2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering

Diffusion Convolutional Recurrent Neural Network with Rank Influence Learning for Traffic Forecasting

Yujun Huang
Sun Yat-Sen University
Guangzhou, China
huangyj78@mail2.sysu.edu.cn

Yunpeng Weng
Sun Yat-Sen University
Guangzhou, China
wengyp@mail2.sysu.edu.cn

Shuai Yu
Sun Yat-Sen University
Guangzhou, China
yushuai@mail.sysu.edu.cn

Xu Chen*
Sun Yat-Sen University
Guangzhou, China
chenxu35@mail.sysu.edu.cn

***Abstract*—With the rapid development of urban road traffic, accurate and timely road traffic forecasting becomes a critical problem, which is significant for traffic safety and urban transport efficiency. Many methods based on graph convolutional network (GCNs) are proposed to deal with the graph-structured spatio-temporal forecasting problem, since GCNs can model spatial dependency with high efficiency. In order to better capture the complicated dependencies of traffic flow, we introduce rank influence factor to the Diffusion Convolutional Recurrent Neural Network model. The rank influence factor could adjust the importance of neighboring sensor nodes at different proximity ranks with the target node when aggregating neighborhood information. Experiments show a considerable improvement when rank influence factor is used in GCNs with a tolerable time consumption.**

***Index Terms*—Graph Convolutional Network, Spatio-temporal model, Traffic forecasting.**

I. Introduction

Transportation is an indispensable part of people's daily life. The urban road traffic has been developed rapidly for decades and there is an increasing perception of the need for traffic forecasting. Predicting traffic flow accurately could prevent drivers from traffic jam and provide guidance for transportation services. It's also the foundation of building intelligent city. Real-time speed and flow data can be recorded by sensors, but traffic flow data has complicated spatial and temporal dependency, which can not be simply modeled by linear model.

Although some roads are close geographically, they have very different traffic flow patterns. Moreover, the future traffic flow is influenced more by the downstream traffic than the upstream one. Thus, it is unreasonable to represent the spatial structure in traffic data with geographic distance. To this end, we use road network distance according to the directed graph of road network instead of geographic distance [1]. Road network distance is directional and non-Euclidean, which is calculated based on geographic distance and road network connection. Note that roads behave similarly when they have closed road network distance, and behave differently when they are far from each other. So we may pay attention to the roads from long-distance because of the diversity and restrain information from closed road. Fig 1 shows an example, in which we use the radial basis function of the road network distance between roads to measure the degree of proximity between roads. Road 1 and road 2 have 0.8 degree of proximity while road 1 and road 3 have 0.1 degree of proximity. The curves of road 1 and road 2 have more similar shape than curves of road 1 and road 3. But when we predict the traffic flow of road 1, the information from road 2 is less useful because the information from road 2 is largely contained in information from road 1, while information from road 3 have more diversity. To address this problem, we introduce the trainable influence factor to the DCRNN model, which adjusts the importance of neighboring nodes at different proximity rank during the information aggregation for a target sensor

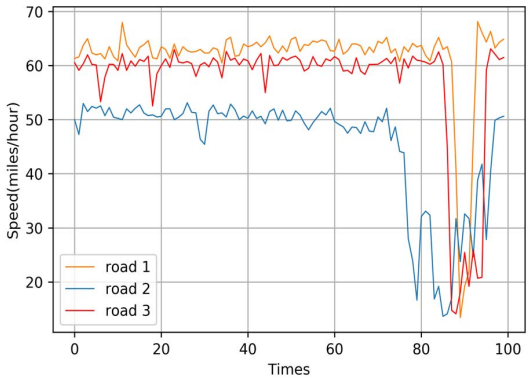

Fig. 1. Road 1 and road 2 have 0.8 degree of proximity while road 1 and road 3 have a 0.1 degree of proximity. The curves of road 1 and road 2 have more similar shape than curves of road 1 and road 3

*Corresponding author. This work is supported in part by the National Science Foundation of China (No.U1711265, No.61802449); the National Key Research and Development Program of China under grant No.2017YFB1001703; the Fundamental Research Funds for the Central Universities (No.17lgjc40), and the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (No.2017ZT07X355)

2324-9013/19/$31.00 ©2019 IEEE
DOI 10.1109/TrustCom/BigDataSE.2019.00096

node.

Dynamical modeling and data-driven methods are two widely used methods for tackling traffic prediction problem [2]. Dynamical modeling predicts traffic flow using mathematic model by computational simulation. For example, car following model is often used to simulate the car flow [30]. Specifically, car-following model contains general motor model [31], linear model [32], safe distance model [33], AP model [31], fuzzy inference model [31]. Highly accurate simulation needs the details of road situation, which sometime couldn't be obtained. Besides, these models sometimes couldn't be applied in all places, because model couldn't adapt to all situation in different place. To reduce the calculation overhead of dynamical modeling, researchers often make some simplified assumptions, which make it difficult to predict the real world situation [4]. Data-driven methods like autoregressive integrated moving average model [5] (ARIMA) is a classical statistics method based on normal distribution and stationarity assumption. However, this type of model is limited by the statistics assumption of time sequences and only uses time dimension information. Linear relationship between inputs and outputs also limits the performance of prediction. Other data-driven methods like the Kalman filtering model [6] , support vector regression machine model ([7], [8], [9]) , k-nearest neighbor model [10], Bayesian model [11], and partial neural network model ([12], [13]) also mainly focus on time dependency, overlook the spatial relationship between nodes.

