Make More Connections: Urban Traffic Flow Forecasting with Spatiotemporal Adaptive Gated Graph Convolution Network

BIN LU, XIAOYING GAN, HAIMING JIN, LUOYI FU, and XINBING WANG, Shanghai Jiao Tong University, China
HAISONG ZHANG, Tencent AI Lab, China

Urban traffic flow forecasting is a critical issue in intelligent transportation systems. Due to the complexity and uncertainty of urban road conditions, how to capture the dynamic spatiotemporal correlation and make accurate predictions is very challenging. In most of existing works, urban road network is often modeled as a fixed graph based on local proximity. However, such modeling is not sufficient to describe the dynamics of the road network and capture the global contextual information. In this paper, we consider constructing the road network as a dynamic weighted graph through attention mechanism. Furthermore, we propose to seek both spatial neighbors and semantic neighbors to make more connections between road nodes. We propose a novel ***Spatiotemporal Adaptive Gated Graph Convolution Network*** (**STAG-GCN**) to predict traffic conditions for several time steps ahead. STAG-GCN mainly consists of two major components: (1) multivariate self-attention **Temporal Convolution Network** (**TCN**) is utilized to capture local and long-range temporal dependencies across recent, daily-periodic and weekly-periodic observations; (2) mix-hop AG-GCN extracts selective spatial and semantic dependencies within multi-layer stacking through adaptive graph gating mechanism and mix-hop propagation mechanism. The output of different components are weighted fused to generate the final prediction results. Extensive experiments on two real-world large scale urban traffic dataset have verified the effectiveness, and the multi-step forecasting performance of our proposed models outperforms the state-of-the-art baselines.

CCS Concepts: • **Information systems → Spatial-temporal systems;**

Additional Key Words and Phrases: Traffic forecasting, spatiotemporal data, graph neural network, urban computing

This work was supported by National Key R&D Program of China 2018AAA0101200; in part by National Natural Science Foundation of China under Grants No. 42050105, No. 62061146002, No. 62032020, No. 61829201, No. 61960206002; in part by Tencent AI Lab Rhino-Bird Focused Research Program JR202034; in part by the Science and Technology Innovation Program of Shanghai, Grant No. 18XD1401800.

Authors' addresses: B. Lu, X. Gan (corresponding author), H. Jin, L. Fu, and X. Wang, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, China; emails: {robinlu1209, ganxiaoying, jinhaiming, yiluofu, xwang8}@sjtu.edu.cn; H. Zhang, Tencent AI Lab, Science and technology Zhongyi Road, Shenzhen, Guangdong, China; email: hansonzhang@tencent.com.

2157-6904/2022/01-ART28 $15.00
https://doi.org/10.1145/3488902

ACM Reference format:
Bin Lu, Xiaoying Gan, Haiming Jin, Luoyi Fu, Xinbing Wang, and Haisong Zhang. 2022. Make More Connections: Urban Traffic Flow Forecasting with Spatiotemporal Adaptive Gated Graph Convolution Network. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 28 (January 2022), 25 pages.
https://doi.org/10.1145/3488902

1 INTRODUCTION

With the vigorous development of data acquisition technologies, traffic data, such as vehicle trajectory, road sensor data, etc., are exploding. Accurate and timely traffic flow forecasting according to historical observations helps road users make better travel plans, alleviate traffic congestion, and improve traffic operation efficiency [1–3]. Thus, urban traffic flow forecasting has gained more and more attention with the rapid development of intelligent transportation systems.

Early approaches for traffic flow prediction are usually developed on time series with statistical methods or traditional machine learning models. However, these approaches [4, 5] only consider temporal information and ignore the importance of spatial information for accurate prediction. According to our daily life experience, road conditions near roads have strong causality for traffic prediction. Recently, deep learning based methods for traffic forecasting have been extensively studied [6–10]. Some researchers [11, 12] model the traffic network as grids and use **convolution neural network** (**CNN**) to capture the spatial correlations. However, modeling with grids will inevitably lose the topology information within the traffic network due to the irregularity of the roads. To deal with this problem, some researchers consider modeling roads as nodes, and constructing edges according to the relationship between road networks as shown in Figure 1. Then use **graph neural network** (**GNN**), which is efficient in capturing non-Euclidean correlation, to further embed the prior knowledge of traffic network and capture the complex spatiotemporal correlations by aggregating neighbor nodes' information.

In spite of the promising performance of GNN-based methods, there are several important aspects that previous methods have overlooked. First, these methods only consider local spatial proximity when using GNNs to aggregate neighbor information, but ignore global contextual features. For some nodes, their features have strong similarities, such as roads belonging to the same type of functional area, although they are not close geographically. Figure 2 shows the road condition of one target road segment and its spatial/semantic neighbors in one week. The semantic neighbor has more similar periodic characteristics with the target segment and facilitates long-term traffic prediction. The spatial neighbor is capable of capturing dynamics in accidental situation. The combination of these two kinds of features can better capture the complexity and uncertainty of urban road conditions. Second, some previous methods mainly model spatial dependencies by constructing a fixed graph. However, urban traffic conditions change at any time, and the interaction with roads should also be dynamic. Finally, in order to avoid the over-smoothing problem, existing GNN models often get the best results when cascading two-layer graph convolution networks. However, shallow networks cannot obtain deeper and richer spatial features. How to aggregate the results of multi-layer GNNs to achieve better performance has not been involved in previous studies.

To address the aforementioned problems, we model the spatial dependency as a dynamic weighted graph according to the input traffic conditions. In order to describe spatial correlations more comprehensively, we define spatial neighbors and semantic neighbors of road nodes, which respectively represent the connectivity of roads and the contextual similarity of traffic flow features. By making more connections to the nodes, the useful edge information is expanded. For the performance degradation of cascaded GNNs, we propose an adaptive gating mechanism to selectively update and forget high-dimensional features, and a mix-hop propagation mechanism

Road	Coordinates	Time	Velocity
Liangui West Road	104.09464,30.63786	2020/5/8 8:00-8:05	43.2m/s
Hongji Middle Road	104.09697 30.64012	2020/5/8 8:00-8:05	39.4m/s
Shunjiang Road	104.09201 30.64147	2020/5/8 8:00-8:05	32.8m/s
......	......	......	......

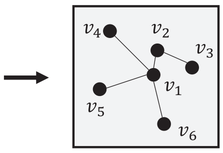


Fig. 1. Make more connections: use graph data structure to establish connections between originally isolated nodes.

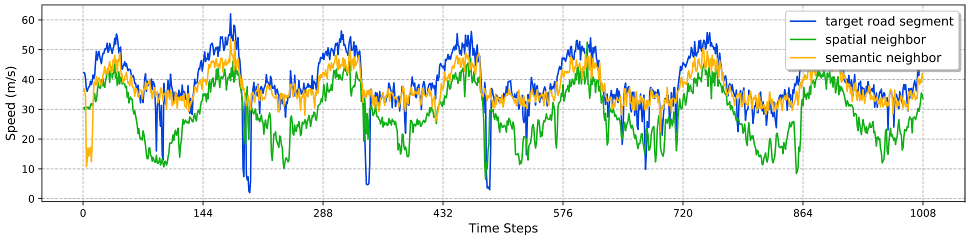


Fig. 2. Illustration of traffic condition in 1 week in terms of different road segments.

to aggregate the results of multi-layer GNN. In general, we propose a deep learning model called **SpatioTemporal Adaptive Gated Graph Convolution Network** (**STAG-GCN**) to collectively predict traffic flow at every location on the traffic network.

Our main contributions are summarized as follows:

— We propose to make more connections in graph by seeking the spatial neighbors and semantic neighbors of road nodes. A multi-head graph attention mechanism is utilized to formulate the road relationship as a dynamic weighted graph.
— We propose STAG-GCN, a holistic approach that captures temporal, spatial, and semantic dependencies among time series. A novel adaptive graph gating mechanism is proposed to selectively update and forget the high-order neighbor information of nodes within the multi-layer stacking. In addition, the mix-hop propagation mechanism is designed to aggregate the results of multi-layer GNN.
— We conduct extensive experiments on two real-world urban traffic datasets by using our proposed model, STAG-GCN, and outperform the performance compared to existing baselines. We also carry out rich experiments on the model itself, and explain the performance and design rationale in detail.

This article is structured as follows. In Section 2, we concisely review the existing approaches for traffic forecasting and GNN. In Section 3, we formally formulate the multi-step urban traffic flow forecasting problem and discuss the spatial dependency modeling. We then present the proposed STAG-GCN model and its components in detail in Section 4. In Section 5, we conduct extensive experiments on real-world datasets and in-depth experimental analysis of the model. Finally, we conclude our article and present our future work in Section 6.

A preliminary version of this work appeared in the 29th ACM International Conference on Information Knowledge Management [13].

2 RELATED WORK

In this section, we briefly review the methods of traffic forecasting in recent years and existing models of GNN.

2.1 Traffic Forecasting

The traffic forecasting problem has been widely studied for a long time [14–17]. The majority of existing methods follow a statistical method, such as **auto-regressive integrated moving average** (**ARIMA**), vector auto-regressive moving average, and Gaussian process [18, 19]. However, a statistical method predicts each road traffic separately, ignoring the impact of spatial relationships on traffic conditions. In the past 2 years, researchers have tended to model the road network as a graph to capture the spatiotemporal correlations and make more accurate traffic forecasting [20–24]. Li et al. [25] proposed to model the traffic flow as a diffusion process on a directed graph and introduced **Diffusion Convolutional Recurrent Neural Network (DCRNN)** inspired by **Gate Recurrent Unit (GRU)** [26] structure. Yu et al. [27] combined a graph convolution with a 1D convolution and proposed STGCN model, which is much more computationally efficient than Recurrent Neural Network (RNN) structure. Lv et al. [28] encoded the heterogeneous spatial and semantic correlations using multiple graphs. However, these models have a common disadvantage. Their spatial dependency modeling is fixed once trained, ignoring dynamic changes in traffic conditions.

