2019 IEEE Intelligent Transportation Systems Conference (ITSC)
Auckland, NZ, October 27-30, 2019

A Hybrid Deep Learning Approach with GCN and LSTM for Traffic Flow Prediction*

Zhishuai Li, Gang Xiong, Yuanyuan Chen, Yisheng Lv†, Bin Hu, Fenghua Zhu and Fei-Yue Wang

Abstract— **Traffic flow prediction is an important functional component of Intelligent Transportation Systems (ITS). In this paper, we propose a hybrid deep learning approach, called graph and attention-based long short-term memory network (GLA), to efficiently capture the spatial-temporal features in traffic flow. Firstly, we apply graph convolutional network (GCN) to mine the spatial relationships of traffic flow over multiple observation stations, in which the adjacent matrix is determined by a data-driven approach. Then, we feed the output of the GCN model to the long short-term memory (LSTM) model which extracts temporal features embedded in traffic flow. Further, we implement a soft attention mechanism on the extracted spatial-temporal traffic features to make final prediction. We test the proposed method over the PeMS data sets. Experimental results show that the proposed model performs better than the competing methods.**

I. Introduction

ACCURATE and effective traffic flow prediction is crucial for traffic management and control, which can help alleviate urban traffic congestion, save energy and reduce emissions [1]–[3]. Traffic flow prediction has a long research history, and various traffic flow prediction methods have been developed.

Recently, deep learning methods are widely used in traffic prediction and have achieved good performance. Lv *et al.* used stacked autoencoders to extract spatial-temporal traffic flow features and make traffic flow prediction [4]. The long short-term memory (LSTM) models can capture temporal dependencies of time series data [5]. Fu *et al.* applied the LSTM model and its variant GRU model [6] to predict short-term traffic flow [7]. Yu *et al.* proposed a graph convolutional network model (GCN) to tackle the traffic prediction problem [8]. Chen *et al.* used convolutional neural networks (CNN) to predict traffic flow where they used time series folding and multi-grained learning techniques [9]. Duan *et al.* combined CNN and LSTM networks to predict traffic flow with trajectory data [10].

Traffic flow dynamics is a typical spatial and temporal process. Therefore, it has both spatial features and temporal features. In this paper, we propose a hybrid deep learning framework called graph and attention-based long short-term memory network (**GLA**) to learn both the spatial and temporal features of traffic flow for prediction. More specifically, we use the graph convolutional network (GCN) to extract spatial relationships of traffic flow between observation locations, in which the adjacent matrix $\mathcal{A}$ is composed of trainable parameters and obtained through the learning over data sets. The GCN is then connected to the LSTM module which captures the temporal relations of traffic flow. Also, we use the soft attention mechanism when designing the proposed hybrid network. We test the proposed method on the PeMS data sets and experiments show that the proposed method performs better than some commonly used traffic flow prediction approaches.

The contribution of this paper is threefold.

- We propose a hybrid deep learning framework for traffic flow prediction, in which GCN is to capture the spatial relationships of traffic flow between adjacent traffic observation stations, and the LSTM model is to capture temporal dependency of traffic flow data. Thus, the proposed model inherently considers the spatial and temporal correlations of traffic flow.
- A data-driven approach is used to learn the adjacent matrix $\mathcal{A}$, which represents an influence metric of traffic flow among traffic observation stations.A soft attention mechanism is added to capture important features and thus assign different weights to the elements of output vectors.
- We compare the proposed method with existing popular methods on traffic flow prediction, and experimental results show that the proposed method has superior performance.

*This work was supported partially by the National Natural Science Foundation of China under Grants U1811463, 61773381, 61876011 and 61533019; Dongguan's Innovation Talents Project (Gang Xiong); China Railway under Grant 2017X001-B; the 2018-2019 joint project between Beijing Ten Dimensions Technology Co. Ltd. and CASIA; Chinese Guangdong's S&T project (2017B090912001); 2017 Special Cooperative Project of Hubei Province and Chinese Academy of Sciences.

