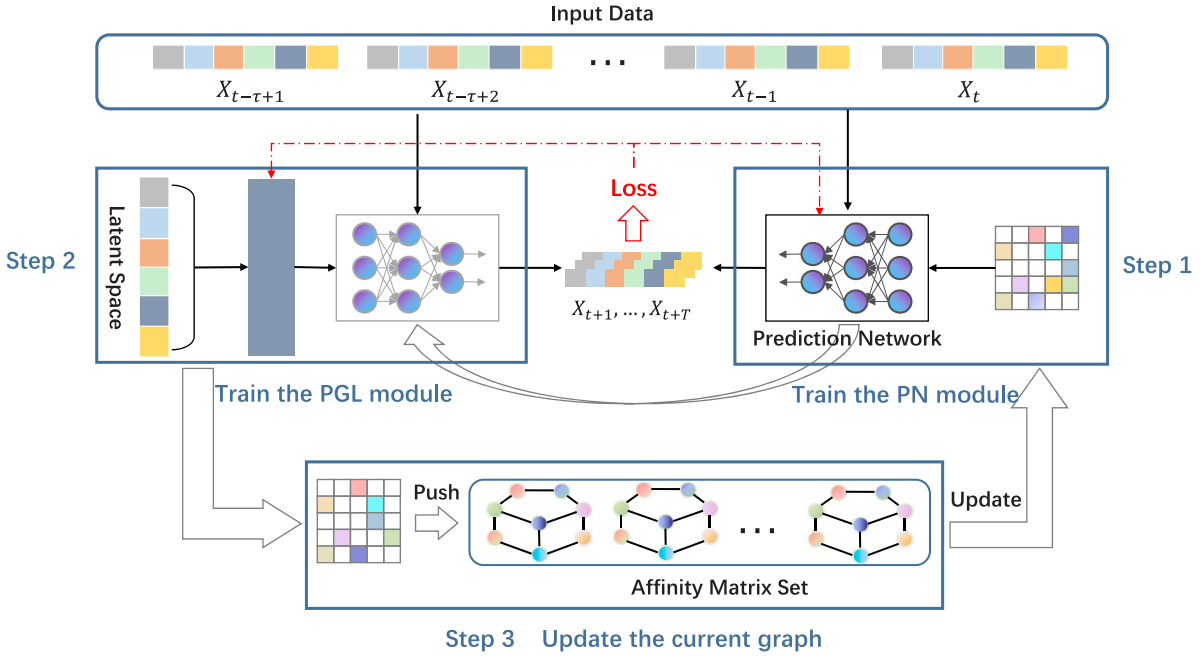


Fig. 2. The overall framework of AdapGL.

3.5. Problem definition

The geometric structure of traffic network can be denoted as $\mathcal{G} = (V, \mathbb{A})$, where V is a set of nodes that represent the sources of time series and $|V| = N$. $\mathbb{A} = \{A_k | k = 1, 2, \ldots, N_r\}$ is composed of different types of relationships (i.e., OD and correlation coefficient) between the nodes, where $A_k \in \mathbb{R}^{N \times N}$ is the affinity matrix dictated by the kth type, and N_r is the number of graphs.

The multivariate traffic data of one time step can be represented as $X_t = (\vec{x_{t,1}}, \vec{x_{t,2}}, \ldots, \vec{x_{t,N}})^T$, where $\vec{x_{t,i}} \in \mathbb{R}^F$ denotes the observed information (i.e., flow volume and average speed) of the ith node at time step t. Take traffic flow prediction as an example, the problem can be formulated as

$$[X_{t-\tau+1}, \ldots, X_{t-1}, X_t; \mathcal{G}] \xrightarrow{P_\Theta(\cdot)} [Y_{t+1}, \ldots, Y_{t+T}] \tag{8}$$

where $P_\Theta(\cdot) : \mathbb{R}^{\tau \times N \times F} \to \mathbb{R}^{N \times T}$ is the target function maps from time series of the past τ steps to the traffic flow volume in the next T steps, Θ is the learnable parameters, and $Y_t \in R^N$ is the traffic flow of all nodes at step t.

4. Adaptive graph learning algorithm based on spatiotemporal neural networks

4.1. Overall structure

The overall structure of AdapGL proposed in this paper is shown in Fig. 2. Let A^* be the optimal affinity matrix that accurately reflects the spatial dependencies between the nodes of the studied area, and Θ^* be the optimal parameters of the prediction network module based on graph convolution. For the optimization of the PN module and the PGL module, the training process is divided into three steps:

1. Taking the current affinity matrix as the estimation of A^*, train the PN module. According to the affinity matrix set, the estimation of A^* can be obtained through the approach provided by step 3. Then, the new matrix is used for the optimization of the prediction network module. As a result, the maximum likelihood estimation of Θ^* at the current step can be acquired.
2. Taking the current parameters of the PN module as the estimation of Θ^*, train the PGL module. As the estimation of Θ^*, the parameters of the PN module are fixed to train the graph learning module. To ensure the sparsity of the generated relations, a well-designed loss function is leveraged for the optimization. As a result, a new affinity matrix is generated through the optimized PGL module, which reweights the spatial dependencies between nodes and contains valuable information.

3. Update the current affinity matrix. First, the generated adjacent matrix is pushed into the affinity matrix set $\mathbb{A}$. Then, the graphs in the set will be fed into the PN module respectively, and their weights can be computed by the corresponding performance. Finally, a weighted sum approach is utilized to update the current affinity matrix, which will be the estimation of A^* in the next iteration.

4.2. Initialization of the current graph

The initialization of the current affinity matrix plays a significant role for AdapGL, which is the same as EM algorithm. The accuracy and reliability of the initial correlations between nodes can greatly affect the optimization of the PN module, and then impact greatly on the performance of both modules. Thus, instead of random initialization, prior information is leveraged to build an affinity matrix set, which includes different types of spatial dependencies. And the current affinity matrix A is initialized by union operation $\cup$, as

$$A_k \Leftarrow \widetilde{D}_k^{-\frac{1}{2}} \widetilde{A}_k \widetilde{D}_k^{-\frac{1}{2}} \tag{9}$$

$$A^{(i,j)} = \frac{\sum_{k=1}^{N_r} A_k^{(i,j)}}{\sum_{k=1}^{N_r} \mathcal{I}[A_k^{(i,j)}]} \tag{10}$$

where $\widetilde{A}_k = A_k + I_N$ and $\widetilde{D}_k^{(ii)} = \sum_j \widetilde{A}_k^{(ij)}$. Eq. (9) is the renormalization trick proposed by Kipf and Welling (2017), which can ensure the comparability of different affinity matrices. $\mathcal{I}$ denotes the indicator function, as

$$\mathcal{I}(x) = \begin{cases} 1 & \text{if } x \neq 0 \\ 0 & \text{else} \end{cases} \tag{11}$$

4.3. Parameterized graph learning module

For the dynamics of the traffic flow, the spatial dependencies between nodes are often not bi-directional. Take the passenger flow of the metro stations as an example, the change of passenger flow at the upstream station will quickly affect the passenger flow at the downstream station, but not vice versa. Therefore, the proposed parameterized graph learning module aims to extract the uni-directional relationships between nodes. First, a coarse affinity matrix is generated:

$$A_1 = ReLU(M_1 M_2^T - M_2 M_1^T + Diag(\Lambda)) \tag{12}$$

where $M_1, M_2 \in \mathbb{R}^{N \times F_0}$ $(F_0 \ll N), \Lambda \in \mathbb{R}^N$ are the learnable parameters, $Diag(\Lambda)$ diagonizes Λ. As $(M_1 M_2^T - M_2 M_1^T)$ is a skew-symmetric matrix, the activation function $ReLU$ sets the diagonal positions and half of the other positions to zero, which can enfore the sparsity of A_1. $Diag(\Lambda)$ is utilized to generate the weights of diagonal positions. Subsequently, an adaptive aggregation module is leveraged to combine the old relationships and the new spatial dependencies, as Eqs. (13) and (14).

$$S = g(h([A_1, A_{old}])) \tag{13}$$

$$A_2 = S \odot A_1 + (1 - S) \odot A_{old} \tag{14}$$

where g denotes a non-linear activation function (e.g. Sigmoid), h is one or more 1×1 convolutional layers, and $\odot$ is the element-wise multiplication. Finally, to further strengthen the sparsity of the generated matrix, A_{new} is computed as

$$A_3 = ReLU(D_2^{-\frac{1}{2}} A_2 D_2^{-\frac{1}{2}} - \epsilon) \tag{15}$$

$$A_{new} = D_3^{-\frac{1}{2}} A_3 D_3^{-\frac{1}{2}} \tag{16}$$

where D_2, D_3 are diagonal matrices and $D_2^{(i,j)} = \sum_{j=1}^{N} A_2^{(i,j)}$, $D_3^{(i,j)} = \sum_{j=1}^{N} A_3^{(i,j)}$. $\epsilon \in (0, 1)$ is a threshold used to filter out some weak relations of A_2.