Recently, deep learning methods which use convolution network and recurrent neural network, make considerable improvement in spatio-temporal forecasting ([1], [2], [3], [14], [15], [17], [18], [19]). Most of these works use graph convolutional network to model spatial dependency. But there is a constraint in regular graph convolutional network. The importance weights of neighboring nodes are fixed during information aggregation and GCNs usually assigns larger weights to the nodes with high proximity to the target node, which leads to the neglect of neighboring information with high diversity.

In this work, we adjust the weight of neighboring nodes during aggregation procedure by learning rank influence factor, which make graph convolutional network more flexible to aggregate information target nodes' neighborhoods. Based on the result of experiment, we find that this mechanism will make model pay more attention to less similar nodes, which may merge more information with high diversity. Based on this mechanism and DCRNN [1], we propose diffusion convolutional recurrent neural network with rank influence learning(DCRNN-RIL). We also use this rank influence learning in other spatial-temporal models and improve the models with a little more time consumption. The major contributions of this paper are summarised as follows:

• We propose rank influence learning mechanism which makes GCNs more flexible. The rank influence learning has intuitive interpretation with tolerable time consumption. Based on this method, we propose graph convolutional network with rank influence learning (GCNs-RIL). Experiments show that GCNs with rank influence learning will pay attention to the nodes which bring more diversity, and merge more useful information to help the model obtain more accurate prediction.

• We propose diffusion convolutional recurrent neural network with rank influence learning (DCRNN-RIL), which integrates the rank influence learning to state-of-the-art model DCRNN. Experiment results show that our DCRNN-RIL usually outperforms other baseline schemes in the precision of prediction. At last, we also use the rank influence learning in other spatial-temporal models with GCNs structure and obtain considerable improvement with tolerable time consumption.

II. Related Work

Spatio-temporal Model for Traffic Forecasting. The state-of-the-art data-driven neural network methods model spatio-temporal data mainly divides into two parts: i) modeling the temporal dynamics with Neural Network, such as, RNN, 1D CNN ([2], [3]), and ii) modeling the spatial dependency with graph convolutional network with a state transition matrix [1] or a Laplacian matrix ([2], [3]). DCRNN [1] models the spatial dependency with diffusion convolution network and models the temporal dependency using a Recurrent Neural Network (RNN) variant: Gated Recurrent Units. CGRNN [3] models the spatial dependency with Laplacian graph convolution operation and models the time dependency with RNN. CGRNN also has a multi-graph structure for non-Euclidean data and global contextual information when modeling the temporal dependencies.

STGCN [2] models the spatial dependency like CGRNN, but models the temporal dependency with 1D Gated CNN. 1D Gated CNN can be trained faster than RNN and reach state-of-the-art result. T-GCN [19] only leverages 1st order diffusion convolution as merging function of spatial dependency model and leverages full connected network instead of RNN to model temporal dependency. In most case, researchers model the spatial dependency by GCNs on graph-structured data, but the merging function of graph convolutional network is fixed, which may limit the model's flexibility. Thus, it makes sense to make the merging function trainable, like things has been done in Graph Neural Network (GNN) ([21], [22]). If we make the merging function trainable, like 2D-CNN, a node will merge the information from its neighboring nodes with reasonable weights and the model will be more flexible to capture the spatial dependency.

Diffusion Convolution Network. Graph convolutional network mainly divided into two categories: spatial domain and spectral domain. Graph convolutional network using spatial domain analogizing the classical convolutional aggregation [23]. DCRNN [1] leverages spatial domain graph convolution, which is called diffusion convolution neural networks [29]. DCRNN use $(D_0^{-1}W)^T$ as the state transition matrix A, here $W \in R^{N \times N}$ denotes weighted adjacency matrix, N is the number of road. Elements of W: $w_{ij} = exp(-\frac{dist(v_i,v_j)^2}{\sigma^2})\ if\ dist(vi,vj) \leq k,\ else\ 0$. $D_0 = diag(W \cdot 1)$ is the out-degree diagonal matrix, and 1

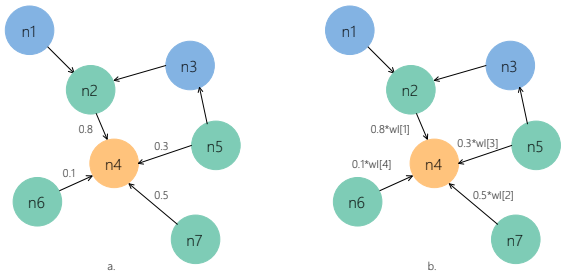

Fig. 2. a. Regular GCNs merge information from neighboring nodes. Neighboring nodes' importance is proportion to proximity between target node and its neighboring nodes. Yellow node n4 is the target node. Green nodes n2, n5, n6, n7 are its neighboring nodes. Blue nodes n1, n3 are irrelevant to target node. b. GCNs with rank influence learning merge information from neighboring nodes. Neighboring nodes' importance is proportion to proximity between target node and its neighboring nodes multiply relative rank influence factor.

