Spatiotemporal Adaptive Gated Graph Convolution Network for Urban Traffic Flow Forecasting

Bin Lu
Shanghai Jiao Tong University
robinlu1209@sjtu.edu.cn

Xiaoying Gan
Shanghai Jiao Tong University
ganxiaoying@sjtu.edu.cn

Haiming Jin
Shanghai Jiao Tong University
jinhaiming@sjtu.edu.cn

Luoyi Fu
Shanghai Jiao Tong University
yiluofu@sjtu.edu.cn

Haisong Zhang
Tencent AI Lab
hansonzhang@tencent.com

ABSTRACT

Urban traffic flow forecasting is a critical issue in intelligent transportation systems. It is quite challenging due to the complicated spatiotemporal dependency and essential uncertainty brought about by the dynamic urban traffic conditions. In most of existing methods, the spatial correlation is captured by utilizing graph neural networks (GNNs) throughout a fixed graph based on local spatial proximity. However, urban road conditions are complex and changeable, which leads to the interactions between roads should also be dynamic over time. In addition, the global contextual information of roads are also crucial for accurate forecasting. In this paper, we exploit spatiotemporal correlation of urban traffic flow and construct a dynamic weighted graph by seeking both spatial neighbors and semantic neighbors of road nodes. Multi-head self-attention temporal convolution network is utilized to capture local and long-range temporal dependencies across historical observations. Besides, we propose an adaptive graph gating mechanism to extract selective spatial dependencies within multi-layer stacking and correct information deviations caused by artificially defined spatial correlation. Extensive experiments on real world urban traffic dataset from Didi Chuxing GAIA Initiative have verified the effectiveness, and the multi-step forecasting performance of our proposed models outperforms the state-of-the-art baselines. The source code of our model is publicly available at https://github.com/RobinLu1209/STAG-GCN.

CCS CONCEPTS

• **Information systems → Spatial-temporal systems**.

KEYWORDS

Traffic forecasting, Spatiotemporal data, Graph neural network, Urban computing

ACM Reference Format:
Bin Lu, Xiaoying Gan, Haiming Jin, Luoyi Fu, and Haisong Zhang. 2020. Spatiotemporal Adaptive Gated Graph Convolution Network for Urban Traffic Flow Forecasting. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411894

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3411894

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]. Thus, urban traffic flow forecasting has gained more and more attention with the rapid development of intelligent transportation systems.

Traffic flow forecasting is a widely studied problem and researchers always analyze the traffic data from two aspects: temporal feature extraction and spatial feature extraction. Early approaches for traffic flow prediction are usually developed on time series with statistical methods [2] or simple machine learning models [3]. However, these approaches only consider temporal information and ignore the importance of spatial information for accurate prediction. Recently, deep learning based methods for traffic forecasting have been extensively studied [4]. Some researchers [5, 6] 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, graph neural network (GNN), which is efficient in capturing non-Euclidean correlation, is integrated into recurrent neural network (RNN) [7] or CNN [8] to 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 introducing GNN, we argue that there are several important aspects that previous methods have overlooked. First, these 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. Second, some previous 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. 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. For the performance degradation of cascaded GNNs, we propose an adaptive gating mechanism to selectively update and forget high-dimensional features. We capture the spatiotemporal correlation and propose a deep learning model called **S**patio**T**emporal **A**daptive **G**ated **G**raph **C**onvolution **N**etwork **(STAG-GCN)**.

Our main contributions are summarized as follows:

- We consider both the local and contextual spatial information, and define the spatial neighbors and semantic neighbors of the road nodes. Multi-head graph attention mechanism is utilized to model the road relationship as a dynamic weighted graph.
- We propose a novel adaptive graph gating mechanism to selectively update and forget the high-order neighbor information of nodes within the multi-layer stacking. GNN based on adaptive adjacency matrix can identify deviations caused by artificially defined spatial relationships and characterize global spatial correlations.
- We conduct extensive experiments on 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.

