LSGCN: Long Short-Term Traffic Prediction with Graph Convolutional Networks

Rongzhou Huang[1] , **Chuyin Huang**[1] , **Yubao Liu** [*1,2] , **Genan Dai**[1] and **Weiyang Kong**[1]
[1]Sun Yat-Sen University, Guangzhou, China
[2]Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
liuyubao@mail.sysu.edu.cn, {huangrzh6, huangchy78, daign, kongwy3}@mail2.sysu.edu.cn

Abstract

Traffic prediction is a classical spatial-temporal prediction problem with many real-world applications such as intelligent route planning, dynamic traffic management, and smart location-based applications. Due to the high nonlinearity and complexity of traffic data, deep learning approaches have attracted much interest in recent years. However, few methods are satisfied with both long and short-term prediction tasks. Target at the shortcomings of existing studies, in this paper, we propose a novel deep learning framework called Long Short-term Graph Convolutional Networks (LSGCN) to tackle both traffic prediction tasks. In our framework, we propose a new graph attention network called cosAtt, and integrate both cosAtt and graph convolution networks (GCN) into a spatial gated block. By the spatial gated block and gated linear units convolution (GLU), LSGCN can efficiently capture complex spatial-temporal features and obtain stable prediction results. Experiments with three real-world traffic datasets verify the effectiveness of LSGCN.

1 Introduction

Traffic prediction is a classical spatial-temporal prediction problem whose purpose is to predict the traffic conditions (e.g. vehicle speed) of several future times based on the historical traffic observations (e.g. recorded via sensors of traffic network). The problem has been found useful in many real-world applications such as intelligent route planning, dynamic traffic management, and smart location-based applications [Wu and Tan, 2016]. In general, there are two kinds of traffic prediction tasks according to the length of prediction time, namely the short-term (5∼30 minutes) and long-term (30∼60 minutes) [Ostring and Sirisena, 2001; Vlahogianni *et al.*, 2005]. The traditional methods often adopt the queuing theory, computational simulation and statistics [Ahmed and Cook, 1979; Williams and Hoel, 2003]. Since these methods rely on the stationarity assumption, they fail to capture complex spatial-temporal features for the prediction tasks.

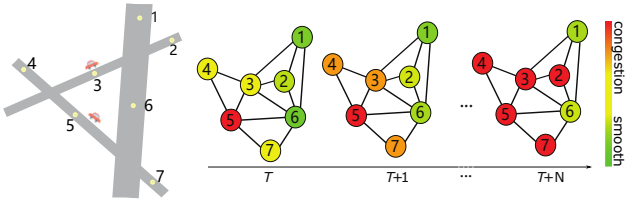

Figure 1: The left picture shows a road network with seven sensors numbered from 1 to 7. The right pictures denote the traffic networks over time from T to $T+N$. The different color of the sensors represents the degree of road congestion. For example, red means that the corresponding roads are congested and green shows the roads are smooth. If there is a sudden accident on the road with sensor 5 at time T, the road becomes congested. Then, at time $T+1$, the adjacent roads with sensors 3, 4 and 7 may be affected by the accident and also become congested. Instead, sensors 1, 2 and 6 may be affected less since they are located at the farther roads. Moreover, since the roads with sensors 2 and 5 are similar, which are narrower than the road with sensors 1 and 6, sensor 2 may be affected more than sensors 1 and 6.

Due to the high nonlinearity and complexity of traffic data, deep learning approaches have attracted much interest in recent years such as convolutional neural network (CNN) and recurrent neural networks(RNN) [Wu and Tan, 2016; Ma *et al.*, 2017; Hochreiter and Schmidhuber, 1997; Xingjian *et al.*, 2015]. In particular, graph convolution networks(GCN) and its variants have been used to traffic prediction tasks and often obtain promising prediction results [Yu *et al.*, 2018; Li *et al.*, 2018; Guo *et al.*, 2019]. However, few methods are satisfied with both long and short-term prediction tasks. For example, STGCN [Yu *et al.*, 2018] adopts the mechanism of iterative prediction. The traffic conditions predicated in the previous iterations are used as the historical observations for the next iterations. Since the historical traffic observations may be not real values, there are errors accumulated for the prediction task. In general, the more network structure layers, the more errors accumulated, especially for the long-term task. On the other hand, ASTGCN [Guo *et al.*, 2019] and D-CRNN [Li *et al.*, 2018] adopt the mechanism of non-iterative prediction, where all prediction values for multiple time steps are obtained by one unified evaluation instead of multiple iterations. So, it is hard to give considerations to both long and short-term tasks.

*Yubao Liu is the corresponding author.