For the parameterized graph learning module, the following characteristics of the generated matrix can be guaranteed:

1. Sparsity. Eqs. (12) ~ (15) ensure the overall sparsity of the generated matrix, while the number of related nodes for each node is not restricted. It should be noted that some nodes may be strongly correlated with other nodes, while some others are relatively isolated. Therefore, the PGL module can generate more effective relationship between the studied nodes.
2. Similarity. As the new spatial dependencies are generated on the basis of the previous graph, the similarity between A_{new} and A_{old} can be guaranteed. This means that the prior information and the generated information can be continuously utilized in the iterative process, which is significant for accelerating the speed of convergence.

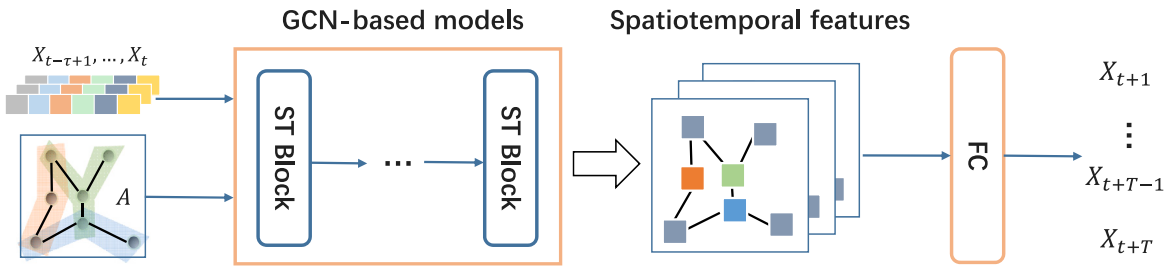


Fig. 3. The general structure of GCN-based spatiotemporal neural networks for traffic prediction. ST Block is a spatiotemporal module, which aims to extract the high-dimensional features of the traffic time series. And FC denotes one or more fully connected layers.

4.4. Prediction network module

Generally, the graph convolution based spatiotemporal neural networks for traffic prediction takes the historical time series and one or more adjacency matrices as the input, which aims to extract the spatial and temporal features hidden in the historical data. As is shown in Fig. 3, the networks are usually stacked by multiple spatiotemporal blocks. For each of the blocks, the high-dimensional spatial features are exploited by a GCN-based approach.

For all of the tested models in this paper, the spectral graph convolution based on Chebyshev polynomials approximation (ChebConv) up to the Kth order is used to capture the spatial dependencies between nodes, as

$$H^{(l)} = ChebConv(\hat{A}, H^{(l-1)}; \Theta^{(l)}) = \sigma(\sum_{k=0}^{K} T_k(\hat{A}) H^{(l-1)} \Theta^{(l)}) \tag{17}$$

where σ is the activation function (i.e., ReLU), $T_k(x)$ denotes the Chebyshev polynomials that are recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. $\hat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}$ is the normalized adjacent matrix, where $\widetilde{A} = A + I$ and $\widetilde{D}$ is a diagonal matrix with $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. $H^{(l)}$ is the hidden features of the lth layer and $\Theta^{(l)}$ is the corresponding learnable parameters.

As the generated relationships are uni-directional, we use the following approach to process inflow and outflow information, as

$$H^{(l+1)} = Concat[ChebConv(\hat{A}, H^{(l)}; \Theta_P), ChebConv(\hat{A}^T, H^{(l)}; \Theta_N)] \tag{18}$$

where Θ_P, Θ_N are the learnable parameters, $\hat{A}^T$ is the transpose of $\hat{A}$ and $Concat$ is the concatenation of different data.

4.5. Loss functions for optimization

For the training of the PN module, the minimization of L1 loss between the predicted value and the ground truth is selected as our training objective for multi-step traffic prediction, as

$$L_P(Y, \hat{Y}) = |Y - \hat{Y}|_1 \tag{19}$$

where $Y = (X_{t+1}, \ldots, X_{t+T})$ is the ground truth and $\hat{Y}$ is the predicted value.

For the training of the PGL module, a novel loss function is proposed to further ensure the sparsity of the generated adjacency matrix, as

$$\Delta A = ReLU[\mathcal{I}(A_{new}) - \mathcal{I}(A_{old})] \tag{20}$$

$$L_G(Y, \hat{Y}) = L_P(Y, \hat{Y}) + ReLU(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \Delta A^{(i,j)}}{N^2} - \delta) \ / \ \delta \tag{21}$$

where $\delta \in (0, 1)$ is a hyper-parameter to control the sparsity rate of A_{new}. The smaller the value of δ, the stronger the restriction on the sparsity of the A_{new}. The proposed loss function has the following characteristics:

1. The number of new edges compared to A_{old} is restricted. For ΔA, the value of positions where $A_{new}^{(i,j)} > 0$ and $A_{old}^{(i,j)} = 0$ will be set to 1, while 0 for other positions. As a result, $\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \Delta A^{(i,j)}}{N^2}$ denotes the proportion of the new edges to the maximum edges of the graph, and L_G will rise linearly when it exceeds δ.
2. The weight change of the old relationships will not affect the right half part of L_G. The spatial dependencies between nodes can be adaptively amplified or attenuated according to the value of L_P. After several iterations, the stronger correlations will be highlighted, while the weaker ones will be gradually erased.

4.6. Update the current affinity matrix

As the subgraphs in the affinity matrix set $\mathbb{A}$ are not accurate and effective enough, we use the prediction error to weight the importance of different subgraphs, to obtain the best estimate of A^*. First, each subgraph in $\mathbb{A}$ and the validation set will be fed into the PN module to calculate the corresponding prediction loss, as

$$L_k = L_P[P(X|A_k, \Theta), Y] \tag{22}$$

where P is the forecasting function and Θ is the learnable parameters of the PN module. Let $\vec{l} = (L_1, L_2, \dots, L_{N_r})^T$ be a vector that is composed of all prediction losses, and $L_{max} = \max_{1 \le i \le N_r} L_i$ be the maximum value, the weight vector $w = (w_1, w_2, \dots, w_{N_r})^T$ can be formulated as

$$\vec{w} = softmax(L_{max} - \vec{l}) \tag{23}$$

where the function $softmax$ is defined as

$$softmax(\vec{x}) = \frac{e^{\vec{x}}}{\sum_{c=1}^{C} e^{x_i}} \tag{24}$$

The current affinity matrix is updated by the weighted sum of all subgraphs, as

$$A = \sum_{i=1}^{N_r} w_i A_i \tag{25}$$

and A will be normalized analogous to Eqs. (15) $\sim$ (16) eventually.

4.7. The training algorithm

For the training of the PN module and the PGL module, Kingma and Ba (2015) is leveraged to optimize the parameters. To control the amount of subgraphs in $\mathbb{A}$, the subgraph with the maximum prediction error will be removed from $\mathbb{A}$ when certain conditions are met. The specific process of AdapGL is demonstrated as below.