denotes the all one vector. Diffusion convolution uses the state transition matrix A to modeling the state transition. 1-st order feature $X_{(1)} \in R^{N\times P}$ can be calculated based on A and feature vector $X \in R^{N\times P}$. P is the number of feature of road signal. n^{th} order feature $X_{(n)} \in R^{N\times P}$ can be calculated recursively:

$$X_{(1)} = A \cdot X,$$
$$X_{(n)} = A \cdot X_{(n-1)}.$$

And some method use spectral domain operation. They calculate graph Laplacian matrix ([2], [3]):

$$L = I - D^{1/2} \cdot W \cdot D^{1/2}$$

, and they use Chebyshev polynomial to calculate the n^{th} order feature:

$$X_{(n)} = 2 \cdot L \cdot X_{(n-1)} - X_{(n-2)}.$$

DCRNN [1] gets the q^{th} feature output of spatial model as follow:

$$X^*_{G:,q} = \sum_{p=1}^{P} \theta_{0,p,q} X_{:,p} + \sum_{k=1}^{K-1} \sum_{p=1}^{P} \theta_{k,p,q} X_{(k):,p}.$$

This is the complete formula of diffusion convolution operation of DCRNN. $X^*_{G:,q}$ is the q^{th} feature of diffusion convolution output. $X_{(k):,p}$ is the p^{th} feature of $X_{(k)}$, all $\theta_{k,p,q}$ constitude a diffusion convolution parameters tensor $\Theta \in R^{K\times P\times Q}$.

The k^{th} order merging function of diffusion convolution and the diffusion convolution can be formulated as:

$$m_k(x) = A \cdot m_{k-1}(x),$$
$$y_j = \delta(X \cdot \theta_{0j} + m_1(x) \cdot \theta_{1j} + ... + m_{k-1}(x) \cdot \theta_{k-1j} + b_j).$$

y_j is the j^{th} feature of output. The formula of spectral convolution is similar to formula of spatial convolution introduced above. The merging function of both of them is fixed which is calculated on data preprocessing. It's worth noting that state transition matrix is an important module in merging function. Fixed State transition matrix leads to fixed merging function.

III. Methodology

A. Problem & Notation

Traffic forecasting is a problem using historic information, such as, speed, time and traffic flow of roads, to predict future traffic situation: speed, traffic flow. Road is treated as node in a graph. Because of complicated spatial dependency of roads in the city, graph is usually not grid-structured, which means 2D CNN is difficult to use in graph-structured data.

We can represent the traffic network as a weighted directed graph $G = (V, E, W)$, where V is a set of nodes $|V| = N$, E is a set of edges present the spatial dependency of streets and $W \in R^{N\times N}$ is a weighted adjacency matrix representing the nodes proximity (radial basis function from the road network distance between two streets to their proximity). Denote the traffic-flow observed on G as a graph signal $X \in R^{N\times P}$, where P is the number of features of each node (e.g., speed, traffic volume). Let $X^t \in R^{N\times P}$ represent the traffic flow observed at time t , the goal of traffic forecasting problem is to learn a function $h(\cdot)$ which takes T_h historical traffic flow as inputs and outputs future T_f traffic flow, based on the graph-structured road network G [1]:

$$h([X^{t-T_h+1}, ..., X^t]|G) = [X^{t+1}, ..., X^{t+T_f}].$$

B. Modeling Spatial Dependency: Graph Convolutional Network with Rank Influence Learning

In order to better capture the spatial dependency, we introduce a rank influence factor in GCNs. Fig 2 shows difference between regular GCNs and GCNs-RIL. Yellow node n4 is the target node. Green nodes n2, n5, n6, n7 are its neighboring nodes. Blue nodes n1, n3 are irrelevant to the target node. When regular GCNs merges information from the target node's neighboring nodes, neighboring nodes' importance is proportion to proximity between the target node and its neighboring nodes. When GCNs adding rank influence factor, neighboring nodes' importance is proportion to proximity between target node and its neighboring nodes multiply relative rank influence factor. Rank influence factor helps GCNs control information that input from different proximity. Formulating adding rank influence factor: l^{th} order diffusion convolution operation in GCNs: $X_{(l)} = A \cdot X_{(l-1)}$ is replaced by $X_{(l)} = A \odot W_I \cdot X_{(l-1)}$ in GCNs-RIL. $\odot$ is element wise multiplication between two matrices, $W_I \in R^{N\times N}$ is a rank influence factor matrix (construction method is introduced below). A is the state transition matrix. When rank influence factor adds to DCNNs [29], it becomes Diffusion Neural Networks with Rank Influence Learning (DCNNs-RIL). When using Laplacian matrix to merge information, we treat Laplacian matrix as the state transition matrix.

Rank Influence Factor Matrix. Rank influence factor $wI \in R^N$ is an ordered vector. The k^{th} element of rank

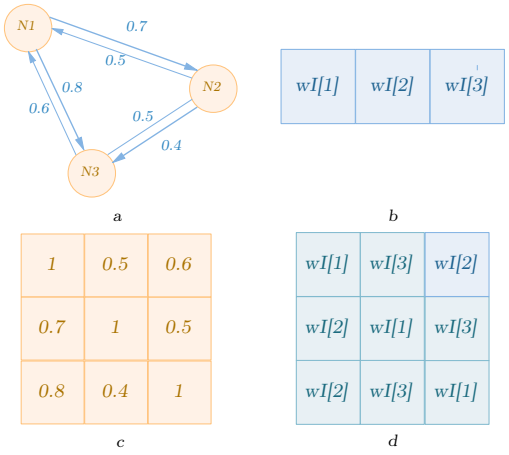

Fig. 3. a. Directed graph with 3 nodes, values on the edges represents the proximity between two nodes. b. A rank influence factor whose size equals to the number of nodes. c. Adjacent matrix A. d. Rank influence factor matrix results from reordering rank influence factor base on A.

