A Deep Learning Traffic Forecasting Based on Edge-Based Spatial Temporal Graph Convolution Networks

1st Zexin Xia
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
xiazexin@sjtu.edu.cn

2nd Puyue Hou
School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, China
houpy2016@sjtu.edu.cn

***Abstract*—Citizens are beginning to attach much more importance to the traffic problems. A good traffic environment can have a positive effect on building a city brand. When it comes to traffic prediction, accuracy, timeliness and efficiency of prediction need to be taken into consideration.This paper is the project proposal of the course "wireless communication and mobile Internet" of Shanghai Jiaotong university, aiming to set up a reasonable and more accurate training model and forecast traffic flow through machine learning methods. In this paper, a new deep learning method has been proposed based on spatial temporal graph convolution networks (STGCN), called edge-based spatial temporal graph convolution networks.**

***Index Terms*—traffic flow, EB-STGCN,traffic jam forecast, deep learning, intelligent transportation system**

I. Introduction

Intelligent transportation system (ITS) is an essential component of intelligent city, especially in modern cities. In recent years, many researchers have studied in the field of traffic flow detection or traffic flow prediction. However, previous researchers have often used historical traffic data from years ago or hardware installed on the ground to obtain traffic flow information.

To take full advantage of spatial features, some researchers use convolutional neural network (CNN) to capture adjacent relations among the traffic network, along with employing recurrent neural network (RNN) on time axis. By combining long short-term memory (LSTM) network [Hochreiter and Schmidhuber, 1997] and 1-D CNN, Wu and Tan [2016] presented a feature-level fused architecture CLTFP for shortterm traffic forecast. Although it adopted a straightforward strategy, CLTFP still made the first attempt to align spatial and temporal regularities. Afterwards, Shi et al. [2015] proposed the convolutional LSTM, which is an extended fullyconnected LSTM (FC-LSTM) with embedded convolutional layers. However, the normal convolutional operation applied restricts the model to only process grid structures (e.g. images, videos) rather than general domains. Meanwhile, recurrent networks for sequence learning require iterative training, which introduces error accumulation by steps. Additionally, RNN-based networks (including LSTM) are widely known to be difficult to train and computationally heavy.

There are several recent deep learning frameworks for this problem, such as DC-RNN and ST-GCN. Most of the previous studies neglect spatial attributes of traffic networks, while researchers from PKU propose a new framework, STGCN, which utilize graph convolution to extract spatial features of traffic networks. Our work is based on this framework.

For overcoming these issues, we first establish our own model based on some public APIs such as Gaode map. For the API, we can directly get average velocity speed of each road at any time to set up a detailed enough dataset. In this data set, we collected traffic data every two minutes, and collected a month's traffic data in total, which provided a basis for model training and traffic prediction.

In order to better use of space and time characteristics of traffic data, the model of training in our model is based on Spatial-Temporal Graph convolution networks. But since people often use sections to show whether a road is congested or not, rather than a latitude and longitude coordinates or a certain point, we put forward a new figure convolution model based on the edge of space and time, called edge-based spatial-temporal graph convolution networks(EB-STGCN).

II. Preliminary

A. traffic prediction

Traffic forecasting is a typical time-series prediction problem. At each time step, the traffic condition can be represented by a graph, which is a set of vertices, corresponding to the observation. Our purpose is predicting the most likely traffic speed in the next H time steps based on the previous M traffic observations.

$$\hat{v}_{t+1}, ...\hat{v}_{t+H} = \underset{v_{t+1},...v_{t+H}}{\operatorname{argmax}} logP(v_{t+1}, ...v_{t+H}|v_{t-M+1}, ...v_t), \tag{1}$$

where $v_t \in \mathbb{R}$ is an observation vector of n road segments at time step t, each element of which records historical observation for a single road segment.

