DSTAGCN: Dynamic Spatial-Temporal Adjacent Graph Convolutional Network for Traffic Forecasting

Qi Zheng and Yaying Zhang

Abstract—Capturing complex and dynamic spatial-temporal dependencies of traffic data is of great importance for accurate and real-time traffic forecasting in intelligent transportation systems. The spatial-temporal dependency between traffic locations is often dynamic, which means the correlation between the traffic status of different locations changes jointly over their spatial distance and the time slice they are in. Most of the existing Graph Convolutional Network-based methods usually capture spatial and temporal dependencies separately and then combine them in a parallel or serial mechanism to capture the spatial-temporal features. They always utilize the predefined static graph structure to capture both local correlations and global dependencies in the same time slice. These methods are incapable of directly learning dynamic spatial-temporal dependency across time slices. Meanwhile, it is challenging to learn the spatiotemporal correlation knowledge among traffic locations only by using neural networks. To address these issues, we propose a novel Dynamic Spatial-Temporal Adjacent Graph Convolutional Network (DSTAGCN), which connects the latest time slice with each past time slice to construct the spatial-temporal graph. DSTAGCN can directly learn the global spatial dependency across time and simultaneously capture the spatial-temporal dependencies through graph convolution. Since fuzzy theory make it possible to represent uncertain relationships, a simplified fuzzy neural network that integrates fuzzy systems and neural networks is designed to help generating the graph adjacency matrix representing the dynamic adjacency correlations. Experiments on public datasets show that our method outperforms baselines with fast convergence.

1 Introduction

Traffic forecasting plays a crucial role in the Intelligent Transportation System (ITS). Accurate real-time prediction of massive traffic data is conducive to alleviating traffic congestion, optimizing traffic configuration, and improving the effectiveness of traffic control. With the significance, traffic forecasting has attracted more and more attention in academia and industry. Traffic forecasting is challenging due to the interaction of spatial-temporal dependencies and the dynamic properties of the adjacency relationship:

(1) Strong *interactive* spatial and temporal dependencies. Here *interactive* means that the spatial correlation and the temporal correlation between traffic status in different locations will influence each other. As traffic information is spread around the road network over time dynamically, the mutual influence between traffic locations is simultaneously related to their spatial distance and the time slice they were in. The traffic state of one location in the upcoming moment depends not only on the recent states of itself and its local neighbors but also on the past states of locations far away from it. From a local space-time point of view, as shown in Fig. 1, the present traffic state (e.g. congestion) of one location in the road network will affect the adjacent locations' near future through spatial propagation. Based on this propagation mode, the traffic state of the target location (blue rectangle) at time $t+3$ can be potentially attributed to the traffic state of the distant locations at time t (red circles). Therefore, when it comes to a larger time scale and space scale, the past states of distant locations also have global impacts on the future states of the target location. Specifically, for a target location in the predicted time slice, spatially nearby neighbours and distant locations will have different impacts in the prediction. Its adjacent locations will have a greater short-term impact, while its distant locations might have a greater long-term impact [1], [2], as illustrated in Fig. 2. This demonstrates the interaction of spatial-temporal dependency, which inspires us that considering the relationship between the two dependencies simultaneously may help to understand the process of traffic state evolution.

(2) *Dynamic* and *uncertain* mutual influence of traffic locations. The spatial correlations between different traffic locations are time-varying. For example, the same area shows different traffic patterns during peak hours and off-peak hours; Secondly, due to the traffic propagation, the spatial influence on different time spans is different. For example, in Fig. 2, local spatial dependencies tend to occur in the most recent periods, and global spatial dependencies tend to be across time rather than at the same time slice. Moreover, due to the existence of various internal and external factors (such as weather, events, emergencies, and road constructions, etc), the degree of interaction among traffic locations becomes complex and thus hard to be clearly identified. Modeling of this uncertain correlation in different time slices could contribute to the efficient traffic prediction.

• *The authors are with the Key Laboratory of Embedded System and Service Computing, MoE, Tongji University, Shanghai 200092, China. E-mail: {zhengqi97, yaying.zhang}@tongji.edu.cn.*

Manuscript received 23 July 2021; revised 25 January 2022; accepted 28 February 2022. Date of publication 7 March 2022; date of current version 16 January 2023.
This work was supported by the National Key Research and Development Program of China under Grant 2018YFB2100800.
(Corresponding author: Yaying Zhang.)
Recommended for acceptance by L. Tang.
Digital Object Identifier no. 10.1109/TBDATA.2022.3156366

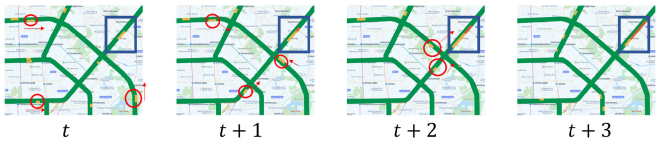


Fig. 1. An example of local traffic state propagation demonstrating the local spatial dependency in recent periods on Beijing [3]. The red circles indicate the traffic congestion on road locations and their moving direction, while the area inside the blue rectangle indicates the target location to be investigated.

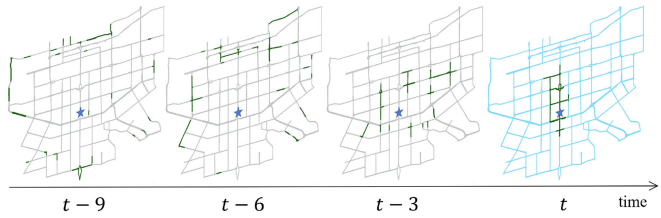


Fig. 2. An example of global spatial dependencies across distant time slices on Xi'an [2]. Traffic conditions on the road network can be aggregated at regular time slices and displayed as snapshots on the timeline. This figure illustrates road networks in different time slices, the blue one is in the predicted time slice t, and the blue star node is the predicted target. The dark green lines on the road network represent the sections with the greatest influence on the predicted target node in the past time slices.

To capture these complex spatial-temporal correlations of traffic data, more and more deep learning methods have been proposed. Most of these methods focus on the modeling of spatial and temporal characteristics respectively. Many studies use the Graph Convolutional Network (GCN) to learn the spatial relationship between traffic graph nodes in each time slice, and also use the Recurrent Neural Network (RNN) or the Convolutional Neural Network (CNN) to learn the time-varying dependency of each node in the past time slices. Then these two features were combined either in parallel or serial mechanisms to capture the spatial-temporal dependency. Despite remarkable results achieved by these methods, capturing the interactivity of spatial and temporal dependencies well is still challenging. Besides, in many existing GCN-based methods, the adjacency matrix representing the correlation among traffic locations is generally predefined in terms of the original topological structure of the road network which is static over time. The unprocessed raw adjacency matrix manifests only the local relationship but neglects the global dependency. And the static adjacency matrix fails to reflect the dynamic spatial dependencies. Therefore, these methods need to stack many convolution layers or increase the length of the input sequence when capturing the above-mentioned global spatial dependencies across distant time, which leads to an increasing cost in model training and the possibility of error accumulation, especially in RNNs.

To address these issues, we propose a Dynamic Spatial-Temporal Adjacent Graph Convolutional Network (DSTAGCN) for traffic forecasting. Inspired by STSGCN [4], which utilizes a localized spatial-temporal subgraph module to model localized correlations synchronously, we construct spatial-temporal graphs to simultaneously capture the spatial-temporal dependencies. Different from STSGCN, we connect the latest time slice with each time slice in the input sequence to explore the potential impact of different locations at each time slice on the upcoming time slice. Thus, the adjacency matrix of each spatial-temporal graph reflects the direct spatial-temporal correlation of two different time slices. Moreover, since fuzzy theory make it possible to represent uncertain relationships, it will be beneficial to combine fuzzy reasoning with neural networks to express uncertain spatial-temporal relations. So, to effectively capture the adjacency correlation between traffic locations, a data-driven method that combines fuzzy systems and neural networks is designed to generate the dynamic adjacency matrices. Furthermore, a new graph convolution module is proposed to extract the hidden features. With this framework, the proposed model can dynamically mine the interactive spatial-temporal dependencies through the reconstructed graph structure, and improve the prediction effectiveness through a simplified fuzzy neural network.

The main contributions of this work are as follows:

1) The two graphs from the latest time slice and each past time slice are connected to construct the spatial-temporal graph. The Spatial-Temporal Adjacency Matrix (STAM) of the reconstructed graph reflects direct global spatial correlation across time slices. In this way, the proposed model can capture the spatial and temporal dependency simultaneously, especially the long-term global spatial dependency, by only one layer of graph convolution, which would avoid the excessive error accumulation and consumption in RNN methods in sequence learning.
2) A novel data-driven module based on the fuzzy neural network is proposed to learn the adjacency matrices of traffic graphs. This work is conducive to mine the potential and uncertain interconnections between traffic data. To the best of our knowledge, our proposed model is the first to leverage the fuzzy neural network to learn the adjacency matrix in graph convolution methods.
3) The proposed model is evaluated on six public real-world traffic datasets. The experiments show that our approach outperforms baseline methods.