2 RELATED WORKS

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

2.1 Traffic Forecasting

In the past two years, researchers have tended to model the road network as a graph to capture the spatiotemporal correlations and make traffic forecasting. Li et al. [9] proposed to model the traffic flow as a diffusion process on a directed graph and introduced DCRNN inspired by GRU [10] structure. Yu et al. [8] combined a graph convolution with a 1D convolution and proposed STGCN model, which is much more computationally efficient than RNN structure. 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 correlations in traffic forecasting, Fang et al. [11] 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. [12] proposed to randomly group the nodes, then calculate the global dependencies by group, and use inter-group attention mechanism to combine the outputs. However, directly learning the global feature information lacks the guidance of domain knowledge and is prone to over-fitting.

For the extraction of temporal features, RNN-based models are widely used. Yao et al. [13] used node features and external information as input to LSTM to extract temporal dependencies. Wang et al. [14] 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. However, 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.

2.2 Graph Neural Networks

Graph neural networks are building blocks for learning graph-structure data. They are widely used in node classification [15], link prediction [16], graph clustering [17] and other graph theory problems. Advances in this direction are often categorized as spectral approaches and spatial approaches [18]. 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 [19] and LGCN [20]. 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 [21], stacking multiple GNN layers will result in 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. [22] 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 paper, we have studied how to aggregate multi-layer GNNs, and the performance of 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, and in this paper we focus on speed prediction. 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

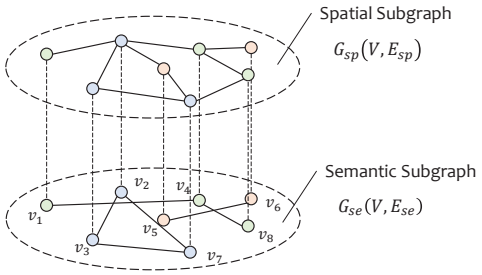


Figure 1: Spatial dependency modeling for urban traffic flow forecasting: spatial neighbors and semantic neighbors.

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 1 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 $\mathcal{G}$:

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

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 Dynamic Time Warping (DTW) Algorithm [23]. 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, because some nodes may have a quite low degree under single neighborhood definition.

3.1 Spatial Neighbor 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, 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 Neighbor 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 Dynamic Time Warping (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 2. Two signatures have a strong similarity, but they are not aligned on the time axis. 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 DTW algorithm have a stronger similarity compared to Euclidean distance pair, $v_A(t_i)$ and $v_B(t_i)$.

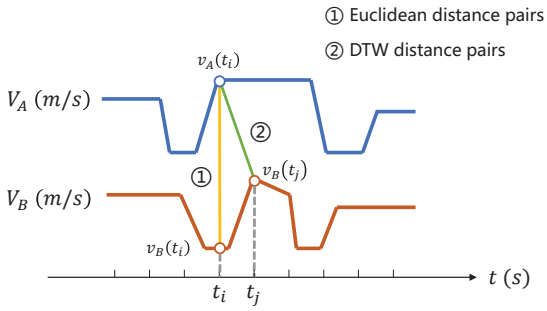


Figure 2: Example of the Euclidean vs. the DTW distance between two speed signatures.

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

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

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 multi-head attention mechanism to derive the dynamic weights of edges.

4 METHODOLOGY

Figure 3 shows the framework of our model, STAG-GCN. It mainly consists of three components: Self-Attention Temporal Convolution Network (Self-Attention TCN), Adaptive Gated Graph Convolution Network (AG-GCN) and Spatiotemporal Fusion Layer. These modules will be introduced in detail below.

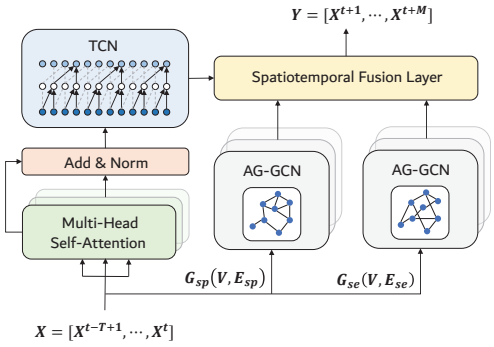


Figure 3: Model Overview of Spatiotemporal Adaptive Gated Graph Convolution Network (STAG-GCN).