influence factor represents the importance of the k^{th} approximate neighboring nodes to the target node. Rank influence factor matrix $W_I \in R^{N\times N}$ is composed of rank influence factor. Each row in rank influence factor matrix is a rearrangement of relative rank influence factor. We rearrange *wI* with an ordering rule based on proximity between the target node and its neighboring nodes. The proximity can be represented by the elements in state transition matrix *A*.

$$W_{Ii,:} = R(wI|A_{i,:}).$$

Where $R(\cdot)$ is a function for rearrangement. It rearranges *wI* based on the order of elements in $A_{i,:}$. With this method, weights of differently approximate nodes can be adjusted. Fig 3 shows a simple example to construct rank influence factor matrix. Fig 3-a is the graph, Fig 3-c is the state transition matrix based on the graph. Fig 3-b is a rank influence factor *wI*. Fig 3-d is a rank influence factor matrix W_I. Every element in the first row of state transition matrix represents the proximity between the first node and the other node. We rearrange *wI* to construct the first row of W_I based on all nodes' proximity with the first node. One is the largest value in the first row of *A* which means the first node is the most similar to itself. So we assign the first parameter of *wI*: *wI*[1] to the element in the first row and the first column of W_I. Node 3 is the second similar to Node 1. So we assign the second parameter of *wI*: *wI*[2] to the element in the first row and the third column of W_I. Other elements in W_I can be assigned using the same method.

Element wise multiplication with a rank influence factor matrix can be considered as a mechanism to modify the weights to aggregate the information from target node's neighboring nodes. [25] leverage a sparse matrix *F* with nonzero trainable elements where the state transition matrix is nonzero to construct a trainable state transition matrix. They use the sparse matrix and state transition matrix to get k^{th} order feature:

$$x_k = L_{k-1} \cdot \delta(F \cdot x_{k-1}),$$

where $\delta(\cdot)$ is an activate function. Multiplying a rank influence factor matrix is a kind of modification of the state transition matrix, while their method construct a fully trainable state transition matrix to merge neighboring nodes' information, which is more flexible but more easy to over-fitting to the training data.

Graph Convolution with Rank Influence Learning from *k*-*I*th Order to *k*th Order. Because the merging function of GCNs is untrainable, k^{th} order state transition could only have a single channel: $X_{(k)} \in R^{N\times P}$. With the trainable rank influence factor, each order feature could have more than one channel. Then transition from $k-1^{th}$ order with c_{k-1} channels to k^{th} order with c_k channels can be formulated as :

$$X_{(k)}^{(i)} = \sum_{j=1}^{c_{k-1}} A \odot W_{I,k-1}^{(i,j)} \cdot X_{(k-1)}^{(j)} (i = 1, ..., c_k), \qquad (1)$$

where $X_{(k)}^{(i)} \in R^{N\times P}$ is the i^{th} channel of the k^{th} order feature, $W_{I,k-1}^{(i,j)}$ is the rank influence factor matrix corresponding to $X_{(k-1)}^{(j)}$ and $X_{(k)}^{(i)}$. c_k is the number of channel of k^{th} order feature. Then we obtain the output of GCNs-RIL: X_G^*, based on all orders feature. The q^{th} feature of X_G^* can be calculated as:

$$X_{G:,q}^* = \sum_{p=1}^{P} \theta_{0,p,q} X_{:,p} + \sum_{k=1}^{K-1} \sum_{i=1}^{c_i} \sum_{p=1}^{P} \theta_{k,i,p,q} X_{(k):,p}^{(i)}. \qquad (2)$$

C. Algorithm Complexity Analysis

Because in most case state transition matrix is a sparse matrix, asymptotic complexity of calculating $A \odot W_I$ is $O(|E|)$. And $A \odot W_I$ is a sparse matrix. So adding W_I to the model will increase tolerable time consumption. And rank influence factor whose size equals to the number of node only uses a little memory.

D. Modeling Temporal Dependency

We leverage the Gated Recurrent Units (GRU [25]) to model the temporal dependency [1]. We replace the matrix multiplications $X \cdot W$ in GRU with the diffusion convolution with rank influence learning $G_\theta(X)$, which leads to Diffusion Convolutional Gated Recurrent Unit with rank influence learning (DCGRU-RIL) based on the temporal model in DCRNN [1].

$$\begin{aligned} r^t &= \delta(G_{\theta_r}([X^t, H^{t-1}]) + b_r), \\ u^t &= \delta(G_{\theta_u}([X^t, H^{t-1}]) + b_u), \\ C^t &= \delta(G_{\theta_C}([X^t, (r^t \odot H^{t-1})]) + b_C), \\ H^t &= u^t \odot H^{t-1} + (1 - u^t) \odot C^t. \end{aligned}$$