In this work, we define the traffic network on a graph and focus on structured traffic time series. The observation v_t is not independent but linked by pairwise connection in graph. Therefore, the data point v_t can be regarded as a graph signal that is defined on an undirected graph (or directed one) G with weights w_{ij} as shown in Fig.1. At the t^{th} time step, in graph $G_t = (V_t; \epsilon; W)$, V_t is a finite set of vertices, corresponding to the observations from n monitor stations in a traffic network; ϵ is a set of edges, indicating the connectedness between stations; while $W \in \mathbb{R}^{nxn}$ denotes the weighted adjacency matrix of G_t.

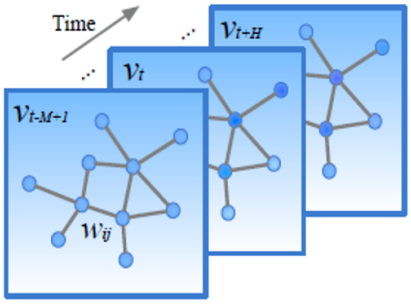

Fig. 1. Graph-structured traffic data.

B. Graph Convolutions

A standard convolution for regular grids is clearly not applicable to general graphs. Two basic approaches currently explore how to generalize CNNs to structured data forms. One is to expand the spatial definition of a convolution [Niepert et al., 2016], and the other is to manipulate in the spectral domain with graph Fourier transforms [Bruna et al., 2013]. The former approach rearranges the vertices into certain grid forms which can be processed by normal convolutional operations. The latter one introduces the spectral framework to apply convolutions in spectral domains, often named as the spectral graph convolution. Several followingup studies make the graph convolution more promising by reducing the computational complexity from O(n2) to linear [Defferrard et al., 2016; Kipf and Welling, 2016].

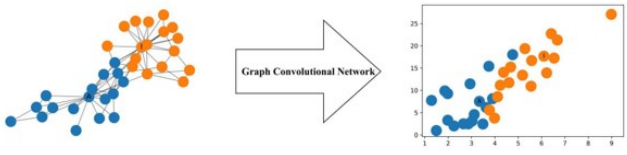

Fig. 2. Sketch map of graph convolution networks

The basic architecture of graph convolution networks is shown in Fig.2 and Fig.2. The input includes a N x F^o feature matrix and an N x N adjacent matrix As for the feature matrix, N is the number of nodes and F^o is the number of input features for each node Meanwhile the adjacent matrix represents the graph structure, such as the relationship among points on road.

Graph convolution mainly consists of three steps. First, select the node sequences; next, collect the neighborhood nodes; finally, do the subgraph normalization.

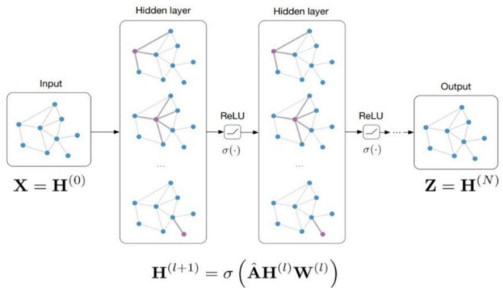

Fig. 3. Graph-structured traffic data.

III. Proposed Model

A. Network Architecture of STGCN

The framework STGCN consists of two spatio-temporal convolution blocks and a fully connected output layer in the end. Each spatio-temporal convolution block contains two temporal gated convolution layers and one spatial graph convolution in the middle.

The residual connection and bottleneck strategy are applied inside each block. The input $v_{t-M-1}, ...v_t$ is uniformly processed by ST-Conv blocks to explore spatial and temporal dependencies coherently. Comperhensive features are integrated by an output layer to generate the final prediction.

STGCN is a universal framework to process structured time series. It is not only able to tackle traffic network modeling and prediction issues but also to be applied to more general spatio-temporal sequence learning tasks. The spatio-temporal block combines graph convolutions and gated temporal convolutions, which can extract the most useful spatial features and capture the most essential temporal features coherently. The model is entirely composed of convolutional structures and therefore achieves parallelization over input with fewer parameters and faster training speed. More importantly, this economic architecture allows the model to handle large-scale networks with more efficiency.