Algorithm 1 AdapGL

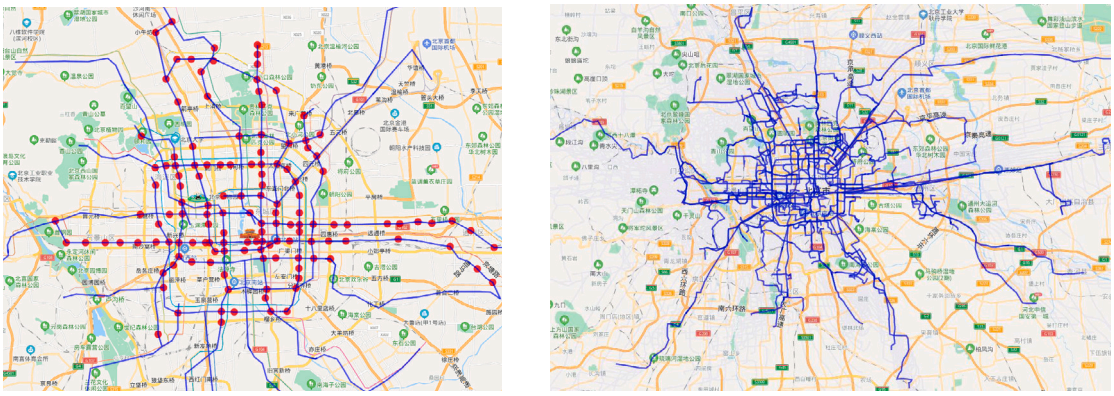
Input The initial affinity matrix set $\mathbb{A} = \{A_k | k = 1, 2, ..., N_r\}$, a GCN-based spatiotemporal neural networks for traffic prediction P_Θ, the parameters for the PGL module $M_1, M_2 \in R^{N \times F_0}, \Lambda \in R^N$, the capacity of $\mathbb{A}$ N_{max}, hyper-parameters $\epsilon, \delta \in (0, 1)$, input data X, target data Y, number of epoch N_{epoch}.

Output Well-trained parameters of the PN module Θ, the maximum likelihood estimation of A^*.

$A_{old} = \bigcup_{i=1}^{N_r} A_i$
repeat
  // Taking the current affinity matrix as the estimation of A^*, train the PN module
  Acquire the prediction result $\hat{Y} = P(A_{old}, X; \Theta)$
  Compute $L_P(\hat{Y}, Y)$ according to Section 4.5
  for $i = 1 : N_{epoch}$ **do**
    Train the parameters of the PN module Θ through Adam
  end for
  // Taking the current parameters of the PN module as the estimation of Θ^*, train the PGL module.
  Generate a new affinity matrix $A_{new} = G(A_{old}; M_1, M_2, \Lambda)$ according to Section 4.3
  Obtain the prediction result $\hat{Y} = P(A_{new}, X; \Theta)$
  Compute $L_G(\hat{Y}, Y)$ according to Section 4.5
  for $i = 1 : N_{epoch}$ **do**
    Fix Θ, optimize the parameters of the PGL module through Adam
  end for
  // Update the current affinity matrix.
  Push A_{new} into $\mathbb{A}$, and compute the prediction loss of all subgraphs in $\mathbb{A}$ according to Eq. (22)
  if $N_r > N_{max}$ **then**
    Remove the subgraph with the maximum prediction error
  end if
  Compute A according to Section 4.6
  $A_{old} = A$
until The stop criterion is met

Table 1
Dataset description used for experiments.

Datasets	N	τ	T	F	#TimeSteps	Used graphs
PeMSD4	307	12	12	3	16 992	A_D, A_C
PeMSD8	170	12	12	3	12 345	A_D, A_C
MetroBJ	140	12	6	1	4053	A_{OD}, A_C
BusBJ	139	12	6	1	4053	A_{Tr}, A_C



(a) (b)

Fig. 4. The geographical distribution of the research objects. (a) Metro stations of MetroBJ dataset. The red points are the studied metro stations, and the blue lines are different metro lines. (b) Bus lines of BusBJ dataset. The blue lines are different bus lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5. Experiments

5.1. Dataset description

we evaluate the performance of AdapGL on four real-world datasets: PeMSD4, PeMSD8, MetroBJ and BusBJ. The detailed information of the datasets are shown in Table 1, which can be illustrated as follows:

1. **PeMSD4**. It denotes the traffic data in San Francisco Bay Area collected by the Caltrans Performance Measurements Systems (PeMS). The measurements of one sensor are aggregated into 5-minutes windows, including traffic flow, average speed and average occupancy. And the dataset consists of records of 307 sensors within the period from 1/Jan/2018 to 28/Feb/2018. In this paper, the distance relationship A_D and the correlation coefficient relationship A_C are used as the initial graphs.
2. **PeMSD8**. This dataset consists of traffic information on the San Bernardino area from 1/Jul/2016 to 31/Aug/2016, including 170 loop detectors. Analogous to PeMSD4, there are three kinds of measurements, and A_D, A_C are utilized as the initial graphs.
3. **MetroBJ**. It contains the metro passenger flow data for part of the time in Beijing in 2019. The research object is the passenger flow of multiple metro stations, whose geographical distributions are shown in Fig. 4(a). For each metro station, the passenger flow of one time step is counted by the records per 10 min, which means there are 144 points for one day. According to the operating hours of the subway, the data before 5:00 am and after 23:00 pm are removed. In this paper, the OD relationship A_{OD} and the correlation coefficient relationship A_C are used for the initialization of the current graph.
4. **BusBJ**. It contains the bus passenger flow data of Beijing Public Transport, the time span of which is the same as MetroBJ. The research object is the passenger flow of multiple bus lines, which are counted by the boarding records of one time step corresponding to different bus lines. The geographical distributions of the lines are shown in Fig. 4(b), and A_T, A_C are used for the initialization.

For PeMSD4 and PeMSD8, the targets include traffic flow prediction and traffic speed forecast respectively. To facilitate comparison with the published experimental results, we split them into training set, validation set and test set with the ratio 6:2:2 according to the chronological order for traffic flow prediction. For traffic speed prediction, the setting is the same as Huang et al. (2020): The first 47 days are used as training set and the rest as validation and test set for PeMSD4, while the first 50 days are used as training set and the remaining as validation and test set for PeMSD8. The target is to predict the future ingoing passenger flow for MetroBJ and BusBJ, which are divided with the ratio 6:1:3.