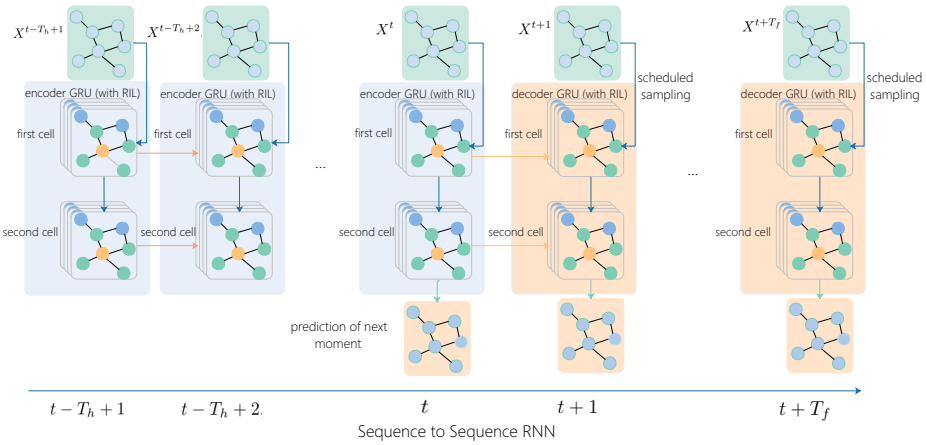

Fig. 4. The structure of DCRNN-RIL canbe dividied into encoder model and decoder model. We input traffic speed and time to encoder ans input only ground truth or prediction of traffic speed to decoder. Gated Recurrent Units (with RIL) is the basal module of encoder and decoder. Two GRU cells construct a muti-cell unit for encoder and decoder. The last output H_t of encoder is treated as the initial state of decoder.

Where X^t, H^t denote the input and output of at time *t* , r^t and u^t are reset gate and update gate at time *t* , respectively. $G_\theta(\cdot)$ denotes the diffusion convolution with rank influence learning defined in equation 2, and θ_r, θ_u, θ_C represent parameters for relative diffusion convolution with rank influence learning. We stack two GRU cells as a two-cell GRU. The second cell of this unit outputs one channel for every node in the graph, which be treated as the prediction of traffic speed. Sequence to Sequence architecture [26] and scheduled sampling [27] are also integrated in temporal model. Sequence to Sequence architecture is divided into encoder model and decoder model, encoder receives history information of traffic speed and time from *t-*T_h*+1* to *t*, and outputs H^t: the last state of encoder. The prediction of time *t* is obtained from the last state of encoder. Decoder receives this output as initial state. Decoder receives traffic speed of ground truth or prediction from *t+1* to *t+*T_f and predicts traffic speed from *t+2* to *t+*T_f*+1*. The last prediction of decoder will be neglected. Scheduled sampling is a method to determine whether the input to the decoder is the ground truth or the result of model prediction at previous time step. At the beginning, because prediction of model is poor, using the prediction as inputs to the decoder will make training difficult. So scheduled sampling sets high probability inputing ground truth to decoder at the beginning and lets the probability decrease continuously. Specifically, $p = \frac{c}{c+exp(i/c)}$, *p* is the pobability, c is a parameter to control the decreasing ratio. *i* is the number of iter.

Fig 4 shows the structure of DCRNN-RIL. DCRNN-RIL could divide into encoder model and decoder model. Gated Recurrent Units (with RIL) is the basal module of the encoder and the decoder.

IV. Experiments

A. Experimental Setting

We conducted experiments on three traffic datasets: METR-LA, PEMS-BAY, SZ-taxi. METR-LA contains traffic flow in the highway of Los Angeles County [28] collected by road sensors from Mar 1st 2012 to Jun 30th 2012. Dataset has 207 road sensors and time span of 4 months [1]. The adjacency matrix is calculated using radial basis function of the road network distance between sensors in the traffic networks: $w_{ij} = exp(-\frac{dist(v_i,v_j)^2}{\sigma^2})\ if\ dist(vi,vj) \leq k,\ else\ 0$. w_{ij} is element in the i^{th} row and the j^{th} column of the adjacency matrix. PEMS-BAY is collected by California Transportation Agencies (CalTrans) Performance Measurement System (PEMS) from Jan 1st 2017 to May 31th 2017. Dataset has 325 road sensors and time span of 6 months. The adjacency matrix is calculated by the same method of METR-LA. SZ-taxi contains the traffic flow of taxi of Shenzhen from Jan. 1 to Jan. 31, 2015 [19]. Datasets has traffic flow of 156 major roads of Luohu District as the study area and time span of 1 month. The 156*156 adjacency matrix, which describes the spatial relationship between roads, sets element to one if two roads are connected and to zero if two roads aren't connected.

We use 60min history traffic information to predict 60-min future traffic flows on METR-LA and PEMS-BAY datasets. And we use 180min history traffic information to predict 180-min future traffic flow on SZ datasets.

Three metrics are used to evaluate the performance: i) Mean Absolute Error (MAE), ii) Mean Absolute Percentage Error(MAPE), and iii) Root Mean Squared Error (RMSE).

Four models: DCRNN, CGRNN, STGCN, T-GCN are used in experiments ([1], [2], [3], [19]), which have been introduced in Section II. We replaced RNN with its variant: GRU when we model CGRNN [3]. We don't use time information as part of inputs when modeling STGCN [2] (different from that we used

TABLE I
EXPRIMENTS RESULTS ON METR-LA DATASET. APPROACHES WITH RANK INFLUENCE LEARNING OBTAIN BETTER PERFORMANCE THAN APPROACHES WITHOUT IT

METR-LA	15min			30min			60min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	2.77	5.38	7.3%	3.15	6.45	8.8%	3.60	7.59	10.5%
DCRNN-RIL	**2.46**	**4.50**	**6.3**%	**2.87**	**5.42**	**7.8**%	**3.32**	**6.36**	**9.7**%
CGRNN	2.83	5.22	7.5%	3.45	6.30	9.9%	4.37	7.82	13.5%
CGRNN-RIL	2.73	4.93	7.3%	3.22	5.96	9.3%	3.95	7.30	12.2%
STGCN	4.10	9.79	—	5.34	12.71	—	7.16	15.91	—
STGCN-RIL	3.94	10.08	—	5.16	12.89	—	6.97	15.83	—
T-GCN	3.79	6.07	—	4.41	7.20	—	5.31	8.71	—
T-GCN-RIL	3.71	6.00	—	4.26	7.08	—	5.19	8.60	—

TABLE II
EXPRIMENTS RESULTS ON PEMS-BAY DATASET.