Zhishuai Li is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. He is also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, 100049, China.

Gang Xiong is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China. He is also with the Cloud Computing Center, Chinese Academy of Sciences, Dongguan, 523808, China.

Yuanyuan Chen is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

Yisheng Lv is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

Bin Hu is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

Fenghua Zhu is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

Fei-Yue Wang is with the State Key Laboratory for Management and Control of Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China.

†Corresponding Author. E-mail: yisheng.lv@ia.ac.cn

The rest of this paper is organized as follows. Section II introduces the proposed hybrid deep learning model. Section III presents the experimental results. Section IV concludes this paper.

II. Methodology

In this section, we present the proposed hybrid deep learning approach for traffic flow prediction. We begin by describing the problem definition of traffic flow prediction, and then we introduce the basics of GCN and soft attention mechanism. Finally, we present the framework of the proposed method, which is the combination of the GCN, the LSTM module, and the soft attention mechanism.

A. Problem Definition

The traffic flow prediction task can be described as follows. Given the sequence of observed traffic flow $[X_{t-S}, \ldots, X_{t-2}, X_{t-1}]$, the task is to predict traffic flow X_t at time step t, which can be written as (1).

$$X_t = \mathcal{F}([X_{t-S}, \ldots, X_{t-2}, X_{t-1}]) \tag{1}$$

Here, S is called window size, i.e. the size of previous time steps. X_t is a vector representing traffic flow data at time step t across observation stations.

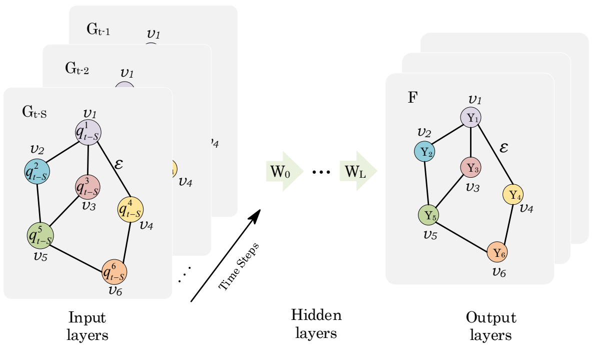


Fig. 1: Schematic depiction of regression problem on graph with S input channels and F feature maps in output layers.

As Fig. 1 shows, the traffic observation stations can be described as a graph $G = \{q, \mathcal{E}, \mathcal{V}, \mathcal{A}\}$. $\mathcal{V}$ is the vertex set denoting observation stations, and $q \in \mathbb{R}$ is a scalar associated with every vertex $\nu_i \in \mathcal{V}$. Suppose the number of detector stations is $\mathcal{N}_s$. Then we can obtain that the feature vector of traffic flow at time step t is $X_t = \{q_t^1, q_t^2, \ldots, q_t^{\mathcal{N}_s}\}$. The traffic flow relationship between adjacent observation s-tations can be described as the interactions between vertexes, which is denoted as the adjacent matrix $\mathcal{A} \in \mathbb{R}^{\mathcal{N}_s \times \mathcal{N}_s}$. Each entry $\mathcal{A}_{ij}$ represents the influence degree of two stations with a directed edge $\varepsilon \in \mathcal{E}$ from ν_i to ν_j.

B. GCN to Capture Spatial Information

Traffic flow prediction benefits from the use of both temporal information and neighboring traffic information. Previous studies have shown that using traffic flow data from neighboring observation stations can help improve traffic prediction [11]. In the proposed method, we use GCN to extract the spatial relationships of traffic flow among observation stations. We introduce the GCN model as follows.

The Laplace matrix $\mathcal{L}$ for a graph can be defined as (2).

$$\mathcal{L} = D^{-1/2}(D - \mathcal{A})D^{-1/2} = I_N - D^{-1/2}\mathcal{A}D^{-1/2} \tag{2}$$

Where I_N is the identity matrix with the size of $N \times N$. The degree matrix is defined as $D_{ii} = \sum_j \mathcal{A}_{ij}$.