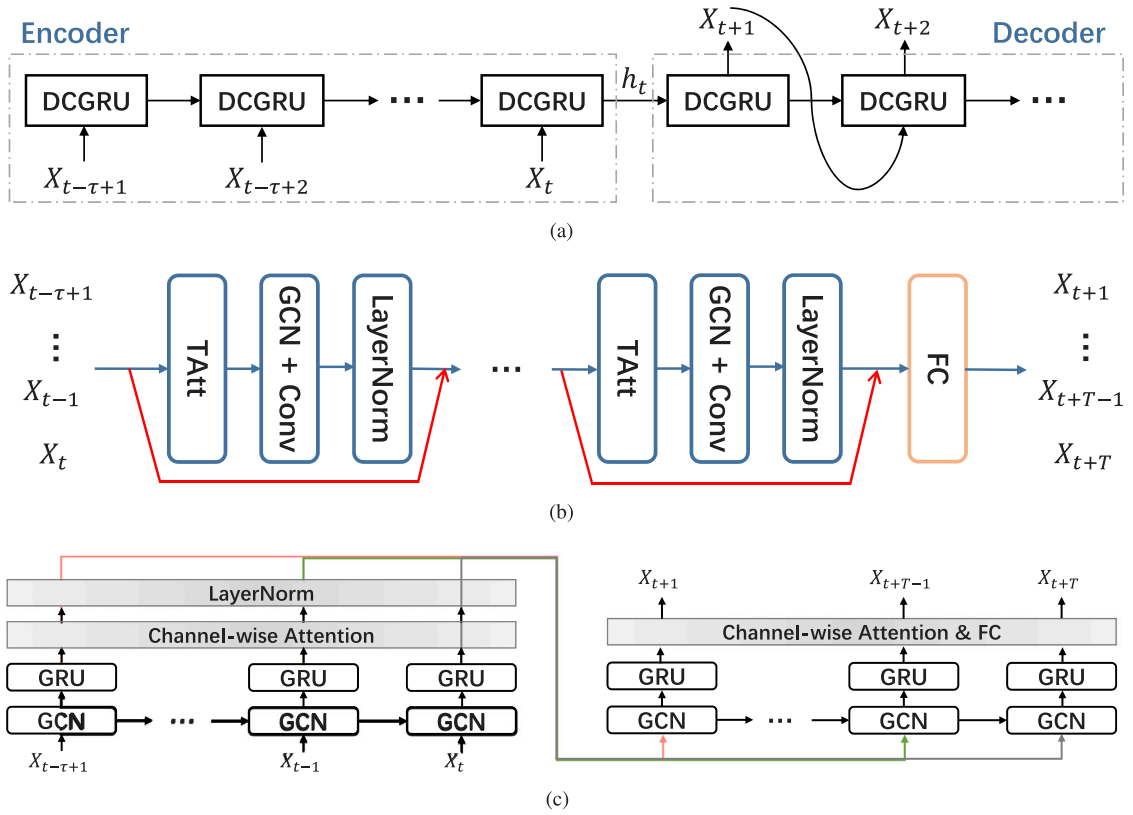


Fig. 5. The structures of the tested models. (a) The structure of DCRNN. DCGRU is a module that integrates graph convolution into GRU. (b) The structure of ASTGCN. Tatt is the temporal attention module, GCN+Conv is a block that extracts high-dimensional features from data through graph convolution and standard convolution. (c) The structure of TGCN. Channel-wise Attention is a module that weights the importance of different features extracted from GRU.

5.2. Evaluation metrics

In this paper, we use the mean absolute error (MAE), root mean square error (RMSE) and mean percentage error (MAPE) to evaluate the performance of different methods. The metrics are formulated as follows:

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(y_i - \hat{y}_i)^2} \tag{26}$$

$$MAE = \frac{1}{M}\sum_{i=1}^{M}|y_i - \hat{y}_i| \tag{27}$$

$$MAPE = \frac{1}{M}\sum_{i=1}^{M}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{28}$$

where y_i is the ground truth, $\hat{y}_i$ is the value of prediction and M is the number of tested samples. MAE is used to measure the overall error of the forecasting result, which is not sensitive to the outliers. RMSE is sensitive to large or small values and can be used to analyze the stability of the forecasting results. MAPE can reflect the degree of deviation, but it is susceptible to the small values of the ground truth.

5.3. Tested frameworks and baseline methods

As shown in Fig. 5, three models are leveraged to evaluate the performance of our method. The models represent different types of GCN-based frameworks for multi-step traffic prediction, as follows:

1. **DCRNN** (Li et al., 2018): Diffusion Convolution Recurrent Neural Network, which replaces the matrix multiplications in recurrent models (i.e., GRU and LSTM) with graph convolution, and simultaneously extracts the spatial and temporal features in an encoder–decoder manner. Some other works like AGCRN (Bai et al., 2020) also follow this framework.

Table 2
Performance comparison of different methods on PeMSD4 and PeMSD8 for traffic flow prediction. +AdapGL indicates that the model is trained by AdapGL, while others are trained directly.

Model	Dataset	PeMSD4			PeMSD8		
	Metrics	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
HA		38.52	56.77	28.20	32.00	47.44	20.28
SVR		24.54	37.53	19.37	19.89	30.47	16.78
FC-LSTM		27.14	41.59	18.20	22.20	34.06	14.20
STSGCN		21.19	33.65	**13.90**	17.13	26.80	10.96
DCRNN		23.67	36.58	15.61	18.22	28.29	11.56
DCRNN + AdapGL		**20.67**	**32.23**	14.22	**16.25**	**25.32**	10.91
Improvements		+12.67%	+11.89%	+8.90%	+10.81%	+10.50%	+5.62%
ASTGCN		23.05	35.98	15.14	18.39	28.75	12.08
ASTGCN + AdapGL		20.90	32.64	14.40	16.45	25.70	**10.68**
Improvements		+9.33%	+9.28%	+4.89%	+10.55%	+10.61%	+11.59%
TGCN		23.76	36.84	16.33	18.05	28.19	11.47
TGCN + AdapGL		20.68	32.38	14.29	16.67	25.77	11.07
Improvements		+12.96%	+12.11%	+12.49%	+7.65%	+8.58%	+3.61%

2. **ASTGCN** (Guo et al., 2019): Attention Based Spatial–Temporal Graph Convolutional Networks, which combines graph convolution with spatial attention to capture the spatial patterns, and leverages standard convolution and temporal attention to extract the temporal features. STGCN (Yu et al., 2017), MTGNN (Wu et al., 2020) and some other models can be classified as this type of framework. In this paper, the spatial attention is removed as the pre-defined graph is not accurate, and only the recent components is taken to ensure the fair comparison.
3. **TGCN** (Zhao et al., 2020): Temporal Graph Convolution network model, which separately utilizes the recurrent models to extract the temporal features and graph convolution to capture the spatial dependencies. Multi-GCN (Geng et al., 2019), GLA (Li et al., 2019b) and some other networks also belong to this framework. According to our previous work, a channel-wise attention module is integrated into the network in this paper.

The following methods are selected as the baselines to ensure the reasonableness of the tested models:

1. **HA**: Historical Average. We use the average values of the historical traffic flow data to achieve the task.
2. **SVR** (Drucker et al., 1996): Support Vector Regression. It is an extension of Support Vector Machine (SVM), which is trained by minimizing the distance from the sampled point to the hyperplane using kernel functions.
3. **FC-LSTM** (Sutskever et al., 2014): Long Short-Term Memory Network with fully connected LSTM hidden units.
4. **STGCN** (Yu et al., 2017): Spatial–Temporal Graph Convolutional Network. Is utilizes the gated temporal convolution and graph convolution to extract the spatiotemporal features.
5. **STSGCN** (Song et al., 2020): Spatial–Temporal Synchronous Graph Convolutional Network. It uses a localized spatial–temporal graph to synchronously capture the complex correlations between nodes.
6. **LSGCN** (Huang et al., 2020): Long Short-term Graph Convolutional Networks. It presents a new graph attention network called CosAtt.

5.4. Experimental settings

All of the experiments are executed under a platform with one NVIDIA TESLA V100 GPU 32 GB card. For all of the deep-learning based models, the training processes are implemented in Python with Pytorch 1.5.0, Adam with learning rate 0.001 is used as the optimization approach and an early stop strategy is utilized to decide whether the stop criterion is met. Furthermore, the optimal parameters are dictated by the performance on the validation dataset.

For the parameter settings of AdapGL, the dimension of M_1, M_2 are set as 64 for all of the datasets. The maximum capacity of $\mathbb{A}$ is set as 3, δ is 0.02, and ϵ is set as $\frac{1}{2N}$, where N is the number of nodes. For each iteration, the training times N_{epoch} can be set to a value between 5 and 10 according to the convergence speed of the PGL module.

5.5. Experimental results

Tables 3 ∼ 4 show the overall performances of different baselines and tested models. The results indicate that our proposed method can significantly reduce the prediction error of the tested models. For PeMS datasets, the average prediction error of each model is reduced from 7% to 12% in terms of all evaluation metrics, while the improvement on the other datasets is slightly smaller limited by the size of data. And it can be observed that the GCN-based frameworks outperform the other baselines, which demonstrates the significance of considering the spatial dependencies between nodes. Fig. 6 shows the performance of different models at each horizon on PeMSD4 dataset for traffic flow prediction. The accuracy of the affinity matrix impacts greatly on the performance of the tested frameworks for both short-term and long-term prediction, and the improvement is more apparent with the increase of time horizon.

Table 3
Performance comparison of different methods on the MetroBJ dataset and BusBJ dataset. +AdapGL indicates that the model is trained by AdapGL, while others are trained directly.

