Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting

Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang*

***Abstract*— Traffic forecasting is a particularly challenging application of spatiotemporal forecasting, due to the complicated spatial dependencies on roadway networks and the time-varying traffic patterns. To address this challenge, we learn the traffic network as a graph and propose a novel deep learning framework, Traffic Graph Convolutional Long Short-Term Memory Neural Network (TGC-LSTM), to learn the interactions between roadways in the traffic network and forecast the network-wide traffic state. We define the traffic graph convolution based on the physical network topology. The relationship between traffic graph convolution and the spectral graph convolution is also discussed. The proposed model employs L1-norms on the graph convolution weights and L2-norms on the extracted features to identify the most influential roadways in the traffic network. Experiments show that our TGC-LSTM network is able to capture the complex spatial-temporal dependencies efficiently present in a vehicle traffic network and consistently outperforms state-of-the-art baseline methods on two heterogeneous real-world traffic datasets. The visualization of graph convolution weights shows that the proposed framework can accurately recognize the most influential roadway segments in real-world traffic networks.**

***Index Terms*— Traffic forecasting, Spatial-temporal, Graph convolution, LSTM, Recurrent neural network**

I. Introduction

TRAFFIC forecasting is one of the most challenging components of Intelligent Transportation Systems (ITS). The goal of traffic forecasting is to predict future traffic states in the traffic network given a sequence of historical traffic states and the physical roadway network. Due to the increasing volume and variety of traffic data that has become available in recent years, data-driven traffic forecasting methods have shown considerable promise in their ability to outperform conventional and simulation-based methods [1].

Previous work [2] on this topic roughly categorizes existing models into two categories: classical statistical methods and machine learning models. Much of the work in statistical methods for traffic forecasting was developed years ago, when traffic systems were less complex and transportation datasets were (by necessity) relatively small in size. The capability of such classical methods to deal with high dimensional, complex, and dynamic time series data is quite limited. With the more recent rapid development in computational power, as well as growth in traffic data volume, much of the more recent work on this topic focuses on machine learning methods for traffic forecasting.

With the ability to address high dimensional data and the capability of capturing complex non-linear relationships, machine learning methods, like support vector regression (SVR) [3], tend to outperform the statistical methods, such as autoregressive integrated moving average (ARIMA) [4] and its many variants, with respect to handling complex traffic forecasting problems [5]. However, the full potential of artificial intelligence approaches to traffic forecasting was not exploited until the rise of deep neural network (NN) models (also referred to as deep learning models). Following early works [2], [6] applying NNs to the traffic prediction problem, many NN-based methods have been adopted for traffic forecasting.

Deep learning models for traffic forecasting, such as deep belief networks (DBN) [7] and stacked auto-encoders [8], can effectively learn high dimensional features and achieve good forecasting performance. Due to their efficient use of temporal dependencies, RNN and its variants (long short-term memory (LSTM) [9] and gated recurrent unit (GRU) [10] recurrent neural networks) show excellent potential for traffic forecasting [5], [11], [12]. Although previous RNN-based methods can learn the spatial dependencies, they tend to be over-complex and inevitably capture a certain amount of noise and spurious relationships which likely do not represent the true causal structure in a physical traffic network. Moreover, interpreting the network parameters in terms of real-world spatial dependencies is most often impossible. To address this, other works [13], [14], [15] attempt to model spatial dependencies with convolutional neural network (CNN). However, conventional CNNs are most appropriate for spatial relationships in Euclidean space as represented by 2D matrices

The manuscript was submitted on September 24, 2018. This work is partially supported by the NSF (National Science Foundation) of China under grant number 51138003 and 51329801.

Z. Cui, K. Henrickson, R. Ke, and Y. Wang are with the Department of Civil and Environmental Engineering, University of Washington, Seattle, WA 98195 USA e-mail: (zhiyongc, henr2237, ker27, yinhai@uw.edu).

or images. Thus, spatial features learned in a CNN are not optimal for representing the traffic network structure.

Recently, substantial research has focused on extending the convolution operator to more general, graph-structured data, which can be applied to capture the spatial relationships present in a traffic network. There are two primary ways to conduct the graph convolution. The first class of methods [16], [17], [18], [19] makes use of spectral graph theory, by designing spectral filter/convolutions based on the graph Laplacian matrix. Spectral-based graph convolution has been adopted and combined with RNN [20] and CNN [1] to forecast traffic states. These models successfully apply convolution to graph-structured data, but they do not fully capture the unique properties of graphs [21], like traffic networks. These models [18], [22] usually adopt multiple graph convolution layers, and thus, their learned spatial dependencies are hard to interpret. The other form of graph convolution proposed in several newly-published studies is conducted on graph data dynamically, for example, the dynamic edge-conditioned filters in graph convolution [23], the high-order adaptive graph convolutional network [21]. Still, these methods are not capable of fully accommodating the physical specialties of traffic networks.

One of the deficiencies of the previous graph convolution based models is that the receptive field of the convolution operators are not confined in the graph according to the real structure of the traffic network. The traffic states of two locations far apart from each other in the traffic network should not be influenced by each other in a short time period. Though the spectral graph convolution models [18], [24] can capture features from K-localized neighbors of a vertex in the graph, how to choose the value of K and whether the localized neighbors truly affect the vertex are still questions to be answered. Thus, we propose a free-flow matrix based on the free-flow speed of the real traffic and apply it on the graph convolution operator to learn features from truly influential neighborhood in the traffic network.

In this work, we learn the traffic network as a graph and conduct convolution on the traffic network graph based on the physical roadway characteristics. We propose a traffic graph convolutional LSTM (TGC-LSTM) to model the dynamics of the traffic flow and capture the spatial dependencies. Evaluation results show that the proposed TGC-LSTM outperforms multiple state-of-the-art traffic forecasting baselines. More importantly, the proposed model turns out to be capable of identifying the most influential roadway segments in the real-world traffic networks.

The main contributions of our work include:

1. A traffic graph convolution operator is proposed to accommodate physical specialties of traffic networks and extract comprehensive features.
2. A traffic graph convolutional LSTM neural network is proposed to learn the complex spatial and dynamic temporal dependencies presented in traffic data.
3. L1-norms on traffic graph convolution weights and L2-norms on traffic graph convolution features are added to the model's loss function as two regularization terms to make the graph convolution weight more stable and interpretable.
4. The real-world traffic speed data, including the graph structure of the traffic network, used in this study is published via a publicly available website[1] to facilitate further research on this problem.