Target at the shortcomings of existing studies, we propose a novel deep learning framework called Long Short-term Graph Convolutional Networks (LSGCN) to tackle both traffic prediction tasks. Intuitively, the mechanism of iterative prediction is good for the simulation of dynamic traffic conditions over time.

So, we also adopt the mechanism of iterative prediction for LSGCN. We try to enhance the performance of LSGCN from two aspects. On the one hand, we construct a simplified network structure with fewer layers and reduce the accumulated errors in each iteration. On the other hand, we design some new network layers to efficiently capture complex spatial-temporal features. Specifically, we propose a spatial gated block in which a new graph attention network called cosAtt and graph convolution networks (GCN) are integrated by the gated form. By cosAtt and GCN, the spatial gated block can precisely extract the spatial location adjacency and similar road conditions of traffic networks. For example, Figure 1 shows an application scenario can be captured by the spatial gated block. Moreover, the graph attention cosAtt enables more stable prediction results.

Our contributions are summarized as follows. (1) We propose a novel deep learning framework LSGCN for both long and short-term traffic prediction tasks. In LSGCN, we adopt the spatial gated block and gated linear units convolution (GLU) to capture the spatial and temporal features, respectively. We propose a new graph attention network cosAtt and integrate both GCN and cosAtt into the spatial gated block. (2) We evaluate our LSGCN on three real-world traffic datasets and the experimental results show that LSGCN outperforms state-of-the-art baselines.

2 Preliminary

2.1 Traffic Prediction Problem

Traffic Network. Traffic network is defined as an undirected graph $G = (V, E, W)$, where V is the set of nodes, E is the set of edges between two nodes, and $W \in \mathbb{R}^{N\times N}$ corresponds to the adjacency matrix of G. In practice, a node may represent a sensor located at the corresponding road of the traffic networks. Each node records some traffic features, such as traffic flow, vehicle speed, and road occupancy, etc.

Problem Definition. For a traffic network, let $x_t^i \in \mathbb{R}$ represents the traffic feature value of the i^{th} node at time step t, and $x_t \in \mathbb{R}^N$ represents the traffic feature values of all nodes at time step t. Given history traffic data $X = (x_1, x_2, ..., x_\tau)^T \in \mathbb{R}^{N\times\tau}$, the purpose of traffic prediction is to predict the traffic features of all nodes in future T_p time steps, namely $Y^* = (y_1, y_2, ..., y_{T_p})^T \in \mathbb{R}^{N\times T_p}$. Without loss of generality, we consider the feature of vehicle speed in the following.

2.2 Attention on Graphs

The input to graph attention networks (GAT) is a set of node features, $\mathcal{Q} = \{\vec{q_1}, \vec{q_2}, ..., \vec{q_N}\}$, where $\vec{q_i} \in \mathbb{R}^F$ and N is the number of nodes, and F is the number of features in each node. The output is also a new set of node features (of potentially different cardinality F'), $\mathcal{Q}' = \{\vec{q_1'}, \vec{q_2'}, ..., \vec{q_N'}\}$, where $\vec{q_i'} \in \mathbb{R}^{F'}$. The attention coefficient for any pair of $\vec{q_i}$ and $\vec{q_j}$ is given as follows.

$$e_{ij} = A(\vec{q_i}w_i, \vec{q_j}w_j), w_i, w_j \in F \times F' \tag{1}$$

where w_i and w_j are weight matrix components, and $A(\cdot)$ is the attention mechanism operation function. Then, we normalize the attention coefficient e_{ij} among node neighborhoods and achieve the output as follows.

$$a_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}_i} exp(e_{ik})} \tag{2}$$

$$\vec{q_i'} = \sigma(\sum_{j\in\mathcal{N}_i} a_{ij}\vec{q_j}w_j)) \tag{3}$$

where $\mathcal{N}_i$ is the set of neighbours of node i in the graph and σ is an activation function [Velickovic *et al.*, 2017].