The rest of the paper is organized as follows. Section 2 introduces problem definition and reviews the spatial-temporal modeling methods for traffic forecasting, especially the GCN-based methods, and the related theory of the fuzzy neural network. The general framework and details of the proposed method DSTAGCN are introduced in Section 3. We conduct the experiments and analyze the result in Section 4 and the conclusion is made in Section 5.

2 Problem Definition and Related Work

2.1 Problem Definition

The traffic forecasting of the road network aims to predict the future traffic states (e.g. flow, speed, etc.). Given a road network, we can represent it as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$, where $\mathcal{V}$ is a set of nodes, $|\mathcal{V}| = N$, $\mathcal{E}$ is a set of edges and $\mathcal{A} \in \mathbb{R}^{N \times N}$is the original adjacency matrix. Each entry $\mathcal{A}_{ij}$ denotes the influence degree of two nodes with an edge $e \in \mathcal{E}$ from v_i to v_j. Here, the nodes in the graph represent the observation point on the road section, and the edges in the graph represent the connection relationship between each node. Denote the traffic data observed on $\mathcal{G}$ at time slice t as

a feature matrix $X_t \in \mathbb{R}^{N \times C}$, where C is the number of features of each node. The problem of spatial-temporal traffic forecasting on road network can be formulated as learning a mapping function f on the road network $\mathcal{G}$ and corresponding historical observation X in the past P time slices and then predict the traffic data in the next Q time slices, which is shown as follows:

$$f(X_{t-p}, \ldots, X_{t-1}, X_t; \mathcal{G}) \rightarrow (\hat{X}_{t+1}, \hat{X}_{t+2}, \ldots, \hat{X}_{t+Q}) \quad (1)$$

where $p = P - 1$ for briefness, P is the length of historical time slices and Q is the length of the time series that need to be predicted.

From the perspective of time, the data of each node itself is a time-varying traffic sequence. From the perspective of space, within a time slice, the traffic state of each node spatially affects others. From the perspective of traffic propagation, the traffic state is simultaneously affected by time and space factors, presenting interactive temporal and spatial characteristics. Hence, capturing these complex dynamic spatial-temporal dependencies is crucial and has drawn a lot of attention in the literature. Besides, the fuzzy neural network helps to mine uncertain adjacency correlations with the integration of the good knowledge representation ability of the fuzzy system and the self-supervised learning ability of the neural network.

2.2 Spatial-Temporal Modeling in Traffic Forecasting

The flourishing research work on spatial-temporal modeling in traffic forecasting work can be generally grouped into three categories: classical statistical methods, machine learning methods, and deep learning methods. Many classical statistical methods are devoted to summarizing the temporal characteristics of traffic flow changes through a statistical model based on data, such as Historical Average (HA), Auto-Regressive Integrated Moving Average (ARIMA) [5], and Vector Auto-Regressive (VAR) [6]. Later, many machine learning methods, such as Support Vector Regression (SVR) [7] were proposed for traffic forecasting. Although such methods can process high-dimensional data and capture nonlinear relationships, they are still labored to capture the complex and dynamic spatial-temporal dependencies. With the improvement of data acquisition capability and hardware computing power, the deep-learning method develops rapidly. Generally, for gird-like data, the spatial correlation can be modeled by Convolutional Neural Network (CNN) and the temporal correlation can be extracted by Recurrent Neural Networks (RNN) (e.g. GRU [8] and LSTM [9]).

In recent years, Graph Convolutional Network (GCN) extends the convolution methods to non-Euclidean data, which can incorporate the topology of a transportation network into a deep neural network. Hence, many GCN-based traffic forecasting methods have been proposed to capture the spatial-temporal dependency.

(1) Most GCN-based methods focus on modeling spatial and temporal characteristics separately. In general, many studies use the GCN to learn the spatial relationship between traffic graph nodes in each time slice, and also use the Recurrent Neural Network (RNN) or the Convolutional Neural Network (CNN) to learn the time-varying dependency of each node in the past several time slices. They then combine these two features either in parallel or serial mechanisms to capture the spatial-temporal dependency. STGCN [10] used Gated CNN and spectral-domain GCN respectively to extract time and space features and then combined them into a spatial-temporal convolutional block. DCRNN [11] introduced the diffusion graph convolution (spatial domain) replacing the matrix multiplications in GRU to model spatial-temporal dependency. T-GCN [12] utilized spectral-domain GCN to capture spatial dependency and then fed these hidden features into a GRU layer. Some other methods [13], [14], [15], with the encoder-decoder structure, used the attention mechanism to capture dynamic changes in space or time dimension and get improvements. STSGCN [4] connected graphs from different time slices to capture spatial-temporal features simultaneously. It has a better performance in capturing short-time dependency with the localized spatial-temporal graph, but it does not consider the long-term global spatial dependency.

(2) The adjacency matrix of a GCN is usually used to recognize the connectivity of a road network. It can also indicate the degree of correlation between traffic locations. Adjacency matrix plays a vital role in both spectral-domain and spatial-domain methods of graph convolution. In a general spectral-domain method [16], the Laplacian matrix is computed from the adjacency matrix. In spatial-domain methods [11], [17], [18], the aggregation of neighbor characteristics is crucial, and such neighbor relationships are usually defined or precomputed by the adjacency matrix. Many GCN-based methods use a predefined adjacency matrix based on the original road network topology. This predefined graph structure tends to be static in the network and always manifests a local spatial correlation, which means a location is always most strongly associated with its immediate neighbors. However, the adjacency relationship of each location is prone to be dynamically changeable and exists in the global graph structure. Based on this, many researchers try to dynamically learn the graph structure. STGNN [19] and GAT [20] utilized the attention mechanism to adaptively adjust the degree of connection between traffic locations but they were still based on the predefined local adjacency relation. GLA [21] used GCN to extract spatial relationships of traffic flow, in which the adjacency matrix is composed of trainable parameters and is obtained through the learning over datasets. Graph Wavenet [22] proposed a self-adaptive adjacency matrix to discover hidden spatial dependencies. SLCNN [23] proposed two data-driven and time-varying graph structure learning modules to capture the global and local structure, respectively. Each module in SLCNN learns a static graph adjacency matrix and a dynamic one from data. These methods have made considerable progress. They all aim to the variable adjacency relationship in each time slice but neglect the spatial relation across time. In a recent study, STFGNN [24] also plays attention to exploiting the spatial and temporal properties of the traffic data in the graph construction. It propose a novel spatial-temporal fusion graph module inspired by Dynamic Time Wrapping algorithm to construct spatial-temporal graph adjacency matrix.

2.3 Fuzzy Neural Network

The fuzzy system is closer to human-like thinking, which uses fuzzy rules to represent inexact data and knowledge

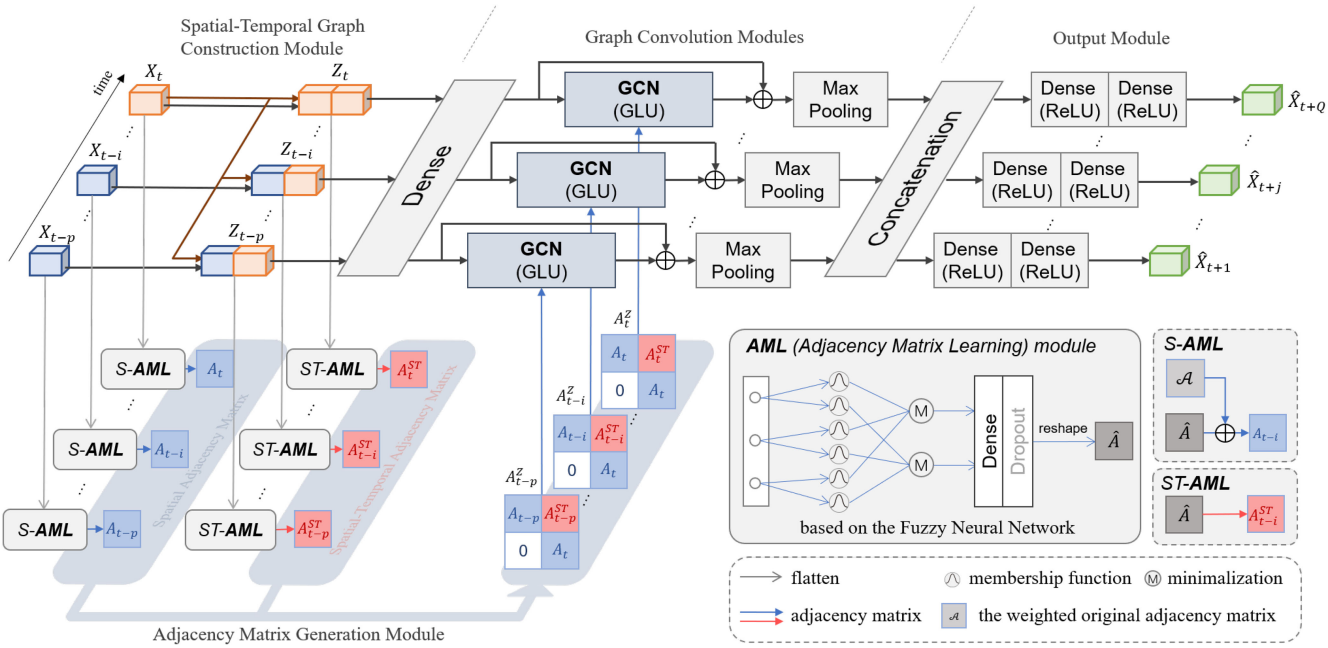


Fig. 3. The overall framework of the proposed DSTAGCN model.

and applies fuzzy inference to data. Fuzzy systems provide convenient and flexible reasoning methods to realize fuzzy but robust and efficient reasoning [25]. Different from traditional fuzzy reasoning systems that require manual rule-setting which takes much time, fuzzy neural networks can learn fuzzy rules and tune membership functions. Since fuzzy relations make it possible to represent uncertain relationships, it might be helpful to integrate fuzzy reasoning into the model to express uncertain spatial-temporal relations.