II. Literature Review

A. *Deep Learning based Traffic Forecasting*

Deep learning models have shown their superior capabilities of traffic forecasting. Ever since the precursory study [25] using the feed-forward NN for vehicle travel time estimation was proposed, many other NN-based models, including fuzzy NN [26], recurrent NN [6], convolution NN [13], deep belief networks [7], and autoencoders [8], and combinations of these models have been applied to forecast traffic states. With the ability to capture temporal dependencies, the recurrent NN or its variants, like LSTM [9] and GRU [10], is widely adopted as a component of a traffic forecasting model to forecast traffic speed [5], travel time [27], and traffic flow [28] in recent years. Further, bidirectional LSTM layers [29] and shared hidden LSTM layers [30] are designed to capture comprehensive temporal dependencies and predict traffic mobility. To capture spatial relationships present in traffic networks, many forecasting models [13], [31] incorporating CNNs to extract spatial features from 2D spatial-temporal traffic data. Due to the traffic structure is hard to be depicted by 2D spatial-temporal data, studies [15] tried to convert traffic network structures to images and use CNNs to learn spatial features. However, these converted images have a certain amount of noise, inevitably resulting in spurious spatial relationships captured by CNNs. To solve this problem, studies [1], [20] learn the traffic network as a graph and adopt the graph-based convolution operator to extract features from the graph-structured traffic network.

B. *Graph Convolution Networks*

In the last couple of years, many studies attempt to generalize neural networks to work on arbitrarily structured graphs by designing graph convolutional networks. Generally, the graph convolutional networks utilize the adjacency matrix or the Laplacian matrix to depict the structure of a graph. The Laplacian matrix based graph convolution [17], [22] are designed based on the spectral graph theory [32]. As an extension, a localized spectral graph convolution [18] is also proposed to reduce the learning complexity. The adjacency matrix based graph convolution neural networks [19], [21] incorporate the adjacency matrix and their network structures are more flexible. As a typical type of graph, traffic networks have been process and learnt by the spectral graph convolutions [1] and the diffusion graph convolution [16], which is designed for directed graphs. Although these existing methods can extract spatial features from neighborhoods in the traffic network, the physical specialties of roadways, like length, speed limits, and number of lanes, are normally neglected.

[1] https://github.com/zhiyongc/Seattle-Loop-Data

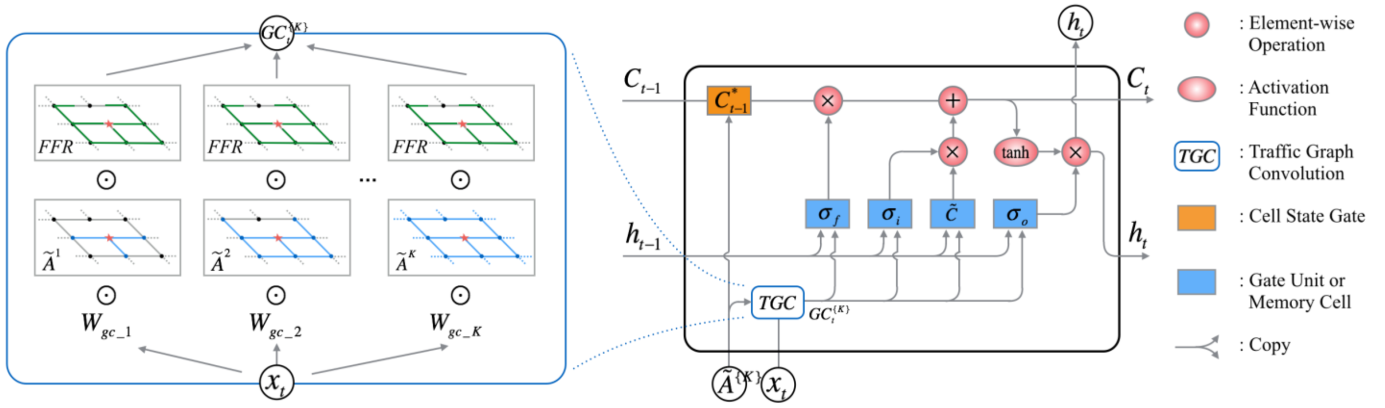


Fig. 1. The architecture of the proposed Traffic Graph Convolution LSTM is shown on the right side. The traffic graph convolution (TGC) as a component of the proposed model is shown on the left side in detail by unfolding the traffic graph convolution at time $\boldsymbol{t}$, in which $\tilde{\boldsymbol{A}}^{\boldsymbol{k}}$s and $\boldsymbol{\mathcal{FFR}}$ with respect to a red star node are demonstrated.

III. Methodology

A. Traffic Forecasting Problem

Traffic forecasting refers to predicting future traffic states, in terms of speed or volume, given previously observed traffic states from a roadway network consisting of N sensor locations and road segments connecting the sensors. The traffic network and the relationship between sensor locations can be represented by an undirected graph $\mathcal{G}$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E}, A)$ with N nodes (vertices) $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$. The connectedness of nodes is represented by an adjacency matrix $A \in \mathbb{R}^{N\times N}$, in which each element $A_{i,j} = 1$ if there is a link connecting node i and node j and $A_{i,j} = 0$ otherwise ($A_{i,i} = 0$). Based on the adjacency matrix, a link counting function $d(v_i, v_j)$ can be defined as counting the minimum number of links traversed from node i to node j. The edges (links) in a graph representing a traffic network are distinct from social network graphs, document citation graphs, or molecule graphs, in several respects: 1) there are no isolated nodes/edges; 2) the traffic status of each link in a traffic network varies with time; and 3) links in a traffic graph have meaningful physical characteristics, such as length of the road segment represented by a link, type of roadway, speed limit, and number of traffic lanes. Thus, we define a distance adjacency matrix, $D \in \mathbb{R}^{N\times N}$, where each element $D_{i,j}$ represents the real roadway distance from node i to j ($D_{i,i} = 0$). Let $X_t \in \mathbb{R}^{N\times P}$ be the graph signals, namely traffic state, for each node at time t, where P is the number of features associated with each node. In this study, we only include the traffic speed feature, and thus, $P = 1$. The traffic forecasting problem aims to learn a function $F(\cdot)$ to map T' time steps of historical graph signals to the next subsequent T time step of graph signals:

$$F\big([X_{t-T'+1}, \dots, X_t]; \mathcal{G}(\mathcal{V}, \mathcal{E}, A, D)\big) = [X_{t+1}, \dots, X_{t+T}] \quad (1)$$

During the forecasting process, we also want to learn the traffic impact transmission between adjacent and neighboring nodes in a traffic network graph.