3 Proposed Model

3.1 Network Architecture

LSGCN is composed of five layers, namely two gated linear units convolutional layers (GLUs), spatial gated block, convolution-unified layer and fully connection layer. The spatial gated block is between two GLUs as shown in Figure 2. Our network structure has fewer layers compared with STGCN [Yu *et al.*, 2018]. LSGCN adopts the mechanism of iterative prediction as follows. In each iteration, we first obtain $H' = (h'_1, h'_2, ..., h'_{\tau-K_T+1})^T \in \mathbb{R}^{N\times(\tau-K_T+1)\times C_1}$ by the first GLU layer, where K_T is the width of 1D convolution kernel and C_1 is the channel numbers. Then, we can obtain $H'' = (h''_1, h''_2, ..., h''_{\tau-K_T+1})^T \in \mathbb{R}^{N\times(\tau-K_T+1)\times C_1}$ by spatial gated block, which is based on H' and the adjacency matrix W. Next, H'' is processed by the second GLU, and we can obtain $H''' = (h'''_1, h'''_2, ..., h'''_{\tau-2\times(K_T-1)})^T \in \mathbb{R}^{N\times(\tau-2\times(K_T-1))\times C_2}$, where C_2 is the second channel numbers. Finally, the output is achieved by the convolution unified layer and fully connection layer, namely $Y = (y_1)^T \in \mathbb{R}^{N\times 1\times 1}$. Note that the obtained Y is added into the input for next iteration. For example, the input for the second iteration is $\{x_2, ..., x_i, ..., x_\tau, y_1\}$. The iteration process is halted after T_p iterations. The adjacent road conditions are important in short-term task, the similar road conditions more so in long-term task. They can be captured by GCN and cosAtt of the network model, respectively.

3.2 Spatial Gated Block for Extracting Spatial Features

We consider two main spatial features, namely spatial location adjacency and road condition similarity. We use GCN to extract the spatial location adjacency and a new graph attention network cosAtt to extract road condition similarity.

Inspired by the sequential gate structure, such as LSTM [Hochreiter and Schmidhuber, 1997] and GRU [Chung *et al.*, 2014], we adopt the gate structure to integrate both components. We use the *sigmoid* function to map the results of GCN to the interval $(0, 1)$ and control the output of cosAtt. Intuitively, GCN determines how much information generated by cosAtt can flow into the next layer based on the

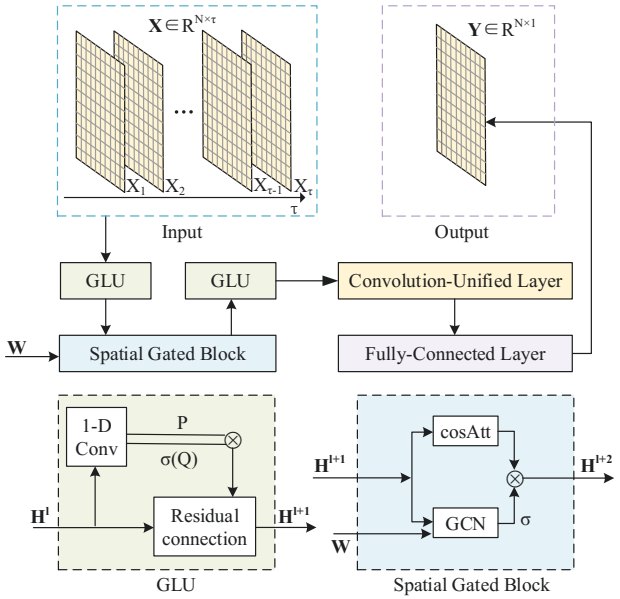

Figure 2: Network architecture of LSGCN. P and Q are the outputs of 1-D Conv, σ represents the $sigmoid$ activation function and $\otimes$ represents the element-wise Hadamard product between two branches.

topological structure (i.e., spatial location adjacency) of traffic networks.

As shown in Figure 2, the spatial gated block works as follows. First, we pass the H^{l+1} generated by the first GLU and the adjacency matrix W into GCN. The results of GCN then go through the *sigmoid* activation function. On the other hand, we make a copy of H^{l+1} and pass it to cosAtt. Finally, H^{l+2} is achieved by the element-wise Hadamard product on the results generated before.

Graph Attention cosAtt. Based on the observation that the roads adjacent to each other in a traffic network often have similar road conditions as shown in Figure 1. We design a new graph attention cosAtt to extract the similar road conditions of traffic networks. In cosAtt, we adopt global graph attention networks to learn the similar conditions of any two roads of traffic network instead of neighbours in graph attention networks (GAT).

Initially, node $H = \{h_1, ..., h_i, ..., h_N\}, h_i \in \mathbb{R}^{N \times T \times C_i}$ is passed into cosAtt, where T represents time steps and C_i is input channels. We define cosAtt as follows.

$$e_{ij} = cos(h_i, h_j) \cdot w_{ij} = \frac{h_i \times h_j^T \cdot w_{ij}}{\|h_i\| \times \|h_j\|} \tag{4}$$

$$a_{ij} = sigmoid(e_{ij}) \tag{5}$$

$$cosAtt_i = \sum_{j \in \widetilde{N}_i} a_{ij} h_j w_{ij} \tag{6}$$

where $\widetilde{\mathcal{N}}_i$ is all nodes in graph except node i.