The fuzzy neural network [26], [27], [28], [29] combines the fuzzy system with the neural network, which has both the reasoning ability of the fuzzy inference system in an environment of uncertainty and imprecision, and the self-learning ability of the neural network. The fuzzy neural network is based on the simple "IF-THEN" rules, like "IF x1 AND x2 AND..., THEN y". A typical fuzzy neural network has three main parts: a fuzzification layer, a rules layer, and a defuzzification layer. In the fuzzification layer, the input variables are fuzzified by appropriate membership functions to obtain the membership values which are the reasoning antecedents for the "IF" part. These values are combined in the rules layer through a specific T-norm operator (usually multiplication or minimalization) to act the "AND" operation of each fuzzy rule. In the defuzzification layer, the qualified consequents of each rule are aggregated to produce an output for the "THEN" part.

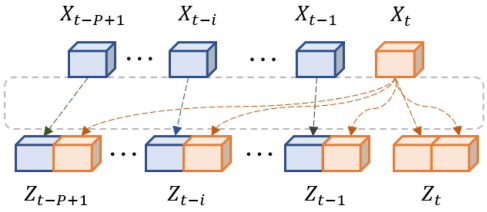
Fuzzy neural networks have been utilized in traffic forecasting. Quek *et al.* [30] described the application of the pseudo-outer-product fuzzy neural network using the truth-value-restriction method for short-term traffic flow prediction. Zhang *et al.* [31] proposed a hierarchical fuzzy system that combined the genetic algorithm (GA) to optimize the rule base for traffic congestion prediction. Tang *et al.* [32] proposed a new method in the construction of a fuzzy neural network to forecast the travel speed. Chen *et al.* [33] presented a fuzzy deep-learning approach to deal with the challenge of uncertainty in large-scale traffic flow prediction. In general, the experimental results showed that the fuzzy neural network was an effective method in traffic state estimation and prediction.

In a nutshell, we aim to capture the spatial-temporal dependency of traffic locations simultaneously from the perspective of traffic state propagation. Different from most existing methods, our model can capture the global spatial correlations between the past time slice and the latest time slice through the spatial-temporal adjacency matrix. And we learn the inexact adjacency matrices dynamically in a data-driven way through a re-designed fuzzy neural network.

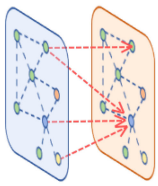
3 Methodology

Considering the unique spatial-temporal dependencies of traffic data during the dynamical traffic propagation, we propose the Dynamic Spatial-Temporal Adjacent Graph Convolutional Network (DSTAGCN) model to address the traffic forecasting issue on the traffic network. Fig. 3 illustrates the overall framework of the proposed DSTAGCN model. Specifically, the model consists of four main parts: *1) Spatial-Temporal Graph Construction Module* for constructing the spatial-temporal graphs which connected each past time slice and the latest time slice. *2) Adjacency Matrix Generation Module* for learning and constructing the adjacency matrix of the spatial-temporal graph. In our work, the fuzzy neural network is employed in generating the adjacency matrix. *3) Graph Convolution Module* for extracting the spatial-temporal features from the spatial-temporal graph with dynamic learnable adjacency matrices. *4) Output Module* for predicting the expected future traffic sequence.

The traffic graph in each time slice $t-i \quad (i = \{0, \ldots, P-1\})$ in the input sequence is connected with that of the latest time slice t to construct a spatial-temporal graph. The feature matrix Z_{t-i} of the spatial-temporal graph is connected by X_{t-i} and X_t correspondingly in the *Spatial-Temporal Graph Construction Module*. The adjacency matrix A^Z_{t-i} of the



(a) Feature matrix construction.



(b) The spatial-temporal graph of Z_{t-i}.

Fig. 4. The Spatial-Temporal Graph Construction Module.

spatial-temporal graph is learned from the input feature matrix X_{t-i} and the constructed feature matrix Z_{t-i} in the *Adjacency Matrix Generation Module*. Then these P transformed spatial-temporal feature matrices and P adjacency matrices are sent into P *Graph Convolution Modules* respectively to capture the hidden features. All hidden features are then concatenated and used to generate the predicted sequence in the *Output Module*.

3.1 Spatial-Temporal Graph Construction Module

To capture the global spatial influence across time between locations, we reconstruct the input sequence. We connect two time slices to construct a spatial-temporal graph. Instead of connecting all nodes with themselves at adjacent time slices, we connect the past time slices and the latest time slice, to directly capture the spatial-temporal dependencies on different time spans. The spatial-temporal graph can be regarded as having a node-set twice the size. The feature matrix on the spatial-temporal graph is constructed by the feature matrices in respective time slice as in Fig. 4a.

$$Z_{t-i} = [X_{t-i}, X_t] \in \mathbb{R}^{2N \times C} \tag{2}$$

Where [] denotes concatenation, $X_{t-i}, X_t \in \mathbb{R}^{N \times C}$, and $Z_{t-i} \in \mathbb{R}^{2N \times C}$ is the spatial-temporal feature matrix.

Fig. 4b illustrates an example of the spatial-temporal graph structure, the orange and blue slices represent the latest time slice t and one of the past time slices $t-i$, respectively, and the dotted red arrows represent spatial correlation across time between different nodes. It is noteworthy that this spatial connection across time is unidirectional and is not limited to the same nodes. We use $A^Z_{t-i} \in \mathbb{R}^{2N \times 2N}$ to denote the adjacency matrix of the spatial-temporal graph which connects the time slice $t-i$ and time slice t. It is defined as:

$$A^Z_{t-i} = \begin{pmatrix} A_{t-i} & A^{ST}_{t-i} \\ 0 & A_t \end{pmatrix} \in \mathbb{R}^{2N \times 2N} \tag{3}$$

where A_{t-i}, A_t are the spatial adjacency matrices of the networks of the past time slice $t-i$ and the latest time slice t. The Spatial-Temporal Adjacency Matrix (STAM) A^{ST}_{t-i} reflects the across-time spatial influence from nodes in past time slice $t-i$ to the nodes in latest time slice t, which recognizes the spatial-temporal correlation. With this structure, A^Z_{t-i} can recognize not only the spatial correlation in time slice t and $t-i$ but also the spatial-temporal correlation between the two time slices.

3.2 Adjacency Matrix Generation Module

The original adjacency matrix of the road network traffic graph represents the spatial association between nodes, which is static. The adjacency matrix of the spatial-temporal graph represents the spatial relations across time.

Instead of using predefined or fixed adjacency matrices, we adopt a data-driven approach to dynamically generate the adjacency matrices. As shown in Fig. 5, we feed the feature matrix X_{t-i} into the *S-AML (Spatial Adjacency Matrix Learning)* module to generate the spatial adjacency matrix A_{t-i} at each time slice $t-i$, and feed the reconstructed feature matrix Z_{t-i} on the spatial-temporal graph into the *ST-AML (Spatial-Temporal Adjacency Matrix Learning)* module to generate the spatial-temporal adjacency matrix A^{ST}_{t-i}, and then combine these two types of adjacency matrices according to Eq. (3) to get A^Z_{t-i}.