B. Traffic Graph Convolution

Firstly, we define the k-hop (k-th order) neighborhood $\mathcal{NB}_i = \{v_i \in \mathcal{V} \,|\, d(v_i, v_j) \le k\}$ for each node i. The one-hop neighborhood matrix for a graph $\mathcal{G}$ is exactly the adjacency matrix. Then, the k-hop adjacency matrix can be acquired by calculating the k-th product of A. Here, we define that k-th order adjacency matrix as follows

$$\tilde{A}^k = \mathrm{Bi}(\textstyle\prod_{i=1}^{k} A + I) = \mathrm{Bi}(A^k + I) \quad (2)$$

where $\mathrm{Bi}(\cdot)$ is a function to clip the values of all elements of $A^k + I$ to 1, and thus, $A^k + I \in \{0,1\}^{N\times N}$. The identity matrix I added to A^k makes the nodes self-accessible in the graph. An example $\tilde{A}^k$ with respect to a node (a red star) is shown by blue points on the left side of Fig. 1.

Then, the k-hop graph convolution can be defined as follows

$$GC^k = (W_{gc\_k} \odot \tilde{A}^k) X_t \quad (3)$$

where $\odot$ is the element-wise matrix multiplication operator, W_{gc_k} is the k-hop weight matrix for the k-hop adjacency matrix, and $X_t \in \mathbb{R}^{N\times 1}$ is the traffic feature at time t. However, when taking the underlying physics of vehicle traffic on a road network into consideration, we need to understand that the impact of a roadway segment on adjacent segments is transmitted in two primary ways: 1) slowdowns and/or blockages propagating upstream; and 2) driver behavior and vehicle characteristics associated with a particular group of vehicles traveling downstream. Thus, for a traffic network-based graph or other similar graphs, the impact transmission between non-adjacent nodes cannot bypass the intermediate node/nodes, and thus, we need to consider the reachability of the impact between adjacent and nearby node pairs in the graph convolution. We define a free-flow reachable matrix, $\mathcal{FFR} \in \mathbb{R}^{N\times N}$, that

$$\mathcal{FFR}_{i,j} = \begin{cases} 1, & S_{i,j}^{\mathcal{FF}} m\Delta t - D_{i,j} \ge 0 \\ 0, & \text{otherwise} \end{cases}, \forall\, v_i, v_j \in \mathcal{V} \quad (4)$$

where $S_{i,j}^{\mathcal{FF}}$ is the free-flow speed between node i and j, and free-flow speed refers to the average speed that a motorist would travel if there were no congestion or other adverse conditions (such as severe weather). Δt is the duration of time quantum and m is a number counting how many time intervals are considered to calculate the distance travelled under free-flow speed. Thus, m determines the temporal influence of formulating the $\mathcal{FFR}$. Each element $\mathcal{FFR}_{i,j}$ equals one if vehicles can traverse from node i to j in m time-step, $m \cdot \Delta t$, with free-flow speed, and $\mathcal{FFR}_{i,j} = 0$ otherwise. All diagonal values of $\mathcal{FFR}$ are set as one. An example $\mathcal{FFR}$ with respect to a node (a red star) is shown by green lines on the left side of Fig. 1. Thus, the traffic graph convolution is defined as

$$GC^k = (W_{gc\_k} \odot \tilde{A}^k \odot \mathcal{FFR}) X_t \quad (5)$$

where the k-hop adjacency matrix is multiplied element-wise with $\mathcal{FFR}$ to ensure the traffic impact transmission between k-hop adjacent nodes follow established traffic flow theory [33]. For a specific graph, when we increase k, the $\tilde{A}^k \odot \mathcal{FFR}$ will eventually converge such that $k = K$ and $\tilde{A}^K \odot \mathcal{FFR} = \mathcal{FFR}$.

Thus, at most, only K hops of graph convolution features need to be extracted from the data X_t. Features extracted by the graph convolution within the K hops adjacent nodes with respect to time t are concatenated together as follows

$$\boldsymbol{GC}_t^{\{K\}} = [GC_t^1, GC_t^2, \dots, GC_t^K] \quad (6)$$

The $\boldsymbol{GC}^{\{K\}} \in \mathbb{R}^{N \times K}$ is set of K-th order traffic graph convolutional features, shown in the left part of Fig. 1, that can be fed to the network models described in the following subsection.

1) Relationship to Spectral Graph Convolution

The spectral graph convolution (SGC) is defined based on the graph Laplacian L, whose combinatorial form is defined as

$$L = D - A \quad (7)$$

where $D \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix with $D_{ii} = \sum_j A_{ij}$. The Laplacian matrix L is symmetric positive semi-definite such that it can be diagonalized via eigen-decomposition, $L = U \Lambda U^T$, where Λ is a diagonal matrix containing the eigenvalues and U consists of the eigenvectors.

The spectral graph convolution operator $*_{\mathcal{G}}$ is defined in the Fourier domain [32]. The operator $*_{\mathcal{G}}$ can be described as

$$x *_{\mathcal{G}} h = U\big((U^T x) \odot (U^T h)\big) = U \hat{h}(\Lambda) U^T x \quad (8)$$

where x is the input graph signal and h is a filtering function that can be considered as a convolutional kernel. $\hat{h}(\cdot)$ is the filtering function in the Fourier frequency domain. When applying the SGC into neural networks, the Λ will be replaced by a diagonal parameter matrix:

$$x *_{\mathcal{G}} h = U \begin{bmatrix} \theta_0 & & \\ & \ddots & \\ & & \theta_{N-1} \end{bmatrix} U^T x = U \hat{h}_\theta(\Lambda_\theta) U^T x \quad (9)$$

Further, Defferrard et al. 2016 proposed to employ a polynomial filter $\hat{h}_\theta(\Lambda_\theta) = \sum_{j=0}^{K-1} \theta_j \Lambda^j$ to defined the localized spectral graph convolution (LSGC), which is formulated as:

$$\begin{aligned} x *_{\mathcal{G}} h &= U \begin{bmatrix} \sum_{j=0}^{K-1} \theta_j \lambda_0^j & & \\ & \ddots & \\ & & \sum_{j=0}^{K-1} \theta_j \lambda_{N-1}^j \end{bmatrix} U^T x \\ &= U \sum_{j=0}^{K-1} \theta_j \Lambda^j U^T x \\ &= \sum_{j=0}^{K-1} \theta_j L^j x \quad (10) \end{aligned}$$

In the neural network, each parameter θ_i can be updated via the back-propagation process. The advantages of the LSGC is that it only has K parameters and does not need eigen-decomposition. It is well spatial localized and each convolution operation on a centered vertex extracts the summed weighted feature of the vertex's K-hop neighbors.