Next, we show the key differences between cosAtt and traditional GAT. First, the weight distribution of cosAtt is used as the output, whereas the weight distribution of GAT is added into the original features. On the other hand, for any given two features h_i and h_j, the similarity value between them is constant (i.e., $cos(h_i, h_j)$). The value of e_{ij} is relatively stable and only adjusted by the weight matrix W (i.e., w_{ij}). In contrary to cosAtt, the value of e_{ij} in GAT may be unstable since we have to learn them by the attention function and the weight matrix W as shown in Eq.(1).

Note that we use the *sigmoid* instead of the *softmax* activation function in our model. This is because that the *softmax* activation function often results in denominator underflow and program crash since the number of neighbors (i.e., all nodes except the target node itself) for each node may be huge.

The details on GCN are as follows. Based on the conception of spectral graph convolution, the graph convolution operator "$*_{\mathcal{G}}$" can be represented as the multiplication of a signal $x \in \mathbb{R}^N$ with a kernel Θ.

$$\Theta *_{\mathcal{G}} x = \Theta(L)x = \Theta\left(U \Lambda U^T\right) x = U \Theta(\Lambda) U^T x \tag{7}$$

In the equation, graph Fourier basis $U \in \mathbb{R}^{N \times N}$ corresponds to the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}} W D^{-\frac{1}{2}} = U \Lambda U^T \in \mathbb{R}^{N \times N}$, where I_N is an identity matrix and $D \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix with $D_{ii} = \sum_j W_{ij}$. $\Lambda \in \mathbb{R}^{N \times N}$ corresponds to the diagonal matrix of eigenvalues of L and filter $\Theta(\Lambda)$ is also a diagonal matrix [Shuman *et al.*, 2013].

Since the computation of Eq.(7) is expensive, we adopt the Chebyshev polynomials approximation in our implementation to speed up the computation of GCN.

$$\Theta *_{\mathcal{G}} x = \Theta(L)x \approx \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L}) x \tag{8}$$

where $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients, K is the kernel size of graph convolution, and $T_k(\tilde{L}) \in \mathbb{R}^{N \times N}$ corresponds to the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{L} = 2L/\lambda_{max} - I_N$. The computation of Eq. (8) takes $\mathcal{O}(K \cdot |E|)$ time [Defferrard *et al.*, 2016].

3.3 Gated Linear Units Convolution Layer for Extracting Temporal Features

We use GLU to capture dynamic behaviors of temporal features, which is similar to STGCN [Yu *et al.*, 2018]. As shown in Figure 2, there are two GLUs each of which consists of a 1D convolution with a width-K_T kernel and a residual connection.

In general, GLU works as follows. The input is $X = \{x_1, x_2, ..., x_i, ..., x_N\}, x_i \in \mathbb{R}^{N \times \tau \times C_i}$ ($C_i = 1$ for the first GLU). Next, we use convolutional kernels $\Theta \in \mathbb{R}^{K_t \times C_i \times 2C_o}$ to obtain the convolutional result $Z = [PQ] \in \mathbb{R}^{N \times (\tau - K_T + 1) \times 2C_o}$, where C_o is the size of the feature set generated by GLU. Note that the number of 1D convolutional kernels is set as $2 \cdot C_o$. The output through 1D convolutional layer is divided into two equal parts, namely P and Q. P is the output of the first half convolution kernels and Q is of the second half convolution kernels. Then, the element-wise Hadamard product of P and $\sigma(Q)$ is computed and added into the input of the residual connection. If the number of input

	PeMSD4	PeMSD7	PeMSD8
#Nodes	307	228	170
#Edges	340	832	295
#Time steps	16992	12672	17856
Time span	2018/1 - 2018/2	2012/5 - 2012/6 (weekdays)	2016/7 - 2016/8
Time interval	5mins		
Daily range	0:00 - 24:00		

Table 1: The details for the datasets

dimension is less than that of the output dimension, the input dimension is filled with zero, and vice versa. Finally, we can reduce the time steps to $\tau - 2 \times (K_T - 1)$ (> 0) based on two GLUs.

3.4 Convolution Unified Layer

In the mechanism of iterative prediction, we need to output single prediction value for each time step in each iteration process. So, it is necessary to reduce dimension on time axis. As shown Figure 2, LSGCN adopts the convolution unified layer to merge the $\tau - 2 \times (K_T - 1)$ time steps into single time step with casual convolution. In general, the sequence with $\tau - 2 \times (K_T - 1)$ time steps is a relatively long sequence. For a long sequence, the casual convolution fuse time steps in proportion, so we can obtain a reasonable time step as the output of this layer.