B. Network Architecture of EB-STGCN

Based on this framework, we make some improvement. In STGCN, the adjacency matrix is computed based on the distances among points in the traffic network. As this formula shows, the matrix is a function of straight-line distances; if the distance is larger than threshold, the weight is set to 0. This matrix does not encompass the logical connections between each road segment.

Our model EB-STGCN is similar to STGCN, but different from it, we have made some modifications to the rules of adjacency matrix. We propose another method to generate graph matrix, which is based on edge – road segment. Each

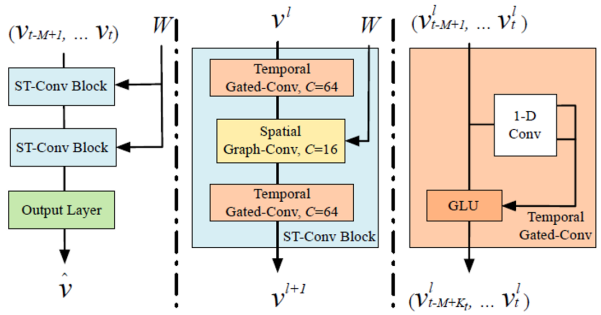

Fig. 4. Architecture of spatio-temporal graph convolutional networks.

weight in the adjacency matrix represents the topological relationship between two road segments. If there is only one turn between two road, the weight is 1, if more turns are needed, the weight is less than 1. Our matrix shows the topological connections between two road, thus closer to the real situation. The concrete determination method of adjacency matrix in our experiment will be explained in detail in the next part.

IV. Experiments

A. Dataset Description

In previous work, the traffic data is recorded by hardware, such as cameras or loop detector. Those methods need extra construction in hardware equipment, which is expensive for large scale implement. Thus, we use the traffic API from internet map service provider Gaode. We built the database using MongoDB and obtained data from Gaode api. We have three generation of traffic dataset, and use the final one to do experiment.

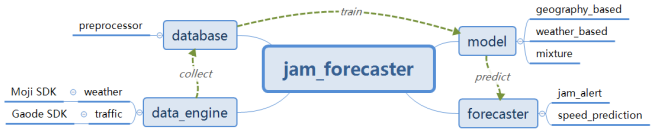

Fig. 5. Workflow of Dataset SJAM1.

SJAM1: In the first dataset SJAM1, our original plan was to take the weather factor as one of the important variables affecting traffic conditions into consideration in the traffic forecast. So we have chosen 50 distribute points in Shanghai by latitude and longitude, and get weather data from Moji api at the same time. The data from two APIs is shown in Fig.6. However, we found that in this dataset, there is little relationship between two points, and its hard to integrate weather data to our dataset.

SJAM2: In the second data set, we believe that the influence of weather factors on road traffic prediction is not obvious enough. On the contrary, compared with the weather factors, the traffic condition of a road has a very close relationship with the surrounding road conditions. Therefore, in the second data

traffic

parameter	meaning	example
road	road name	沪金高速
position_id	observation position id(1~300)	3
position	latitude and altitude	(121.4299106598, 31.0308171524
date	observe date(month, day)	(04, 03)
time	observe time(hour, minute)	(18, 05)
l_code	road section id(Gaode standard)	(2109,8110)
angle	angle of car flow (0~359)	267
direction	direction of car flow	从G 1 5沈海高速出入口到莘庄立
speed	average moving speed (km/h)	75

weather

parameter	meaning	example
area_name	district name	闵行区
area_id	observation area id(0~5)	2
date	observe date(month, day)observe date	(04, 03)
time	observe time(hour, minute)	(18, 05)
condition	weather condition(Chinese)	晴
condition_id	weather condition(code)	1
humidity	air humidity	28
pressure	air pressure	1017
temp	outside temperature	10.3
temp_feel	real feel temperature	11.5
wind_dir	wind direction	东南风
wind_level	wind speed level	1

Fig. 6. Data from Gaode API and Moji API.

set, our goal is to predict the road conditions of a road section in the next few minutes. We select 50 road sections centered on Shanghai railway station in this data set, and read Gaode API every 5 minutes to obtain the current traffic conditions. In this trial, we still use the method of determining points by longitude and latitude. The points we have selected is shown in Fig.7.