THE COMPARISON BETWEEN TGC, SGC, AND LSGC IN TERMS OF THE NUMBER OF PARAMETERS, COMPUTATIONAL TIME, AND LOCALIZED FEATURE EXTRACTION, IS SHOWN IN

TABLE I. Comparing to SGC and LSGC, TGC is better spatial localized because it can extract local features based on physical properties of roadways by incorporating the $\mathcal{FFR}$. TGC with more parameters has better capabilities of representing the relationships between connected nodes in the graph. Further, SGC and LSGC normally need multiple convolutional layers, which leads the SGC and LSGC lose their interpretability. However, TGC only needs one convolution layer and its parameters can be well interpreted.

C. Traffic Graph Convolutional LSTM

We propose a Traffic Graph Convolutional LSTM (TGC-LSTM) recurrent neural network, as shown on the right side of the Fig. 1, which learns both the complex spatial dependencies

TABLE I
COMPARISON BETWEEN TGC, SGC, AND LSGC

Model	TGC	SGC	LSGC
Equation of GC Weights	$W_{gc\_k} \odot \tilde{A}^k \odot \mathcal{FFR}$	$U \hat{h}_\theta(\Lambda_\theta) U^T$	$\sum_{j=0}^{k-1} \theta_j L^j$
Number of Parameters	N^2	N	k
Computational Time	O(N^2)	O(N^2)	O(N^2) (The computation complexity of all these three methods can be reduced via sparse matrices multiplication)
Extract Localized features	Yes. It is k-localized incorporating roadway physical properties.	No	Yes. It is exactly k-localized.

and the dynamic temporal dependencies presented in traffic data. In this model, the gates structure in the vanilla LSTM [9] and the hidden state are unchanged, but the input is replaced by the graph convolution features, which are reshaped into a vector $\boldsymbol{GC}^{\{K\}} \in \mathbb{R}^{KN}$.The forget gate f_t, the input gate i_t, the output gate o_t, and the input cell state $\tilde{C}_t$, at time t are defined as follows

$$\mathrm{f}_t = \sigma_g\left(W_f \cdot \boldsymbol{GC}_t^{\{K\}} + U_f \cdot h_{t-1} + b_f\right) \tag{11}$$

$$\mathrm{i}_t = \sigma_g\left(W_i \cdot \boldsymbol{GC}_t^{\{K\}} + U_i \cdot h_{t-1} + b_i\right) \tag{12}$$

$$\mathrm{o}_t = \sigma_g\left(W_o \cdot \boldsymbol{GC}_t^{\{K\}} + U_o \cdot h_{t-1} + b_o\right) \tag{13}$$

$$\tilde{C}_t = tanh\left(W_C \cdot \boldsymbol{GC}_t^{\{K\}} + U_C \cdot h_{t-1} + b_C\right) \tag{14}$$

where $\cdot$ is the matrix multiplication operator. W_f, W_i, W_o, and $W_C \in \mathbb{R}^{KN\times N}$ are the weight matrices, mapping the input to the three gates and the input cell state, while U_f, U_i, U_o, and $U_C \in \mathbb{R}^{N\times N}$ are the weight matrices for the preceding hidden state. b_f, b_i, b_o, and $b_C \in \mathbb{R}^N$ are four bias vectors. The σ_g is the gate activation function, which typically is the sigmoid function, and tanh is the hyperbolic tangent function.

Due to each node in a traffic network graph is influenced by the preceding states of itself and its neighboring nodes, the LSTM cell state of each node in the graph should also be affected by neighboring cell states. Thus, a cell state gate is designed and added in the LSTM cell. The cell state gate, as shown in Fig. 1, is defined as follows

$$C_{t-1}^{*} = W_{\mathcal{N}} \odot (\tilde{A}^K \odot \mathcal{FFR}) \cdot C_{t-1} \tag{15}$$

where $W_{\mathcal{N}}$ is a weight matrix to measure the contributions of neighboring cell states. To correctly reflect the traffic network structure, the $W_{\mathcal{N}}$ is constrained by multiplying a $\mathcal{FFR}$ based K -hop adjacency matrix, $\tilde{A}^K \odot \mathcal{FFR}$. With this gate, the influence of neighboring cell states will be considered when the cell state is recurrently input to the subsequent time step. Then, the final cell state and the hidden state are calculated as follows

$$C_t = \mathrm{f}_t \odot C_{t-1}^{*} + \mathrm{i}_t \odot \tilde{C}_t \tag{16}$$

$$h_t = \mathrm{o}_t \odot \tanh(C_t) \tag{17}$$

D. Loss and Graph Convolution Regularization

As mentioned in the previous section, the TGC-LSTM output at time t is h_t, namely the predicted value $\hat{Y}_t = h_t$. Let Y_t denote the label at time t, and thus, the loss is defined as

$$L_t = \mathrm{Loss}\left(\hat{Y}_t - Y_t\right) \tag{18}$$

where $\mathrm{Loss}(\cdot)$ is a function to calculate the error between the predicted value $\hat{Y}_t$ and the label/true value Y_t. Normally, the $\mathrm{Loss}(\cdot)$ function is a Mean Squared Error (MSE) function for predicting continuous values or a Cross Entropy function for classification problems.

For a time sequence, the label of time step t is the input of the next step, $t+1$, such that $Y_t = X_{t+1}$, and thus, the loss can be defined as

$$L_t = \mathrm{Loss}\left(\hat{Y}_t - Y_t\right) = \mathrm{Loss}(h_t - X_{t+1}) \tag{19}$$

To make the graph convolution feature more stable and interpretable, we add two regulation terms on the loss function.

1) Regularization on Graph Convolution weights

Because the graph convolution weights are not confined to be positive and each node's extracted features are influenced by multiple neighboring nodes, the graph convolution weights can vary a lot while training. Ideally, the convolution weights would be themselves informative, so that the relationships between different nodes in the network could be interpreted and visualized by plotting the convolution weights. This is not likely to be possible without regularization, because very high or low weights tend to appear somewhat randomly, with the result that high/low weights tend to cancel each other out. In combination, such weights can still represent informative features for the network, but they cannot reflect the true relationship between nodes in the graph. Thus, we add L1-norm of the graph convolution weight matrices to the loss function as a regularization term to make these weight matrices as sparse as possible. The L1 regularization term is defined as follows

$$R^{\{1\}} = \left\|\boldsymbol{W}_{gc}\right\|_1 = \sum_{i=1}^{K} \left|W_{gc\_i}\right| \tag{20}$$

In this way, the trained graph convolution weight can be sparse and stable, and thus, it will be more intuitive to distinguish which neighboring node or group of nodes contribute most.