3.5 Loss Function

We use L2 loss to measure the performance of our model which is defined as follows,

$$L\left(\hat{y}; W_\theta\right) = \sum_t \left\| \hat{y}\left(x_{t-\tau}, \ldots, x_t, W_\theta\right) - x_{t+1} \right\|^2 \qquad (9)$$

where $\hat{y}(\cdot)$ is the prediction value, x_{t+1} denotes the ground truth and W_θ represents all trainable parameters.

4 Experiment

4.1 Dataset Description

In the experiment, we use three real-world traffic datasets, namely PeMSD4, PeMSD7 and PeMSD8, which are collected by California Performance of Transportation(PeMS) [Chen *et al.*, 2001] and widely used in the previous studies such as STGCN and ASTGCN [Yu *et al.*, 2018; Guo *et al.*, 2019]. More details for the datasets are in Table 1. The PeMS sensor network is shown in Figure 3.

PeMSD4. It refers to the traffic data in San Francisco Bay Area, with 307 sensors on 29 roads. The dataset spanned from January to February in 2018. The first 47 days are used as training set, and the remaining as validation and test set.

PeMSD7. It refers to the traffic data in District 7 of California, with 228 sensors and the time range is in the weekdays from May to June in 2012. The first 34 days are used as training set, and the remaining as validation and test set.

PeMSD8. It refers to the traffic data in San Bernardino from July to August in 2016, with 170 detectors on 8 roads. The first 50 days are used as training set, and the remaining as validation and test set.

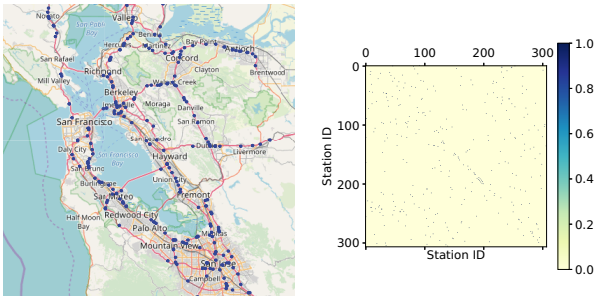

Figure 3: PeMS sensor network in District 4 of California (left). Sensor stations are denoted by dots. Heat map of weighted adjacency matrix in PeMSD4 (right).

4.2 Data Preprocessing

We set the standard time interval as 5 minutes. Therefore, every sensor contains 288 traffic data points per day. We use linear interpolation method to fill the missing values and Z-Score method to normalize the input data.

For PeMSD4 and PeMSD8, the weighted adjacency matrix W can be formed as,

$$w_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{\delta}\right), \text{if } i, j \text{ are neighbours} \\ 0\ , \text{ if } i, j \text{ aren't neighbours} \end{cases} \qquad (10)$$

where δ is the threshold to control the distribution of W and is set to be 0.1, w_{ij} is the edge weight which is related to d_{ij} (the distance between station i and j).

For PeMSD7, the weighted adjacency matrix W can be formed as,

$$w_{ij} = \begin{cases} \exp\left(-\frac{d_{ij}^2}{\delta}\right), i \neq j \text{ and } \exp\left(-\frac{d_{ij}^2}{\delta}\right) \geq \epsilon \\ 0\ , \text{ otherwise} \end{cases} \qquad (11)$$

where δ and ϵ are the thresholds to control the distribution and sparsity of W and are set to be $\delta = 0.1$ and $\epsilon = 0.5$, respectively. w_{ij} is the edge weight which is related to d_{ij} (the distance between station i and j).

4.3 Experimental Settings

All experiments are performed on a Linux server (CPU: Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, GPU: GeForce RTX 2080 Ti). The grid search strategy is executed to locate the best parameters on validations. All the tests adopt 60 minutes as the history time window, a.k.a. 12 observed data points (τ = 12) are used to forecast traffic conditions in the next 15, 30, 45 and 60 minutes (T_p = 3, 6, 9, 12).

Evaluation Metric & Baselines. We adopt Mean Absolute Errors (MAE), Mean Absolute Percentage Errors (MAPE), and Root Mean Squared Errors (RMSE) to measure the performance of different methods. The baselines are as follows.

- HA: Historical Average. Here, we use the average value of the last 12 times slices to predict the next value.
- ARIMA [Williams and Hoel, 2003]: Auto-Regressive Integrated Moving Average method, which is widely used in time series prediction.