To further model the complicated spatial correlations in traffic forecasting, Fang et al. [29] proposed to compute the global spatial dependency, which is to calculate the correlation between all nodes in the graph for feature aggregation. However, for large-scale graphs, this method has high computational complexity, large memory consumption, and slow training speed. To solve this problem, Zheng et al. [30] proposed to randomly group the nodes, then calculate the global dependencies by group, and use an inter-group attention mechanism to combine the outputs. Zhang et al. [31] proposed a hierarchical attention network to capture both the multi-level temporal relations and cross-region traffic dependencies. However, directly learning the global feature information lacks the guidance of domain knowledge and is prone to over-fitting. Yin et al. [32] utilized the adjacency relations as a prior to divide all nodes into different sets and learned the global spatial correlation. VGRAN [33] proposed by Zhou et al. extended the GNNs with uncertainty of node latent representations, which overcame the problem of deterministic vector embedding in previous work. Fang et al. [34] constructed the graph of the road network at the turn level, and improved the accuracy of traffic information detection by estimating the queue starting point.

For the extraction of temporal features, RNN-based models are widely used [35]. Yao et al. [36] used node features and external information as input to **long short-term memory** (**LSTM**) to extract temporal dependencies. Wang et al. [37] proposed a Periodic-Skip LSTM, which is different from LSTM sequential input structure. In order to better model the periodicity, the authors choose to periodically skip irrelevant inputs to reduce the noise caused by useless inputs. Wang et al. [38] combined a recurrent network and a Transformer layer to capture the local and global temporal dependence. However, a RNN model is time-consuming in training process, not capable of capturing dynamic changes, and is sensitive to cause error propagation during multi-step prediction. Zhang et al. [39] proposed a hierarchically structured graph neural architecture for exploring the multi-resolution traffic transitional information, which we also considered in our proposed multivariate self-attention **Temporal Convolution Network** (**TCN**).

2.2 GNNs

GNNs are building blocks for learning graph-structure data. They are widely used in node classification [40], link prediction [41], graph clustering [42], and other graph theory problems [43, 44]. Advances in this direction are often categorized as spectral approaches and spatial approaches [45]. Spectral approaches smooth a node's input signals using graph spectral filters depending on the Laplacian eigenbasis. However, the spectral-based method needs to load the entire graph into memory during the operation. It is not efficient for large-scale graph, and even cannot be trained when resources are limited. From a spatial-based method, it extracts a node's high-level representation by aggregating feature information from neighbors. There have been many variants of GNN with various kinds of aggregators and updaters proposed these days, like GraphSage [46] and LGCN [47]. However, these methods do not design the aggregation method between the layers of GNN. The output of the previous GNN layer is usually the input of the latter, which makes the useful elementary representation information gradually forgotten.

According to past studies, we find that GNN is very shallow and two-layer GNN is usually used in most applications. As shown in [48, 49], stacking multiple GNN layers will result in an over-smoothing problem. That is to say, all vertices will converge to the same value. Part of the reason is that the simple cascade of GNNs loses useful representation. Hence, Xu et al. [50] paid attention to this issue and proposed the Jumping Knowledge Network which could learn structure-aware representations based on all stacking layer outputs. However, this is still one of the core problems that GNN currently faces and deserves more extensive research. In this article, we have studied how to aggregate multi-layer GNNs, and the performance of the proposed adaptive gating mechanism exceeds Jumping Knowledge Network.

3 PROBLEM FORMULATION

Traffic flow forecasting is a classical spatiotemporal prediction problem given previously observed traffic flow data in the city. Here, traffic flow refers to observations over a traffic system that can be reported in numeric values. To illustrate this point, we will focus on the prediction of traffic speed in the article, although our model can be applied to the prediction of other numerical traffic data.

Suppose there are N roads in the area, and we represent each road section as a node in a traffic graph $\mathcal{G} = (V, E_{sp}, E_{se})$, where V is a set of road nodes with $|V| = N$. E_{sp} and E_{se} are two sets of edges. E_{sp} represents the connectivity between spatial neighbors, and E_{se} represents the correlation between the semantic neighbors. Meanwhile, the adjacency matrices derived from the aforementioned two kinds of edges are denoted by $\mathbf{A}_{sp} \in \mathbb{R}^{N\times N}$ and $\mathbf{A}_{se} \in \mathbb{R}^{N\times N}$. Figure 3 shows the nodes and two types of edges in the traffic graph modeling. The nodes in subgraphs $G_{sp}(V, E_{sp})$ and $G_{se}(V, E_{se})$ are the same and connected with dotted lines. The solid lines within each subgraph represent the edges defined by road connectivity or contextual similarity. Since many roads have two-way lanes, the traffic flow of the two opposite lanes is often not the same. Therefore, we treat two-way lanes as two nodes in the graph for analysis.

Denote the traffic flow observed on $\mathcal{G}$ at time t as a graph signal $\mathbf{X}^t \in \mathbb{R}^{N\times P}$, where P is the number of features of each node. Suppose we have T historical graph signals, and we want to predict future M graph signals. The urban traffic flow forecasting is formulated as learning a function $f(\cdot)$ given a traffic graph $\mathcal{G}$:

$$[\mathbf{X}^{t-T+1}, \ldots, \mathbf{X}^t; \mathcal{G}] \xrightarrow{f(\cdot)} [\mathbf{X}^{t+1}, \ldots, \mathbf{X}^{t+M}]. \tag{1}$$

Two kind of edges represent two neighbor relationships, and thus form two subgraphs, including (1) Spatial Neighbor Subgraph $G_{sp}(V, E_{sp})$, which encodes the local connectivity of adjacent roads, and (2) Semantic Neighbor Subgraph $G_{se}(V, E_{se})$, which encodes the global contextual similarity of roads by the **Dynamic Time Warping** (**DTW**) Algorithm [51]. The DTW algorithm, which is originally for speech recognition, is used to measure the similarity between two temporal sequences by calculating an optimal match. By expanding the definition of neighbors, more spatial features can be aggregated to improve the accuracy of prediction. At the same time, it can make up for the occurrence of information islands by making more connections, because some nodes may have a quite low degree under single neighborhood definition.

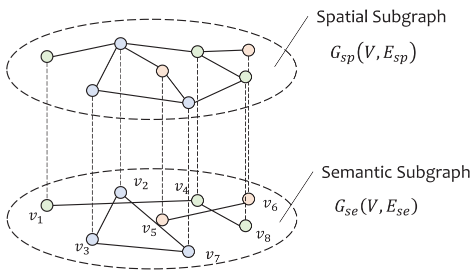


Fig. 3. Spatial dependency modeling for urban traffic flow forecasting: spatial neighbors and semantic neighbors.

3.1 Spatial Subgraph

The connection of roads is an important factor in spatial dependency modeling. If road v_i and road v_j share the same traffic intersection, we call them spatial neighbors and all the spatial neighbors form the spatial neighbor subgraph $G_{sp}(V, E_{sp})$. A^{sp}_{ij} in the adjacency matrix $\mathbf{A}_{sp}$ is equal to 1. Otherwise, it is equal to 0.

$$A^{sp}_{ij} = \begin{cases} 1, & v_i \text{ and } v_j \text{ share the same intersection,} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In a traffic network, each traffic intersection causes traffic flow to either diverge or converge. In previous research, the Euclidean distance of spatial location was commonly used to characterize the correlation of different roads as edge weight. However, using a fixed Euclidean distance to characterize the correlation is inaccurate. On one hand, although some road nodes are close, the interaction between two roads is weak, such as two opposite lanes. On the other hand, the correlation of roads should change dynamically with the input, which means the weights of the edges should be different at different times. Therefore, we use the multi-head attention mechanism to calculate the attention score between connected nodes as the dynamic weight of the edges based on the input at different moments, which will be explained in detail in Section 4.2.1.

3.2 Semantic Subgraph

When we predict the traffic of a certain road section, not only is it directly related to the connectivity, but roads with high contextual similarities also should be considered. For example, the roads near two commercial areas have close similarities. Although they are geographically far apart, they are similar in characteristics, and they can be called semantic neighbors. To capture the changes in speed signatures, we use the DTW algorithm to calculate the similarity of two time series. The reason why we choose DTW distance as the metric is that we pay more attention to finding the cause and effect of the traffic flow changes between two speed signatures rather than traffic flow itself. Consider, for example, the two speed signatures V_A and V_B in Figure 4. Two signatures have a strong similarity, but they are not aligned on the time axis. The DTW algorithm calculates the distance by finding a suitable matching pair according to the signature features. For example, at time t_i, the DTW distance pair, $v_A(t_i)$ and $v_B(t_j)$, obtained by the DTW algorithm have a stronger similarity compared to the Euclidean distance pair, $v_A(t_i)$ and $v_B(t_i)$.

Accordingly, we define the semantic neighbor subgraph $G_{se}(V, E_{se})$ according to the DTW distance of the speed signatures among the roads. We use the average weekly traffic flow series as the speed signatures to calculate the DTW distance. A^{se}_{ij} represents the element in semantic neighbor

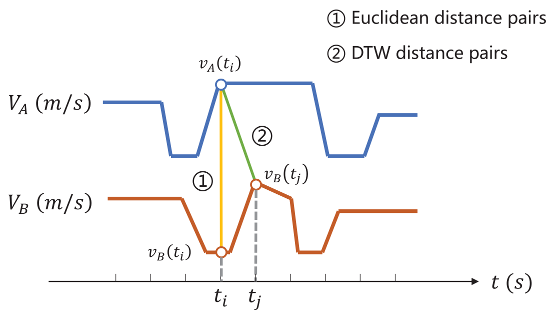


Fig. 4. Example of the Euclidean vs. the DTW distance between two speed signatures.