In this work, we propose a data-driven method based on the fuzzy neural network to learn the adjacency matrix in both *S-AML* and *ST-AML* modules. The core idea of two *AML* modules is to extract the global interactive correlations

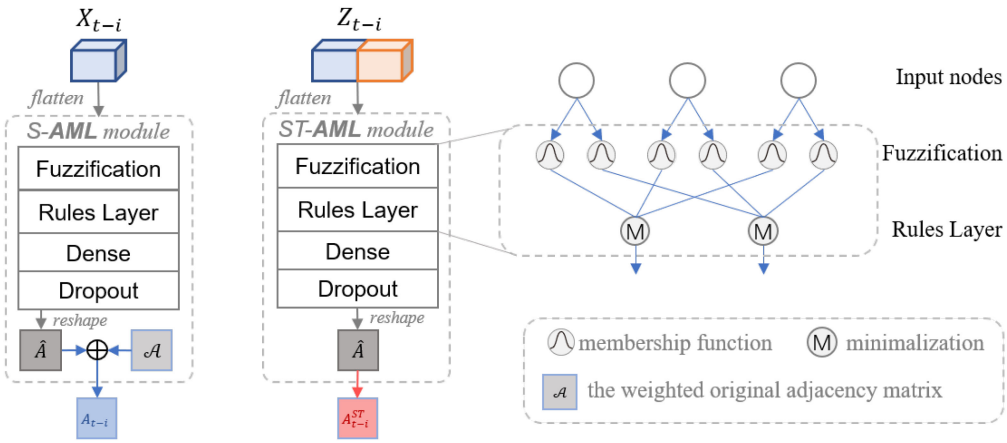


Fig. 5. **Left**: The AML (Adjacency Matrix Learning) modules based on the fuzzy neural network. **Right**: An example of the fuzzification and the rules layer with 3 input nodes and 2 membership functions for each input node.

by examining the data of all traffic locations, and map them to the adjacency matrix. Therefore, the inputs of this fuzzy network are flattened, which investigates all traffic nodes in an individual way. Specifically, for an input feature matrix X_{in}, it can be reshaped into a vector $x_{in} \in \mathbb{R}^{n_{in}*C}$, here n_{in} is the number of input nodes (n_{in} equals N for X_{t-i} and 2 N for Z_{t-i}), C is the number of input feature channels (Here in this paper, the input feature is either traffic flow or traffic speed, i.e. $C = 1$). The n_{in} input nodes are fuzzified in the fuzzification layer. Gaussian function is employed as the membership function in the fuzzification layer. The fuzzification can be formulated as:

$$u_{ki} = MF_{ki}(x_k) = e^{-(x_k-\mu_{ki})^2/\sigma_{ki}^2}, i \in \{1, \ldots, m\} \tag{4}$$

where MF denotes the membership function, m is the number of membership functions for each input, x_k indicates the k^{th} input node, $k \in \{1, \ldots, n_{in}\}$. μ_{ki}, σ_{ki} are learnable parameters in the Gaussian function, and u are outputs of the fuzzification. Different from the learnable parameters which directly weight the input in the hidden layer of a general neural network, the parameters in the fuzzification layer here are only used to define the membership function. The output value of the membership function shows the extent to which a value of input variable is included in a fuzzy set. These values are the reasoning antecedents for the "IF" part in fuzzy rules, they all contribute to the fuzzy reference process in the following layer.

In the rules layer of a conventional fuzzy neural network, the number of fuzzy rules n_F is determined by the combination of the membership functions for input variables [26], which is $n_F = m^{n_{in}}$. In our work, all input nodes are investigated. As n_{in} is relatively high in a citywide traffic network, $n_F = m^{n_{in}}$ will be extremely large which is unacceptable in this model. So, considering the feature homogeneity of the input ($C = 1$), instead of covering each combination of membership functions, we select just one combination for membership functions, which can be formulated as:

$$d_i = min\{u_{1i}, u_{2i}, \ldots, u_{ki}\}, i \in \{1, \ldots, m\} \tag{5}$$

Here, the min function is selected to combine the membership values as a usual T-norm operator and acts the "AND" operation in fuzzy rules. In this way, the number of rules n_F is significantly reduced to m. The output $D = [d_1, \ldots, d_i, \ldots, d_m] \in \mathbb{R}^m$ of the rules layer can be treated as hidden fuzzy features. The right part of Fig. 5 illustrates the detail of an example of these two layers with 3 input nodes ($n = 3$) and 2 membership functions for each input node ($m = 2$).

The defuzzification layer in a general fuzzy neural network consists of normalization and aggregation. Different from the general fuzzy neural network which outputs only one numerical value, we expect to obtain the influence relationship among multiple nodes. Therefore, in our work, a fully connected layer is employed to make defuzzification with a linear projection:

$$A_d = Dense(D) \in \mathbb{R}^{N^2} \tag{6}$$

where $Dense$ means the fully connected layer, D is the fuzzy features. This layer generates the deduction for the "THEN" part in fuzzy rules. Besides, a dropout layer is stacked behind to reduce overfitting. Note that this data-driven module does not directly calculate the adjacency matrix from the input data through certain connection weights, but first uses the fuzzy layers to extract the fuzzy features of the input data and then maps them to the target matrix through a fully connected layer. The output feature A_d are then reshaped into $\hat{A} \in \mathbb{R}^{N \times N}$.

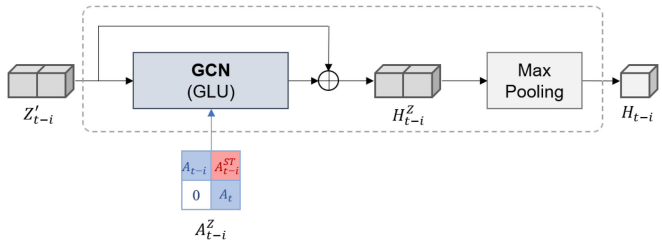


Fig. 6. The Graph Convolution Module.

In the *S-AML* module, since the original adjacency matrix represents a good prior and will have impacts in a single time slice, the original adjacency matrix $\mathcal{A}$ is weighted and then added to the learned one. So the spatial adjacency matrix is calculated as follows:

$$A_{t-i} = \hat{A} + W_{mask} \otimes \mathcal{A} \tag{7}$$

where W_{mask} is the weight of the original adjacency matrix and $\otimes$ denotes element-wise product. This transforms the learned adjacency matrix $\hat{A}$ in each time slice into a residual part of the original one, which is the dynamic correction of the correlation while maintaining a good prior for the short-term spatial dependency. Differently, in the *ST-AML* module, $A^{ST}_{t-i} = \hat{A}$ since the correlation across time is hard to define in advance.

3.3 Graph Convolution Module

After the construction of the spatial-temporal graph, we use a fully connected layer to transform the reconstructed feature sequence $Z = [Z_{t-P+1}, ..., Z_{t-1}, Z_t] \in \mathbb{R}^{P \times 2N \times C}$ into a high-dimension space:

$$Z' = Dense(Z) \in \mathbb{R}^{P \times 2N \times C'} \tag{8}$$

where Z' is the processed feature matrix sequence. As shown in Fig. 6, the graph convolution module is employed to capture the spatial-temporal dependency between nodes in the constructed graph. According to the established spatial-temporal adjacency matrix, we define a convolution method in the spatial domain to aggregate the information of the adjacency nodes of each node on the spatial-temporal graph. Meanwhile, GLU [34] is selected as the activation function. This graph convolution can be formulated as:

$$H^{\mathcal{G}}_{t-i} = (A^Z_{t-i} Z'_{t-i} W_1 + b_1) \otimes \sigma(A^Z_{t-i} Z'_{t-i} W_2 + b_2) \tag{9}$$

where $W_1, W_2 \in \mathbb{R}^{C' \times C_{out}}$, $b_1, b_2 \in \mathbb{R}^{C_{out}}$ are learnable parameters, σ denotes the *sigmoid* activation function, and $H^{\mathcal{G}}_{t-i}$ is the output of the convolution layer. It is worth noting that only one such convolutional layer is adopted here. Due to the construction form of the spatial-temporal graph, the connection of the distant location in the earlier time can be directly reflected by the spatial-temporal adjacency matrix.

Therefore, one layer of convolution is sufficient for aggregating the adjacent features of each node. Besides, we add a skip connection before and after the convolution layer, taking the result of the convolution as the residual part, to facilitate network training:

$$H^Z_{t-i} = H^{\mathcal{G}}_{t-i} + Z'_{t-i} \in \mathbb{R}^{2N \times C_{out}} \tag{10}$$

here $C' = C_{out}$ in practice. Theoretically, these two hyperparameters can be different. In this case, the operation of dimension change needs to be carried out which has been omitted here for brevity. As for the result of the convolution $H^Z_{t-i} \in \mathbb{R}^{2N \times C_{out}}$, we use the max-pooling to compress it into the node feature:

$$(H_{t-i})_{m,:} = max((H^Z_{t-i})_{m,:}, (H^Z_{t-i})_{m+N,:}) \tag{11}$$

where $H_{t-i} \in \mathbb{R}^{N \times C_{out}}$ is the output node feature.

Collectively, here we use one graph convolution layer (based on the spatial domain) and combine the residual connection to aggregate the features of the spatial-temporal adjacent features of each node, and finally use the max-pooling to transform it into the hidden node feature.

3.4 Output Module

The output layer converts the feature matrices output by the graph convolution module into the final expected targets. The output of the graph convolution module on each time slice is $H_{t-i} \in \mathbb{R}^{N \times C_{out}}$. We merge the past P time slice output into $H_o \in \mathbb{R}^{N \times PC_{out}}$. Two fully connected layers are used to generate the predicted output for each predicted time slice, then the final prediction matrix can be concatenated by the output of each time slice:

$$\hat{Y}_{t+j} = ReLU(Dense(ReLU(Dense(H_o)))) \in \mathbb{R}^{N \times C} \tag{12}$$

$$\hat{Y} = [\hat{Y}_{t+1}, \hat{Y}_{t+2}, \ldots, \hat{Y}_{t+Q}] \in \mathbb{R}^{Q \times N \times C} \tag{13}$$

where $ReLU$ is the activation function, and $\hat{Y}$ is the output of the proposed model.