Model	PeMSD4 (15/ 30/ 45/ 60 min)		
	MAE	MAPE (%)	RMSE
HA	2.54	5.56	4.96
ARIMA	2.51/ 2.75/ 2.96/ 3.21	5.32/ 5.69/ 6.01/ 6.56	5.72/ 6.34/ 6.85/ 7.36
DCRNN	**1.35/ 1.77/** 2.04/ 2.26	**2.68/ 3.71/** 4.48/ 5.10	2.94/ 4.06/ 4.77/ 5.28
STGCN	1.47/ 1.93/ 2.26/ 2.55	2.92/ 3.98/ 4.73/ 5.39	3.01/ 4.21/ 5.01/ 5.65
ASTGCN	2.12/ 2.42/ 2.60/ 2.73	4.16/ 4.80/ 5.20/ 5.46	3.96/ 4.59/ 4.97/ 5.21
LSGCN	1.45/ 1.82/ **2.04/ 2.22**	2.90/ 3.84/ **4.42/ 4.85**	**2.93/ 3.92/ 4.47/ 4.83**
Model	PeMSD7 (15/ 30/ 45/ 60 min)		
	MAE	MAPE (%)	RMSE
HA	4.01	10.61	7.20
ARIMA	5.57/ 5.94/ 6.27/ 6.68	13.04/ 14.01/ 15.01/ 16.78	9.00/ 9.22/ 9.43/ 9.68
DCRNN	2.22/ 3.04/ 3.64/ 4.15	5.16/ 7.46/ 9.26/ 10.82	4.25/ 6.02/ 7.24/ 8.20
STGCN	2.24/ 3.04/ 3.61/ 4.08	5.28/ 7.46/ 9.00/ 10.23	4.01/ 5.74/ 6.85/ 7.69
ASTGCN	2.85/ 3.35/ 3.70/ 3.96	7.25/ 8.67/ 9.73/ 10.53	5.15/ 6.12/ 6.77/ 7.20
LSGCN	**2.22/ 2.96/ 3.43/ 3.81**	**5.14/ 7.18/ 8.51/ 9.60**	**3.98/ 5.47/ 6.39/ 7.09**
Model	PeMSD8 (15/ 30/ 45/ 60 min)		
	MAE	MAPE (%)	RMSE
HA	1.98	3.94	4.11
ARIMA	1.90/ 2.12/ 2.43/ 2.79	5.11/ 5.21/ 5.46/ 5.62	4.87/ 5.24/ 5.63/ 6.22
DCRNN	1.17/ 1.49/ 1.71/ 1.87	2.32/ 3.21/ 3.83/ 4.28	2.59/ 3.56/ 4.13/ 4.50
STGCN	1.19/ 1.59/ 1.92/ 2.25	2.34/ 3.24/ 3.91/ 4.54	2.62/ 3.61/ 4.21/ 4.68
ASTGCN	1.49/ 1.67/ 1.81/ 1.89	3.16/ 3.59/ 3.98/ 4.22	3.18/ 3.69/ 3.92/ 4.13
LSGCN	**1.16/ 1.46/ 1.66/ 1.81**	**2.24/ 3.02/ 3.51/ 3.89**	**2.45/ 3.28/ 3.75/ 4.11**

Table 2: Performance comparison of different approaches on the datasets PeMSD4, PeMSD7 and PeMSD8.

- DCRNN [Li *et al.*, 2018]: Diffusion Convolution Recurrent Neural Network, which combines graph convolution with recurrent neural networks in an encoder-decoder manner.
- STGCN [Yu *et al.*, 2018]: Spatial-Temporal Graph Convolutional Network, which combines graph convolution with gated temporal convolution.
- ASTGCN [Guo *et al.*, 2019]: Attention based Spatial-Temporal Graph Convolution Network, which combines the spatial-temporal attention mechanism and the spatial-temporal convolution.

LSGCN Model. The batch size of PeMSD4, PeMSD7 and PeMSD8 are 32, 32 and 16, respectively. The hyperparameters are set as follows. For all datasets, the channels of the first GLU, GCN, cosAtt, the second GLU are 32, 32, 32 and 64, respectively. We set both the graph convolution kernel size K and GLU convolution kernel size K_t to 3. Our model is trained by minimizing the mean square error with RMSprop optimizer for 60 epochs. The initial learning rate is set as 10^{-3} with a decay rate of 0.7 after every 5 epochs.

4.4 Experiment Results

As shown in Table 2, LSGCN performs well in both long-term and short-term prediction for three evaluation metrics. In detail, in the three datasets, it performs very well, especially in the long-term prediction of PeMSD4, both long-term prediction and short-term prediction of PeMSD7 and PeMSD8. In the short-term prediction of PeMSD4, it also achieves the second best. The results show that LSGCN can efficiently capture the spatial-temporal features for the prediction tasks. In particular, STGCN is also based on the mechanism of iterative prediction, but it causes more accumulation errors than LSGCN since it consists of more network structure layers.