The eigendecomposition of the matrix $\mathcal{L}$ can be described as $\mathcal{L} = U\Lambda U^T$, where $\Lambda = \text{diag}\left([\lambda_0, \lambda_1, \cdots, \lambda_{N-1}]\right)$ and λ_i is the eigenvalue of $\mathcal{L}$. U is the matrix whose columns are the eigenvectors of $\mathcal{L}$.

Generalizing convolutions to the graph domain has an increasing interest, and one approach is called the spectral method [12], [13]. The convolution operation is done in the Fourier domain and defined as multiplication of an input signal x with a filter g, as (3).

$$g * x = U\left((U^T g) \odot (U^T x)\right) = U g_\theta(\Lambda) U^T x \tag{3}$$

That is, the input x is converted into $U^T x$ which falls into the spectral space U whose basis is $[u_0, u_2, \ldots, u_{N-1}]$. Here $\odot$ represents the hadamard product, and $g_\theta(\Lambda) = U^T g = diag(\theta)$, where $\theta \in \mathbb{R}^N$. Generally, the computation overhead of the convolution kernel $g_\theta(\Lambda)$ is expensive (the computational complexity is $\mathrm{O}(N^3)$). Therefore, some approximate methods are proposed such as polynomials and Chebyshev polynomials [12]. The calculation of $g * x$ can be simplified by using only the first-order polynomial [13], which is shown as (4).

$$g * x \approx \theta\left(I_N + D^{-1/2}\mathcal{A}D^{-1/2}\right)x \tag{4}$$

We use (4) for the approximate calculation in this paper. We know that: $I_N + D^{-1/2}\mathcal{A}D^{-1/2} = \tilde{D}^{-1/2}\tilde{\mathcal{A}}\tilde{D}^{-1/2}$, with $\tilde{\mathcal{A}} = \mathcal{A} + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{\mathcal{A}}_{ij}$. (This is also called the renormalization trick.) Therefore, the output of the lth layer $H^{(l)}$ can be written as (5).

$$H^{(l)} = \sigma\left(\tilde{D}^{-1/2}\tilde{\mathcal{A}}\tilde{D}^{-1/2}H^{(l-1)}\Theta^{(l-1)}W^{(l)}\right) \tag{5}$$

Where σ is the activation function such as sigmoid. Let $\Theta^{(l-1)} \times W^{(l)}$ be $\tilde{W}^{(l)}$, where $\Theta^{(l-1)} \in \mathbb{R}^{C^{(l\ 1)} \times F^{(l\ 1)}}$, $W^{(l)} \in \mathbb{R}^{F^{(l\ 1)} \times C^{(l)}}$ and $\tilde{W}^{(l)} \in \mathbb{R}^{C^{(l\ 1)} \times C^{(l)}}$. $C^{(l-1)}$ is the dimension of the $(l-1)$th layer's output and $F^{(l-1)}$ is the size of feature vectors in each dimension. Then, (5) can be rewritten as (6).

$$H^{(l)} = \sigma\left(\tilde{D}^{-1/2}\tilde{\mathcal{A}}\tilde{D}^{-1/2}H^{(l-1)}\tilde{W}^{(l)}\right) \tag{6}$$

An intuitive way to determine $\tilde{\mathcal{A}}$ is to use the distance between two stations. Many researchers used a threshold Gaussian kernel method to calculate $\tilde{\mathcal{A}}$ [8] [14] . However, such a simplified way cannot easily represent traffic flow relationships appropriately. Here, let $\hat{\mathcal{A}} = \tilde{D}^{-1/2}\tilde{\mathcal{A}}\tilde{D}^{-1/2}$, then (6) can be rewritten as (7), and the entries in $\hat{\mathcal{A}}$ can be learned from data. That is, $\hat{\mathcal{A}}$ is a matrix composed of trainable parameters.