Fig. 7. The points that we collected in the dataset SJAM2.

But in this dataset, we found that due to the defects of Gaode API, data of some points are missing in part or a large number, which brings great difficulties to our next step of data cleaning and model training. Moreover, some points on one road segment have exactly the same data, which causes redundancy and inaccuracy.

SJAM3: Given that traffic data read from the audur API is given in the unit of road segment and each segment produces equal data with each otherwe chose the method of obtaining the speed and congestion information of the road section by the name of it in the third data set.

In the third dataset, we still chose several road segments around Shanghai railway station. SJAM3 includes 43 road segments concentrated upon Shanghai Railway station, and we collected traffic data every two minutes, which enlarges our amount of data. The distribution of 43 roads is shown in Fig.8. Finally, we choose to use this dataset in further experiment.

Fig. 8. The distribution of 43 roads in SJAM3.

B. Data Preprocessing

After obtaining reliable data sets, we find that the data we collected is not pure and clean, and there are some problematic points and data, so data preprocessing is an essential step. Our work of data preprocessing can be mainly divided into two parts, location filter, data regularization and determination of adjacency matrix.

Location Filter

As it turns out, there are some defects in the Gaode API, which sometimes results in some lack of data. At the beginning, we have passed 71 road sections of information to Gaode API, but from the data collected at last, there were 28 road sections with a large amount of data loss, and the proportion of data loss at the most missing segment was more than 50%. In this case, if we fill in the missing data in traditional way, the accuracy of the final dataset will be certainly reduced. So we chose to discard the 28 road sections, and finally we built a data set of 43 road sections.

Data Regularization

After deleting the sections with high data loss rate, we need to complete the remaining data with some loss and remove the duplicates returned by the API. Since our data is collected every two minutes, the percentage of missing data in the remaining 43 roads is very small. So our data completion method is to use linear interpolation method to fill the missing values.

We have written a script file in python to do these two-step preprocessings. See the Github url at the end of this paper for the code.

Determination of adjacency matrix

In the previous method, a good way to determine the adjacency matrix is to judge the relationship between points by the distance between them. Their weighted adjacency matrix W can be formed as,

$$w_{ij} = \begin{cases} exp(-\frac{d_{ij}^2}{\sigma}^2), & i \neq jandexp(-\frac{d_{ij}^2}{\sigma}^2) \geq \epsilon \\ 0, & otherwise. \end{cases} \tag{2}$$

where w_{ij} is the weight of edge which is decided by d_{ij} (the distance between point i and j). σ^2 and ϵ are thresholds to control the distribution and sparsity of matrix W, assigned to 10 and 0:5, respectively.

This method of determining matrix above has certain rationality, for example, the closer the road is, the higher the correlation degree will be. But from our point of view, their judgment method is a little rough. For instance, if we take two near short sections of the same road and assume that the two sections are in opposite directions. In reality, the correlation between the two sections is very small, except when the car turns around. However, in the above judgment method, the two roads will be defined as very relevant which is not consistent with the fact. Therefore, we modify the judgment method of adjacency matrix, and the judgment criteria are as follows;

$$w_{ij} = \begin{cases} 1 : less\ than\ one\ turn \\ 0.8 : less\ than\ one\ turn\ with\ a\ short\ interval\ of\ one\ other \\ 0.5 : two\ turns\ while\ d_{ij}\ is\ relatively\ small \\ 0.3 : two\ turns\ while\ d_{ij}\ is\ medium \\ 0.1 : two\ turns\ while\ d_{ij}\ is\ large \\ 0 : more\ than\ two\ turns \end{cases} \tag{3}$$

Although our judgment criteria is still not detailed enough, we propose a new way to measure the relationship between two road sections. In the future work, we can further refine the final w_{ij} determination on this basis.