We can easily observe that the traditional statistical methods (i.e., HA and ARIMA) often have poor performance since they cannot efficiently handle the complex spatial-temporal data.

In general, since there are lots of accumulated errors in STGCN for the prediction tasks, ASTGCN and DCRNN have better performance than STGCN. In detail, STGCN is not better than DCRNN in the short-term tasks and not better than ASTGCN in the long-term tasks. DCRNN adopts diffusion convolution which is beneficial to capture complex spatial topology information. Besides, it can accurately calculate the probability from the target sensor to neighbour sensors by using a finite step random walk to express the diffusion process. Thus, DCRNN has the best performance in the metrics of MAE and MAPE for the complex traffic networks of PeMSD4. But, DCRNN also adopts GRU structure for time series which often causes information loss in the long-term tasks.

ASTGCN considers the periodicity in prediction tasks and various traffic characteristics related to the tasks. So, it performs well in the long-term prediction of PeMSD4 and PeMSD8. But, ASTGCN is not better than STGCN in the long-term prediction of PeMSD7. This is because that the neighbor information is beneficial to the prediction tasks, which is removed by the threshold from PeMSD7, and saved in PeMSD4 and PeMSD8.

Test in extreme case. In particular, we test our LSGCN in some extreme cases, namely morning peak and evening rush

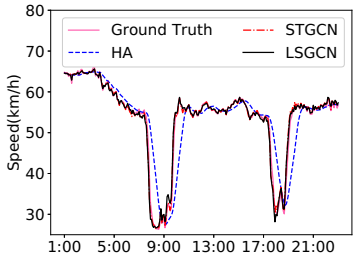

Figure 4: Speed prediction in the morning peak and evening rush hours of the dataset PeMSD4.

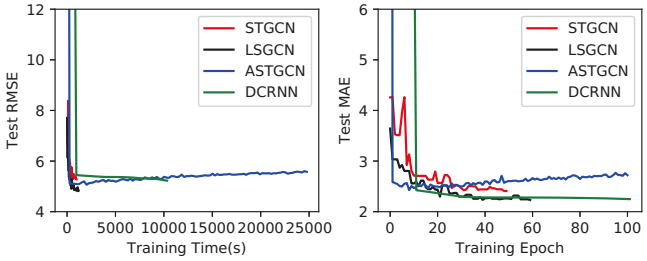

Figure 5: Test RMSE versus the training time(left); Test MAE versus the number of training epochs(right).(PeMSD4)

hours. As shown in Figure 4, Due to the efficient spatial gated block, LSGCN is capable of fast response to the dynamic changes in extreme cases. The trend of rush hours captured by LSGCN is more accurate than the other methods. Besides, the ending of the rush hours detected by LSGCN is earlier than the others.

Training Efficiency and Generalization. We plot the RMSE and MAE of the test set of PeMSD4 during the training process to further investigate the performance of different deep learning models. As shown in Figure 5, it is obvious that our LSGCN can achieve faster training and easier convergence than DCRNN and ASTGCN. Specifically, LSGCN and STGCN need almost the same training time, about 1,000 seconds, whereas DCRNN and ASTGCN need about 10,000 and 25,000 seconds, respectively. This is because that more parameters are used for the training process of DCRNN and ASTGCN.

Effect of cosAtt. By replacing the proposed cosAtt of LSGCN with traditional graph attention, we can obtain a modified version called LSGCN(GAT). We compare LSGCN and LSGCN(GAT) to test the changes in prediction results. For each prediction task, both methods are executed 10 times with the same hyperparameters. Then, the maximum and minimum values among all evaluation results for each metric are reported, respectively. As shown in Table 3, the metric value changes for LSGCN are often smaller than LSGCN(GAT), so cosAtt enables more stable prediction results.