4.1 Self-Attention TCN

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 [24] 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 single-head attention model, given input $\mathbf{X}^{t-T+1:t} = [\mathbf{X}^{t-T+1}, \cdots, \mathbf{X}^{t}] \in \mathbb{R}^{T\times N\times P}$, simplifying the notation as $\mathbf{X}$, three subspaces are obtained, namely, query subspace $Q \in \mathbb{R}^{N\times d_q}$, key subspace $K \in \mathbb{R}^{N\times d_k}$ and value subspace $V \in \mathbb{R}^{N\times d_v}$. The latent subspace learning process can be formulated as:

$$Q = XW^{Q}, \quad K = XW^{K}, \quad V = XW^{V}. \tag{4}$$

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

$$Attention(Q, K, V) = softmax(\frac{QK^{T}}{\sqrt{d_k}})V. \tag{5}$$

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:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^{O}, \\ \text{where } head_i &= Attention(Q_i, K_i, V_i), \end{aligned} \tag{6}$$

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 temporal convolution network (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:

$$y_{i,t,p} = \sum_{k=1}^{K_\tau}\sum_{m=1}^{M} w_{k,m,p} \cdot x_{i,t-d(k-1),m}, \tag{7}$$

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:

$$\mathcal{Y}_{tcn} = \mathcal{W} *_d \mathcal{X}. \tag{8}$$

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:

$$\mathcal{Y}_{tcn}^{l} = \begin{cases} \mathcal{X}, & l = 0, \\ \sigma(\mathcal{W}^{l} *_{d^l} \mathcal{Y}_{tcn}^{l-1}), & l = 1, 2, \ldots, L, \end{cases} \tag{9}$$

where $\mathcal{X} \in \mathbb{R}^{N\times T\times P_{in}}$ is the output of mult-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 Adaptive Gated Graph Convolution Network (AG-GCN)

In this part, we present the Adaptive Gated Graph Convolution Network in Figure 4 in detail. There are two 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.

Graph convolution network 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:

$$Z = \widetilde{A}XW. \tag{10}$$

In our module, two different graph convolution methods are used and the results of the two graph convolutions are integrated using

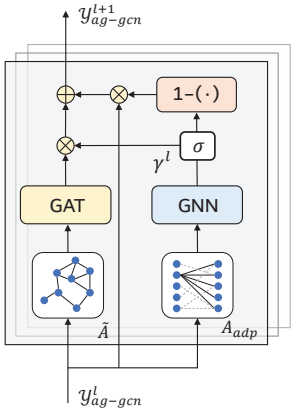


Figure 4: Adaptive Gated Graph Convolution Network

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 [25] 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 graph.

$$e_{ij} = a(Wh_i, Wh_j), \ j \in \mathcal{N}_i, \tag{11}$$

where weight matrix is $W \in \mathbb{R}^{D\times M}$, 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 choices of j using the $softmax$ function:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}_i} exp(e_{ik})}. \tag{12}$$

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

$$h'_i = \Big\Vert_{k=1}^{K} \sigma(\sum\nolimits_{j\in\mathcal{N}_i} \alpha_{ij} W^k h_j), \tag{13}$$

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

$$h'_i = \sigma(\frac{1}{K}\sum\nolimits_{k=1}^{K}\sum\nolimits_{j\in\mathcal{N}_i} \alpha_{ij} W^k h_j). \tag{14}$$

When we write the process of graph attention network as a matrix operation, it is

$$Z^l = (\widetilde{A} \odot M) Z^{l-1} W, \tag{15}$$

where $\widetilde{A} = A_{sp} + I$ or $\widetilde{A} = A_{se} + I$, $\odot$ is the element-wise product, 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 $l = 0$, $Z^0 = X$.

4.2.2 Adaptive Graph Gating Mechanism. Graph neural networks 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. [22] studied this issue and found that after passing through multi-layer GNN, the nodes with high degrees are prone to oversmoothing, 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.

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.

$$A_{adp} = softmax(ReLU(E_s E_t^T)). \tag{16}$$

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

$$\gamma = \sigma(A_{adp} X \Theta), \tag{17}$$

where X denote the input signal, Θ denote the adaptive graph convolution parameter matrix, 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.

$$\boldsymbol{y}^l_{ag-gcn} = \begin{cases} X, & l = 0, \\ \gamma^l \odot g(Z^l) + (1-\gamma^l) \odot \boldsymbol{y}^{l-1}_{ag-gcn}, & l = 1, 2, \ldots, L. \end{cases} \tag{18}$$

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

4.3 Spatiotemporal Fusion Network

After we get the high-dimensional features in the temporal, spatial and semantic domain, how we aggregate the spatiotemporal information for multi-step prediction is a problem we need to consider. Our model permits general module-aggregation mechanisms. We explore three mechanisms, *concatenation*, *max-pooling* and *LSTM-attention*. We will compare these three 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. Generate forward-LSTM and backward-LSTM hidden feature and use linear transformation to calculate the attention score. Finally, we use $softmax$ function to yield the attention of node v on three modules and get the weighted-sum final representation.