2) Regularization on Graph Convolution features

Considering that the impact of neighboring nodes with respect to a specific node must be transmitted through all nodes between the node of interest and the influencing node, features extracted from different hops in the graph convolution should not vary dramatically. Thus, to restrict the difference between features extracted from adjacent hops of graph convolution, an L2-norm based TGC feature regularization term is added on the loss function at each time step. The regularization term is defined as follows

$$R_t^{\{2\}} = \left\|\boldsymbol{GC}_t^{\{K\}}\right\|_2 = \sqrt{\sum_{i=1}^{K-1} \left(GC_t^i - GC_t^{i+1}\right)^2} \tag{21}$$

In this way, the features extracted from adjacent hops of graph convolution should not differ dramatically, and thus, the graph convolution operator should be more in keeping with the physical realities of the relationships present in a traffic network.

Then, the total loss function at time t can be defined as follows

$$L_t = \mathrm{Loss}(h_t - X_{t+1}) + \lambda_1 R^{\{1\}} + \lambda_2 R_t^{\{2\}} \tag{22}$$

where λ_1 and λ_2 are penalty terms to control the weight magnitude of the regularization terms on graph convolution weights and features.

IV. Experiments

A. Dataset Description

In this study, two real-world network-scale traffic speed datasets are utilized. The first contains data collected from inductive loop detectors deployed on four connected freeways (I-5, I-405, I-90, and SR-520) in the Greater Seattle Area, shown in Fig. 2 (a). This dataset, which is publicly accessible, contains traffic state data from 323 sensor stations over the entirety of 2015 at 5-minute intervals. The second contains road link-level traffic speeds aggregated from GPS probe data collected by commercial vehicle fleets and mobile apps

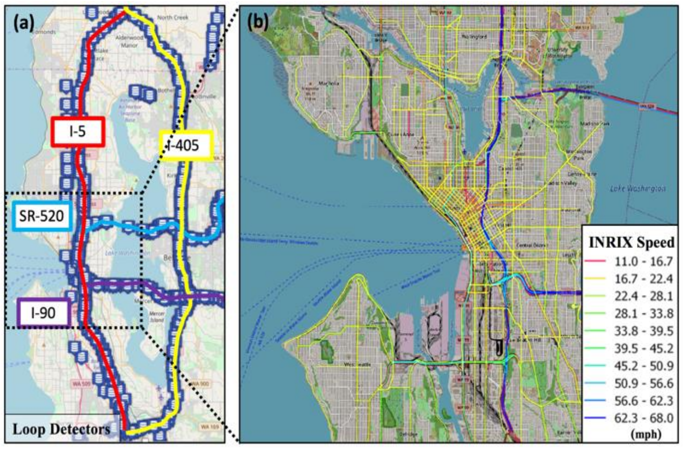


Fig. 2. (a) LOOP dataset covering the freeway network in Seattle area; (b) INRIX dataset covering the downtown Seattle area, where traffic segments are plotted with colors.

provided by the company INRIX. The INRIX traffic network covers the Seattle downtown area, shown in Fig. 2 (b). This dataset describes the traffic state at 5-minute intervals for 1014 road segments and covers the entire year of 2012. We use LOOP data and INRIX data to denote these two datasets, respectively, in this study.

We adopt the speed limit as the free-flow speed, which for the segments in the LOOP traffic network is 60mph in all cases. The INRIX traffic network contains freeways, ramps, arterials and urban corridors, and so the free-flow speeds of INRIX traffic network range from 20mph to 60mph. The distance adjacency matrices D and free-flow reachable matrices $\mathcal{FFR}$ for both datasets are calculated based on the roadway characteristics and topology.

B. *Experimental Settings*

1) *Baselines*

We compare TGC-LSTM with the following baseline models, (1) ARIMA: Auto-Regressive Integrated Moving Average model [4]; (2) SVR: Support Vector Regression [3]; (3) FNN: Feed forward neural network with two hidden layers. (4) LSTM: Long Short-Term Memory recurrent neural network [9]; (5) SGC+LSTM: stacking one-layer spectral graph convolution [22] and LSTM; (6) LSGC+LSTM: stacking one-layer localized spectral graph convolution [18] and LSTM. All the neural networks are implemented based on PyTorch 0.3.1 and they are trained and evaluated on a single NVIDIA GeForece GTX 1080 Ti with 11GB memory.

2) *TGC-LSTM Model*

For both datasets, the dimensions of the hidden states of the TGC-LSTM are set as the amount of the nodes in the traffic network graphs. The size of hops in the graph convolution can vary, but we set it as 3, $K = 3$, for the model evaluation and comparison in this experiment. In this case, the $\mathcal{FFR}$ is calculated based on three time steps. The learning rates for the two regularization terms are all set as 0.01. We train our model by minimizing the mean square error using RMSProp [34] with the batch size of 10 and the initial learning rate of 10^{-5}.

3) *Evaluation*

In this study, the samples of the input are traffic time series data with 10 time steps. The output/label is the next subsequent data of the input sequence. The performance of the proposed and the compared models are evaluated by three commonly used metrics in traffic forecasting, including 1) Mean Absolute Error (MAE), 2) Mean Absolute Percentage Error (MAPE), and 3) Root Mean Squared Error (RMSE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|Y_t - \hat{Y}_t\right| \tag{23}$$

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{Y_t - \hat{Y}_t}{Y_t}\right| * 100\% \tag{24}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_t - \hat{Y}_t\right)^2} \tag{25}$$

C. *Experimental Results*

TABLE II demonstrates the results of the TGC-LSTM and other baseline models on the two datasets. The proposed method outperforms other models with all the three metrics on the two datasets. The ARIMA and SVR cannot compete with other methods, which suggest that non-neural-network approaches are less appropriate for this network-wide prediction task, due to the complex spatiotemporal dependencies and the high dimension features in the datasets. The basic FNN does not perform well on predicting spatial-temporal sequence. The SGC+LSTM performs better than vanilla LSTM, which demonstrates the feature extraction by using spectral graph convolution is beneficial for traffic forecasting. However, the LSGC+LSTM does not outperform LSTM resulting from utilizing one-layer LSGC, whose parameters is not enough for representing the network features.