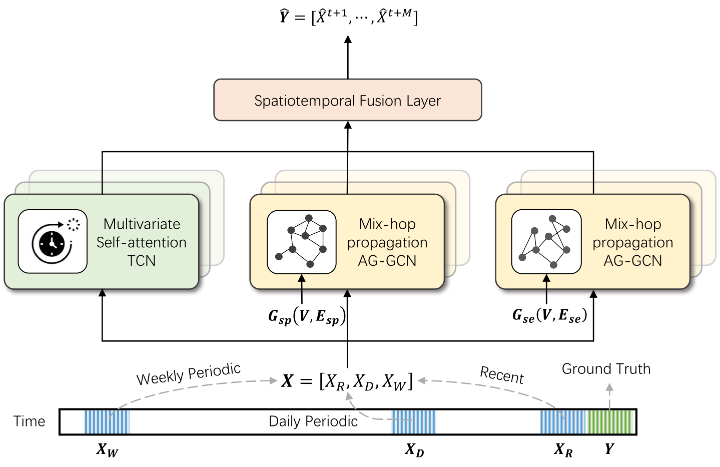

$\hat{Y} = [\hat{X}^{t+1}, \cdots, \hat{X}^{t+M}]$
Spatiotemporal Fusion Layer
Multivariate Self-attention TCN
Mix-hop propagation AG-GCN
Mix-hop propagation AG-GCN
$G_{sp}(V, E_{sp})$
$G_{se}(V, E_{se})$
Weekly Periodic
$X = [X_R, X_D, X_W]$
Recent
Ground Truth
Time
Daily Periodic
X_W
X_D
X_R
Y

Fig. 5. Model overview of spatiotemporal adaptive gated graph convolution network (STAG-GCN).

adjacency matrix $\mathbf{A}_{se}$, and can be calculated according to the following equation:

$$A_{ij}^{se} = \begin{cases} 1, & \mathrm{DTW}(v_i, v_j) > \epsilon, \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

where ϵ is the threshold to determine the sparsity of adjacency matrix $\mathbf{A}_{se}$. $\mathbf{A}_{se}$ indicates whether the nodes are semantic neighbors and we also use a multi-head attention mechanism to derive the dynamic weights of edges.

4 METHODOLOGY

Figure 5 shows the framework of our model, STAG-GCN. It mainly consists of three components: **Multivariate Self-attention Temporal Convolution Network** (**Multivariate Self-attention TCN**), **Mix-hop Adaptive Gated Graph Convolution Network** (**Mix-hop AG-GCN**) and Spatiotemporal Fusion Layer. These modules will be introduced in detail below. The input of different components can be divided into three parts: recent, daily periodic and weekly periodic dependencies of the historical data.

Suppose the sampling frequency of daily traffic is q minutes. As shown in Figure 5, we intercept three time series segments along the time axis as the input of the recent, daily periodic, and weekly periodic component, respectively, denoted as X_W, X_D, and X_R. The recent segment X_R is a

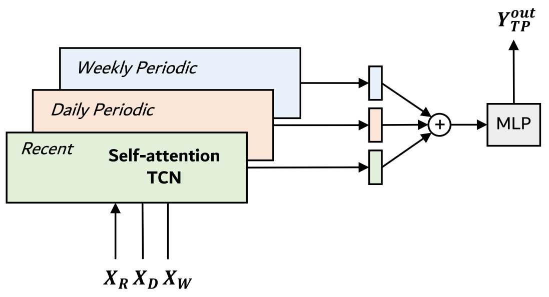


Fig. 6. Multivariate self-attention TCN.

historical observation directly adjacent to the predicting period Y. Intuitively, the characteristics of road conditions are continuous, and recent road information plays a vital role in the accurate traffic flow prediction. Daily-periodic segment X_D records the same time period as the predicting period of the past day. Due to the regular daily routine of people, traffic data may show repeated patterns, such as daily rush hours. Therefore, the daily-periodic component is a traffic flow data worthy of reference. Weekly periodic segment X_W is composed of the segment on last week, which has the same time intervals as the forecasting period. Usually, the road conditions are often very cyclical, and the road characteristics of this Wednesday are very similar to those of last Wednesday. Therefore, weekly periodic segment X_W is used to dig out periodic features of traffic flow data.

4.1 Multivariate Self-Attention TCN

In this part, we present the structure of self-attention TCN and how to fuse the multivariate time series results. As shown in Figure 6, we send the input of three time segments, X_W, X_D, and X_R, into the self-attention TCN to extract different features in temporal domain, respectively, and then concatenate the results together into a **Multi-Layer Perceptron** (**MLP**) to obtain the time dimension features:

$$Y_{TP}^{out} = MLP\left(\mathcal{Y}_{tcn}^{R} \oplus \mathcal{Y}_{tcn}^{D} \oplus \mathcal{Y}_{tcn}^{W}\right), \tag{4}$$

where $\oplus$ means the concatenation of multivariate temporal domain features. $\mathcal{Y}_{tcn}^{R}$, $\mathcal{Y}_{tcn}^{D}$, and $\mathcal{Y}_{tcn}^{W}$ is the self-attention TCN output of X_R, X_D, and X_W. Through this method, the similarity and periodicity at different times can be effectively extracted and fully integrated, so that the mining in the temporal dimension is more sufficient and effective. Then we will introduce the specific structure of self-attention TCN in Figure 7 in detail.

Although RNN-based models, like LSTM and GRU, become widespread in time-series analysis, recurrent network still suffers from time-consuming iterations, unstable gradient, and slow response to dynamic changes. Meanwhile, attention mechanism has become an integral part of compelling sequence modeling and transductive models in various tasks. Especially in the Transformer [52] structure proposed by Google, the multi-head attention mechanism is used extensively, and it has achieved remarkable results in the translation tasks. Therefore, we propose a novel **Self-attention Temporal Convolution Network** (**Self-attention TCN**) to capture both local and long-range temporal dependencies.

In Self-attention TCN, we first use Multi-Head Self-attention to jointly attend to information from different representation subspaces at different historical observations. Mathematically, for a single-head attention model, given input $\mathbf{X}^{t-T+1:t} = [\mathbf{X}^{t-T+1}, \ldots, \mathbf{X}^{t}] \in \mathbb{R}^{T\times N\times P}$, simplifying the notation as $\mathbf{X}$, three subspaces are obtained, namely, query subspace $Q \in \mathbb{R}^{N\times d_q}$, key subspace

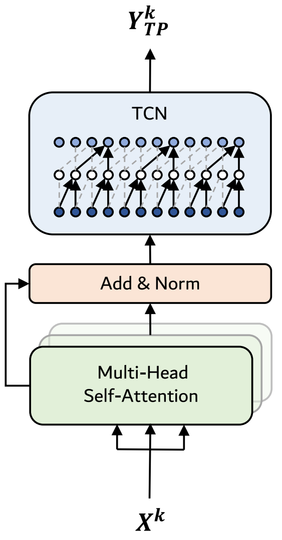


Fig. 7. Self-attention TCN.

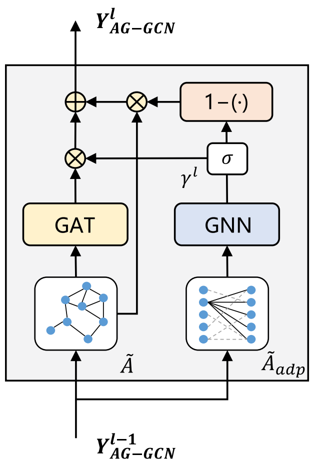


Fig. 8. Adaptive gated graph convolution network.

$K \in \mathbb{R}^{N\times d_k}$, and value subspace $V \in \mathbb{R}^{N\times d_v}$. The latent subspace learning process can be formulated as

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V. \tag{5}$$

Scaled Dot-product attention is used to calculate the attention output:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{6}$$

Furthermore, when we use more heads to jointly attend to input from different representation subspaces, we will get richer latent information. Concatenate the heads together and project again to get the final values:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, \ldots, head_h)W^O, \\ \text{where } head_i &= Attention(Q_i, K_i, V_i), \end{aligned} \tag{7}$$

where $Q_i = XW_i^Q$, $K_i = XW_i^K$, $V_i = XW_i^V$, and $W^O \in \mathbb{R}^{hd_v\times d_{model}}$. After the Multi-Head Self-attention, we employ a residual connection around two sub-layers, followed by a layer normalization. The multi-head self-attention mechanism is good at capturing the long-range dependencies of time series and the correlation within the data or feature, which is equivalent to a preliminary and efficient embedding of the series.

In order to consider both local and long-range temporal dependencies simultaneously, the output of the multi-head self-attention network is input to a dilated TCN stacking multiple layers. The TCN result of node v_i on the p-th channel at time t is $y_{i,t,p}$, which can be calculated as follows:

$$y_{i,t,p} = \sum_{k=1}^{K_\tau}\sum_{m=1}^{M} w_{k,m,p} \cdot x_{i,t-d(k-1),m}, \tag{8}$$

where d is the dilation rate, and $w_{k,m,p}$ is the element of the convolution kernel. Moreover, all the elements of $w_{k,m,p}$ constitute the temporal convolution kernel $\mathcal{W} \in \mathbb{R}^{K_\tau\times M\times P}$, where K_τ denotes the kernel length, and P represents the number of output channels. In the process, zero-padding strategy is utilized to keep the time series length unchanged. Applying the same convolution kernel to all nodes in graph yields the following formulation:

$$\mathcal{Y}_{tcn} = \mathcal{W} *_d \mathcal{X}. \tag{9}$$

Multi-layer dilated TCN stacking can not only expand the receptive field on the temporal axis, but also obtain multi-resolution output. To further expand the receptive field, the dilation rate increases with an exponential speed, i.e., $d^l = 2^{l-1}$. The output of self-attention TCN on the l-th layer is $\mathcal{Y}_{tcn}^l \in \mathbb{R}^{N \times T \times P_{out}}$, which can be calculated as follows:

$$\mathcal{Y}_{tcn}^l = \begin{cases} \mathcal{X}, & l = 0, \\ \sigma(\mathcal{W}^l *_{d^l} \mathcal{Y}_{tcn}^{l-1}), & l = 1, 2, \ldots, L, \end{cases} \tag{10}$$

where $\mathcal{X} \in \mathbb{R}^{N \times T \times P_{in}}$ is the output of the multi-head self-attention network, $\mathcal{W}^l \in \mathbb{R}^{K_\tau \times M \times P_{out}}$ is the dilated TCN kernel, and $\sigma(\cdot)$ is the non-linear function.

4.2 Mix-hop AG-GCN

In order to capture complex and dynamic spatial relationships from traffic flow data, mix-hop AG-GCN, as shown in Figure 8, uses a multi-head graph attention mechanism for each layer to construct dynamic edge weights. A novel adaptive graph gating mechanism is utilized to selectively update and forget high-order neighbor information of nodes within multi-layer stacking. The output layer of each AG-GCN adopts mix-hop propagation to fully aggregate the effective hidden information of each AG-GCN layer. There are two mix-hop AG-GCN modules in our model. The difference between them is that the spatial dependencies are different. One is based on the spatial neighbors, and the other is based on semantic neighbors, but their main structures are consistent.