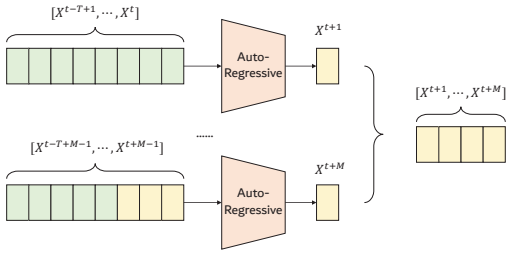
When we get the high-dimensional spatiotemporal features, we make a multi-step prediction through a linear layer. One prediction method is an auto-regressive prediction method, as shown in Figure 5(a). Error-prone predictions are used as input with previous observations for further predictions, resulting 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. [26] to directly predict the future M steps, as shown in Figure5(b), 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:

$$L(\mathbf{Y}, \mathbf{Y}^{'}) = \frac{1}{n} \sum_{i=1}^{n} \|Y_i - Y_i^{'}\|^2, \tag{19}$$

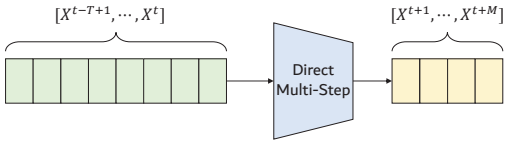
where $\mathbf{Y} \in \mathbb{R}^{n \times N \times M}$ is the ground truth, $\mathbf{Y}^{'} \in \mathbb{R}^{n \times N \times M}$ is the prediction, and n is the batch number.

5 EXPERIMENT

In this section, we evaluate the performance of proposed model, STAG-GCN, on the urban traffic index dataset of Chengdu, China, provided by Didi Chuxing GAIA Initiative [27]. We visualize the road map in the dataset in Figure 6. This public dataset contains the average speed of major roads in Chengdu, China in 2018. 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]. DiDi_Chengdu[M] dataset contains 524 roads in the core urban area of Chengdu, while DiDi_Chengdu[L] dataset contains 1343 roads in a wider area.



(a) Auto-regressive Multi-step Prediction



(b) Direct Multi-step Prediction

Figure 5: Two multi-step prediction methods: Auto-regressive and direct multi-step prediction

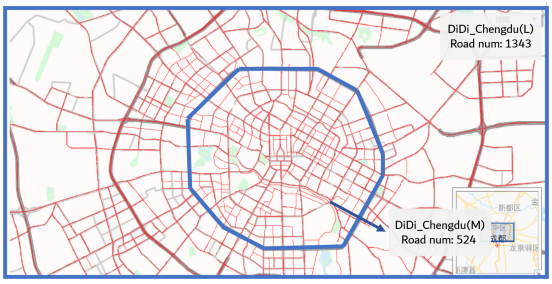


Figure 6: Road map visualization for DiDi_Chengdu[M] and DiDi_Chengdu[L] datasets

5.1 Experiment Settings

In both datasets, the interval between the collected speed is 10 minutes. We apply Z-Score normalization for data preprocessing. 70% of data are used for training, 20% are used for testing while the remaining 10% for validation.

We compare our model, STAG-GCN, with 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 average of previous seasons as the prediction.
- **ARIMA**: Auto-Regressive Integrated Moving Average model, which is widely used in time series prediction.
- **KNN**: K-Nearest Neighbor algorithm, which classifies nodes by measuring the feature distance between different nodes, and adopts its closest K neighbors to represent the features.
- **RF**: Random Forrest, a tree-based ensemble learning algorithm.
- **FNN**: Feed forward neural network with two hidden layers and L2 regularization.

Table 1: Performance comparison of STAG-GCN and other baseline models on DiDi_Chengdu datasets. STAG-GCN achieves the best performance with all three metrics for all forecasting horizons.