In this work, Huber loss [35] is selected as the loss function due to its robustness to outliers:

$$\mathcal{L}(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2, & |y - \hat{y}| \leq \delta \\ \delta|y - \hat{y}| - \frac{1}{2}\delta^2, & \text{otherwise} \end{cases} \tag{14}$$

where y and $\hat{y}$ represent the ground truth and predictions respectively, and δ is a threshold parameter (1 by default).

In summary, the *Spatial-Temporal Graph Construction Module* provides a novel approach to directly capture spatial associations across distant time by linking the past and the latest time slices. In the *Adjacency Matrix Generation Module*, the adjacency matrices learned on each time slice can reflect the dynamic and heterogeneous spatiotemporal correlation, and the fuzzy network can extract the uncertain adjacency correlations flexibly. Based on the above framework, the *Graph Convolution Module* can extract long-term spatial-temporal features with only one layer of convolution.

4 Experiments

4.1 Datasets and Preprocessing

We evaluate our model on six public traffic network datasets.

TABLE 1
The Detail Statistics of Datasets

Dataset	#Nodes	#Time slices	Data Type
PEMS03	358	26208	flow
PEMS04	307	16992	flow
PEMS07	883	28224	flow
PEMS08	170	17856	flow
METR-LA	207	34272	speed
PEMS-BAY	325	52116	speed

- PEMS03, PEMS04, PEMS07, and PEMS08 published by [4]. These four datasets record traffic flow in four districts in California respectively.
- METR-LA and PEMS-BAY released by [11]. METR-LA contains 4 months of traffic speed from 207 sensors in the highway of Los Angeles County. PEMS-BAY records 6 months of traffic speed from 325 sensors in the Bay Area.

The detailed statistics are shown in Table 1. All data are collected every 5 minutes, which means there are 12 time slices for each hour. Following [4] and [11], we split the first four datasets with a ratio of 6:2:2 into training sets, validation sets, and test sets, and 7:1:2 for the other two datasets. For METR-LA and PEMS-BAY datasets, the missing values are filled with the mean value of each location in the training process. For all datasets, we employ the Z-score normalization to standardize the features:

$$X_{norm} = \frac{X - mean(X)}{std(X)} \tag{15}$$

4.2 Baselines

- **VAR [6]**: Vector Auto Regression, a time series model, can capture the pairwise relationships among traffics.
- **SVR [7]**: Support Vector Regression uses the linear support vector machine for a regression problem.
- **LSTM [9] and FC-LSTM [36]**: Long Short-Term Memory network and Fully Connected Long Short Term Memory network for time series prediction.
- **STGCN [10]**: Spatial-Temporal Graph Convolution Network. STGCN uses the graph convolution method based on spectral domain, integrated with 1D CNN to capture the spatial-temporal correlation of traffic data.
- **DCRNN [11]**: Diffusion Convolutional Recurrent Neural Network. DCRNN introduces diffusion graph convolution to capture the spatial features and uses a variant of RNN to predict the future sequence.
- **T-GCN [12]**: Temporal Graph Convolutional Network. T-GCN uses spectral-domain GCN to capture spatial dependency and then feed these hidden features into GRU to make the prediction.
- **STSGCN [4]**: Spatial-Temporal Synchronous Graph Convolutional Networks, which connect different time slices to form a localized spatial-temporal graph and capture the local spatial-temporal relationship synchronously.

TABLE 2
Performance Comparison of the DSTAGCN and Baselines on 1-Hour Prediction on PEMS03, PEMS04, PEMS07 and PEMS08 Datasets. the Best Results are Shown in **Bold**, the Results in Italics Come From a Re-Implementation in Our Environment, and Other Baseline Results are Cited from [4] and [24]

Models	PEMS03			PEMS04			PEMS07			PEMS08		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
VAR	23.65	24.51	38.26	23.75	18.09	36.66	75.63	32.22	115.24	23.46	15.42	36.33
SVR	21.97	21.51	35.29	28.70	19.20	44.56	32.49	14.26	50.22	23.25	14.64	36.16
LSTM	21.33	23.33	35.11	27.14	18.20	41.59	29.98	13.20	45.84	22.20	14.20	34.06
T-GCN	*20.62*	*20.20*	*35.30*	*24.02*	*15.54*	*39.68*	*31.83*	*13.76*	*58.19*	*22.83*	*13.45*	*38.03*
DCRNN	18.18	18.91	30.31	24.70	17.12	38.12	25.30	11.66	38.58	17.86	11.45	27.83
STGCN	17.49	17.15	30.12	22.70	14.59	35.55	25.38	11.08	38.78	18.02	11.40	27.83
STSGCN	17.48	16.78	29.21	21.19	13.90	33.65	24.26	10.21	39.03	17.13	10.96	26.80
STFGNN	16.77	16.30	28.34	19.83	13.02	31.88	22.07	9.21	35.80	16.64	10.60	26.22
DSTAGCN	**15.31**	**14.91**	**25.30**	**19.48**	**12.93**	**30.98**	**21.62**	**9.10**	**34.87**	**15.83**	**10.03**	**24.70**

TABLE 3
Performance Comparison of the DSTAGCN and Baselines on METR-LA and PEMS-BAY Datasets. The Best Results are Shown in **Bold**, the Results in Italics Come From a Re-Implementation in Our Environment, and Other Baseline Results are Cited from [11] and [22]

Dataset	Models	15min			30min			60min		
		MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
METR-LA	VAR	4.42	10.20	7.89	5.41	12.70	9.13	6.52	15.80	10.11
	SVR	3.99	9.30	8.45	5.05	12.10	10.87	6.72	16.70	13.76
	FC-LSTM	3.44	9.60	6.30	3.77	10.90	7.23	4.37	13.20	8.69
	T-GCN	*4.12*	*12.60*	*7.82*	*4.13*	*12.68*	*7.91*	*4.14*	*12.78*	*7.94*
	STGCN	2.88	7.62	5.74	3.47	9.57	7.24	4.59	12.70	9.40
	STSGCN	*2.85*	*7.57*	*5.40*	*3.30*	*9.30*	*6.53*	*3.96*	*11.95*	*7.97*
	DCRNN	2.77	7.30	5.38	3.15	8.80	6.45	3.60	10.50	7.59
	DSTAGCN	**2.74**	**7.12**	**5.24**	**3.14**	**8.65**	**6.27**	**3.59**	**10.26**	**7.33**
PEMS-BAY	VAR	1.74	3.60	3.16	2.32	5.00	4.25	2.93	6.50	5.44
	SVR	1.85	3.80	3.59	2.48	5.50	5.18	3.28	8.00	7.08
	FC-LSTM	2.05	4.80	4.19	2.20	5.20	4.55	2.37	5.70	4.96
	T-GCN	*2.70*	*6.37*	*5.47*	*2.70*	*6.35*	*5.46*	*2.70*	*6.35*	*5.43*
	STGCN	1.36	2.90	2.96	1.81	4.17	4.27	2.49	5.79	5.69
	STSGCN	*1.46*	*3.09*	*3.00*	*1.86*	*4.17*	*4.12*	*2.33*	*5.43*	*5.26*
	DCRNN	1.38	2.90	2.95	1.74	3.90	3.97	2.07	4.90	4.74
	DSTAGCN	**1.36**	**2.88**	**2.85**	**1.70**	**3.83**	**3.84**	**2.01**	**4.71**	**4.60**

- **STFGNN [24]**: Spatial-Temporal Fusion Graph Neural Networks, which propose a novel spatial-temporal fusion graph module to capture spatial-temporal dependencies synchronously.

4.3 Experiment Settings

Our experiments are conducted on a Linux server with one Intel(R) Xeon(R) Gold 6230 CPU @ 2.10 GHz and one NVIDIA Tesla V100-SXM2 GPU card. We implement the proposed model with Keras based on TensorFlow. For all datasets, the input sequence length P and the output sequence length Q are both 12, which means to use one-hour historical data to predict the next hour data. The number of membership functions m for each input node in the fuzzification layer is 10 and the dropout rate in the *AML* module is set as 0.8. And just 1 graph convolution layer that has 64 filters is employed. In our experiments, we use Adam optimizer to train the model with a learning rate of 0.001 for 200 epochs. The batch size is 32.

We use three metrics to evaluate the prediction performance of the proposed model: Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Error (RMSE). All missing values are excluded from testing. For the METR-LA and PEMS-BAY datasets, following the implementation of [11] and [22], the metrics are calculated from one single predicted time slice and the truth value, that is, the 15, 30, 60 min in table 3 means the 3 rd, 6th, 12th time slice respectively. For the other four PEMS datasets, following the implementation of [4], the metrics are calculated from the whole 12 time slices.

4.4 Experiment Results

The performance comparison of various methods is shown in Table 2 and Table 3. The proposed DSTAGCN achieves better performance than baseline methods consistently on these six datasets.