Model	Dataset	MetroBJ			BusBJ		
	Metrics	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
HA		81.52	143.52	57.57	53.17	73.56	59.01
SVR		29.27	51.17	24.06	24.90	33.32	21.97
FC-LSTM		24.73	41.20	19.42	22.60	31.58	**17.98**
DCRNN		24.57	41.72	18.89	22.63	30.93	19.03
DCRNN + AdapGL		**22.68**	**38.93**	**16.96**	**21.96**	**29.89**	18.08
Improvements		+7.69%	+6.69%	+10.22%	+2.96%	+3.36%	+4.99%
ASTGCN		25.90	43.44	20.72	23.77	32.31	20.71
ASTGCN + AdapGL		24.21	41.51	19.01	23.30	31.43	19.74
Improvements		+6.56%	+4.44%	+10.71%	+1.98%	+2.72%	+4.68%
TGCN		26.52	44.86	21.63	23.63	32.56	20.00
TGCN + AdapGL		23.53	39.49	18.49	22.63	30.94	19.24
Improvements		+11.27%	+11.97%	+14.52%	+4.23%	+4.98%	+3.80%

Table 4
Performance comparison of different methods on PeMSD4 and PeMSD8 for traffic speed prediction. +AdapGL indicates that the model is trained by AdapGL, while others are trained directly.

Model	Dataset	PeMSD4 (15/30/45/60 min)		
	Metrics	MAE	RMSE	MAPE (%)
HA		2.54	4.96	5.56
STGCN		1.47/ 1.93 /2.26 /2.55	3.01/ 4.21/ 5.01/ 5.65	2.92/ 3.98/ 4.73/ 5.39
ASTGCN		2.12/ 2.42/ 2.60/ 2.73	3.96/ 4.59/ 4.97/ 5.21	4.16/ 4.80/ 5.20/ 5.46
TGCN		1.39/ 1.79/ 2.09/ 2.34	2.96/ 4.07/ 4.80/ 5.34	2.72/ 3.78/ 4.63/ 5.35
DCRNN		1.35/ 1.77/ 2.04/ 2.26	2.94/ 4.06/ 4.77/ 5.28	2.68/ 3.71/ 4.48/ 5.10
LSGCN		1.45/ 1.82/ 2.04/ 2.22	2.93/ 3.92/ 4.47/ 4.83	2.90/ 3.84/ 4.42/ 4.85
ASTGCN + AdapGL		1.36/ 1.69/ 1.88/ 2.05	2.88/ 3.78/ 4.27/ 4.62	2.70/ 3.60/ 4.16/ 4.59
TGCN + AdapGL		**1.35**/ 1.68/ 1.89/ 2.06	**2.84**/ 3.72/ 4.24/ 4.63	**2.66**/ 3.54/ 4.11/ 4.55
DCRNN + AdapGL		1.36/ **1.68**/ **1.86**/ **1.99**	2.85/ **3.72**/ **4.19**/ **4.51**	2.67/ **3.51**/ **4.02**/ **4.37**
Model	Dataset	PeMSD8 (15/30/45/60 min)		
	Metrics	MAE	RMSE	MAPE (%)
HA		1.98	4.11	3.94
STGCN		1.19/ 1.59 /1.92 /2.25	2.62/ 3.61/ 4.21/ 4.68	2.34/ 3.24/ 3.91/ 4.54
ASTGCN		1.49/ 1.67/ 1.81/ 1.89	3.18/ 3.69/ 3.92/ 4.13	3.16/ 3.59/ 3.98/ 4.22
TGCN		1.15/ 1.47/ 1.69/ 1.88	2.52/ 3.40/ 3.95/ 4.36	2.20/ 2.97/ 3.55/ 4.05
DCRNN		1.17/ 1.49/ 1.71/ 1.87	2.59/ 3.56/ 4.13/ 4.50	2.32/ 3.21/ 3.83/ 4.28
LSGCN		1.16/ 1.46/ 1.66/ 1.81	**2.45**/ 3.28/ 3.75/ 4.11	2.24/ 3.02/ 3.51/ 3.89
ASTGCN + AdapGL		**1.12**/ **1.38**/ 1.54/ 1.66	2.49/ 3.30/ 3.78/ 4.10	2.16/ 2.87/ 3.35/ 3.70
TGCN + AdapGL		1.12/ 1.39/ 1.57/ 1.72	2.46/ **3.25**/ **3.75**/ 4.11	**2.13**/ **2.84**/ 3.35/ 3.73
DCRNN + AdapGL		1.14/ 1.39/ **1.54**/ **1.65**	2.48/ 3.30/ 3.77/ **4.09**	2.20/ 2.90/ **3.35**/ **3.68**

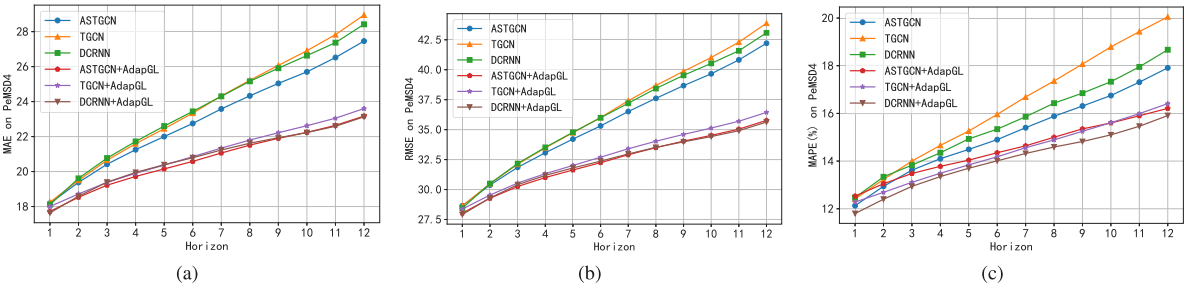


Fig. 6. Performance comparison of the tested models at each horizon on PeMSD4 dataset for traffic flow prediction, where one horizon denotes 5 min. (a) Comparison of MAE. (b) Comparison of RMSE. (c) Comparison of MAPE.

Both the alternate training and the adaptively learned graph can improve the performance of the tested models, as shown in Fig. 7. Ori means that the model is trained directly with the pre-defined graph. -A denotes that the parameters are optimized directly with the affinity matrix adaptively obtained by ASTGCN+AdapGL, which means that the pre-defined graph is replaced by A^* obtained by ASTGCN+AdapGL while the other settings are the same as Ori. Analogous to -A, -D is DCRNN+AdapGL, and -T is TGCN+AdapGL.