$$H^{(l)} = \sigma\left(\hat{\mathcal{A}}H^{(l-1)}\tilde{W}^{(l)}\right) \tag{7}$$

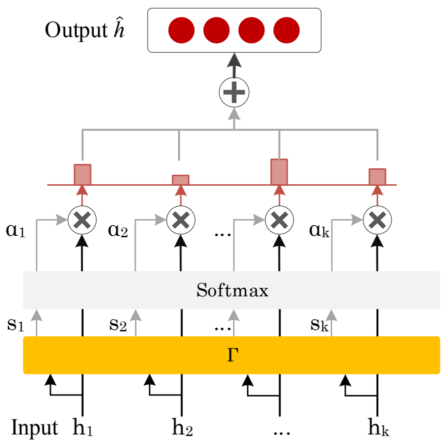


Fig. 2: The diagrammatic view of soft attention.

C. Attention to Assign Weights of Sequence Output

Inspired by recent successes of attention mechanism in natural language analysis and image processing, we use a soft attention mechanism in the proposed traffic flow prediction methods [15]–[17]. In our proposed framework, the output of GCN is fed into the LSTM module, and then the soft attention mechanism is implemented over the output of the LSTM, which helps the model focus on more important features and make the prediction more accurate.

The diagrammatic view of the soft attention mechanism is shown in Fig. 2. Suppose there are k feature vectors with d dimensions, denoted as $h_i = \{h_i^1, h_i^2, \ldots, h_i^d\}(i = 1, 2, \ldots, k)$. The output $\hat{h}$ (also with d dimensions) is calculated in a weighted average way as follows.

$$\hat{h} = \sum_{i=1}^{k} \alpha_i h_i \tag{8}$$

Where α_i is the weight. h_i needs to be scored to evaluate its effect on $\hat{h}$. We train a fully connected network to calculate a score s_i for each h_i, and it should be noted that other functions except neural networks also can be used. The output of the network is (9).

$$s_i = \Gamma(h_i) = \tanh\left(w^T h_i + b_i\right) \tag{9}$$

Where s_i represents the correlation coefficient between h_i and $\hat{h}$. Then we use the softmax function to normalize the score s_i and get the final weight α_i, as (10).

$$\alpha_i = \mathrm{softmax}\,(s_i) = \mathrm{softmax}\,(\Gamma(h_i)) \tag{10}$$

The attention mechanism can be seen as producing a fixed-length embedding $\hat{h}$ of the input sequence h_i by computing an adaptive weight α_i.

D. The Hybrid GLA Network

The proposed hybrid deep learning network contains the graph convolution network, the LSTM model, and the soft attention mechanism, which is shown in Fig. 3.

The GCN is to extract spatial features and relationships of traffic flow between observation stations mapped into a graph. Then the output of the GCN is fed into the LSTM module which has the good ability of capture temporal dependencies. Finally, the soft attention mechanism is put over the output of the LSTM module, which further combines the spatial and temporal information.

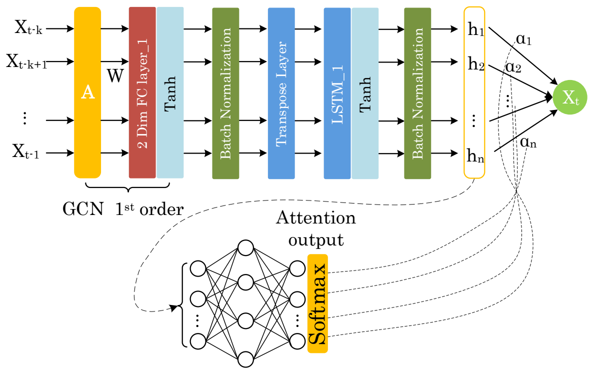


Fig. 3: The architecture of the proposed GLA model.