TABLE II

PERFORMANCE COMPARISON OF DIFFERENT APPROACHES. (THE NUMBER OF HOPS K IS SET AS 3 IN THE GRAPH CONVOLUTION RELATED MODEL)

Model	Loop Data			INRIX Data		
	MAE (mph)±STD	MAPE	RMSE	MAE (mph)±STD	MAPE	RMSE
ARIMA	6.10± 1.09	13.85%	10.65	4.80 ± 0.32	13.51%	10.85
SVR	6.85± 1.17	14.39%	11.12	4.78 ± 0.37	13.37%	10.44
FNN	4.45± 0.81	10.19%	7.83	2.31 ± 0.17	8.35%	5.92
LSTM	2.70± 0.18	6.83%	4.97	1.14 ± 0.09	3.88%	2.43
LSGC+LSTM	3.16± 0.23	7.51%	6.18	1.38 ± 0.12	4.54%	2.82
SGC+LSTM	2.64± 0.12	6.52%	4.80	1.07 ± 0.08	3.74%	2.28
TGC-LSTM	2.57± 0.10	6.01%	4.63	1.02 ± 0.07	3.28%	2.18

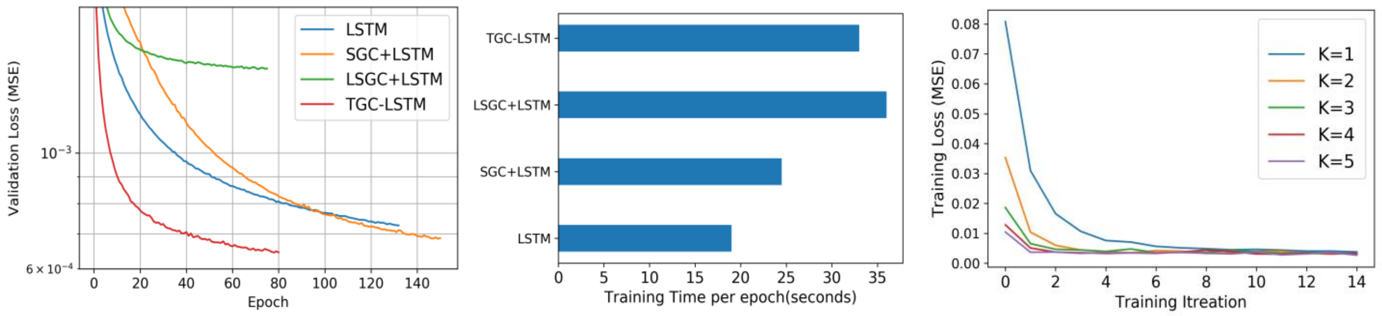


Fig. 4. (a) Validation loss versus training epoch (batch size = 40 and early stopping patience = 10 epochs). (b) Histogram of model's training time per epochs. (c) Compare training efficiency with different K hops of TGC: training loss versus training iteration (batch size = 40). (The figures are generated based on the LOOP data.)

The proposed TGC-LSTM, which capture graph-based features while accommodating the physical specialties of traffic networks, performs better than all other approaches. It should be noted that the INRIX data misses a number of observations during nighttime and off-peak hours and these missing observations are filled with free-flow speed. Thus, there are few variations at the non-peak hours in the INRIX data. Further, the speed values in the INRIX data are all integers. Therefore, the calculated errors of the INRIX data is less than that of the LOOP data and the evaluated performance on INRIX data is inflated somewhat.

Fig. 3 shows a histogram of performance comparison on the effects of orders (hops) of the graph convolution in the TGC-LSTM. The model performance is improved when the value of K increases. For the LOOP data, the performance improves slightly when K is gradually increased. But for the INRIX data, there is a big improvement in when K increases to two from one. The complex structure and the various road types in the INRIX traffic network could be the main reason for this performing difference. Further, when K is larger than two, the improvement of the prediction is quite limited. This is also the reason why we choose K=3 in the model comparison part, as shown in TABLE II.

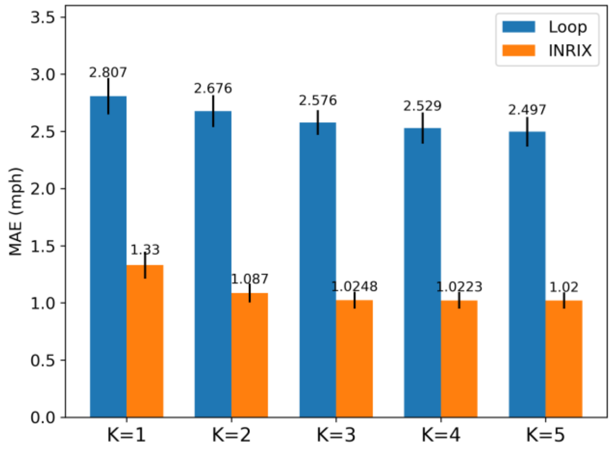


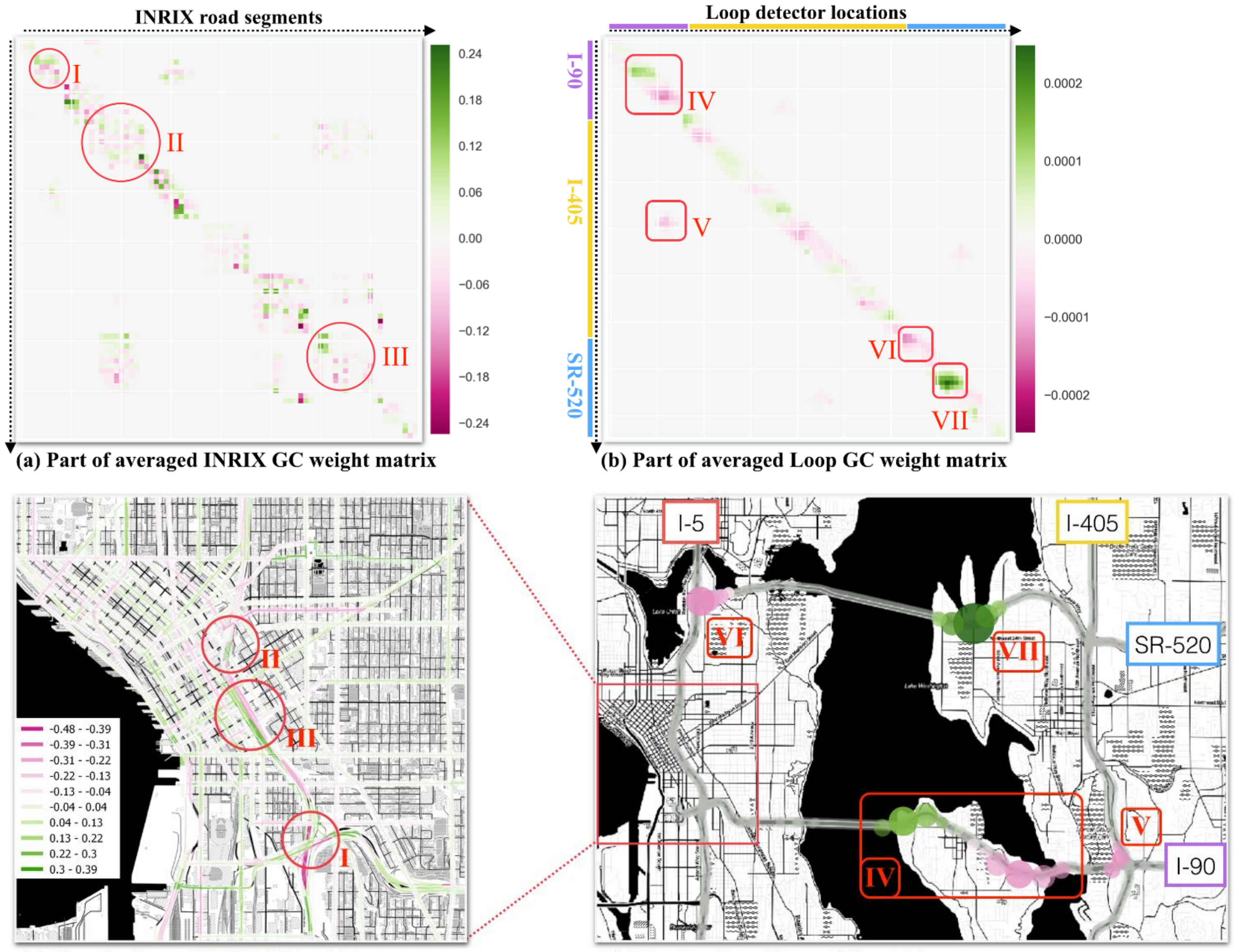
Fig. 3. Histogram of performance comparison for the influence of orders (hops) of graph convolution in the TGC-LSTM on INRIX and LOOP datasets.