DSTAGCN outperforms the traditional time-series models and classic GCN-based models on every dataset. VAR and SVR perform quite poorly because they only take time dependency into account and simply make predictions statistically or analytically from historical sequences. LSTM and FC-LSTM also only consider the time factor and do not achieve comparable performance. Both DCRNN and T-GCN use GCN and GRU to capture spatial and temporal

features respectively. STGCN uses spectral domain graph convolution to capture spatial correlation and 1D CNN to process time series. And the adjacency matrices used by these methods are all predefined and fixed in each time slice, which means they do not consider the dynamic characteristics of traffic data and could not obtain good prediction performance.

Compared with STSGCN, the MAE and RMSE of DSTAGCN are approximately 12.41% and 13.38% lower on the PEMS03 dataset, respectively, and we connect fewer time slices (2 time slices) and fewer convolutional layers (1 layer). STSGCN similarly connects different time slices to form a spatial-temporal graph, but it only captures local spatial-temporal relationships, and the impact on long-term global neighbors needs to be extracted by stacking many convolutional layers. Although weighted by the *Mask* matrix, the adjacency matrix adopted by STSGCN is still limited to the predefined adjacency relation and does not extend to the global neighbor relation.

Compared with STFGNN, which similarly pay attention to the generation of the spatial-temporal adjacency matrices, the MAE and RMSE of DSTAGCN are approximately 8.70% and 12.01% lower on the PEMS03 dataset, respectively. Despite that the data-driven adjacency matrices of the reconstructed graph in STFGNN contribute to the better performance than STSGCN, the connection of continuous adjacent time slices (4 in practice) and multi-layers framework in STFGNN brings higher model complexity and makes it labored to capture the spatial correlation across time (it requires stacking multiple complicated layers).

The consistently better results on these datasets show that this organization of the spatial-temporal graph in our DSTAGCN is more effective due to the more direct way to capture the global spatial-temporal dependency across time with a concise model framework.

4.5 Components Analysis

To further test and verify the components of the DSTAGCN model, variant experiments are conducted and analyzed. Different methods of adjacency matrix generation in the *Adjacency Matrix Generation Module* and the number of convolution layers in the *Graph Convolution Module* are evaluated.

4.5.1 Analysis on the Adjacency Matrix Generation Module

In our method, the adjacency matrix of a spatial-temporal graph composed of two different time slices is an important representation of the spatial-temporal relationship, and we use the fuzzy neural network to generate it in the *Adjacency Matrix Generation Module*. Therefore, we design a comparative experiment on how to generate the adjacency matrix, and also verify the effectiveness of the fuzzy neural network.

The detailed construction of the adjacency matrix A^Z_{t-i} of the spatial-temporal graph in time slice $t-i$ is in *Section 3.1*. Here in this section, five methods of generating the adjacency matrix are tested.

- **Original ADJ**: $A^Z = \begin{pmatrix} \mathcal{A} & I_N \\ 0 & \mathcal{A} \end{pmatrix}$, where $\mathcal{A}$ is the original adjacency matrix of the traffic graph, I_N is the identity matrix. This connection means connecting each node of a graph at one time slice to itself at another time slice, which represents the temporal influence is restricted to the node itself.
- **Mask ADJ**: $A^Z = W_{mask} \otimes \begin{pmatrix} \mathcal{A} & I_N \\ 0 & \mathcal{A} \end{pmatrix}$, where W_{mask} is a learnable parameter and represents the different degrees of influence of the association between nodes. This method is similar to STSGCN's [4] processing of adjacency matrix. The difference is that STSGCN connects local adjacent time slices, while this work connects the past and latest time slices. Both only take the temporal influence of the node itself into account.
- **Learnable Parameters**:

$$A^Z = SoftMax(ReLU(\mathbf{E}_1\mathbf{E}_2^T))$$

  where $\mathbf{E}_1, \mathbf{E}_2$ are two node embeddings with fully learnable parameters. This is similar to the method of Graph Wavenet [22].
- **MLP**: This variant removes the fuzzification layer and rules layer of the proposed DSTAGCN and replaces them with a fully connected layer that outputs the same dimensional features. This constitutes a simple multi-layer perceptron structure.
- **Fuzzy Network**: This is the model we proposed in this work and the detail is described in *Section 3.2*. The adjacency matrix is learned from the input data through a simplified fuzzy neural network.

Based on these variants, we designed two group experiments of *static* and *dynamic* according to whether the spatial-temporal graph on each time slice uses an independent adjacency matrix. This aims to verify the necessity of dynamically learning the adjacency matrix. Specifically, *static* represents that the graph structure on each time slice is the same and invariable, and *dynamic* represents that the graph adjacency matrix on each time slice is independent.

The results of these two groups of experiments on 1-hour prediction are illustrated in Fig. 7. It is noteworthy that the performance of STSGCN (as in Table 2 and Table 3) are plotted as a red dotted line in the figure for illustration since it is the first model to capture the spatial-temporal dependency synchronously and inspires our work. From the results shown in Fig. 7, we can find:

(1) The *Original ADJ* only employs the same and fixed spatial adjacency matrix within different time slices, and only considers the correlation across time of the same traffic node itself, so the performance is the worst. (2) The *Mask ADJ* introduces the weight of the influence of each node based on the *Original ADJ* and has made progress in the comparison, but it is still impossible to capture the global spatial dependency across time only through one convolutional layer. Although this variant adopts an adjacency matrix construction and weighting method similar to STSGCN, its effect is inferior to that of STSGCN because the connected time slices are not locally close, which indicates that the use of local adjacency correlation is not applicable to this spatial dependency across time. (3) When the adjacency matrix is defined as a fully *Learnable Parameters*, we can see that it is no longer limited to capture the localized spatial-temporal relationship, and it performs better than the

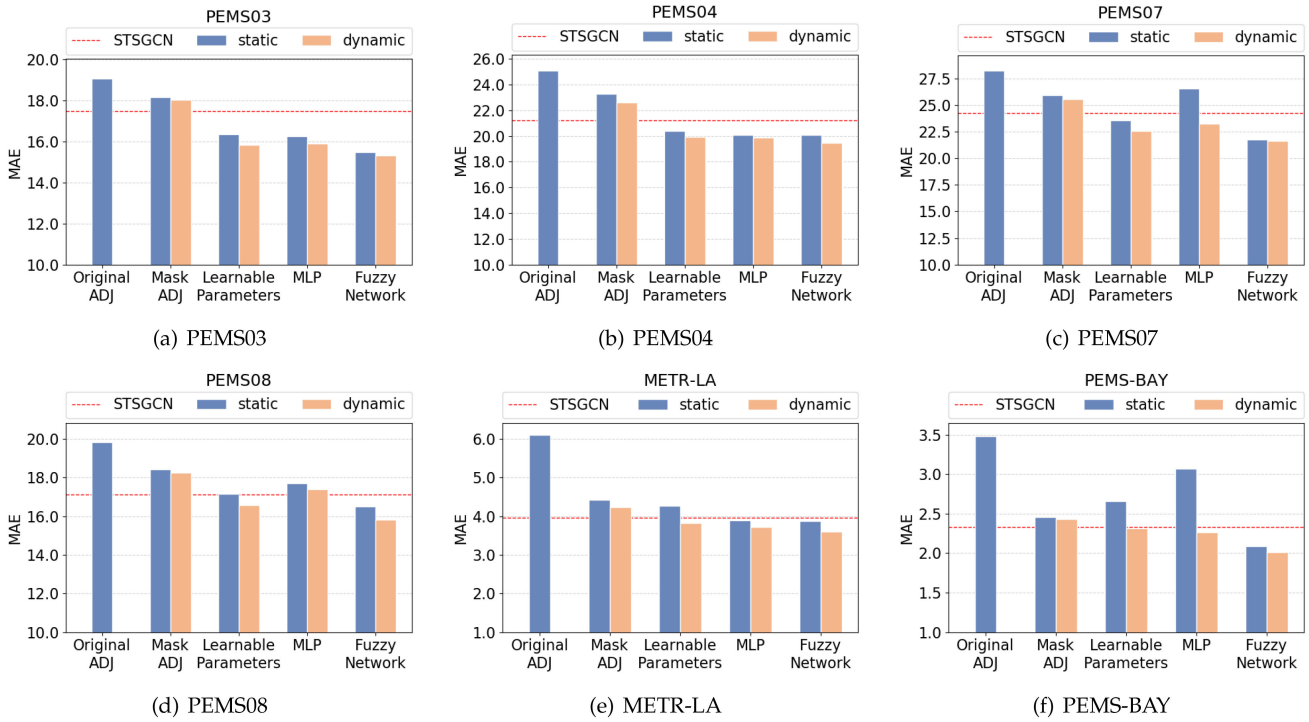


Fig. 7. Performance comparison of different generation methods of the adjacency matrix.