III. Experiments

A. Dataset Description and model training

We test the proposed hybrid model on the data sets extracted from Caltrans Performance Measurement system (PeMS) database. The collected traffic flow data is aggregated in 5-min interval. In the experiment, we set the vertexes of input graph as 100. So the particular data sets used in this paper are collected from 100 detector stations, whose identification numbers are in the range from No.40000 to No.40171 (the station ID is not continuous) in District 4. The sampling period is from April 1, 2016 to August 31, 2016. We regard each detector station as a vertex, then map the observation stations and their neighboring relationship as a graph. Thus, we get the input of the GCN model. The training algorithm is illustrated in Algorithm 1.

B. Index of Performance

In this paper, three commonly used performance indexes are introduced to compare traffic prediction models.

- Root mean square error (RMSE):

$$\mathrm{RMSE} = \left[\frac{1}{\mathcal{N}_s \times n} \sum_{i=1}^{\mathcal{N}_s \times n} (y_i - \hat{y}_i)^2\right]^{1/2} \tag{11}$$

- Mean absolute error (MAE):

$$\mathrm{MAE} = \frac{1}{\mathcal{N}_s \times n} \sum_{i=1}^{\mathcal{N}_s \times n} |y_i - \hat{y}_i| \tag{12}$$

- Mean absolute percentage error (MAPE):

$$\mathrm{MAPE} = \frac{1}{\mathcal{N}_s \times n} \sum_{i=1}^{\mathcal{N}_s \times n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{13}$$

where y_i is the observed traffic flow and $\hat{y}_i$ is the predicted traffic flow. For the MAPE criteria, we only consider the cases when $y_i > 0$.

Algorithm 1 Pseudo-Code of the model Training: $S_l = 40, \mathcal{N}_s = 100, M = 43990, BatchSize = 256$

```
Input: The window size $S_l$, the number of traffic detector
    stations $\mathcal{N}_s$, traffic flow data represented as a matrix $\mathbb{C} \in \mathbb{R}^{\mathcal{N}_s \times M}$.
output: The trained model and the adjacent matrix $\mathcal{A}$.
 1: % Construct data sets $\mathbb{D}$.
 2: $\mathbb{D} \Leftarrow \phi$
 3: All data is scaled range from 0 to 1 for normalization.
 4: for $m < M$ do
 5:     $X_m \Leftarrow [q_m^1, q_m^2, \dots, q_m^{\mathcal{N}_s}]$
 6:     $m \Leftarrow m + 1$.
 7: end for
 8: for $X_{t-S_l}, X_t \in \mathbb{C}$ do
 9:     $\mathbb{D} \Leftarrow \{\mathcal{X} = [X_{t-S_l}, .., X_{t-2}, X_{t-1}], \mathcal{Y} = X_t\}$
10:     $t \Leftarrow t + 1$.
11: end for
12: % Training the model.
13: Construct the network model as shown in Fig. 3.
14: Initialize adjacent matrix $\mathcal{A}$, network parameters $\Theta$.
15: Divide $\mathbb{D}$ into training set $\mathbb{D}_1$ and test set $\mathbb{D}_2$.
16: repeat
17:     A batch of data is randomly selected from $\mathbb{D}_1$ and
        put into the network;
18:     Update the parameters $\mathcal{A}$, $\Theta$ and minimize the MSE
        between $\mathcal{Y}_{pred}$ and $\mathcal{Y}$, using the Adam algorithm;
19: until "The stop criterion is met"
20: return The trained model, the adjacent matrix
```

C. Model structure

The best hyper-parameters are chosen by using the Tree-structured Parzen Estimator (TPE) [18]. The search ranges for hyper-parameters of the proposed model are set as follows: the ranges of hidden layer nodes are within $[32, 64, 100, 128, 150]$, and the window size are selected from $[20, 30, 40, 50]$. Keras [19] is used to build and train the proposed model, and the mean square error (MSE) is set as the loss function. Finally, the best window size is 40 and the best parameters of the proposed architecture are determined, which is shown in Table I.