D. *Training Efficiency*

In this subsection, we compare the training efficiency of the proposed model and other LSTM-based models. Fig. 4 (a) shows the validation loss curves versus the training epoch. Due to the early stopping mechanism is used in the training process, the numbers of training epochs are different. The TGC-LSTM needs less epochs to converge than the SGC+LSTM and the LSGC+LSTM. In addition, the loss of the TGC-LSTM decreases fastest among the compared models. Fig. 4 (b) shows the comparison of the training time per epoch of different models. TGC-LSTM costs twice as much as LSTM does. The time required for SGC+LSTM is less than that for TGC-LSTM, while LSGC+LSTM costs slightly more than TGC-LSTM. Fig. 4 (c) shows the training losses of TGC-LSTM with different hops of graph convolution components. The rate of convergence increases when increasing the number of hops, k. In our experiments, when k is larger than 3, the training and validation results improve only marginally for both INRIX and LOOP datasets.

E. *Effect of Regularization*

The model's loss function contains two regularization terms, the L1-norm on the graph convolution weights and the L2-norm on the graph convolutional features. These regularization terms help the graph convolution weights in the trained model to be sparser, more clustered, and thus, more interpretable. Fig. 5 (a) and (b) show portions of the averaged graph convolution weight matrices for the INRIX data and the LOOP data, respectively, where $K = 3$ and the average weight is calculated by $\frac{1}{K}\sum_{i=1}^{K} W_i \odot \tilde{A}^i \odot \mathcal{FFR}$. The road segment names, which are not displayed, are aligned on the vertical and horizontal axes with the same order in each figure. The colored dots in the matrices in Fig. 5 (a) and (b) illustrate the weight of the contribution of a single node to its neighboring nodes. Since we align the traffic states of roadway segments based on their interconnectedness in the training data, most of the weights are distributed around the diagonal line of the weight matrix. The INRIX network is more complex and the average degree of nodes in the INRIX graph is higher than that in the LOOP graph. Hence, the dots in the average weight matrix of the INRIX graph convolution are



(a) Part of averaged INRIX GC weight matrix **(b) Part of averaged Loop GC weight matrix**

(c) Visualization of INRIX GC weight on map **(d) Visualization of Loop GC weight on map**

Fig. 5. (a) Visualization of a proportion of the INRIX GC weight matrix, in which three representative weight areas are tagged. (b) Visualization of a proportion of the LOOP GC weight matrix, in which four representative weight areas are tagged. (c) Visualization of the INRIX graph convolution weight on the real traffic network using colored lines. (d) Visualization of the four tagged weight areas in the LOOP graph convolution weight on the Seattle freeway network using colorful circles.

more scattered. But these dots still form multiple clusters demonstrating the weights of several nearby or connected road segments. Considering roadway segments are influenced by their neighboring or nearby connected segments, the nodes with the large absolute weight in a cluster are very likely to be key road segments in the local traffic network. In this way, we can infer the bottlenecks of the traffic network from the traffic graph convolution weight matrices.

F. Model Interpretation and Visualization

To better understand the contribution of the graph convolution weight, we mark seven groups of representative weights in Fig. 5 (a) and (b) and visualize their physical locations on the real map in Fig. 5 (c) and (d), by highlighting them with Roman numerals and red boxes. The influence of these marked weights on neighboring nodes in the INRIX and LOOP data are visualized by lines and circles, respectively, considering the INRIX traffic network is too dense to use circles. The darkness of the green and pink colors and the sizes of the circles represent the magnitude of influence. It should be noted that the darkness of colors on lines on the INRIX map and the size of the circles on the LOOP map will change when the model is trained with different scales of regularization terms (λ_1 and λ_2).

From Fig. 5 (c), we can find the marked areas with dark colors in the INRIX GC weight matrix, (I), (II), and (III), are all located at very busy and congested freeway entrance and exit ramps in Seattle downtown area. In Fig. 5 (d), the area tagged with (IV) is quite representative because the two groups of circles are located at the intersections between freeways and two main corridors that represent the entrances to an island (Mercer Island). Areas (V) and (VI) are the intersections between I-90 and I-405 and between I-5 and SR-520, respectively. The VII area located on SR-520 contains a frequent-congested ramp connecting to the city of Bellevue, the location of which is highlighted by the biggest green circle. Additionally, there are many other representative areas in the graph convolution weight matrix, but we cannot show all of them due to space limits. By comparing the weight matrix with the physical realities of the traffic network, it can be shown that the proposed method effectively captures spatial dependencies and helps to identify the most influential points/segments in the traffic network.