GCN is an efficient operation to extract the features of a node through its structural information. Let $\widetilde{A} \in \mathbb{R}^{N \times N}$ denote the normalized adjacency matrix with self-loop, $X \in \mathbb{R}^{N \times D}$ denotes the input graph signal, $Z \in \mathbb{R}^{N \times M}$ represents the output, $W \in \mathbb{R}^{D \times M}$ is the model parameter matrix, and the basic graph convolution layer is defined as

$$Z = \widetilde{A} X W. \tag{11}$$

In our module, two different graph convolution methods are used and the results of the two graph convolutions are integrated using an *adaptive graph gating mechanism* to achieve selective updating and forgetting of knowledge.

4.2.1 Multi-Head Graph Attention Network. According to the spatial modeling dependencies, we can get the graph $G_{sp}(V, E_{sp})$ or $G_{se}(V, E_{se})$, and once again use the multi-head graph attention mechanism [53] to dynamically generate the edge weights and effectively aggregate the neighborhood information. The input of Multi-head Graph Attention Network layer is a set of node features, $\mathbf{h} = \{h_1, h_2, \ldots, h_N\}$, $h_i \in \mathbb{R}^D$, and the layer produces a new set of node representation $\mathbf{h}' = \{h'_1, h'_2 \ldots, h'_N\}$, $h'_i \in \mathbb{R}^M$. As an initial step, a shared linear transform is applied to every node and a shared attention mechanism a is applied to compute the attention coefficients between two neighbor nodes v_i and v_j in a graph. The attention coefficients

$$e_{ij} = a(Wh_i, Wh_j),\ j \in \mathcal{N}_i \tag{12}$$

indicate the importance of node j's feature to node i, where the weight matrix is $W \in \mathbb{R}^{D \times M}$, the attention mechanism is $a : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$, and $\mathcal{N}_i$ is a set of neighbor nodes of node v_i. To make coefficients easily comparable across different nodes, we normalize them across all neighbor choices of j using the $softmax$ function:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k \in \mathcal{N}_i} exp(e_{ik})}. \tag{13}$$

In order to obtain more abundant representation, K independent attention mechanisms execute the transformation, and concatenate to achieve the result.

$$h_i' = \overset{K}{\underset{k=1}{\|}} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} W^k h_j\right), \tag{14}$$

where $\|$ represents concatenation. Generally, the last layer of the multi-head attention mechanism can also employ averaging to get the output:

$$h_i' = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in \mathcal{N}_i} \alpha_{ij} W^k h_j\right). \tag{15}$$

When we transform the process of graph attention network to a matrix operation, it is

$$Z^l = (\widetilde{A} \odot M) Z^{l-1} W, \tag{16}$$

where $\widetilde{A} = A_{sp} + I$ or $\widetilde{A} = A_{se} + I$, $\odot$ is the element-wise product, and the elements in $M \in \mathbb{R}^{N \times N}$ are the dynamic attention coefficients. Thus, $\widetilde{A} \odot M$ constitutes the dynamic edge weights in spatial dependency modeling. $Z^l \in \mathbb{R}^{N \times M}$ is the l-th multi-head graph attention network output, and if $l = 0$, $Z^0 = X$. As a result, we dynamically calculate the weight of the node edge on the original basic graph convolution network, so as to better aggregate the node features.

4.2.2 *Adaptive Graph Gating Mechanism.*

GNNs can effectively represent nodes by aggregating neighbor information, but this network structure also brings some surprises for "deep" learning. Compared to the stacking of dozens of CNN layers in computer vision problems, most GNNs have the best results with two layers. When we increase the number of layers, in principle, it should be possible to aggregate higher-order neighbor information, but the overall performance is worse. Xu et al. [50] studied this issue and found that after passing through multi-layer GNN, the nodes with high degrees are prone to over-smoothing, while nodes with fewer degrees do not fully aggregate the features of the multi-order neighbors. In order to allow each node to be selectively updated and forgotten in each layer of the GNN, we design a novel Adaptive Graph Gating Mechanism.

A gating mechanism plays a vital role in the RNN variant, like LSTM and GRU, and has been proven to have a strong ability to control the flow of information. In the gating mechanism, the same network structure is often used but the activation functions passed are different. In our design, instead of A_{sp} or A_{se}, we use an adaptive adjacency matrix A_{adp} for the gating layer to perform GNN operations. This adaptive adjacency matrix is obtained by two learnable parameters $E_s, E_t \in \mathbb{R}^{N \times c}$, where we name E_s as the source node embedding and E_t as the target node embedding. We multiply E_s and E_t to get the adaptive spatial dependency, use the nonlinear activation function $ReLU$ to eliminate minor dependencies, and finally obtain the adaptive adjacency matrix A_{adp} through the $softmax$ function.

$$A_{adp} = softmax(ReLU(E_s E_t^T)). \tag{17}$$

The reason why A_{sp} or A_{se} is not used as the adjacency matrix in the gating mechanism is because no matter how delicately we design the spatial dependencies, A_{sp} and A_{se} are set by us artificially, which is bound to have a certain gap with the complex spatial dependencies of the traffic network. Therefore, the adaptive adjacency matrix can compensate the information

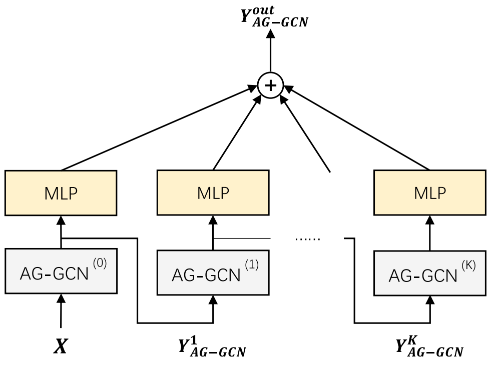


Fig. 9. Mix-hop propagation layer for AG-GCN.

deviation caused by artificial modeling, and further improve the accuracy of the model. Then we use the adaptive adjacency matrix to calculate the result of graph convolution and get the gated value γ.

$$\gamma = \sigma(A_{adp} X \Theta), \tag{18}$$

where X denotes the input signal, Θ denotes the adaptive graph convolution parameter matrix, and σ is the *sigmoid* activation function. We use the gated value γ to indicate how much information is updated in the next layer, and $1-\gamma$ to indicate how much information is forgotten in the next layer and use the result of the previous layer.

$$Y^l_{AG-GCN} = \begin{cases} X, & l = 0, \\ \gamma^l \odot g(Z^l) + (1-\gamma^l) \odot Y^{l-1}_{AG-GCN}, & l = 1, 2, \ldots, L. \end{cases} \tag{19}$$

$g(\cdot)$ is the nonlinear function, like *ReLU*, *tanh*, and so on. In this way, the selective update and forgetting of the multi-layer GNN can be achieved, and each node can obtain a more sufficient and reasonable high-dimensional feature.

4.2.3 Mix-Hop Propagation Layer. AG-GCN is a cascaded network structure; as the number of network layers increases, nodes can aggregate the features of higher-order neighbors. Therefore, the output of each layer of AG-GCN actually represents information perception of different depths. We propose a mix-hop propagation layer, as shown in Figure 9, for information selection and aggregation. In this layer, we use a **parameter-shared** MLP as trainable aggregation parameters to mix the information of multi-hop neighbors. In particular, we also aggregate the initial node features, which is also a manifestation of the idea of residual connection. Specifically, the definition of mix-hop propagation is as follows:

$$Y^{Out}_{AG\text{-}GCN} = \sum_{i=0}^{K} Y^i_{AG\text{-}GCN} W_i, \tag{20}$$

where K is the depth of propagation, which is also equal to the number of layers of AG-GCN. $Y^i_{AG\text{-}GCN}$ is the output result of each layer of AG-GCN and $Y^0_{AG\text{-}GCN}$ is the input X for unified representation. Through this method, it propagates information horizontally and selects information vertically.

Through the above three mechanisms, *Multi-head Graph Attention Network*, *Adaptive Graph Gating Mechanism*, and *Mix-hop propagation layer*, we get the complete algorithm of mix-hop AG-GCN. By selecting the adjacency matrix, A_{sp} or A_{se}, we can extract the spatial and semantic features from the traffic flow data.

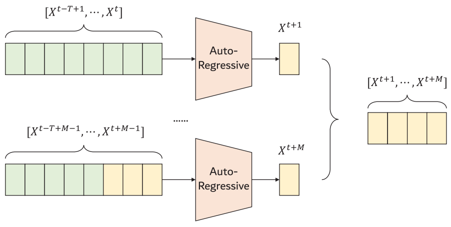

$[X^{t-T+1},\cdots,X^{t}]$
Auto-Regressive
X^{t+1}
$[X^{t+1},\cdots,X^{t+M}]$
......
$[X^{t-T+M-1},\cdots,X^{t+M-1}]$
Auto-Regressive
X^{t+M}

Fig. 10. Auto-regressive multi-step prediction.

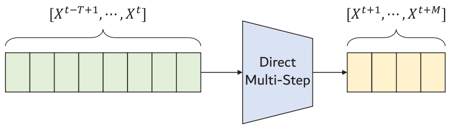


Fig. 11. Direct multi-step prediction.

4.3 Spatiotemporal Fusion Network

After we get the high-dimensional features in the temporal, spatial, and semantic domain, how to aggregate the spatiotemporal information for multi-step prediction is a problem we need to consider. Our model permits general module-aggregation mechanisms. We explore four mechanisms: *concatenation*, *max-pooling*, *LSTM-attention*, and *Weighted-sum*. We will compare these methods in the experiment part. Let $h_i^{tp}, h_i^{sp}, h_i^{dtw}$ be the three spatiotemporal representation of node v_i that are to be aggregated.

Concatenation: A concatenation $[h_i^{tp}, h_i^{sp}, h_i^{dtw}]$ is the most direct way to combine the three parts and can preserve the original spatiotemporal features as completely as possible.

Max-pooling: An element-wise $max(h_i^{tp}, h_i^{sp}, h_i^{dtw})$ selects the most informative representation for each feature coordinate. Max-pooling does not introduce any parameters to learn, and it is widely used in the field of image processing and has achieved good results.

LSTM-attention: A LSTM-based attention mechanism identifies the most useful information for each node v by computing an attention score for three module outputs. For LSTM-attention, $h_i^{tp}, h_i^{sp}, h_i^{dtw}$ will be first input to a bi-directional LSTM. Then we generate the forward-LSTM and backward-LSTM hidden features and use linear transformation to calculate the attention score. Finally, we use the $softmax$ function to yield the attention of node v on three modules and get the weighted-sum final representation.