Model	DiDi_Chengdu[M] (10/30/60 min)			DiDi_Chengdu[L] (10/30/60 min)		
	MAE	MAPE(%)	RMSE	MAE	MAPE(%)	RMSE
HA	3.89	13.42	5.61	3.02	13.24	4.56
ARIMA	2.32/ 3.26/ 4.15	9.80/ 14.21/ 18.23	3.45/ 4.90/ 6.22	2.44/ 3.41/ 4.36	9.00/ 12.92/ 16.67	3.73/ 5.19/ 6.65
KNN	2.31/ 2.89/ 3.32	10.39/ 13.50/ 15.69	3.33/ 4.21/ 4.83	2.42/ 3.01/ 3.45	9.40/ 12.07/ 14.08	3.56/ 4.45/ 5.14
RF	2.29/ 2.94/ 3.37	9.66/ 13.11/ 15.31	3.14/ 4.08/ 4.69	2.40/ 3.05/ 3.52	9.18/ 12.188/ 14.30	3.53/ 4.50/ 5.20
FNN	2.42/ 2.91/ 3.43	10.63/ 12.59/ 14.72	3.35/ 3.97/ 4.65	2.77/ 2.97/ 3.30	11.43/ 12.25/ 13.66	4.12/ 4.35/ 4.79
LSTM-FC	2.37/ 2.52/ 2.91	11.30/ 11.96/ 13.66	3.45/ 3.68/ 4.23	2.60/ 2.72/ 3.09	10.74/ 11.29/ 12.81	3.96/ 4.13/ 4.63
STGCN	2.22/ 2.67/ 3.05	9.94/ 12.67/ 14.65	3.18/ 3.90/ 4.46	2.50/ 2.81/ 3.14	10.07/ 11.56/ 13.02	3.61/ 4.12/ 4.61
DCRNN	2.04/ 2.65/ 3.16	9.00/ 12.34/ 14.83	2.99/ 3.92/ 4.61	2.27/ 2.61/ 2.78	9.22/ 10.97/ 11.78	3.49/ 4.04/ 4.32
ASTGCN	2.05/ 2.44/ 2.70	9.17/ 11.48/ 12.71	2.99/ 3.58/ 3.91	2.20/ 2.66/ 2.95	8.60/ 10.74/ 11.98	3.28/ 3.96/ 4.38
STAG-GCN	**1.98/ 2.36/ 2.54**	**8.84/ 11.05/ 11.90**	**2.89/ 3.46/ 3.69**	**2.08/ 2.52/ 2.79**	**8.07/ 10.04/ 11.02**	**3.12/ 3.75/ 4.11**

- **LSTM-FC**: The Encoder-Decoder framework using fully connected LSTM units.
- **STGCN**: Spatial-temporal graph convolution network [8], which combines graph convolution with 1D convolution.
- **DCRNN**: Diffusion convolution recurrent neural network [9], which combines graph convolution networks with recurrent neural networks in an encoder-decoder manner.
- **ASTGCN**: Attention based spatial-temporal graph convolutional network [28], which design a spatial-temporal attention mechanism to capture the dynamic spatial-temporal correlations.

We implement our model, STAG-GCN, based on the Pytorch framework[1]. Mean square error between the prediction and ground truth is used as the loss function and minimized by back-propagation. During the training phase, the batch size is 32, the learning rate is 0.001 and we adopt ADAM optimizer to train the model. 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. Besides, we use following three 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)

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

5.2 Performance Comparison

Table 1 shows the comparison of different approaches for 10, 30 and 60 minutes forecasting on two datasets. Our proposed model, STAG-GCN, obtains the superior results on both datasets. We observe that: (1) deep learning approaches outperform traditional time series and machine learning methods, demonstrating the strong ability of neural network in extracting nonlinear correlations; (2) among deep learning approaches, STGCN, DCRNN and ASTGCN have a better performance, indicating that traffic graph plays a crucial role in traffic flow forecasting; (3) STAG-GCN achieves state-of-the-art multi-step prediction performances. Our model considers both spatial proximity and contextual similarity, revealing the effectiveness of spatial neighbors and semantic neighbors. The multi-head attention mechanism and proposed adaptive graph gating mechanism help us to obtain richer spatiotemporal features and capture the dynamics of road conditions.