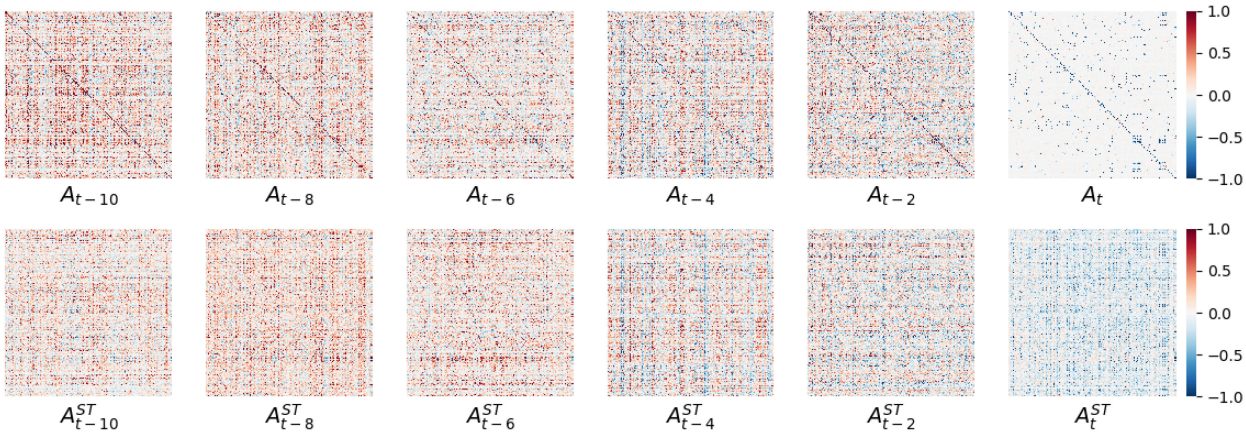


Fig. 8. The estimated adjacency matrices on METR-LA dataset in six certain time slices.

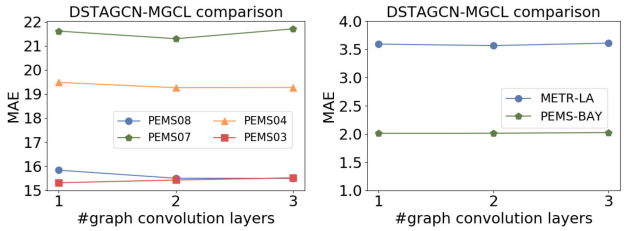
predefined adjacency relationship. This organizational structure of the spatial-temporal graph we proposed begins to exert its effect and is superior to STSGCN. (4) The *MLP* variant handily mines the adjacency relationship from global traffic data while it is not always superior to the *Learnable Parameters* method on all datasets. This demonstrates that the plain data-driven method may have a strong link with the instinct of datasets and be prone to obtain mixed performance. (5) The result of the *Fuzzy Network* variant in the comparison test is the best, which indicates that the fuzzy method performs well not only by simply adding parameters or being driven by data but also by its promising data inference capability for such uncertain adjacency relations. (6) The *dynamic* results are consistently better than *static*, which shows that traffic data tends to have different spatial-temporal correlations at different time slices, i.e., spatial correlations will change over time. This verifies the importance of simultaneously capturing dynamic spatial-temporal correlations.

To intuitively show the dynamic adjacency matrices learned by DSTAGCN, we visualize the spatial adjacency matrices and the spatial-temporal adjacency matrices on the METR-LA dataset in six time slices in Fig. 8. It can be seen that the adjacency matrix changes over time. The proposed method captured the dynamics of the traffic correlation in both spatial and spatial-temporal perspectives.

4.5.2 Other Components Test Analysis

To verify the impact of other components of DSTAGCN, the following three variant experiments were also conducted and analyzed:

- **DSTAGCN-MGCL**: it stacked multiple graph convolutional layers in the *Graph Convolution Module*.
- **DSTAGCN-NDP**: it removes the dropout layer when using a fuzzy neural network to generate an adjacency matrix in the *AML* module.



(a) PEMS03, PEMS04, PEMS07, PEMS08 (b) METR-LA, PEMS-BAY

Fig. 9. Performance comparison on DSTAGCN with the different number of graph convolutional layers.

- **DSTAGCN-NST**: it removes spatial-temporal adjacency matrix, which means $A^{ST}_{t-i} = 0$.

From the test results shown in Fig. 9 and Fig. 10, we can find:

1) The results of DSTAGCN-MGCL variants show that setting the number of convolution layers to be 2 or 3 in the *Graph Convolution Module* does not significantly improve the model performance or even worsen it, while the increasing of parameters brings more training consumption. This indicates that the adjacency matrix of the spatial-temporal graph formed by the connection of the distant time slice and the latest time slice in our proposed model can represent the global spatial relationship between the distant location and the target location, and this relationship can be captured with one graph convolution layer. In most of the existing methods, although the global spatial dependencies can be captured indirectly by increasing the number of convolution layers or the number of diffusion steps (in diffusion graph convolution), the spatial dependencies captured by them are all in the same time slice, and the potential time constraints of global spatial relations are not fully and directly considered.
2) The DSTAGCN-NST simply combines different time slices to capture spatial relationships simultaneously but neglects the potential temporal correlations between them. This variant performs worse than the proposed DSTAGCN due to the vacant spatial-temporal adjacency matrix, which demonstrates that the global spatial dependency across time contributes to the proposed method.
3) The DSTAGCN-NDP model removes the dropout layer in the *Adjacency Matrix Learning* module, and its poor performance on PEMS04, PEMS07, and PEMS08 data sets is due to the serious over-fitting on these datasets. This indicates that the simplified fuzzy neural network shows its effectiveness by using fuzzy theory for data inference on the training set, but it is prone to overfit. In contrast, the DSTAGCN has the best performance in the test set after employing the dropout technique to alleviate the overfitting.

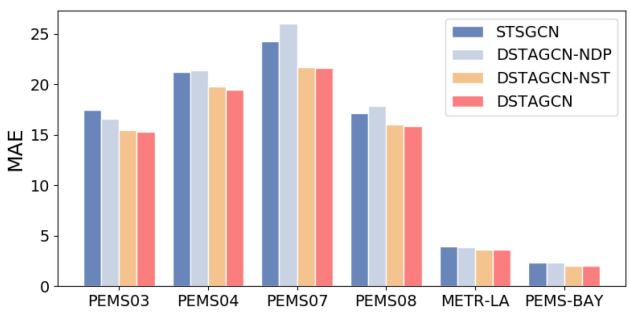


Fig. 10. Performance comparison on ablation experiments.

TABLE 4
Time Consumption on PEMS03 Dataset

	Training time	Inference time
STSGCN	107 seconds/epoch	16.7 seconds
DSTAGCN	53 seconds/epoch	7.6 seconds

4.6 Efficiency Analysis

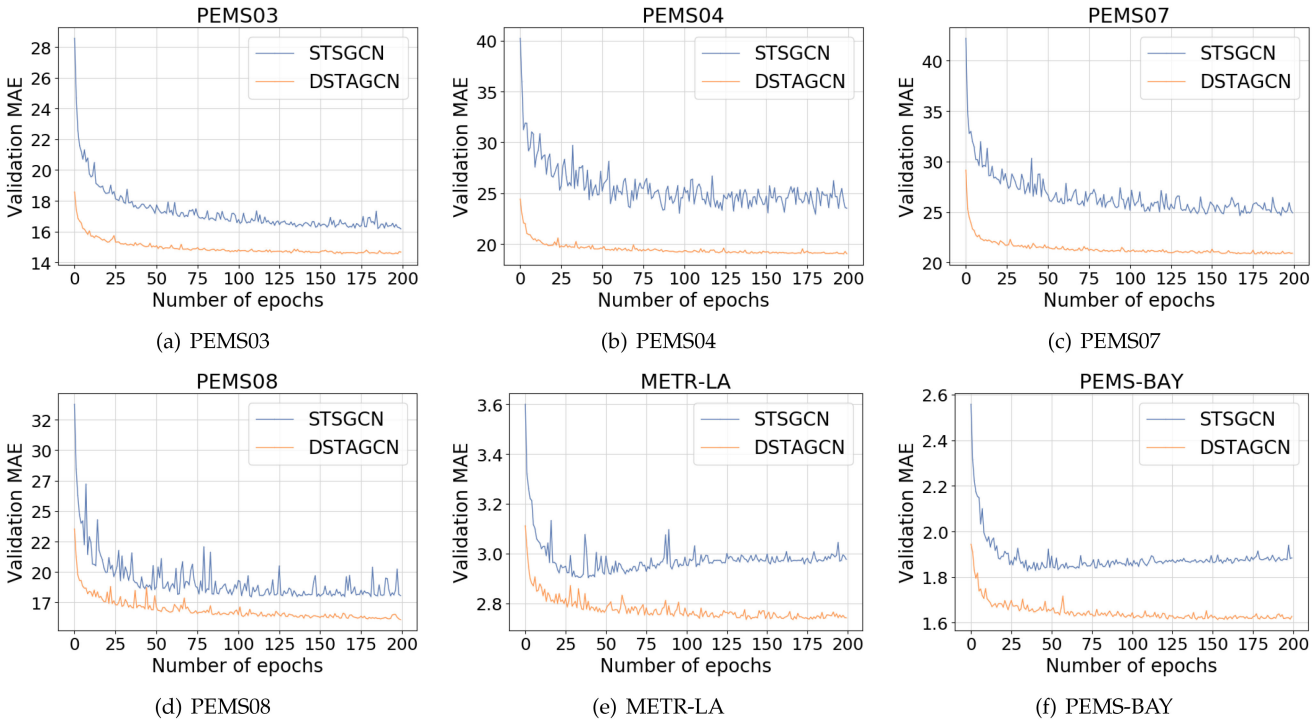
To illustrate the cost consumption and effectiveness of our proposed model, we compare the time consumption and convergence curves of DSTAGCN (the proposed model) and STSGCN since they both connect graphs from multiple different time slices to capture spatial-temporal features and achieve good performance. It should be noted that the experiments of these two models are carried out in the same environment on the same device, and have the same batch size and learning rate.