Weighted-sum: In addition to the several methods mentioned above, using the attention mechanism for weighted summation is also an effective way to aggregate features. Compared with the direct average summation of features, the attention mechanism can discover the importance score among $h_i^{tp}, h_i^{sp}, h_i^{dtw}$ in prediction, so as to achieve the optimal summation. In the model, we use MLP to get an attention score, and use the $softmax$ function to normalize to get the final weight.

When we get the high-dimensional spatiotemporal features, we make a multi-step prediction through a linear layer. One prediction method is an auto-regressive prediction method, as shown in Figure 10. Error-prone predictions are used as input with previous observations for further predictions, resulting in rapid error accumulation, especially for long-term predictions where the error will continue to rise. Therefore, we adopt the same method as Wu et al. [54] to directly predict the future M steps, as shown in Figure 11, to avoid the error dispersion caused by inaccurate prediction. To achieve this, we set the number of output channels of the linear layer as a factor of step length M to get our desired output $\mathbf{Y}'$ and use ***Mean Square Error** (MSE) loss* to train the model, which can be formulated as

$$L(\mathbf{Y}, \mathbf{Y}') = \frac{1}{n} \sum_{i=1}^{n} ||Y_i - Y_i'||^2, \tag{21}$$

where $\mathbf{Y} \in \mathbb{R}^{n \times N \times M}$ is the ground truth, $\mathbf{Y}' \in \mathbb{R}^{n \times N \times M}$ is the prediction, and n is the batch number.

Table 1. Details of the Datasets

Tasks	DiDi_Chengdu[M]	DiDi_Chengdu[L]	METR-LA
Time span	1/1/2018–4/1/2018	1/1/2018–4/1/2018	3/1/2012–6/30/2012
Time interval	10 min	10 min	5 min
# timestamps	17,280	17,280	34,272
# nodes	524	1,343	207

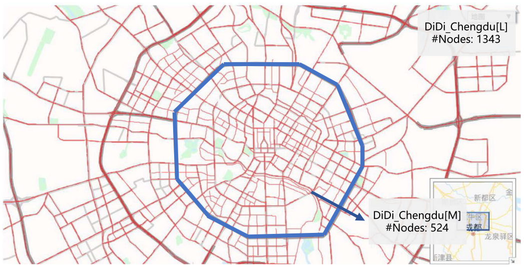


Fig. 12. Road map visualization of DiDi_Chengdu datasets.

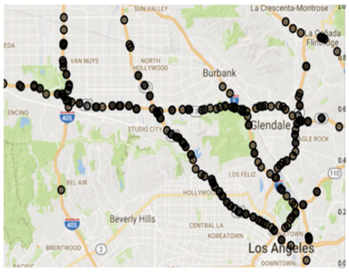


Fig. 13. Sensor networks of METR-LA.

5 EXPERIMENT

5.1 Datasets

We evaluate the performance of the proposed model, STAG-GCN, on two real-world datasets with different network scales: **DiDi_Chengdu** [55] and **METR-LA** [25] shown in Table 1.

— **DiDi_Chengdu:** Traffic index dataset of Chengdu, China, provided by Didi Chuxing GAIA Initiative. We visualize the road map in the dataset in Figure 12. In the experiment, we select data from January to April 2018 and divide it into two datasets based on geographical location. One is the medium dataset, DiDi_Chengdu[M], and the other is the large dataset, DiDi_Chengdu[L]. The DiDi_Chengdu[M] dataset contains 524 roads in the core urban area of Chengdu, while the DiDi_Chengdu[L] dataset contains 1,343 roads in a wider area. The data collection interval is 10 minutes.

— **METR-LA:** Traffic data are collected from observation sensors in the highway of Los Angeles County, as shown in Figure 13. We use 207 sensors and 4 months of data dated from 1st Mar 2012 until 30th Jun 2012 in the experiment. The readings of the sensors are aggregated into 5-minute windows.

5.2 Experiment Settings

In both of those datasets, we apply Z-Score normalization for data preprocessing. 70% of data are used for training, 20% are used for testing, while the remaining 10% are used for validation. The missing values are filled by the linear interpolation.

We compare our model, STAG-GCN, with the following widely used time series regression models and deep learning models in terms of traffic predictions.

— **HA:** Historical Average, which formulates the traffic flow as a seasonal process, and uses the average of previous seasons as the prediction.

— **ARIMA:** Auto-Regressive Integrated Moving Average model, which is widely used in time series prediction.

— **FNN:** Feed-forward neural network with two hidden layers and L2 regularization.

— **LSTM-FC:** The Encoder-Decoder framework using fully connected LSTM units.
— **STGCN:** Spatial-temporal graph convolution network [27], which combines graph convolution with 1D convolution.
— **DCRNN:** Diffusion convolution recurrent neural network [25], which combines GCNs with RNNs in an encoder-decoder manner.
— **ASTGCN:** Attention-based spatial-temporal GCN [56], which designs a spatial-temporal attention mechanism to capture the dynamic spatial-temporal correlations.
— **Graph WaveNet:** A convolution network architecture [54], which introduces a self-adaptive graph to capture the hidden spatial dependency, and uses dilated convolution to capture the temporal dependency.

We implement our model, STAG-GCN, based on the PyTorch framework. MSE between the prediction and ground truth is used as the loss function and minimized by back-propagation. Totally, there are five hyperparameters in our model, i.e., the threshold to determine the sparsity of semantic adjacency matrix ϵ, the number of TCN layer L_{TCN}, the number of AG-GCN layer $L_{AG\text{-}GCN}$, the number of attention head K, and the dimension of hidden representation of AG-GCN $d_{AG\text{-}GCN}$. We tune these parameters on the validation dataset, and observe the best performance on the setting of $\epsilon = 0.85$, $L_{TCN} = 2$, $L_{AG\text{-}GCN} = 3$, $K = 4$, $d_{AG\text{-}GCN} = 36$. During the training phase, we adopt ADAM optimizer for 100 epochs with batch size as 64. The initial learning rate is 1e-2 with a decay rate of 0.6 per 20 epochs. In addition, L2 normalization with a weight decay of 2e-4 is applied for better generalization. All the evaluated models are implemented on a server with two CPUs (Intel Xeon E5-2630 × 2) and four GPUs (NVIDIA GTX 1080 × 4).

In order to fully verify the performance of our model, we forecast the urban traffic flow in the next 10 minutes, 30 minutes, and 60 minutes separately in the DiDi_Chengdu dataset. For the public dataset METR-LA, we predict the road conditions for the next 15 minutes, 30 minutes, and 60 minutes in the usual way of previous researchers. In addition, we apply the following three widely used metrics for evaluation:

— **Mean Absolute Error** (**MAE**):

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

— **Mean Absolute Percentage Error** (**MAPE**):

$$MAPE(y, \hat{y}) = \frac{100\%}{N} \sum_{i=1}^{N} \left| \frac{\hat{y}_i - y_i}{y_i} \right| .$$

— **Root Mean Squared Error** (**RMSE**):

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

5.3 Performance Comparison

Tables 2 and 3 show the comparison of different approaches for multi-step forecasting on two datasets. Our proposed model, STAG-GCN, obtains the superior results on both datasets. We observe that (1) STAG-GCN outperforms the traditional time-series methods (HA, ARIMA, and LSTM-FC) by a large margin, because these models only rely on historical records and overlook spatial features. (2) Both our model and ASTGCN consider the input as different time-series segments, and learn more about periodicity and similarity of the road condition. Compared with the

Table 2. Performance Comparison of STAG-GCN and Other Baseline Models on DiDi_Chengdu Datasets

Data	Models	10 min			30 min			60 min		
		MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
DiDi_Chengdu[M]	HA	3.89	13.42	5.61	3.89	13.42	5.61	3.89	13.42	5.61
	ARIMA	2.32	9.80	3.45	3.26	14.21	4.90	4.15	18.23	6.22
	LSTM-FC	2.37	11.30	3.45	2.52	11.96	3.68	2.91	13.66	4.23
	STGCN	2.22	9.94	3.18	2.67	12.67	3.90	3.05	14.65	4.46
	DCRNN	2.04	9.00	2.99	2.65	12.34	3.92	3.16	14.83	4.61
	ASTGCN	2.05	9.17	2.99	2.44	11.48	3.58	2.70	12.71	3.91
	Graph WaveNet	1.91	**8.43**	2.82	2.25	10.77	3.42	2.38	11.50	3.63
	STAG-GCN	**1.90**	8.45	**2.80**	**2.20**	**10.44**	**3.35**	**2.27**	**10.92**	**3.47**
DiDi_Chengdu[L]	HA	3.02	13.24	4.56	3.02	13.24	4.56	3.02	13.24	4.56
	ARIMA	2.44	9.00	3.73	3.41	12.92	5.19	4.36	16.67	6.65
	LSTM-FC	2.60	10.74	3.96	2.72	11.29	4.13	3.09	12.81	4.63
	STGCN	2.50	10.07	3.61	2.81	11.56	4.12	3.14	13.02	4.61
	DCRNN	2.27	9.22	3.49	2.61	10.97	4.04	2.78	11.78	4.32
	ASTGCN	2.20	8.60	3.28	2.66	10.74	3.96	2.95	11.98	4.38
	Graph WaveNet	**2.09**	8.03	**3.17**	2.62	10.64	4.00	2.95	12.23	4.53
	STAG-GCN	2.12	**7.86**	3.20	**2.50**	**9.43**	**3.75**	**2.65**	**9.98**	**3.96**

Table 3. Performance Comparison of STAG-GCN and Other Baseline Models on METR-LA Datasets

Data	Models	15 min			30 min			60 min		
		MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE	MAE	MAPE (%)	RMSE
METR-LA	HA	4.16	13.00	7.80	4.16	7.80	13.00	4.16	7.80	13.00
	ARIMA	3.99	9.60	8.12	5.15	10.45	12.70	6.90	13.23	17.40
	LSTM-FC	3.44	9.60	6.30	3.77	10.90	7.23	4.37	13.20	8.69
	STGCN	2.88	7.60	5.74	3.47	9.57	7.24	4.59	12.70	9.40
	DCRNN	2.77	7.30	5.38	3.15	8.80	6.45	3.60	10.50	7.60
	Graph WaveNet	2.69	**6.90**	**5.15**	**3.07**	8.37	6.22	3.53	10.01	7.37
	STAG-GCN	**2.67**	7.00	5.23	**3.07**	**8.26**	**6.15**	**3.50**	**9.93**	**7.24**

classic spatiotemporal prediction models DCRNN and STGCN, they have a greater improvement. (3) With respect to the recently proposed model Graph WaveNet, our model has a small improvement in several indicators on the 10-minute horizons in the DiDi_Chengdu dataset and 15-minute horizons in the METR-LA dataset; however, it realizes bigger enhancement on both 30-minute and 60-minute horizons, which indicates our model is more capable of long-term traffic flow prediction. We use the *t-test* to examine the long-term prediction performance of STAG-GCN rather than Graph WaveNet. The p-value is less than 0.01, which demonstrates the superiority of STAG-GCN in long-term prediction.