PEMS-BAY	15min			30min			60min		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
DCRNN	1.38	2.95	2.9%	1.74	3.97	3.9%	2.07	4.74	4.9%
DCRNN-RIL	**1.04**	**1.60**	**1.53**%	**1.20**	**1.80**	**1.87**%	**1.38**	**2.04**	**2.04**%
CGRNN	1.11	1.79	1.70%	1.33	2.09	2.04%	1.60	2.48	2.38%
CGRNN-RIL	1.12	1.79	1.70%	1.33	2.08	2.04%	1.60	2.46	2.38%

time information as part of inputs when modeling DCRNN and CGRNN). So it's better to focus on the improvement result from adding the trainable rank influence factor. T-GCN [19] only leverages 1st order diffusion convolution as merging function of spatial dependency model and leverages fully connected network instead of RNN to model temporal dependency. T-GCN hasn't time message inputs too.

B. Traffic Forecasting Performance Comparison

Table I, II, III show the performance of models with and without rank influence learning on three datasets. In most case, models with rank influence learning outperform models without rank influence learning. Considerable improvement is showed when rank influence factor is used in DCRNN on METR-LA and PEMS-BAY. Although, for some methods such as CGRNN, there is not improvement when rank influence factor is used in GCNs on PEMS-BAY, models with rank influence factor will intuitively perform not worse than models without rank influence factor. Because if all elements in rank influence factor matrix is one and the number of channel in all order is one, model with rank influence learning is the same as the model without rank influence learning. Rank influence learning doesn't obtain better performance when proceeding experiment on SZ dataset. A reasonable explanation is that the adjacent matrix on SZ dataset is not calculated by road network distance, otherwise, it set element to one if two roads are connected and to zero if two roads aren't connected. The order of neighboring nodes' importance doesn't include in this dataset, which make rank influence factor unable to adjust the merging function. So, rank influence learning is more suitable for graphs having ordered neighboring nodes' importance than graphs that doesn't have. Fig 5 shows the prediction of 500 min traffic flow by DCRNN-RIL and DCRNN on test datasets of METR-LA respectively. We randomly select a road from

TABLE III
EXPRIMENTS RESULTS ON SZ DATASET.

SZ	45min		1hour 30min		3hour	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
DCRNN	3.30	4.61	3.41	4.72	3.51	4.81
DCRNN-RIL	3.20	4.48	3.27	4.55	3.32	4.60
CGRNN	3.35	4.69	3.84	4.82	3.64	4.97
CGRNN-RIL	3.29	4.58	3.38	4.66	3.51	4.75
STGCN	3.41	5.35	3.57	5.57	3.76	5.81
STGCN-RIL	3.29	5.13	3.42	5.39	3.56	5.59
T-GCN	2.79	**4.11**	2.82	**4.15**	2.85	**4.18**
T-GCN-RIL	**2.78**	4.13	**2.81**	4.16	**2.80**	**4.18**

graph and a start moment for the figure. In this figure, we find that DCRNN-RIL could roughly predict the tendency of the traffic flow but lose the detail of a specific moment. This is a trade-off between long-term and short-term accuracy. Model pays more attention to long-term accuracy and gives up some short term accuracy.

C. Weights of Rank Influence Factor

Fig 6 shows the first five trained parameters of two rank influence factor of CGRNN model training on METR-LA. Absolute value of weights of the $1^{st}, 2^{nd}, 3^{rd}$ similar nodes are smaller and absolute value of weights of the 4^{th} and 5^{th} similar nodes are much larger. These trained weights of rank influence factor prove that nodes which are more similar to the target node could bring little useful information, while nodes that are less similar to the target node bring more useful information because they bring more diversity.

D. Training Methodology

Adding rank influence factor makes model more flexible to aggregate information but also makes the model more easy to over-fitting to the train dataset. One effective method is

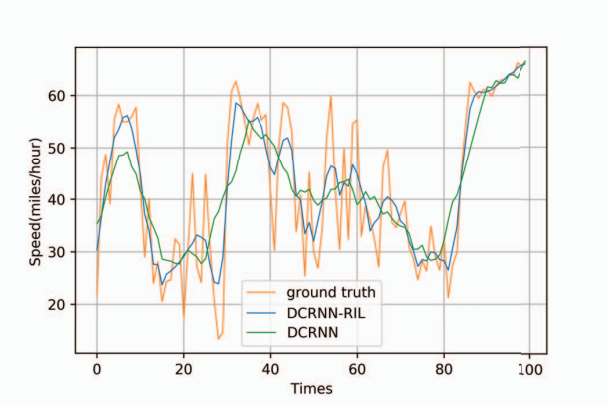

Fig. 5. Prediction of one road's 500 min traffic flow by DCRNN-RIL and DCRNN on test set of METR-LA. The road and start moment are randomly selected.