C. Experiment Settings

Evaluation Metric and Baselines

In this experiment, it is inconvenient to use the loss function to measure the predicted results. Since this problem is not a simple classification problem, the accuracy function is no longer applicable. In order to better represent the prediction results, we used three scales to evaluate the prediction results and training results, Mean Absolute Errors (MAE), Mean Absolute Percentage Errors (MAPE), and Root Mean Squared Errors (RMSE).

- Mean Absolute Errors (MAE)
  Definition: average absolute error is the mean of absolute error;
  The mean absolute error can better reflect the actual situation of the predicted value error.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|(f_i - y_i)| \tag{4}$$

- Mean Absolute Percentage Errors (MAPE)
  Definition: mean absolute percentage error;

$$MAPE = \frac{100\%}{n}\sum_{i=1}^{n}|\frac{\hat{y}_i - y_i}{y_I}| \tag{5}$$

- Root Mean Squared Errors (RMSE).
  Definition: the root mean square error is the arithmetic

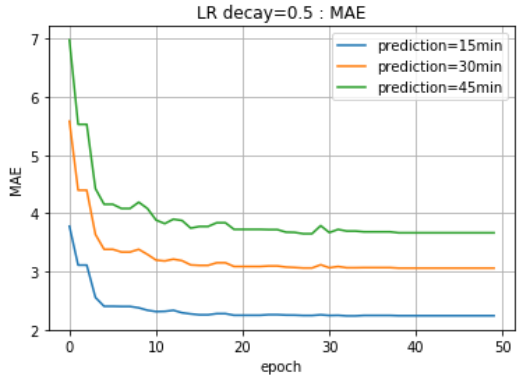

Fig. 9. MAE of different training epoch at the same fixed learning rate

square root of the mean square error.

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}(observed_i - predicted_i)^2} \quad (6)$$

.

D. Experiment Results

Comparison Between Models

We used a total of three training model methods, namely the traditional LSTM, STGCN and our EB-STGCN. We compared the scales of the models trained by three training methods, and obtained the data shown in Table I.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT APPROACHES ON THE DATASET SJAM3

Table Head	SJAM3 (15min/30min/45min)		
	MAE	***MAPE(%)***	***Rmse***
LSTM	3.23/3.45/3.68	9.10/9.43/9.78	5.20/5.65/6.12
STGCN	2.23/2.44/2.50	7.46/8.01/8.14	3.45/3.74/3.87
EBSTGCN	2.21/2.43/2.48	7.48/8.00/8.11	3.45/3.74/3.85

According to the data in the table, it can be clearly seen that among the three models, LSTM performs the worst in all indicators, while STGCN and EB-STGCN have almost the same effect, and EB-STGCN has a slight advantage.

Comparison Between Different Prediction Time

After analyzing the performance of different models, we conducted comparative analysis on the prediction accuracy of 15min, 30min and 45min. For convenience, we first set the learning rate to a fixed value, and among the three indicators, we selected MAE to compare them, as shown Fig.9.

It can be seen from this result that under the same learning rate, the convergence curves of the three basic different target forecasts show a similar trend. However, the closer the prediction time is, the higher the prediction accuracy and smaller the error. It's very cumulative. At the same time, we can also see that when the predicted time point is far away, such as 45 minutes, the obvious error curve fluctuates greatly.