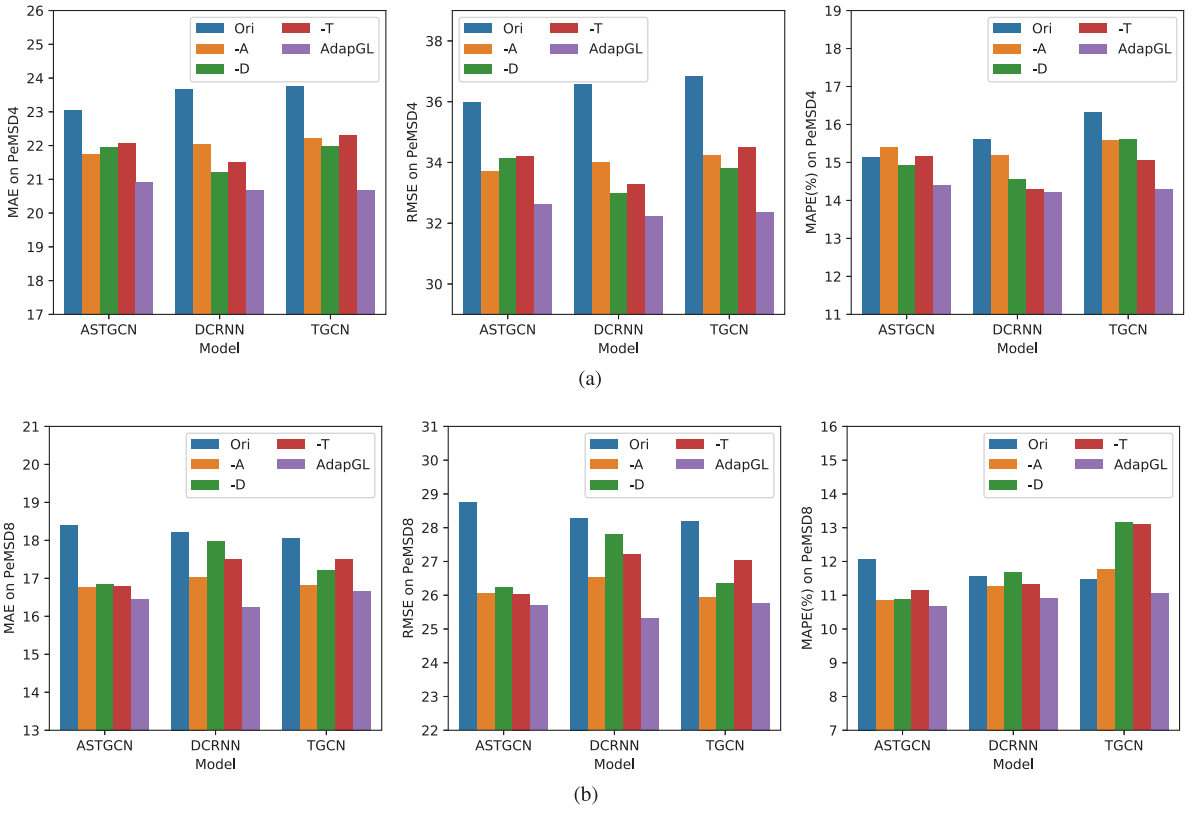


Fig. 7. Performance comparison of the tested models under different training conditions. -A indicates that the model is trained directly with the graph generated by ASTGCN+AdapGL, -D is DCRNN+AdapGL, -T is TGCN+AdapGL, and Ori is the original graph. AdapGL denotes that the model is trained by AdapGL. (a) Performance comparison on PeMSD4 for traffic flow prediction. (b) Performance comparison on PeMSD8 for traffic flow prediction.

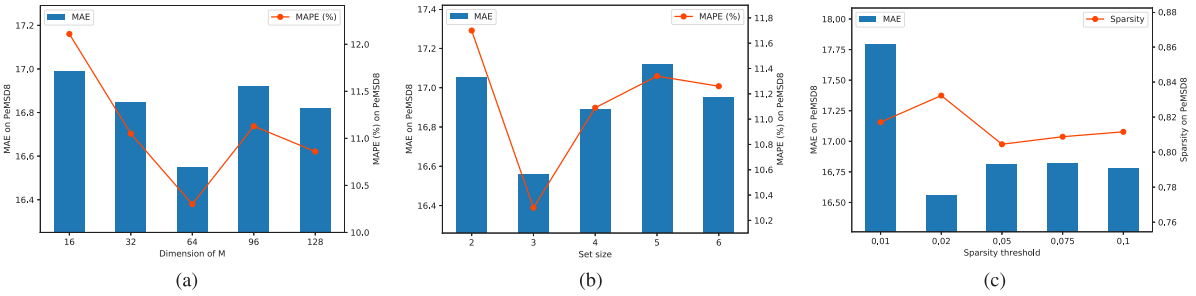
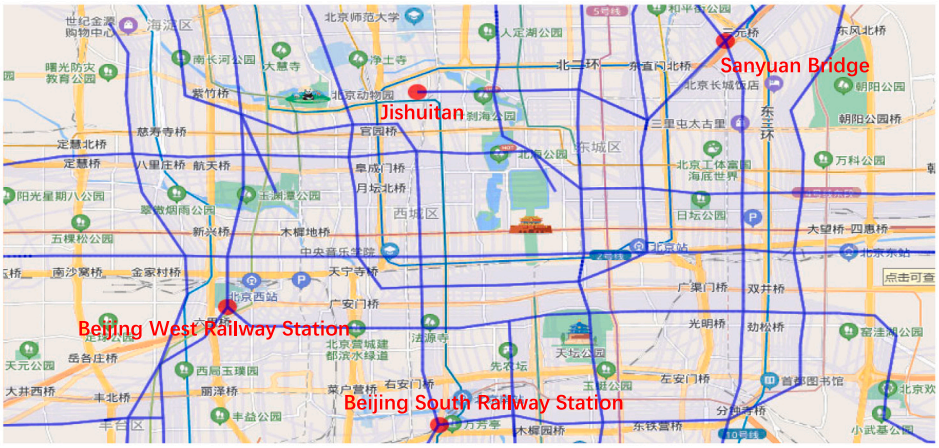


Fig. 8. Influence of different parameter settings. The results are obtained by ASTGCN+AdapGL on PeMSD8 for traffic flow prediction. (a) Influence of the dimension of M_1, M_2. (b) Influence of the capacity of $\mathbb{A}$. (c) Influence of the sparsity threshold δ.

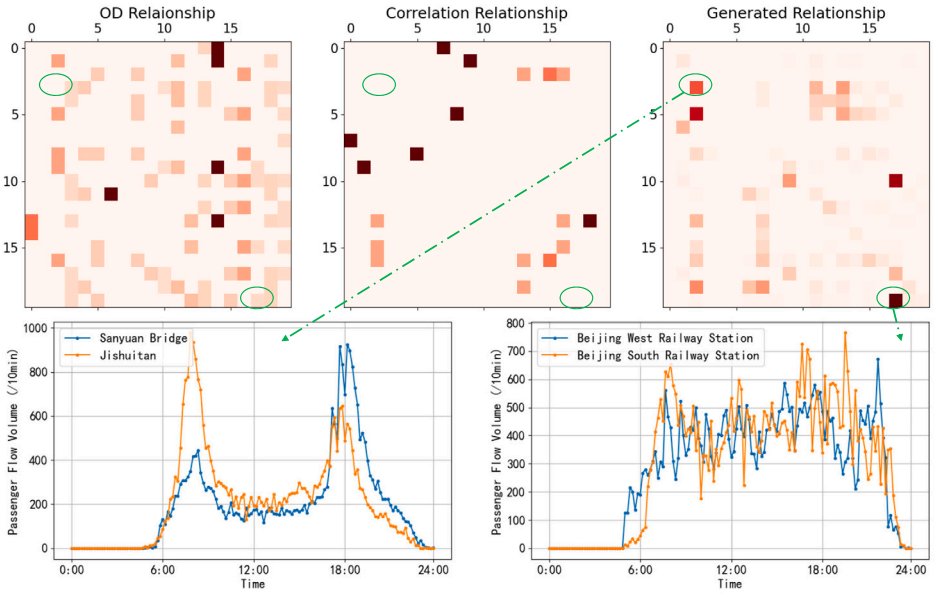
For each of the tested models, the performance for traffic prediction can also be greatly improved with the adaptively learned graph generated by our proposed method, while the other conditions are the same. For the PeMSD4 dataset, the best result obtained by an end-to-end training method is slightly worse than that obtained by AdapGL. It demonstrates that the proposed approach can not only acquire more accurate correlations between nodes, but effectively control the optimization of the GCN-based frameworks.

5.6. Ablation study

Influence of Parameter Settings. The dimension of M_1, M_2, the capacity of $\mathbb{A}$ and the hyper-parameter δ are the key parameters of AdapGL, whose values may significantly influence the performance of the tested models. The expansion of dimension will increase the complexity of the PGL module and enhance the ability of fitting the effective spatial dependencies, while it can also increase the difficulty of optimization. The capacity of $\mathbb{A}$ can affect the stability of the training process, as the affinity matrix will vary a lot if it is too small. And the value of δ decides the sparsity of the generated graph. Fig. 8 presents the performance comparison of



(a)



(b)

Fig. 9. Comparison of the graphs on MetroBJ dataset. OD relationship is A_{OD}, Correlation relationship is A_C computed by Spearman correlation coefficient, and Generated relationship is the affinity matrix generated by TGCN+AdapGL. (a) The geographical locations of the related metro stations. (b) Comparison of different graphs. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ASTGCN+AdapGL with different parameter settings on PeMSD8 for traffic flow prediction. Apparently, the model can achieve the best performance when the dimension of M_1, M_2 is 64, N_{max} is 3 and ϵ is 0.02. Furthermore, the sparsity of the generated graph is maintained at around 0.81 with the increase of δ, which sufficiently demonstrates the stability of our proposed method.