In order to more intuitively display the results of multi-step traffic flow prediction, Figures 14 and 15 show the ground truth and multi-step prediction results on the DiDi_Chengdu[M] dataset and the METR-LA dataset. We can find that our model can better capture the dynamics and variability for the short-term prediction, and many small fluctuation points have good prediction effects. For long-term forecasts, STAG-GCN can also better capture the trends and changes of road conditions. In addition, for the DiDi_Chengdu[M] dataset, we analyze the forecasts for each day of the week and the forecasts for 24 hours a day. Figure 16 shows the average prediction performance of different timesteps during a week. It can be found that both weekdays and weekends have good prediction results. For the sake of model robustness, we compare the average prediction errors of multi-step predictions per hour in Figure 17. It shows that there is a normal performance decline as the forecast time increases. However, due to the strong dynamics during rush hour, 60 minutes

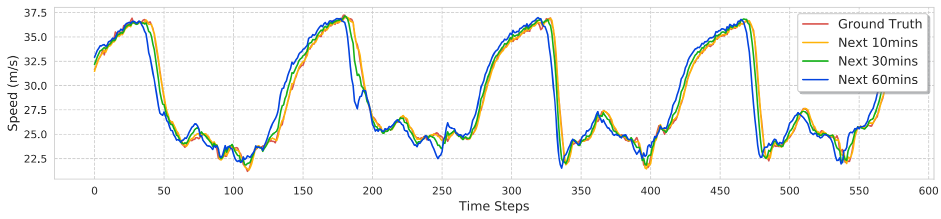


Fig. 14. Visualization of ground truth and multi-step prediction of the average speed on the DiDi_Chengdu[M] dataset.

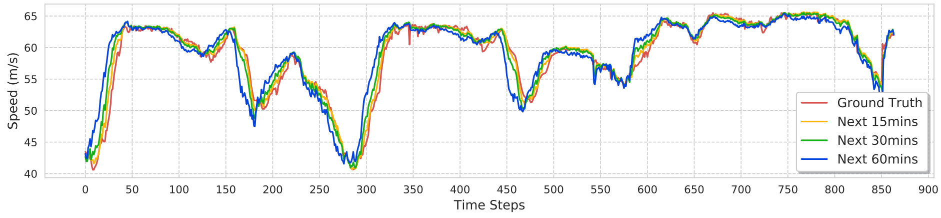


Fig. 15. Visualization of ground truth and multi-step prediction of the average speed on the METR-LA dataset.

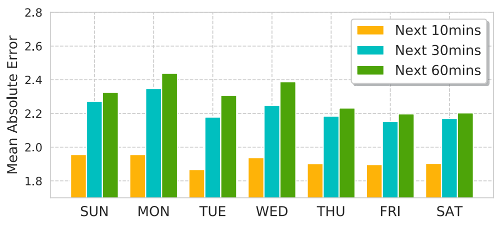


Fig. 16. Prediction error during 1 week on average.

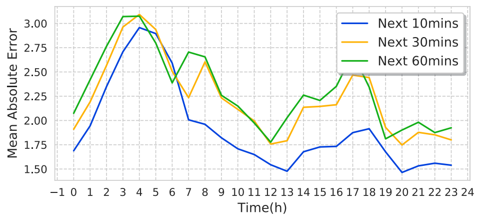


Fig. 17. Prediction error during 1 day on average.

Table 4. The Computational Cost Comparison on Two DiDi_Chengdu Datasets

Method	DiDi_Chengdu(M)		DiDi_Chengdu(L)	
	Training (s/epoch)	Inference (s)	Training (s/epoch)	Inference (s)
STGCN	34.898	3.815	68.229	7.816
DCRNN	199.480	25.378	691.100	78.407
ASTGCN	68.615	10.235	668.839	106.511
Graph WaveNet	57.697	3.719	112.365	5.114
STAG-GCN	37.380	5.046	66.801	7.889

forecasting performance has a significant drop. In addition, owing to less traffic and randomness in early morning, traffic flow forecasting from 2 a.m. to 5 a.m. is not easy to predict as well.

We present the computational cost of different models in Table 4. Our proposed model STAG-GCN has a superior performance on computational cost in both training and inference phases. Compared with our proposed model STAG-GCN, STGCN is competitive with us. Graph WaveNet shows its advantage in inference time, whereas STAG-GCN outperforms on the training time.

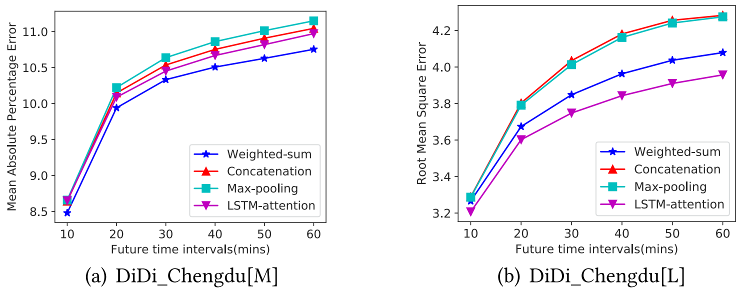


Fig. 18. Module aggregation mechanisms study.

Table 5. Computational Cost Comparison of Different Hyper-Parameter Threshold ϵ

ϵ	Training time (s/epoch)	Inference time(s)
0.5	58.817	7.338
0.75	17.622	2.385
0.8	14.604	2.163
0.85	12.247	1.989
0.9	12.213	1.886

5.4 Component Analysis

After we obtain a high-dimensional spatiotemporal representation, the output of each module will be sent to the fusion layer. As mentioned in Section 4.3, there are four different fusion mechanisms to aggregate multiple features. In order to verify the effects of different feature aggregation methods on short-term and long-term predictions, we conducted experiments on both the DiDi_Chengdu[M] and the DiDi_Chengdu[L] datasets. The experimental results are shown in Figure 18.

On the DiDi_Chengdu[M] dataset, *Weighted-sum* has achieved the best results. By learning the weights between different features through the multi-layer perceptron and then adding them, the importance of different features for the prediction task can be fully considered. On the DiDi_Chengdu[L] dataset, *LSTM-attention* has achieved the best results. Compared with *Weighted-sum*, the LSTM-based aggregation method is more complicated and assigns an attention score to different features. Its better performance is also due to the larger scale of the DiDi_Chengdu[L] dataset. Compared with the simple multi-layer perceptron, *LSTM-attention* learns more accurate aggregation results from high-dimensional features. However, on the two datasets, the *Max-pooling* and *Concatenation* methods have not achieved good results. Although *Max-pooling* can obtain the most informative features, it inadvertently loses some information, which causes performance degradation after several prediction steps. Although the direct concatenation retains the features to the greatest extent, the method of increasing parameters alone cannot achieve a better effect for multi-step prediction tasks.

The hyper-parameter threshold ϵ determines the sparsity of semantic neighbor. Table 5 and Figure 19 show the comparison of computational cost and RMSE performance. When $\epsilon = 0.85$, it has the best performance and moderate computational cost. When the threshold $\epsilon \in [0.8, 0.9]$, the difference of RMSE performance against each other is quite small. As we set the threshold to a quite small value, like $\epsilon = 0.5$, the computational cost has remarkable degradation.

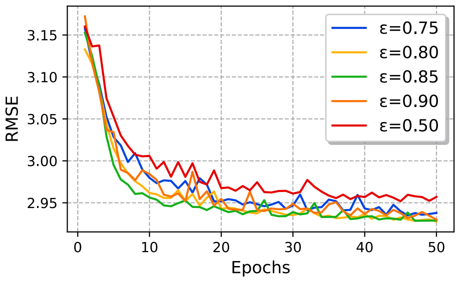
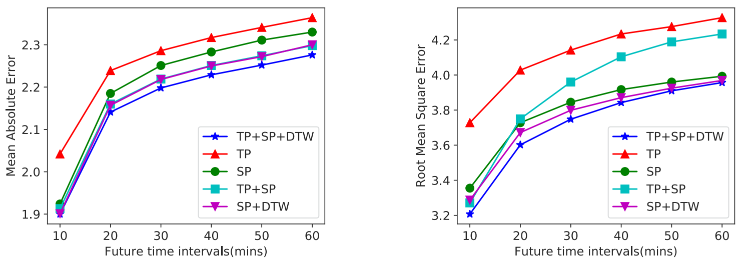


Fig. 19. The comparison of Root Mean Square Error (RMSE) in terms of hyper-parameter threshold ϵ.



(a) DiDi_Chengdu[M] (b) DiDi_Chengdu[L]

Fig. 20. Ablation study for three modules in STAG-GCN.

5.5 Ablation Study

In this section, we verify the effectiveness of each module in STAG-GCN through ablation study, and plot the experimental results in Figure 20. The main part of our model is divided into three parts: Multivariate Self-Attention TCN, mix-hop AG-GCN module for spatial neighbors, and mix-hop AG-GCN module for semantic neighbors. For the convenience of explanation and analysis, we will abbreviate three parts as **TP**, **SP**, and **DTW**. In order to analyze the role of each module in the model, we split and combine them to obtain the following sub-models: (1) TP+SP+DTW, (2) TP, (3) SP, (4) TP+SP, and (5) SP+DTW. We trained these sub-models on the DiDi_Chengdu dataset and performed multi-step prediction.