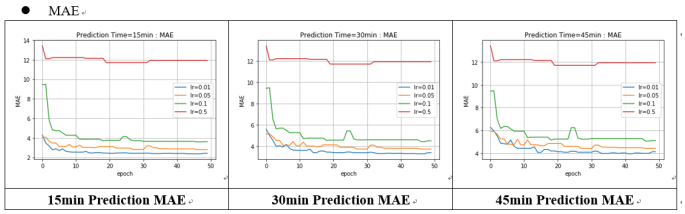

Fig. 10. MAE changes of EB-STGCN at different learning rates

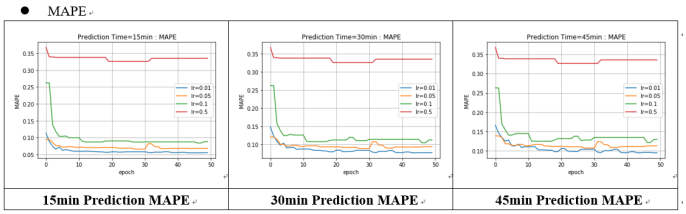

Fig. 11. MAPE changes of EB-STGCN at different learning rates

By contrast, the error curve near the time point, such as 15 minutes, has little fluctuation and soon converges to the final result.We guess that this is because the model is not good enough to predict the distant time point, so the error will be relatively large and the fluctuation will be relatively large. In the end, it is logical that the error of the prediction result is lower than that of the nearest time point.

Influence of Learning Rate on Final Results

Finally, we study the influence of different learning rates on the convergence effect of EB-STGCN. After evaluating the effect of learning rate, we can provide some reference for other users of this model. We set the four groups with fixed learning rates of 0.01, 0.05, 0.1 and 0.5 respectively for comparison, and calculated the measurement indexes that predicted the traffic situation 15 minutes later, 30 minutes later and 45 minutes later. The results are as shown in Fig.10, Fig.11 and Fig.12.

It can be seen from the graph that the convergence effect of the three indicators is gradually getting better with the decrease of the learning rate. In all cases, the lower the learning rate, the faster the convergence rate, and the better the final result. This indicates that the function in this experiment is a function with a small local minimum value, and the gradual convergence will eventually reach the global minimum. Therefore, the smaller the learning rate is, the smaller the step size is, and the easier

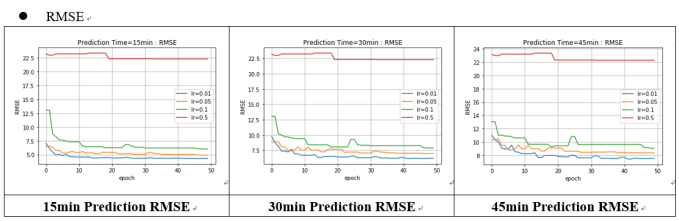

Fig. 12. RMSE changes of EB-STGCN at different learning rates

- MAE

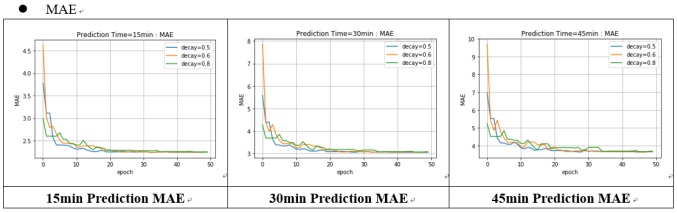

Fig. 13. MAE changes of EB-STGCN at different LR decays

- MAPE

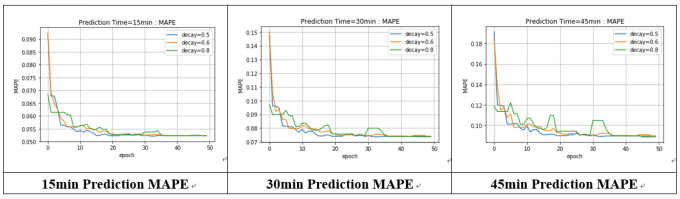

Fig. 14. MAPE changes of EB-STGCN at different LR decays

it is to get close to the lowest point and get the best results.

In order to get a broader rule, we try to use the changing learning rate to get a better convergence effect. In the early epoch, a large learning rate was used to rapidly move the results to the global minimum. As the number of iterations of training increased, we reduced the learning rate, so that the results could converge to the global minimum accurately and quickly.

In this group, we decreased the learning rate after each epoch by 0.5 times, 0.6 times and 0.8 times respectively. The results are as shown in Fig.13, Fig.14 and Fig.15.

It can be seen that when using the variable learning rate, all the initial learning rates can eventually converge to a similar effect. Therefore, the use of variable learning rates can reduce the impact of custom parameters on the final training results. Meanwhile, by observing the above pictures, with the extension of the prediction time point, the fluctuation of parameter convergence also increases gradually, which is the same as the result of fixed learning rate.