Here we show the time consumption of the two models on the PEMS03 dataset, similar results were consistently shown on other five datasets. It can be seen from Table 4 that the training time of each epoch of our proposed model only needs 49.5% of that of STSGCN. For inference, we measure the total time cost on the validation data, DSTAGCN runs twice as fast as STSGCN. This is because our proposed model connects fewer time slices (2 slices) to construct the spatial-temporal graph and sets fewer convolution layers (only 1 layer in total), despite that parameters are added to learn the adjacency matrix in the fuzzy neural network part.

Fig. 11 plots the validation MAE curves of the two models in the training process of one-hour prediction on all datasets, indicating that our DSTAGCN converges faster and has lower validation errors than STSGCN. Although the results compared here are for 200 epochs of training, in the experiment, when we used the early-stopping technique, our model quickly achieved better results than STSGCN after dozens of epochs. The proposed model consistently shows fast convergence and small errors on each dataset.

4.7 Discussion

Compared to other methods, the proposed model aggregates two graph feature matrix from different time slices together simply, and simultaneously captures the dependency between distant time and distant space through convolution, which avoids error accumulation and parameter consumption in the RNN-based models. The fuzzy neural network provides a flexible perspective similar to human reasoning to obtain uncertain spatial-temporal correlation, which fuzzies the data through simple membership functions without introducing many parameters. In this way, the uncertain spatial-temporal dependency can be handily captured as the adjacency matrix. Based on this framework,



(a) PEMS03 (b) PEMS04 (c) PEMS07

(d) PEMS08 (e) METR-LA (f) PEMS-BAY

Fig. 11. Model convergence curves.

only one graph convolution layer is needed to capture the distant temporal and spatial dependencies and the experiments show the improvement. The current time cost of the proposed model grows quadratically with the number of graph nodes. With the number of nodes increases, improving the time performance is our concern.

5 Conclusion

In this work, we proposed a novel Dynamic Spatial-Temporal Adjacent Graph Convolutional Network (DSTAGCN) for traffic forecasting. Considering the spatial-temporal dependency of each node on the traffic network and its propagation mode, we connect graphs from the latest time slice with each past time slice to construct the spatial-temporal graph. The adjacency matrix of this spatial-temporal graph can directly capture the cross-time spatial dependency of the global locations between time slices and the spatial relations in each time slice. We design a simplified fuzzy neural network to learn the above inexact spatial-temporal adjacency matrix from the observations. Extensive experiments on multiple datasets show that our DSTAGCN model is consistently superior to baselines. And the proposed model achieved a more accurate performance with faster convergence.

In our future work, we will explore the influence of more geographic information of the road network (e.g. node distance) and multi-variate features on the generation of the adjacency matrix to get further improvements.

References

[1] B. Yu, Y. Lee, and K. Sohn, "Forecasting road traffic speeds by considering area-wide spatio-temporal dependencies based on a graph convolutional neural network (GCN)," *Transp. Res. Part C: Emerg. Technol.*, vol. 114, pp. 189–204, 2020.

[2] T. Bogaertsa, A. D. Masegosa, J. S. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph CNN-LSTM neural network for short and long-term traffic forecasting based on trajectory data," *Transp. Res. Part C: Emerg. Technol.*, vol. 112, pp. 62–77, 2020.

[3] S. Guo, Y. Lin, S. Li, Z. Chen, and H. Wan, "Deep spatial-temporal 3D convolutional neural networks for traffic data forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3913–3926, Oct. 2019.

[4] C. Song, Y. Lin, S. Guo, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, pp. 914–921.

[5] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *J. Amer. Stat. Assoc.*, vol. 65, no. 332, pp. 1509–1526, 1970.

[6] E. Zivot and J. Wang, "Vector autoregressive models for multivariate time series," *Model. Financial Time Ser. S-PLUS*, pp. 385–429, 2006. [Online]. Available: https://doi.org/10.1007/978-0-387-32348-0_11

[7] G. Ristanoski, W. Liu, and J. Bailey, "Time series forecasting using distribution enhanced linear regression," in *Proc. Pacific-Asia Conf. Knowl. Discov. Data Mining*, Berlin, Heidelberg, Springer, 2013, pp. 484–495.

[8] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data," in *Proc. 10th Int. Conf. Mach. Learn. Comput.*, New York, NY, USA, Association Computing Machinery, 2018, pp. 26–30.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3634–3640.

[11] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–16.

[12] L. Zhao *et al.*, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 9, pp. 3848–3858, Sep. 2020.

[13] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, pp. 1234–1241, 2020.

[14] F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, and A. Khokhar, "Reinforced spatiotemporal attentive graph neural networks for traffic forecasting," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6414–6428, Jul. 2020.

[15] C. Park *et al.*, "ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1215–1224.
[16] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[17] Q. Zhang, Q. Jin, J. Chang, S. Xiang, and C. Pan, "Kernel-weighted graph convolutional network: A deep learning approach for traffic forecasting," in *Proc. 24th Int. Conf. Pattern Recognit.*, 2018, pp. 1018–1023.
[18] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA, Curran Associates Inc., 2017, pp. 1025–1035.
[19] X. Wang *et al.*, "Traffic flow prediction via spatial temporal graph neural network," in *Proc. Web Conf.*, New York, NY, USA, Association Computing Machinery, 2020, pp. 1082–1092. [Online]. Available: https://doi.org/10.1145/3366423.3380186
[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.
[21] Z. Li *et al.*, "A hybrid deep learning approach with GCN and LSTM for traffic flow prediction," in *Proc. IEEE Intell. Transp. Syst. Conf.*, 2019, pp. 1929–1933.
[22] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph waveNet for deep spatial-temporal graph modeling," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 1907–1913.
[23] Q. Zhang, J. Chang, G. Meng, S. Xiang, and C. Pan, "Spatio-temporal graph structure learning for traffic forecasting," *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 1, pp. 1177–1185, 2020.
[24] M. Li and Z. Zhu, "Spatial-temporal fusion graph neural networks for traffic flow forecasting," *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, pp. 4189–4196, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16542
[25] N. K. Kasabov, *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. Cambridge, MA, USA: MIT Press, 1996.
[26] S. Horikawa, T. Furuhashi, S. Okuma, and Y. Uchikawa, "Composition methods of fuzzy neural networks," in *Proc. IECON '90: 16th Annu. Conf. IEEE Ind. Electron. Soc.*, 1990, vol. 2, pp. 1253–1258.
[27] S. Horikawa, T. Furuhashi, and Y. Uchikawa, "On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm," *IEEE Trans. Neural Netw.*, vol. 3, no. 5, pp. 801–806, Sep. 1992.
[28] J. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, pp. 665–685, May/Jun. 1993.
[29] J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets Syst.*, vol. 66, no. 1, pp. 1–13, 1994.
[30] C. Quek, M. Pasquier, and B. B. S. Lim, "POP-TRAFFIC: A novel fuzzy neural approach to road traffic analysis and prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 2, pp. 133–146, Jun. 2006.
[31] X. Zhang, E. Onieva, A. Perallos, E. Osaba, and V. C. S. Lee, "Hierarchical fuzzy rule-based system optimized with genetic algorithms for short term traffic congestion prediction," *Transp. Res. Part C: Emerg. Technol.*, vol. 43, pp. 127–142, 2014.
[32] J. Tang, F. Liu, Y. Zou, W. Zhang, and Y. Wang, "An improved fuzzy neural network for traffic speed prediction considering periodic characteristic," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2340–2350, Sep. 2017.
[33] W. Chen *et al.*, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial-temporal data features," *Future Gener. Comput. Syst.*, vol. 89, pp. 78–88, 2018.
[34] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *Proc. 34th Int. Conf. Mach. Learn.- Volume 70*, 2017, pp. 933–941.
[35] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics: Methodology and Distribution*. New York, NY, USA: Springer, 1992, pp. 492–518.
[36] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA, MIT Press, 2014, vol. 2, pp. 3104–3112.

Qi Zheng received the BS degree in computer science in 2019 from Tongji University, Shanghai, China, where he is currently working toward the postgraduation degree with the Key Laboratory of Embedded System and Service Computing. His research interests include intelligent transportation system and data mining.

Yaying Zhang received the BS degree in computer science and the MS degree in electrical engineering from the Shandong University of Science and Technology, Shandong, China, in 1996 and 1999, respectively, and the PhD degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2004. She is currently a professor with the Key Laboratory of Embedded System and Service Computing, Tongji University, Shanghai China. Her research interests include intelligent transportation system, data integrity, data mining, and traffic control.