The model that only relies on temporal information (TP) for prediction is relatively stable in multi-step prediction, but the overall prediction performance is poor. The model only based on spatial features (SP) plays a key role in the prediction. Mix-hop AG-GCN effectively learns the features of mix-hop node neighbors. In ablation study, it is always better than the model that only captures temporal features. Especially when the number of nodes further increases, the advantages of the AG-GCN model are highlighted. When we consider multiple dimensional features (TP+SP, SP+DTW), the accuracy of prediction is often further improved, especially when we take the features of three dimensions (TP+SP+DTW) into account; the spatiotemporal aggregation layer will use neural network to obtain different attention weights and make the prediction to achieve the best performance. In particular, the performance of TP+SP on the DiDi_Chengdu[L] dataset is worse than SP alone, and SP+DTW almost achieves the same performance as TP+SP+DTW, because when the network structure expands, the rich neighbor information helps the model learn better features, while the temporal domain feature is less helpful.

Through the previous analysis, we found that the AG-GCN module plays a vital role in the model. To further verify the effectiveness of the AG-GCN module in extracting spatial correlations, we replace it with five other models:

Table 6. Ablation Study of Multivariate Self-Attention TCN and Mix-Hop Mechanism

Model	RMSE (10 min)	RMSE (30 min)	RMSE (60 min)
STAG-GCN	2.89	3.46	3.69
STAG-GCN+multivariate	2.85(−1.384%)	3.36(−2.890%)	3.48(−5.69%)
STAG-GCN+mix-hop	2.82(−2.422%)	3.41(−1.445%)	3.53(−4.33%)
STAG-GCN+multivariate+mix-hop	2.80(−3.114%)	3.35(−3.179%)	3.47(−5.96%)

— **GCN:** GCN with pre-defined spatial correlations.
— **GAT:** GAT with pre-defined spatial correlations.
— **Average:** GCN (adaptive spatial correlations) and GAT (pre-defined spatial correlations) are calculated in parallel and then added to get the average.
— **Gated:** GCN and GAT, both with pre-defined spatial correlations, are calculated in parallel and then get the result with the gating mechanism.
— **JKNet:** Jumping Knowledge Network [50] to aggregate GATs with pre-defined spatial correlations.

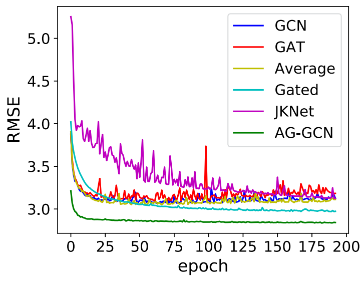
We conduct ablation experiments on the proposed multivariate self-attention TCN and mix-hop propagation layer. Table 6 shows the ablation experiments of different modules on the DiDi_Chengdu[M] dataset. The multivariate interceptions of recent, daily periodic and weekly periodic temporal segments effectively improve the performance of long-term forecasting. The mix-hop mechanism benefits the information aggregation between multiple AG-GCN layers. When combining two modules together, both the performance of short-term and long-term forecasting have been significantly improved. In particular, the long-term forecasting error has been reduced by 5.96%.

Figure 21 shows the RMSE performance of different models on the validation dataset during training. As shown in Figure 21(a), AG-GCN has a strong advantage over other models. In addition, in order to shield the effect of self-attention TCN, we remove it and train again. As shown in Figure 21(b), the **GAT** method that dynamically adjusts the edge weights is better than the **GCN** method with fixed adjacency matrix, which reflects the effectiveness of dynamic weighted graph modeling in our model. In the **AG-GCN** module, the output of the adaptive GCN acts as the gated value of GAT, so as to achieve selective updating and forgetting. When we simply add the two outputs in **Average** method, its performance is improved compared to **GCN** and **GAT**. However, there is still a large gap compared to **AG-GCN**, which highlights the superiority of the gating mechanism. The adaptive adjacency matrix in **AG-GCN** can correct the defects of artificially defined spatial correlations, and the performance has been further improved compared to the basic **Gated** method. With respect to previous state-of-the-art **JKNet**, **AG-GCN** achieves better results, proving the advantages of our design rationale.

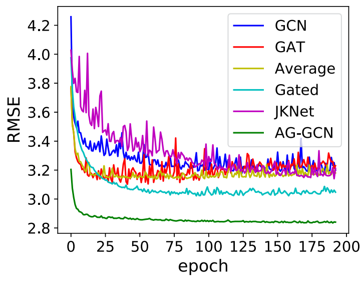
5.6 Model Explanation

In this section, we will analyze the interpretability of the STAG-GCN model. We apply a multi-head graph attention mechanism on dynamic weighted edges, and use an adaptive graph gating mechanism to select node updating and forgetting, which can provide explanations from different aspects.

5.6.1 Attentional Edge Weight. The dynamic attentional edge weight reflects the interaction between two connected road nodes. Figure 22 presents the geographical location of seven road nodes marked on Google Map and the heat map of attentional edge weight at 12 time slots. We mark the road section at the center as 1, and its neighboring roads are labeled 2–7.



(a) With Self-Attention TCN module



(b) Without Self-Attention TCN module

Fig. 21. Ablation study for AG-GCN: RMSE performance of different models on the validation dataset during training.

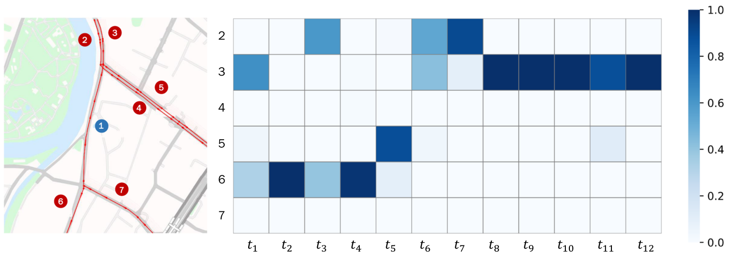


Fig. 22. Geographical location on Google Maps and the heat map of attentional edge weights on different time slots.

As can be seen, the relationships of different nodes are dynamic at different moments, which shows the necessity of dynamic weighted graph modeling compared with fixed graph once used in previous study. In addition, we find that when predicting the traffic flow of Road 1, Road 3 and Road 6 usually play a leading role, because these roads have a direct traffic flow convergence relationship with Road 1. Through the analysis of the attentional edge weights, we can analyze which surrounding roads have caused traffic congestion on a certain road, which provides insights for other intelligent transportation system tasks such as traffic signal control and lane adjustment.

5.6.2 Adaptive Adjacency Matrix. In the AG-GCN module, we use an adaptive adjacency matrix to calculate the gated value, thereby achieving selective updating and forgetting. The adaptive adjacency matrix can learn the node influence between other road nodes under the supervision of pre-defined spatial correlations and correct the information deviation caused by the artificially defined adjacency matrix. We choose first 30 nodes of two adaptive adjacency matrices and draw the heat map based on the gated values as shown in Figures 23 and 24. We select the gated value of column 4 for observation, and mark the nodes with higher gated values on Google Maps.

Figure 23 is the adaptive adjacency matrix of the AG-GCN module based on semantic neighbor, while Figure 24 is based on spatial neighbor. Since two AG-GCN modules extract high-dimensional features from two perspectives separately, the road nodes with higher gated values are quite different. We can find that because the AG-GCN module based on semantic neighbors hopes to aggregate roads with similar contextual information, the road nodes with high gated values are distributed at various locations on the map; while the road nodes with high spatial proximity are concentrated near the analysis road or directly connected to it, which indicates that the road conditions are strongly related to these roads nearby. The results of two completely different adaptive adjacency matrices show that AG-GCN can effectively learn under the guidance of a pre-defined spatial

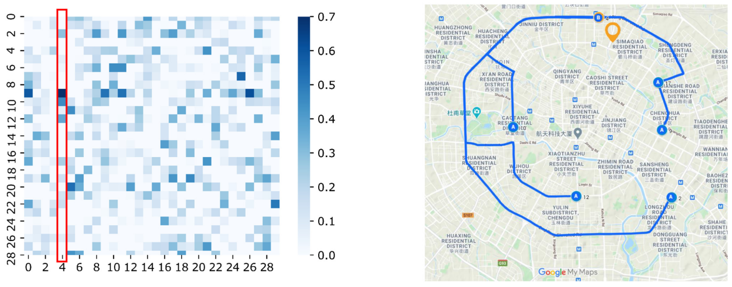


Fig. 23. The learned adaptive adjacency matrix of semantic neighbors and its geographical location marked on Google Maps.

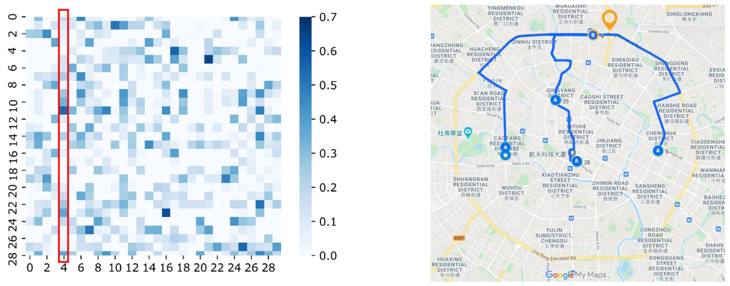


Fig. 24. The learned adaptive adjacency matrix of spatial neighbors and its geographical location marked on Google Maps.

relationship, correct the information deviation to obtain a more accurate adjacency matrix as a gate value to control the flow of information, and learned the correlation between different nodes.

6 CONCLUSION

In this article, we define the road network as a dynamic weighted graph by seeking both spatial neighbors and semantic neighbors of road nodes. A novelSTAG-GCN model is proposed for urban traffic flow forecasting. Multivariate Self-Attention TCN is utilized to extract multi-resolution temporal information. We propose the adaptive graph gating mechanism to improve the performance of multi-layer stacking GNN and mix-hop propagation layer to combine multi-layer stacking AG-GCN results. Experiments on two real-world datasets show that the performance of the proposed model is superior to existing models.

Actually, urban traffic prediction is affected by many external factors, such as social events and road regulations. In future work, we will study how to find anomalies such as traffic accidents and emergencies from traffic data. Furthermore, the proposed AG-GCN module can be generalized into dynamical graph features learning in various applications, and we can apply it to other pragmatic applications in the future.

REFERENCES

[1] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 865–873. http://dx.doi.org/10.1109/TITS.2014.2345663

[2] Zhibin Li, Pan Liu, Chengcheng Xu, Hui Duan, and Wei Wang. 2017. Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks. *IEEE Transactions on Intelligent Transportation Systems* 18, 11 (2017), 3204–3217.