V. Related Works

In recent years, many people have been using graph convolution for traffic prediction, and some modifications and optimizations have been made on the original graph convolution model. Bing Yu et al. proposed a new deep learning framework – spatio temporal graph convolution network (STGCN) for the prediction of time series in the field of transportation. Instead of using conventional convolution and recursion units, they represented the problems in graphs and built a model with complete convolution structure, which made the training faster and with fewer parameters. Experimental results show that STGCN model can effectively capture the comprehensive spatio-temporal correlation and maintain good performance through modeling multi-scale traffic network. However, on the basis of their model, we have studied a method to change the adjacency matrix which results in lower computational complexity and better accuracy.

- RMSE

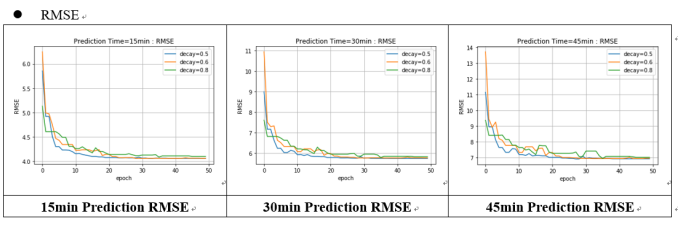

Fig. 15. RMSE changes of EB-STGCN at different LR decays

VI. Conclusion and Future Work

Our work has two innovations and two improvements.

As for the innovations, firstly, our adjacent matrix is based on road segment, not coordinate points, which means our results conform drivers habits. Drivers generally analyze the congestion situation by road segments, not points. Drivers may always say "This road is blocked!" rather than "This point is blocked!". Secondly, the weight of adjacent matrix is based on the topological relation of road segments, rather than the distance between points. Thus, our method considers the direction and turning of the road section, this makes our method more reasonable.

As for the advantage, On one hand, as the experiment shows, the prediction is more accurate. That is because the directions and turnings of road sections are considered. For example, if two observation points are very close on the same road, but on the opposite direction, they will have high weight in point-based adjacency matrix, but in out matrix, their weight is nearly 0. On the other hand, our method is easy to implement. As our data are all from Gaode API, there is no need for hardware implement. If Gaode API allows more detail data to be public for us, our result will be much more accurate.

For more details and code, visit GitHub: https://github.com/martin-xia0/Jam-Forecaster

References

[1] Yu B , Yin H , Zhu Z . Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting[J]. 2017.

[2] Li Y , Yu R , Shahabi C , et al. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting[J]. 2017.

[3] Yao H , Tang X , Wei H , et al. Modeling Spatial-Temporal Dynamics for Traffic Prediction[J]. 2018.

[4] Li C , Cui Z , Zheng W , et al. Spatio-Temporal Graph Convolution for Skeleton Based Action Recognition[J]. 2018.

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

[6] Michael Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NIPS, pages 38443852, 2016.

[7] Koesdwiady A , Soua R , Karray F . Improving Traffic Flow Prediction With Weather Information in Connected Cars: A Deep Learning Approach[J]. IEEE Transactions on Vehicular Technology, 2016, 65(12):1-1.

[8] Chen P , Li K , Sun J . A Method of Traffic Flow Forecast and Management[C]// International Conference on Information Management. IEEE Computer Society, 2008.

[9] Wenhao Huang, Guojie Song, Haikun Hong, and Kunqing Xie. Deep architecture for traffic flow prediction: deep belief networks with multi-task learning. IEEE Transactions on Intelligent Transportation Systems, 15(5):21912201, 2014.

[10] Eleni I Vlahogianni. Computational intelligence and optimization for transportation big data: challenges and opportunities. In Engineering and Applied Sciences Optimization, pages 107128. Springer, 2015.

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

[12] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. Neural computation, 9(8):17351780, 1997

[13] Xingjian Shi, 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, pages 802810, 2015.