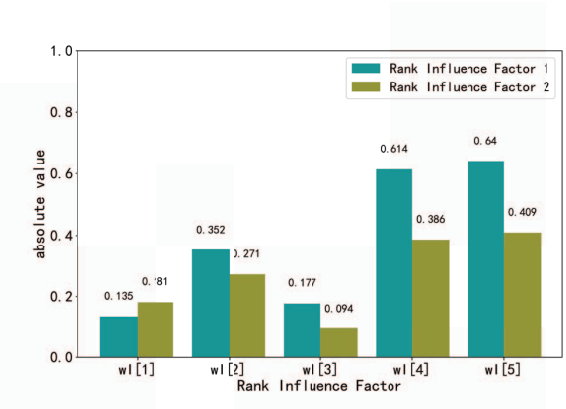

Fig. 6. the first five parameters of two rank influence factor of CGRNN model training on METR-LA.

to reduce the learning rate and restore the parameters of previous epoch every time the validation loss in current epoch is higher than validation loss in previous epoch. Intuitively, validation loss increasing means model is likely to over-fitting to training dataset. Reducing the learning rate could slow down the over-fitting rate. Based on experiments, we find that using this method to control learning rate can keep models from over-fitting to the train datasets. Fig 7, Fig 8 respectively shows performance of 60 min and 15 min traffic flow prediction use different methods on METR-LA. Using this training method obtains best performance in all situation. DCRNN-RIL without this training method mainly perform a little better than DCRNN on test dataset, but we find that it perform much better than DCRNN on train dataset. So this model is probably over-fitting to train dataset. And DCRNN-RIL with this training method perform better than DCRNN both on training and test dataset.

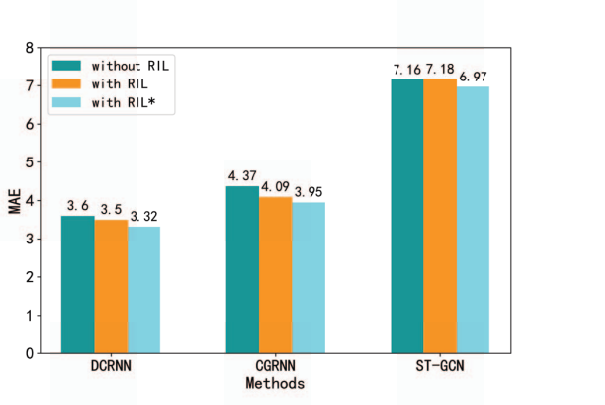

Fig. 7. Performance of 60 min traffic flow prediction use different methods on METR-LA. * means using the training method introduced in section IV-D.

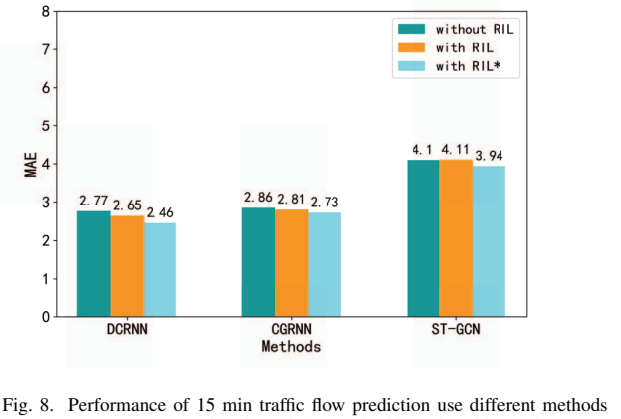

Fig. 8. Performance of 15 min traffic flow prediction use different methods on METR-LA. * means using the training method introduced in section IV-D.

V. Conclusion

In this paper, we propose rank influence learning to make the merging function of graph convolutional network trainable. Rank influence learning could help model to obtain more diversity and useful information. We propose a variant of GCNs: GCNs-RIL, based on the rank influence learning. Spatio-temporal model with GCNs structure could use this variant. We use this method in state-of-the-art traffic forecasting model and obtain the best performance in most case. In addition to this, every model with GCNs structure could use this method to help merge information.

References

[1] Li, Y., Yu, R., Shahabi, C., Liu, Y.: Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: International Conference on Learning Representationse (ICLR 2018).
[2] Yu, B., Yin, H.,Zhu, Z.: Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: IJCAI (2018)
[3] Geng, X., Li, Y.,Wang L., Zhang L., Yang Q., Ye J., Liu Y.: Spatio-temporal multi-Graph Convolution Network for Ride-hailing Demand Forecasting. In: AAAI (2019)

[4] Eleni I Vlahogianni.: Computational intelligence and optimization for transportation big data:challenges and opportunities. In: Engineering and Applied Sciences Optimization, pp. 107C-128. Springer, 2015.

[5] Marco Lippi., Marco Bertini., Paolo Frasconi.: Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. In: IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 871C882, Mar. 2013.

[6] Wei, L., Yu, Z., Sanjay Chawla., Jing, Y., Xie, X.: Discovering spatio-temporal causal interactions In: traffic data streams. In: SIGKDD, pp. 1010C-1018. ACM, 2011.

[7] Wu, C., Ho, J., Lee, D.: Travel-time prediction with support vector regression In: IEEE Transactions on Intelligent Trans-portation Systems, vol. 5, no. 4, pp. 276C281, Dec. 2004.