Effectiveness of the Generated Graph. Fig. 9 compares the initial affinity matrices and the generated graph after the training of TGCN+AdapGL on the MetroBJ dataset. Due to the number of nodes, only the last 20 metro stations are presented in this paper. For Fig. 9(a), the blue lines are part of the lines of Beijing subway system, and the red points are the related metro stations of Fig. 9(b). For Fig. 9(b), the darker the points' color, the closer their relationship is. We can observe that the generated graph is similar with the initial affinity matrices to a certain degree, and the sparsity is preserved. Furthermore, the generated graph has the following properties:

1. New correlations are generated, while some of the old correlations are removed. Under the new spatial dependencies, the lower left corner of Fig. 9(b) presents the distribution of the passenger flow of two correlated metro stations in one day. For Sanyuan Bridge station, the passenger flow volume during the morning peak is much higher than that during the evening peak, while the opposite is true for Jishuitan station. However, their intraday trends of the passenger flow are consistent with each other, which is difficult to obtain via other data analysis methods.

Table 5
Performance comparison of different methods on PeMSD4 dataset and PeMSD8 for traffic flow prediction. +ALT indicates that the model is trained by alternate training, and +E2E denotes that the model is trained by end-to-end training.

Model	Dataset	PeMSD4			PeMSD8		
	Metrics	MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
DCRNN + E2E		21.32	33.21	14.88	17.24	30.04	11.89
DCRNN + ALT		**20.67**	**32.23**	**14.22**	**16.25**	**25.32**	10.91
ASTGCN + E2E		21.00	32.69	14.55	17.24	26.53	11.55
ASTGCN + ALT		20.90	32.64	14.40	16.45	25.70	**10.68**
TGCN + E2E		20.67	32.30	14.38	17.90	27.53	11.93
TGCN + ALT		20.68	32.38	14.29	16.67	25.77	11.07

2. Some relations are amplified or attenuated. As shown in the right corner of Fig. 9(b), the correlation between Beijing West Railway Station and Beijing South Railway Station is relatively weak in the OD affinity matrix, but the weight is significantly strengthened in the generated spatial dependencies. Both are transfer stations and located at the positions that are closed to a train station. This demonstrates that the function similarity between nodes is an important factor to consider the correlations.

6. Conclusion

In this paper, we propose an adaptive graph learning algorithm for traffic prediction based on spatiotemporal neural networks, which is called AdapGL. Confronted with the uncertainty of the graph structure in real traffic systems, the algorithm can adaptively acquire the hidden spatial dependencies between nodes through a novel parameterized graph learning module. Inspired by the EM algorithm, an alternate training approach is leveraged to optimize the parameters of the PN module and the PGL module. And a well-defined graph loss function is used to ensure the sparsity of the generated graph. Extensive experiments on multi-step traffic prediction tasks indicate that the proposed approach can not only improve the performance of different GCN-based frameworks, but effectively exploit the potential correlations hidden in the complex geometric structures of the real-world traffic systems.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (NSFC) under Grant U1811463, Grant U1909204, Grant 62076237 and Grant 61876011; in part by Key-Area Research and Development Program of Guangdong Province 2020B0909050001; in part by CAS STS Dongguan Project under Grant 20201600200132.

Appendix A. Alternate training vs. end-to-end training

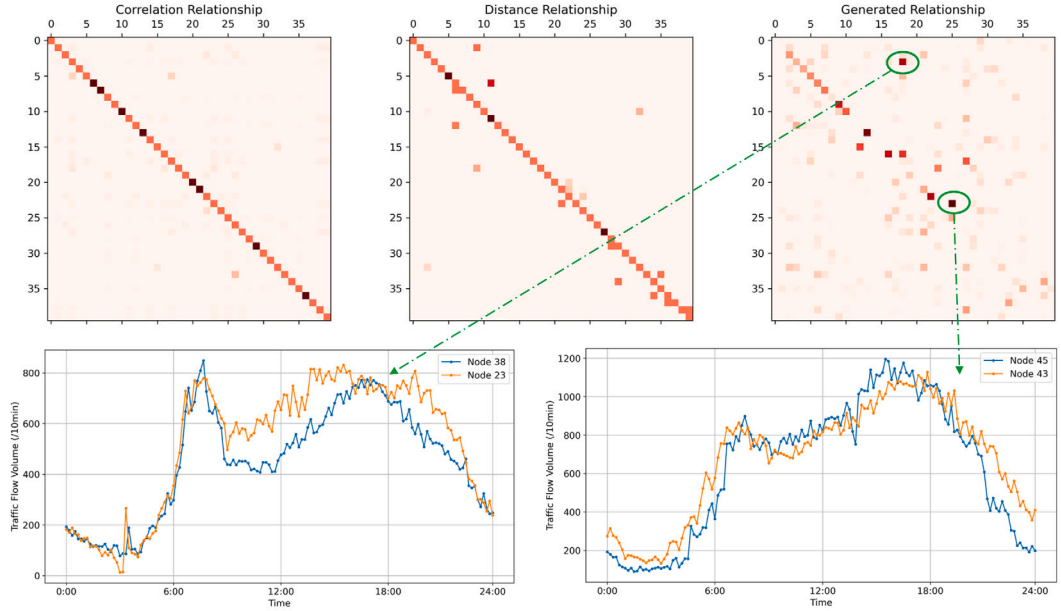
In these experiments, we compare different models' performance in the case of alternate training and end-to-end training. For end-to-end training, the PGL module and PN module are trained simultaneously and Eq. (20) is leveraged as the loss function, while the other conditions are the same as alternate training. We use early stop strategy to decide whether the stop criterion is met. Results are summarized in Table 5. For all of the tested models, we can observe that the performance deteriorates when they are trained with an end-to-end training approach. As it needs to consider the sparsity of the generated affinity matrix and the performance of the PN module, the optimization of parameters can become erratic and difficult, which will affect the effectiveness of the generated graph.

Appendix B. Affinity matrices comparison on other datasets

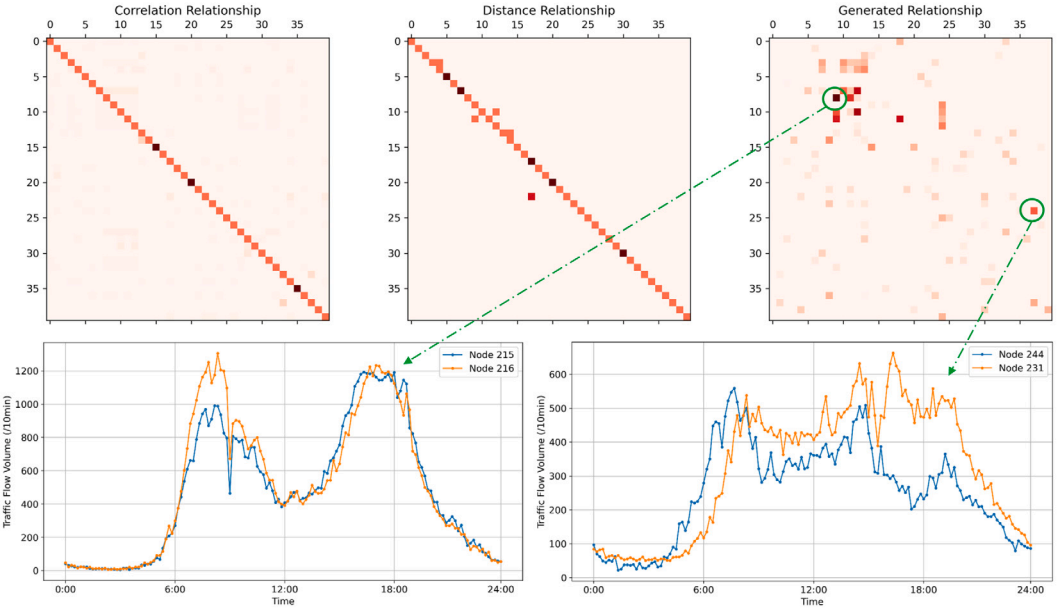
To further show the effectiveness of the generated graph, comparison of different graphs on PeMSD4 and PeMSD8 datasets are investigated here, as shown in Fig. 10. Only 40 nodes are presented for both datasets, and the color depth of a point is proportional to the intensity of the spatial dependencies.