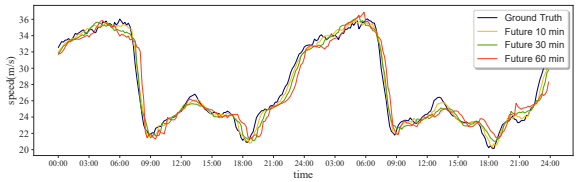
In order to verify the performance of our model under different road conditions, we select two-day snapshot on weekday and weekend from testing dataset. Figure 7 plots the multi-step predictions and ground truth. It can be seen that although the road characteristics on weekdays and weekends are completely different, our prediction results are relatively stable and reliable. For the sake of model robustness, we compare the average prediction errors of multi-step predictions per hour in Figure 8. It shows that there is a normal performance decline as the forecast time increase. However, due to the strong dynamics during rush hour, 60 minutes forecasting performance have 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.

5.3 Component Analysis

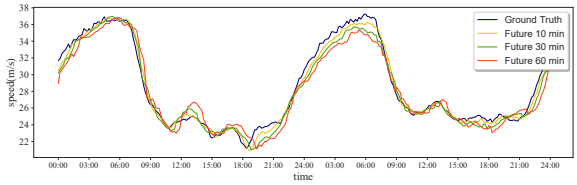
After we obtain a high-dimensional spatiotemporal representation, the output of each module will be sent to the fusion layer. As mentioned in Section 4.3, its design is mainly divided into two problems: one is how to aggregate multiple features, and the other is how to perform multi-step prediction.

5.3.1 Module aggregation mechanisms. Through extensive experiments, Figure 9(a) shows the performance of three module aggregation methods *Concatenation*, *Max-pooling* and *LSTM-attention*. It can be found that *Concatenation* is the best fusion method, because

[1]The implementation code of our model is available at https://github.com/RobinLu1209/STAG-GCN.



(a) Snapshot of two days on weekday



(b) Snapshot of two days on weekend

Figure 7: Visualization of ground truth and multi-step prediction of the average speed on DiDi_Chengdu[M] dataset

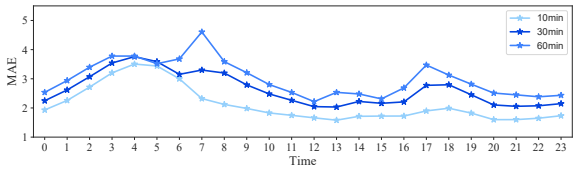


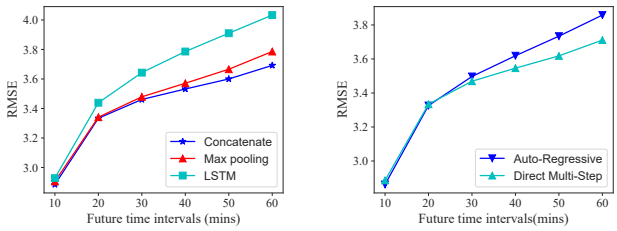
Figure 8: Prediction error during one day on average

it retains the entire high-dimensional spatiotemporal features. *Max-pooling* performs slightly worse than *Concatenation*. Although *Max-pooling* can obtain the most informative features, it inadvertently loses some information, which causes performance degradation after several prediction steps. *LSTM-attention* introduces a large number of parameters to learn. It assigns attention score to different features, but it does not have a good effect on high-dimensional features without time correlation.

5.3.2 Multi-Step prediction method. Figure 9(b) shows the performance of two multi-step prediction methods, auto-regressive and direct multi-step, on DiDi_Chengdu[L] dataset. Auto-regressive multi-step traffic prediction is error-prone.The performance decline drastically due to the superposition of errors, especially after the multi-step prediction. On the contrary, since the direct multi-step prediction does not take the prediction result as input, it avoids the accumulation of errors and obtains better prediction performance.