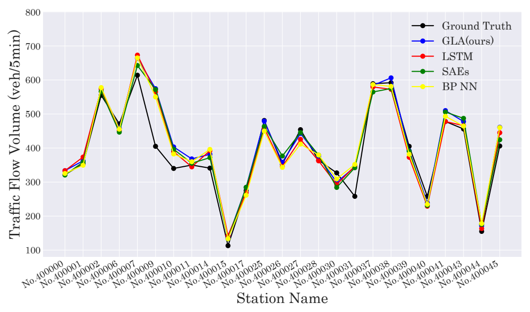
To illustrate the advantages of our model, we compare the proposed model with back propagation neural network (BPNN), LSTM and stacked auto-encoders (SAE). All these three competing methods are finely tuned. During the training process, Batch Normalization (BN) [20] and early stop techniques are applied. All experiments are performed on a desktop with an Intel Core i7-8700 CPU and a Nvidia GeForce GTX 1060 (3G) Graphics Card.

TABLE I: Hypter-parameters of our proposed models

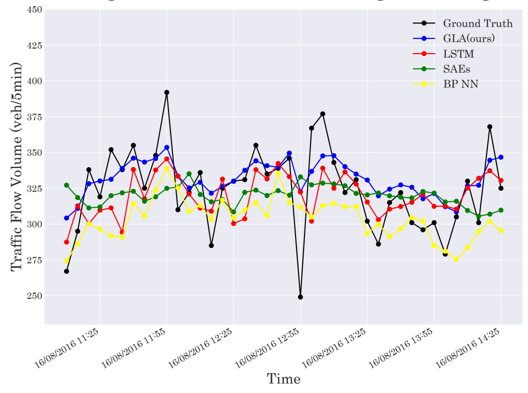
Layer	**Output Shape**	**Hidden Nodes**
GCN	100×64	100×64
Batch normalization	100×64	0
Transpose Layer	64×100	0
LSTM	64×100	100
Batch normalization	64×100	0
Attention	1×100	64

TABLE II: Performance of multiple prediction

Model Name	RMSE	MAE	MAPE%
BP NN	31.39	21.37	13.13
SAE	31.72	20.94	11.94
LSTM	30.88	20.93	11.95
GLA(ours)	**29.41**	**19.95**	**11.79**



(a) Prediction results on first 25 in 100 detector stations at $12:00$, August 16, 2016 (The middle prediction point)

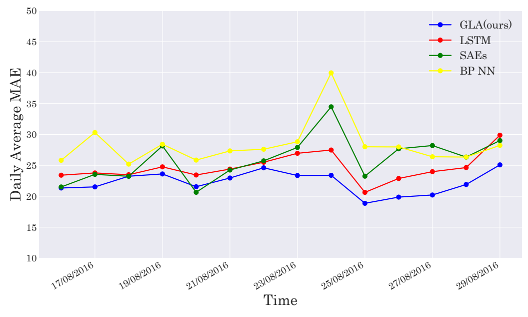


(b) Prediction results from $10:10$, August 16, 2016 to $13:25$, August 16, 2016 on No.40000 detector station

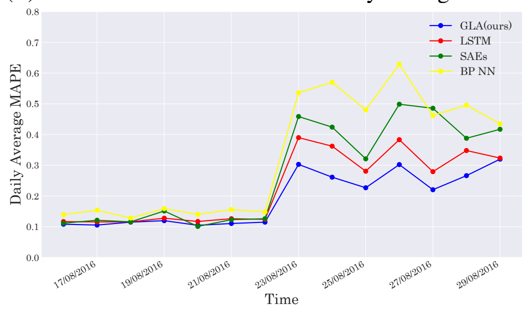
Fig. 4: Performance of prediction on test set.