For PeMSD8 dataset, two new points and the corresponding traffic flow distributions are analyzed. As the adjacency relation between node 38 and node 23 is strong enough in the generated affinity matrix, the lower left corner of Fig. 10(a) clearly shows that their traffic flow dynamics are highly consistent during the morning peak. Similarly, the traffic flows of node 45 and node 43 are close in the whole day. However, the significant information is ignored in the other affinity matrices.

For PeMSD4 dataset, we analyze one of the darker and lighter points. Among nodes that are correlated with node 215, node 216 is the closest. The lower left corner of Fig. 10(b) demonstrates that their intraday traffic flows are almost the same except for the morning peak, while the short-term trends are surprisingly consistent in this period. And for node 244 and node 231, the correlation seems to be not apparent enough, which deserves future research.



(a)



(b)

Fig. 10. Comparison of the graphs on PeMSD4 and PeMSD8 datasets. Correlation relationship is A_C computed by Spearman correlation coefficient, Distance relationship is A_D. (a) Comparison of the graphs on PeMSD8 dataset. Generated relationship is the affinity matrix generated by ASTGCN+AdapGL. Only the spatial dependencies among nodes 20 ∼ 60 are presented. (b) Comparison of the graphs on PeMSD4 dataset. Generated relationship is the affinity matrix generated by DCRNN+AdapGL. Only nodes 207 ∼ 247 are shown. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

Ahmed, M.S., Cook, A.R., 1980. Analysis of freeway traffic time-series data by using box-jenkins techniques. Transp. Res. Rec. 9 (2), 125–143.

Bai, L., Yao, L., Li, C., Wang, X., Wang, C., 2020. Adaptive graph convolutional recurrent network for traffic forecasting. In: International Conference on Learning Representations.

Chen, Y., Lv, Y., Wang, X., Wang, F.-Y., 2018. Traffic flow prediction with parallel data. In: International Conference on Intelligent Transportation Systems. pp. 614–619.

Cui, Y., Jin, B., Zhang, F., Sun, X., 2019. A deep spatio-temporal attention-based neural network for passenger flow prediction. In: EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services. pp. 20–30.
Dai, X., Fu, R., Zhao, E., Zhang, Z., Lin, Y., Wang, F.-Y., Li, L., 2019. DeepTrend 2.0: A light-weighted multi-scale traffic prediction model using detrending. Transp. Res. C 103, 142–157.
Davis, G.A., Nihan, N.L., 1991. Nonparametric regression and short-term freeway traffic forecasting. J. Transp. Eng. 117 (2), 178–188.
Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V., 1996. Support vector regression machines. In: International Conference on Neural Information Processing Systems. pp. 155–161.
Duan, Y., Lv, Y., Wang, F.-Y., 2016. Travel time prediction with LSTM neural network. In: International Conference on Intelligent Transportation Systems. pp. 1053–1058.
Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., Liu, Y., 2019. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In: AAAI Conference on Artificial Intelligence. pp. 3656–3663.
Guo, K., Hu, Y., Qian, Z., Sun, Y., Gao, J., Yin, B., 2020. Dynamic graph convolution network for traffic forecasting based on latent network of Laplace matrix estimation. IEEE Trans. Intell. Transp. Syst. 1–10.
Guo, S., Lin, Y., Feng, N., Song, C., Wan, H., 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: AAAI Conference on Artificial Intelligence. pp. 922–929.
Huang, R., Huang, C., Liu, Y., Dai, G., Kong, W., 2020. LSGCN: Long short-term traffic prediction with graph convolutional networks. In: International Joint Conference on Artificial Intelligence. pp. 2355–2361.
Jeong, Y.-S., Byon, Y.-J., Castro-Neto, M.M., Easa, S.M., 2013. Supervised weighting-online learning algorithm for short-term traffic flow prediction. IEEE Trans. Intell. Transp. Syst. 14 (4), 1700–1707.
Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization. In: International Conference on Learning Representations.
Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations.
Leshem, G., Ritov, Y., 2007. Traffic flow prediction using adaboost algorithm with random forests as a weak learner. World Acad. Sci. Eng. Technol. Int. J. Math. Comput. Phys. Electr. Comput. Eng. 1 (1), 1–6.
Li, Z., Jiang, S., Li, L., Li, Y., 2019a. Building sparse models for traffic flow prediction: an empirical comparison between statistical heuristics and geometric heuristics for Bayesian network approaches. Transp. B 7 (1), 107–123.
Li, Z., Xiong, G., Chen, Y., Lv, Y., Hu, B., Zhu, F., Wang, F.-Y., 2019b. A hybrid deep learning approach with GCN and LSTM for traffic flow prediction. In: International Conference on Intelligent Transportation Systems. pp. 1929–1933.
Li, Y., Yu, R., Shahabi, C., Liu, Y., 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representations.
Li, M., Zhanxing, Z., 2021. Spatial-temporal fusion graph neural networks for traffic flow forecasting. In: AAAI Conference on Artificial Intelligence. pp. 4189–4196.
Liu, Y., Liu, Z., Jia, R., 2019b. DeepPF: A deep learning based architecture for metro passenger flow prediction. Transp. Res. C 101, 18–34.
Liu, L., Qiu, Z., Li, G., Wang, Q., Ouyang, W., Lin, L., 2019a. Contextualized spatial–temporal network for taxi origin-destination demand prediction. IEEE Trans. Intell. Transp. Syst. 20 (10), 3875–3887.
Liu, Y., Zheng, H., Feng, X., Chen, Z., 2017. Short-term traffic flow prediction with Conv-LSTM. In: International Conference on Wireless Communications and Signal Processing. pp. 1–6.
Lv, Y., Chen, Y., Li, L., Wang, F., 2018. Generative adversarial networks for parallel transportation systems. IEEE Intell. Transp. Syst. Mag. 10 (3), 4–10.
Lv, Y., Duan, Y., Kang, W., Li, Z., Wang, F., 2015. Traffic flow prediction with big data: A deep learning approach. IEEE Trans. Intell. Transp. Syst. 16 (2), 865–873.
Shen, T., Hao, K., Gou, C., Wang, F.-Y., 2021. Mass image synthesis in mammogram with contextual information based on GANs. Comput. Methods Programs Biomed. 202, 106019.
Shen, T., Wang, J., Gou, C., Wang, F.-Y., 2020. Hierarchical fused model with deep learning and type-2 fuzzy learning for breast cancer diagnosis. IEEE Trans. Fuzzy Syst. 28 (12), 3204–3218.
Shuman, D.I., Narang, S.K., Frossard, P., Ortega, A., Vandergheynst, P., 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag. 30 (3), 83–98.
Song, C., Lin, Y., Guo, S., Wan, H., 2020. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In: AAAI Conference on Artificial Intelligence. pp. 914–921.
Sun, S., Zhang, C., Yu, G., 2006. A Bayesian network approach to traffic flow forecasting. IEEE Trans. Intell. Transp. Syst. 7 (1), 124–132.
Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks. In: International Conference on Neural Information Processing Systems. pp. 3104–3112.
Wang, F.-Y., 2014. Scanning the issue and beyond: Computational transportation and transportation 5.0. IEEE Trans. Intell. Transp. Syst. 15 (5), 1861–1868.
Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C., 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 753–763.
Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C., 2019. Graph WaveNet for deep spatial-temporal graph modeling. In: International Joint Conference on Artificial Intelligence. pp. 1907–1913.
Yu, B., Yin, H., Zhu, Z., 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: International Joint Conference on Artificial Intelligence. pp. 3634–3640.
Zhang, J., Zheng, Y., Qi, D., 2016. Deep spatio-temporal residual networks for citywide crowd flows prediction. In: AAAI Conference on Artificial Intelligence. pp. 1655–1661.
Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., Li, H., 2020. T-GCN: A temporal graph convolutional network for traffic prediction. IEEE Trans. Intell. Transp. Syst. 21 (9), 3848–3858.