5.4 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 10. The main part of our model is divided into three parts: Self-Attention TCN, AG-GCN module for spatial neighbors and 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,



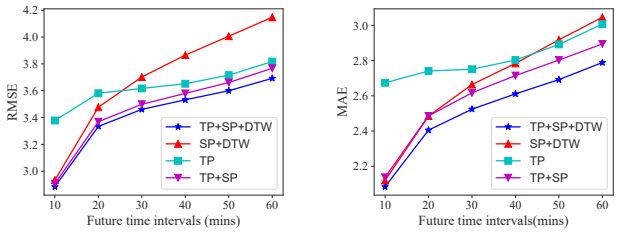
(a) Module aggregation mechanisms study on DiDi_Chengdu[M] dataset

(b) Multi-step prediction method comparison on DiDi_Chengdu[L] dataset

Figure 9: Component analysis of STAG-GCN

we split and combine them to obtain the following sub-models: (1) TP+SP+DTW, (2) SP+DTW, (3) TP, (4) TP+SP. We train these sub-models on two datasets 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 that only relies on spatial information (SP+DTW) has a dramatic performance decline with increasing prediction steps, but its short-term prediction performance is better. When we comprehensively consider the spatiotemporal information (TP+SP), it will help improve the prediction performance. Addition of semantic neighbor considerations (TP+SP+DTW) further improves the prediction accuracy. Especially in the application of large-scale graphs, since nodes can obtain richer neighbor information, the performance improvement is relatively obvious.



(a) DiDi_Chengdu[M] dataset

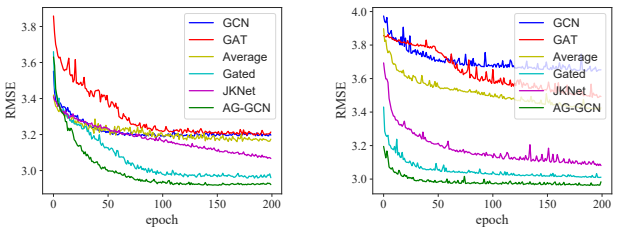
(b) DiDi_Chengdu[L] dataset

Figure 10: Ablation study for three modules in STAG-GCN: Temporal, Spatial and Semantic Feature on two datasets.

To further verify the effectiveness of AG-GCN module in extracting spatial correlations, we replace it with other five models:

- **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 gating mechanism.
- **JKNet**: Jumping Knowledge Network [22] to aggregate GATs with pre-defined spatial correlations.

Figure 11 shows RMSE performance of different models on validation dataset during training. As shown in Figure 11(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 11(b), **GAT** method that dynamically adjusts the edge weights is better than **GCN** method with fixed adjacency matrix, which reflects the effectiveness of dynamic weighted graph modeling in our model. In **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 gating mechanism. 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 result, proving the advantages of our design rationale.



(a) With Self-Attention TCN module (b) Without Self-Attention TCN module

Figure 11: Ablation study for AG-GCN: RMSE performance of different models on validation dataset during training.

5.5 Model Explanation

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

5.5.1 Attentional Edge Weight. The dynamic attentional edge weight reflects the interaction between two connected road nodes. Figure 12 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.

As can be seen, the relationship 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.

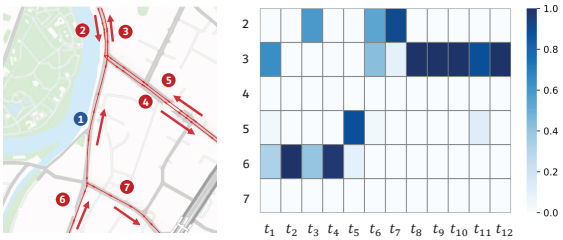


Figure 12: Geographical location on Google map and heat map of attentional edge weights on different time slots.

5.5.2 Adaptive Adjacency Matrix. In 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 matrix and draw the heat map based on the gated values as shown in Figure 13. We select the gated value of column 10 for observation, and mark the nodes with higher gated values on the Google Maps.

Figure 13(a) is the adaptive adjacency matrix of the AG-GCN module based on semantic neighbor, while Figure 13(b) is based on spatial neighbor. Since two AG-GCN module extract high-dimensional features from two perspectives separately, the road nodes with higher gated values are quite different. The road nodes with similar contextual information are distributed at various locations on the map, while the road nodes with spatial proximity are concentrated near the analysis road or directly connected to it.