[8] Yao, Z., Shao, C., Gao, Y.: Research on methods of short-term traffic forecasting based on support vector regression In: Journal of Beijing Jiaotong University, vol. 30, no. 3, pp. 19C22, 2006.

[9] A, J, Smola., B, Schlkopf.: A tutorial on support vector regression In: Statistics and Computing, vol. 14, no. 3, pp. 199C222, Jan. 2004.

[10] Zhang, X., Guo-Guang, H., Hua-Pu, L., Short-term traffic flow forecasting based on k-nearest neighbors non-parametric regression In: Journal of Systems Engineering, vol. 24, no. 2, pp. 178C183, Feb. 2009

[11] S, Sun., Zhang, C., Yu, G.: A bayesian network approach to traffic flow forecasting In: IEEE Transactions on Intelligent Transportation Systems, vol. 7, no. 1, pp. 124C132, Mar. 2006

[12] Huang, W., Song, G., Hong, H., Xie, K.: Deep architecture for traffic flow prediction: Deep belief networks with multitask learning In: IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 5, pp. 2191C2201, Apr. 2014.

[13] Fu, R., Zhang, Z., Li, L.: Using lstm and gru neural network methods for traffic flow prediction In: Youth Academic Conference of Chinese Association of Automation, pp. 324C328, Jan. 2017.

[14] Zhang, J., Zheng, Y., Qi, D., Li, R., Yi, X., Li, T. Predicting citywide crowd flows using deep spatio-temporal residual networks. In : Artificial Intelligence 259:147C166, 2018.

[15] Zhang, X., He, L., Chen, K., Luo, Y., Zhou, J., Wang,F.: Multi-view graph convolutional network and its applications on neuroimage analysis for parkinson's disease. In: arXiv preprint arXiv:1805.08801, 2018.

[16] Ma, X., Dai, Z., He, Z., Ma, J., Wang, Y., Wang, Y. Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed pre- diction. In: Sensors 17(4):818. 2017.

[17] Yao, H., Tang, X., Wei, H., Zheng, G., Yu, Y., Li, Z.: Modeling spatial-temporal dynamics for traffic prediction. In: arXiv preprint arXiv:1803.01254. 2018.

[18] Yao, H., Wu, F., Ke, J., Tang, X., Jia, Y., Lu, S., Gong, P.;Ye, J., Li, Z.: Deep multi-view spatial-temporalnetwork for taxi demand prediction. In: 2018 AAAI Conference on Artificial Intelligence.

[19] Zhao, L; Song, Y; Zhang, C; Liu, Y; Wang, P; Lin, T; Deng, M; Li H 2018.: T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction. In: arXiv preprint arXiv:1811.05320. 2018

[20] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei , Q.: LINE: Large-scale Information Network Embedding. In:24-th International World Wide Web Conference(WWW' 2015).

[21] Bryan Perozzi., Rami Al-Rfou., Steven Skiena.: DeepWalk: Online Learning of Social Representations In: KDD 2014.

[22] Li, Y., Yu, R., Cyrus Shahabi., Liu, Y.: DIFFUSION CONVOLUTIONAL RECURRENT NEURAL NETWORK : DATA-DRIVEN TRAFFIC FORECASTING In: Conference and Workshop on Neural Information Processing Systems(NIPS 2016).

[23] Alex Krizhevsky., I Sutskever., G Hinton 2012.: ImageNet Classification with Deep Convolutional Neural Networks In: Conference and Workshop on Neural Information Processing Systems(NIPS 2012).

[24] Joan Bruna., Wojciech Zaremba., Arthur Szlam., Yann LeCun.: Spectral networks and locally connected networks on graphs. In:International Conference on Learning Representationse(ICLR 2014).

[25] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In: arXiv preprint arXiv:1412.3555, 2014.

[26] Ilya Sutskever., Oriol Vinyals., Quoc V Le.: Sequence to sequence learning with neural networks. In: NIPS, pp. 3104C3112, 2014.

[27] Samy Bengio., Oriol Vinyals., Navdeep Jaitly., Noam Shazeer.: Scheduled sampling for sequence prediction with recurrent neural networks. In: NIPS, pp. 1171C1179, 2015

[28] H. V. Jagadish., Johannes Gehrke., Alexandros Labrinidis., Yannis Papakonstantinou., Jignesh M. Patel., Raghu Ramakrishnan., Cyrus Shahabi.: Big data and its technical challenges. In: Commun. ACM, 57(7):86C94, July 2014.

[29] James Atwood and Don Towsley. Diffusion-Convolutional Neural Networks. In: Neural Information Processing Systems (NIPS 2016).

[30] Braunstein, M. L. Laughery, K. R. and Siegfried, J. B.: Computer Simulation of Driver Behavior During Car Following: A Methodological Study. In: Cornell Aeronautical Lab. Rep. YM-1797-H-1, October, 1963.

[31] P. Chakroborty and S. Kikuchi. : Evaluation of the General Motors based car-following models and a proposed fuzzy inference model. In: Transp. Res., Part C: Emerg. Technol., vol. 7, no. 4, pp. 209C235, 1999.

[32] Panwai, S. and Dia, H.: Comparative evaluation of microscopic car-following behavior. In: IEEE Transactions on Intelligent Transportation Systems, 6(3), 314C 325. 2005

[33] Brackstone M, Mcdonald M.: Car-following: a historical review. In: Transportation Research Part F: Traffic Psychology and Behaviour, pp. 181-196, 1999.