D. Results

We perform 5-min traffic flow prediction task over multiple detector stations (100 stations). The experimental results are shown in Table II. The best results are marked in bold in the table. As can be seen from the table, the GLA method has advantages over the BPNN, the SAE and the LSTM models. The proposed method performs about 4.76%, 4.68% and 1.34% better than the LSTM model, and about 7.28%, 4.73% and 1.28% better than the SAE model for RMSE, MAE and MAPE, respectively.



(a) Performance of models over daily average MAE



(b) Performance of models over daily average MAPE

Fig. 5: Performance of prediction in No.40000 on test set.

Fig. 4 shows the prediction results of the proposed method and competing methods over the data collected on August 16, 2016. We just showed the prediction results over the first 25 detector stations at 12:00, August 16, 2016 in Fig. 4(a). The horizontal axis shows the detector station IDs, while the vertical axis shows the count of vehicles per 5 minutes. In Fig. 4(b), we plot the prediction results on the detector station with No.40000 from 10:10, August 16, 2016, to 13:25, August 16, 2016. It is clearly shown that the GLA model is more accurate than the competing methods.

Fig. 5 illustrated the prediction results on a detector station with No.40000, in which the daily average MAE and MAPE (14 days) as performance indicators. It is also clearly shown that the proposed method performs better.

IV. Conclusion

In this paper, we propose a hybrid deep learning method for traffic flow prediction. The proposed method takes into account both temporal features and spatial features of traffic flow. We use the graph convolutional network to capture spatial interactions of traffic flow over multiple observation stations (100 stations in this paper), where the adjacent matrix $\mathcal{A}$ is learned from data. The output of the GCN model is fed into the LSTM model to capture temporal dependencies of traffic flow. A soft attention mechanism is implemented to combine the extracted spatial and temporal features. We evaluate the performance of the proposed method on the PeMS data sets and compare it with the BP neural network model, the SAE model, and the LSTM model. Experimental results show that the proposed method performs better than the competing methods. We believe there is still room to improve the accuracy of the proposed framework, which is our future work.

References

[1] F. Wang, "Parallel Control and Management for Intelligent Transportation Systems: Concepts, Architectures, and Applications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 630–638, 2010.

[2] J. Zhang, F. Y. Wang, K. Wang, W. H. Lin, X. Xu, and C. Chen, "Data-Driven Intelligent Transportation Systems: A Survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, 2011.

[3] Y. Lv, Y. Chen, X. Zhang, Y. Duan, and N. L. Li, "Social media based transportation research: The state of the work and the networking," *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 1, pp. 19–26, 2017.

[4] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Y. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[7] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Chinese Association of Automation*, 2017, pp. 324–328.

[8] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting," in *International Joint Conference on Artificial Intelligence*, 2018, pp. 3634–3640.

[9] M. Chen, X. Yu, and Y. Liu, "PCNN: Deep Convolutional Networks for Short-Term Traffic Congestion Prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3550–3559, 2018.

[10] Z. Duan, Y. Yang, K. Zhang, Y. Y. Ni, and S. Bajgain, "Improved Deep Hybrid Networks for Urban Traffic Flow Prediction using Trajectory Data," *IEEE Access*, vol. 6, no. 99, pp. 1–1, 2018.

[11] D. Kang, Y. Lv, and Y. Chen, "Short-term traffic flow prediction with lstm recurrent neural network," in *International Conference on Intelligent Transportation Systems*, 2017, pp. 1–6.

[12] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[13] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," *CoRR*, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[14] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[15] C. Raffel and D. P. W. Ellis, "Feed-Forward Networks with Attention Can Solve Some Long-Term Memory Problems," *CoRR*, vol. abs/1512.08756, 2015. [Online]. Available: http://arxiv.org/abs/1512.08756

[16] H. Li, M. R. Min, Y. Ge, and A. Kadav, "A Context-aware Attention Network for Interactive Question Answering," *CoRR*, vol. abs/1612.07411, 2016. [Online]. Available: http://arxiv.org/abs/1612.07411

[17] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration," in *International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 2051–2059.

[18] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[19] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[20] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.