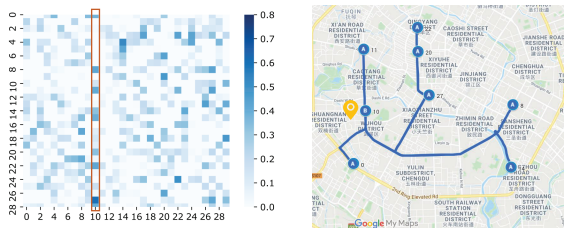
6 CONCLUSION

In this paper, we define the road network as a dynamic weighted graph by seeking both spatial neighbors and semantic neighbors of road nodes. A novel Spatiotemporal Adaptive Gated Graph Convolution Network model is proposed for urban traffic flow forecasting. 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 graph neural network. Experiments on two real-world datasets show that the performance of the proposed model is superior to existing models.

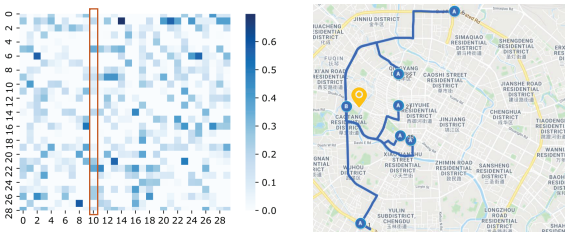
Actually, urban traffic prediction is affected by many external factors, such as social events and road regulations. In future work, we can take these external information into consideration to further improve the accuracy of predictions. Furthermore, the proposed AG-GCN module can be generalized into dynamical graph features learning in various applications, and we can apply it to other pragmatic applications in the future.

ACKNOWLEDGMENTS

This work was supported by National Key R&D Program of China 2018AAA0101200, in part by National Natural Science Foundation



(a) Semantic Neighbor Subgraph



(b) Spatial Neighbor Subgraph

Figure 13: Two kind of adaptive adjacency matrix and its geographical location marked on Google Maps.

of China under Grant (No. 61532012, 61672342, 61671478, 61822206, 61960206002, 61832013, 62041205, 61902244, 61942204), in part by Tencent AI Lab Rhino-Bird Focused Research Program JR202034, in part by Shanghai Municipal Science and Technology Commission Grant 19YF1424600, in part by the Science and Technology Innovation Program of Shanghai (Grant 18XD1401800, 17511105103, 18510761200), in part by Shanghai Key Laboratory of Scalable Computing and Systems. The data source for this study is Didi Chuxing GAIA Initiative.

REFERENCES

[1] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Trans. Intelligent Transportation Systems*, 16(2):865–873, 2015.

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

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

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

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

[6] Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. In Satinder P. Singh and Shaul Markovitch, editors, *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, pages 1655–1661. AAAI Press, 2017.

[7] Weiqi Chen, Ling Chen, Yu Xie, Wei Cao, Yusong Gao, and Xiaojie Feng. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*.

[8] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 3634–3640. ijcai.org, 2018.

[9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[10] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014*, pages 1724–1734. ACL, 2014.

[11] Shen Fang, Qi Zhang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Gstnet: Global spatial-temporal network for traffic flow prediction. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 2286–2293, 2019.

[12] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020*.

[13] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, and Zhenhui Li. Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 5668–5675. AAAI Press, 2019.

[14] Yuandong Wang, Hongzhi Yin, Hongxu Chen, Tianyu Wo, Jie Xu, and Kai Zheng. Origin-destination matrix prediction via graph convolution: a new perspective of passenger demand modeling. In Ankur Teredesai, Vipin Kumar, Ying Li, Rómer Rosales, Evimaria Terzi, and George Karypis, editors, *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019*, pages 1227–1235. ACM, 2019.

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

[16] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pages 5171–5181, 2018.

[17] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. MGAE: marginalized graph autoencoder for graph clustering. In 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, editors, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017*, pages 889–898. ACM, 2017.

[18] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *CoRR*, abs/1901.00596, 2019.

[19] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pages 1024–1034, 2017.

[20] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In Yike Guo and Faisal Farooq, editors, *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018*, pages 1416–1424. ACM, 2018.

[21] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, AAAI, 2018*, pages 3538–3545. AAAI Press, 2018.

[22] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, pages 5449–5458, 2018.

[23] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[25] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[26] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019*, pages 1907–1913. ijcai.org, 2019.

[27] Didi Chuxing. Didi chuxing gaia initiative. https://gaia.didichuxing.com. City Traffic Index.

[28] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 922–929. AAAI Press, 2019.