5 Related Work

Deep learning models. Some solutions for traffic prediction are proposed based on deep learning models. Zhang et al. [2016] convert the road network to a regular 2-D grid and apply CNN to capture adjacent relations among the traffic network. Yao et al. [2018] proposed a method to predict traffic by integrating CNN and long-short term memory (LSTM) to jointly model both spatial and temporal dependencies. Different from these methods, both GCN and GAT are integrated in our proposed model.

	min	MAPE (%)	MAE	RMSE
LSGCN (GAT)	15	2.37±0.06	1.19±0.02	2.62±0.08
	30	3.11±0.08	1.49±0.02	3.46±0.07
	45	3.55±**0.09**	1.68±**0.03**	3.88±0.08
	60	3.87±0.12	1.82±0.06	4.16±0.10
LSGCN	15	2.23±**0.04**	1.17±**0.01**	2.47±**0.03**
	30	2.96±**0.08**	1.48±**0.02**	3.28±**0.05**
	45	3.43±0.10	1.69±0.03	3.77±**0.07**
	60	3.80±**0.11**	1.85±**0.05**	4.14±**0.08**

Table 3: Prediction value changes on PeMSD8

Graph convolutional networks(GCN). Bruna et al. [2013] propose GCN based on the spectral graph theory. Defferard et al. [2016] and Kipf et al. [2016] propose some improve methods for the computation of GCN. Based on GCN, Yu et al. [2018] propose STGCN to tackle the time series prediction problem in the traffic domain. Wu et al.[2019] propose Graph Wavenet to precisely capture the hidden spatial dependency in the data. In addition to GCN, our model contains an additional GAT which is different from these methods.

Attention mechanism. It has been widely utilized in various domains such as natural language processing, speech recognition and image caption [Vaswani *et al.*, 2017; Shen *et al.*, 2018]. Recently, researchers apply attention mechanism to graph [Velickovic *et al.*, 2017; Liu *et al.*, 2018] and traffic prediction. Liang et al. [2018] propose a novel multi-level attention-based network. Guo et al. [2019] propose ASTGCN and Zheng et al. [2019] propose GMAN. Different from these methods, our graph attention is based on the road condition similarity.

6 Conclusion

In this paper, we propose a new graph convolutional networks model LSGCN for long and short-term traffic prediction. In LSGCN, we integrate both a new graph attention network cosAtt and GCN to precisely capture the spatial features and at the same adopt the GLU to capture the temporal features. The experiments on real traffic networks verify the effectiveness of LSGCN. In the future, we will consider our proposed model for more general spatio-temporal structured sequence forecasting such as preference prediction in recommendation systems.

Acknowledgements

This paper is supported by the National Nature Science Foundation of China (61572537, U1501252).

References

[Ahmed and Cook, 1979] Mohammed S Ahmed and Allen R Cook. *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*. Number 722. 1979.

[Bruna *et al.*, 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Chen *et al.*, 2001] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. Freeway performance measurement system: mining loop detector data. *Transportation Research Record*, 1748(1):96–102, 2001.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv: Neural and Evolutionary Computing*, 2014.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[Guo *et al.*, 2019] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, volume 33, pages 922–929, 2019.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Li *et al.*, 2018] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*, 2018.

[Liang *et al.*, 2018] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI*, pages 3428–3434, 2018.

[Liu *et al.*, 2018] Lingbo Liu, Ruimao Zhang, Jiefeng Peng, Guanbin Li, Bowen Du, and Liang Lin. Attentive crowd flow machines. pages 1553–1561, 2018.

[Ma *et al.*, 2017] 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.

[Ostring and Sirisena, 2001] Sven AM Ostring and Harsha Sirisena. The influence of long-range dependence on traffic prediction. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240)*, volume 4, pages 1000–1005. IEEE, 2001.

[Shen *et al.*, 2018] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, 2018.

[Shuman *et al.*, 2013] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[Velickovic *et al.*, 2017] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv: Machine Learning*, 2017.

[Vlahogianni *et al.*, 2005] Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach. *Transportation Research Part C: Emerging Technologies*, 13(3):211–234, 2005.

[Williams and Hoel, 2003] Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.

[Wu and Tan, 2016] Yuankai Wu and Huachun Tan. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework. *arXiv preprint arXiv:1612.01022*, 2016.

[Wu *et al.*, 2019] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. *IJCAI*, 2019.

[Xingjian *et al.*, 2015] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *NIPS*, 2015.

[Yao *et al.*, 2018] Huaxiu Yao, Xianfeng Tang, Hua Wei, Guanjie Zheng, Yanwei Yu, and Zhenhui Li. Modeling spatial-temporal dynamics for traffic prediction. *arXiv preprint arXiv:1803.01254*, 2018.

[Yu *et al.*, 2018] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

[Zhang *et al.*, 2016] Junbo Zhang, Yu Zheng, Dekang Qi, Ruiyuan Li, and Xiuwen Yi. Dnn-based prediction model for spatio-temporal data. In *ACM SIGSPATIAL*, page 92, 2016.

[Zheng *et al.*, 2019] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. Gman: A graph multi-attention network for traffic prediction. *arXiv preprint arXiv:1911.08415*, 2019.