[3] Isaac L. Johnson, J. Henderson, C. Perry, Johannes Schöning, and Brent J. Hecht. 2017. Beautiful...but at what cost?: An examination of externalities in geographic vehicle routing. *Proceedings of the ACM Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 2 (2017), 15:1–15:21.

[4] S. Vasantha Kumar and Lelitha Vanajakshi. 2015. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review* 7, 3 (2015), 21.

[5] Kranti Kumar, M Parida, and V. K. Katiyar 2013. Short term traffic flow prediction for a non urban highway using artificial neural network. *Procedia-Social and Behavioral Sciences* 104, 2 (2013), 755–764.

[6] Nicholas G. Polson and Vadim O. Sokolov. 2017. Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies* 79 (2017), 1–17.

[7] Marco Lippi, Matteo Bertini, and Paolo Frasconi. 2013. Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems* 14, 2 (2013), 871–882.

[8] Haiyang Yu, Zhihai Wu, Shuqin Wang, Yunpeng Wang, and Xiaolei Ma. 2017. Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks. *Sensors* 17, 7 (2017), 1501.

[9] Cen Chen, Kenli Li, Sin G. Teo, Xiaofeng Zou, Kang Wang, Jie Wang, and Zeng Zeng. 2019. Gated residual recurrent graph neural networks for traffic prediction. In *The 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, The 31st Innovative Applications of Artificial Intelligence Conference (IAAI 2019), and The 9th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI 2019)*. AAAI Press, 485–492.

[10] Cen Chen, Kenli Li, Sin G. Teo, Guizi Chen, Xiaofeng Zou, Xulei Yang, Ramaseshan C. Vijay, Jiashi Feng, and Zeng Zeng. 2018. Exploiting spatio-temporal correlations with multiple 3D convolutional neural networks for citywide vehicle flow prediction. In *IEEE International Conference on Data Mining (ICDM 2018)*. IEEE Computer Society, 893–898.

[11] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. 2017. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 17, 4 (2017), 818. http://dx.doi.org/10.3390/s17040818

[12] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, Satinder P. Singh and Shaul Markovitch (Eds.). AAAI Press, 1655–1661.

[13] Bin Lu, Xiaoying Gan, Haiming Jin, Luoyi Fu, and Haisong Zhang. 2020. Spatiotemporal adaptive gated graph convolution network for urban traffic flow forecasting. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, Virtual event. ACM, 1025–1034.

[14] Xinxin Feng, Xianyao Ling, Haifeng Zheng, Zhonghui Chen, and Yiwen Xu. 2019. Adaptive multi-kernel SVM with spatial-temporal correlation for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 20, 6 (2019), 2001–2013.

[15] Kunpeng Zhang, Zijian Liu, and Liang Zheng. 2020. Short-term prediction of passenger demand in multi-zone level: Temporal convolutional neural network with multi-task learning. *IEEE Transactions on Intelligent Transportation Systems* 21, 4 (2020), 1480–1490. http://dx.doi.org/10.1109/TITS.2019.2909571

[16] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'20)*, Virtual event, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 753–763.

[17] Chuanpan Zheng, Xiaoliang Fan, Chenglu Wen, Longbiao Chen, Cheng Wang, and Jonathan Li. 2020. DeepSTD: Mining spatio-temporal disturbances of multiple context factors for citywide traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2020), 3744–3755.

[18] G. E. P. Box and G. M. Jenkins. 2010. Time series analysis: Forecasting and control. *Journal of Time* 31, 3 (2010).

[19] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems* 5, 4 (2004), 276–281.

[20] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 777–785.

[21] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep multi-view spatial-temporal network for taxi demand prediction. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI'18), The 30th Innovative Applications of Artificial Intelligence (IAAI'18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI'18)*. AAAI Press, 2588–2595.

[22] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. AAAI Press, 1655–1661.

[23] Yuxuan Liang, Kun Ouyang, Lin Jing, Sijie Ruan, Ye Liu, Junbo Zhang, David S. Rosenblum, and Yu Zheng. 2019. UrbanFM: Inferring fine-grained urban flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining(KDD 2019).* ACM, 3132–3142.

[24] Junjie Ou, Jiahui Sun, Yichen Zhu, Haiming Jin, Yijuan Liu, Fan Zhang, Jianqiang Huang, and Xinbing Wang. 2020. STP-TrellisNets: Spatial-temporal parallel TrellisNets for metro station passenger flow prediction. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, Virtual Event. ACM, 1185–1194.

[25] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJiHXGWAZ.

[26] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.). ACL, 1724–1734. http://dx.doi.org/10.3115/v1/d14-1179

[27] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI'18)*, Jérôme Lang (Ed.). ijcai.org. 3634–3640. http://dx.doi.org/10.24963/ijcai.2018/505

[28] Mingqi Lv, Zhaoxiong Hong, Ling Chen, Tieming Chen, Tiantian Zhu, and Shouling Ji. 2021. Temporal multi-graph convolutional network for traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems* 22, 6 (2021), 3337–3348.

[29] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2019. GSTNet: Global spatial-temporal network for traffic flow prediction. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence(IJCAI'19)*, Sarit Kraus (Ed.). 2286–2293. http://dx.doi.org/10.24963/ijcai.2019/317

[30] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. GMAN: A graph multi-attention network for traffic prediction. In *The 34th AAAI Conference on Artificial Intelligence (AAAI'20).*

[31] Xiyue Zhang, Chao Huang, Yong Xu, and Lianghao Xia. 2020. Spatial-temporal convolutional graph attention networks for citywide traffic flow forecasting. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*, Virtual Event. ACM, 1853–1862.

[32] Xueyan Yin, Genze Wu, Jinze Wei, Yanming Shen, Heng Qi, and Baocai Yin. 2021. Multi-stage attention spatial-temporal graph networks for traffic prediction. *Neurocomputing* 428 (2021), 42–53.

[33] Fan Zhou, Qing Yang, Ting Zhong, Dajiang Chen, and Ning Zhang. 2021. Variational graph neural networks for road traffic prediction in intelligent transportation systems. *IEEE Transactions on Industrial Informatics* 17, 4 (2021), 2802–2812.

[34] M. Fang, L. Tang, X. Yang, Y. Chen, C. Li, and Q. Li. 2021. FTPG: A fine-grained traffic prediction method with graph attention network using big trace data. *IEEE Transactions on Intelligent Transportation Systems* (2021), 1–13. http://dx.doi.org/10.1109/TITS.2021.3049264

[35] Youngjoo Seo, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. 2018. Structured sequence modeling with graph convolutional recurrent networks. In *Proceedings of the 25th International Conference on Neural Information Processing (ICONIP'18), Part I*, Lecture Notes in Computer Science, Vol. 11301, Long Cheng, Andrew Chi-Sing Leung, and Seiichi Ozawa (Eds.). Springer, 362–373.

[36] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. 2019. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *The 33rd AAAI Conference on Artificial Intelligence (AAAI'19).* AAAI Press, 5668–5675. http://dx.doi.org/10.1609/aaai.v33i01.33015668

[37] Yuandong Wang, Hongzhi Yin, Hongxu Chen, Tianyu Wo, Jie Xu, and Kai Zheng. 2019. Origin-destination matrix prediction via graph convolution: A new perspective of passenger demand modeling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19*), Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis (Eds.). ACM, 1227–1235. http://dx.doi.org/10.1145/3292500.3330877

[38] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *The Web Conference 2020 (WWW'20).* ACM/IW3C2, 1082–1092.

[39] Xiyue Zhang, Chao Huang, Yong Xu, Lianghao Xia, Peng Dai, Liefeng Bo, Junbo Zhang, and Yu Zheng. 2021. Traffic flow forecasting with spatial-temporal graph diffusion network. In *35th AAAI Conference on Artificial Intelligence (AAAI'21), 33rd Conference on Innovative Applications of Artificial Intelligence (IAAI 2021), The 11th Symposium on Educational Advances in Artificial Intelligence (EAAI 2021)*, Virtual Event. AAAI Press, 15008–15015.

[40] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations.*

[41] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018 (NeurIPS'18)*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 5171–5181. http://papers.nips.cc/paper/7763-link-prediction-based-on-graph-neural-networks.

[42] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. 2017. MGAE: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM'17)*, Ee-Peng Lim, Marianne Winslett, Mark Sanderson, Ada Wai-Chee Fu, Jimeng Sun, J. Shane Culpepper, Eric Lo, Joyce C. Ho, Debora Donato, Rakesh Agrawal, Yu Zheng, Carlos Castillo, Aixin Sun, Vincent S. Tseng, and Chenliang Li (Eds.). ACM, 889–898. http://dx.doi.org/10.1145/3132847.3132967

[43] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *CoRR* abs/1812.08434 (2018). arXiv:1812.08434, http://arxiv.org/abs/1812.08434.

[44] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2019. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*. 31, 5 (2019), 833–852.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *CoRR* abs/1901.00596 (2019). arXiv:1901.00596, http://arxiv.org/abs/1901.00596.

[46] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 1024–1034. http://papers.nips.cc/paper/6703-inductive-representation-learning-on-large-graphs.

[47] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*, Yike Guo and Faisal Farooq (Eds.). ACM, 1416–1424. http://dx.doi.org/10.1145/3219819.3219947

[48] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. 2020. When do GNNs work: Understanding and improving neighborhood aggregation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*, Christian Bessiere (Ed.). ijcai.org. 1303–1309.

[49] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI,'18)*, Sheila A. McIlraith and Kilian Q. Weinberger (Eds.). AAAI Press, 3538–3545. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16098.

[50] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*. 5449–5458. http://proceedings.mlr.press/v80/xu18c.html.

[51] Donald J. Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series. In *KDD Workshop*, Vol. 10. 359–370.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

[53] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ.

[54] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence, (IJCAI'19)*, Sarit Kraus (Ed.). ijcai.org. 1907–1913. http://dx.doi.org/10.24963/ijcai.2019/264

[55] Didi Chuxing. [n. d.] Didi Chuxing GAIA Initiative. https://gaia.didichuxing.com. City Traffic Index.

[56] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *The 33rd AAAI Conference on Artificial Intelligence (AAAI'19)*. AAAI Press, 922–929. http://dx.doi.org/10.1609/aaai.v33i01.3301922

Received November 2020; revised July 2021; accepted September 2021