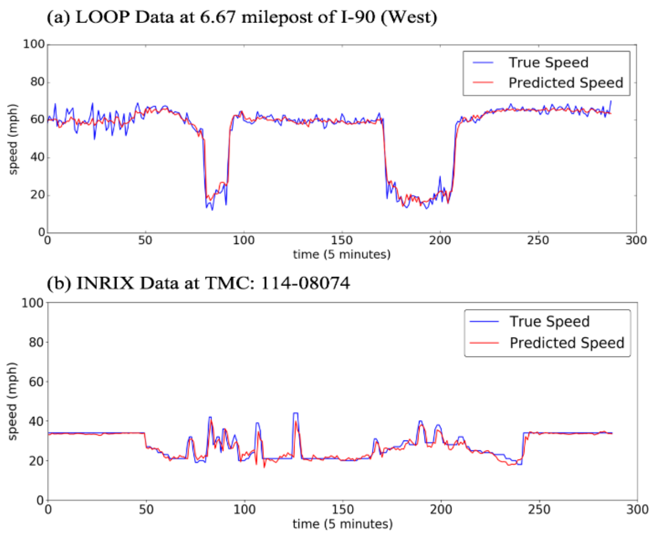


Fig. 6. Traffic time series forecasting visualization for LOOP and INRIX datasets on two randomly selected days. The x-axis is time and the unit is 5-minutes.

Fig. 6 visualizes the predicted traffic speed sequences and the ground truth of two locations selected from the LOOP and INRIX dataset. Though the traffic networks of the two datasets are very different, the curves demonstrate that the trends of the traffic speed are predicted well at both peak traffic and off-peak hours.

V. Conclusion

In this paper, we learn the traffic network as a graph and define a traffic graph convolution operation to capture spatial features from traffic network. We propose a traffic graph convolutional LSTM recurrent neural network for forecasting traffic spatial-temporal data. The regularization terms on the graph convolution help the proposed model be more stable and interpretable. By evaluating on two large-scale real-world traffic datasets, our approach outperforms the baselines. For future work, we will move forward to conduct the convolution on both spatial and temporal dimensions to make the neural network more interpretable.

References

[1] B. Yu, H. Yin, and Z. Zhu, “Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting,” *arXiv Prepr. arXiv1709.04875*, 2017.

[2] D. Park and L. R. Rilett, “Forecasting freeway link travel times with a multilayer feedforward neural network,” *Comput. Civ. Infrastruct. Eng.*, vol. 14, no. 5, pp. 357–367, 1999.

[3] A. J. Smola and B. Schölkopf, “A tutorial on support vector regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, 2004.

[4] M. M. Hamed, H. R. Al-Masaeid, and Z. M. B. Said, “Short-term prediction of traffic volume in urban arterials,” *J. Transp. Eng.*, vol. 121, no. 3, pp. 249–254, 1995.

[5] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, “Long short-term memory neural network for traffic speed prediction using remote microwave sensor data,” *Transp. Res. Part C Emerg. Technol.*, vol. 54, pp. 187–197, 2015.

[6] J. Van Lint, S. Hoogendoorn, and H. Van Zuylen, “Freeway travel time prediction with state-space neural networks: modeling state-space dynamics with recurrent neural networks,” *Transp. Res. Rec. J. Transp. Res. Board*, no. 1811, pp. 30–39, 2002.

[7] W. Huang, G. Song, H. Hong, and K. Xie, “Deep Architecture for Traffic Flow Prediction: Deep Belief Networks With Multitask Learning.,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, 2014.

[8] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, and others, “Traffic flow prediction with big data: A deep learning approach.,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, 2015.

[9] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[10] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” *arXiv Prepr. arXiv1406.1078*, 2014.

[11] Z. Cui, R. Ke, and Y. Wang, “Deep Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction,” *arXiv Prepr. arXiv1801.02143*, 2018.

[12] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, “Deep learning: A generic approach for extreme condition traffic forecasting,” in *Proceedings of the 2017 SIAM International Conference on Data Mining*, 2017, pp. 777–785.

[13] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, “Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction,” *Sensors*, vol. 17, no. 4, p. 818, 2017.

[14] J. Zhang, Y. Zheng, and D. Qi, “Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction.,” in *AAAI*, 2017, pp. 1655–1661.

[15] H. Yu, Z. Wu, S. Wang, Y. Wang, and X. Ma, “Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks,” *Sensors*, vol. 17, no. 7, p. 1501, 2017.

[16] J. Atwood and D. Towsley, “Diffusion-convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 1993–2001.

[17] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral Networks and Locally Connected Networks on Graphs,” *arXiv Prepr. arXiv1312.6203*, Dec. 2013.

[18] 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.

[19] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR ’17)*, 2017.

[20] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion

Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," in *International Conference on Learning Representations (ICLR '18)*, 2018.

[21] Z. Zhou and X. Li, "Convolution on Graph: A High-Order and Adaptive Approach," *arXiv Prepr. arXiv1706.09916*, 2018.

[22] M. Henaff, J. Bruna, and Y. LeCun, "Deep Convolutional Networks on Graph-Structured Data," *arXiv Prepr. arXiv1506.05163*, Jun. 2015.

[23] M. Simonovsky and N. Komodakis, "Dynamic edgeconditioned filters in convolutional neural networks on graphs," in *Proc. CVPR*, 2017.

[24] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting," *arXiv Prepr. arXiv1707.01926*, 2017.

[25] J. Hua and A. Faghri, "Applcations of Artificial Neural Networks to Intelligent Vehicle-Highway Systems," *Record (Washington).*, vol. 1453, p. 83, 1994.

[26] H. Yin, S. Wong, J. Xu, and C. K. Wong, "Urban traffic flow prediction using a fuzzy-neural approach," *Transp. Res. Part C Emerg. Technol.*, vol. 10, no. 2, pp. 85–98, 2002.

[27] Y. Duan, Y. Lv, and F.-Y. Wang, "Travel time prediction with LSTM neural network," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, 2016, pp. 1053–1058.

[28] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Intell. Transp. Syst.*, vol. 11, no. 2, pp. 68–75, 2017.

[29] Z. Cui, R. Ke, and Y. Wang, "Deep Stacked Bidirectional and Unidirectional LSTM Recurrent Neural Network for Network-wide Traffic Speed Prediction," in *6th International Workshop on Urban Computing (UrbComp 2017)*, 2016.

[30] X. Song, H. Kanasugi, and R. Shibasaki, "DeepTransport: Prediction and Simulation of Human Mobility and Transportation Mode at a Citywide Level.," in *IJCAI*, 2016, pp. 2618–2624.

[31] W. Jin, Y. Lin, Z. Wu, and H. Wan, "Spatio-Temporal Recurrent Convolutional Networks for Citywide Short-term Crowd Flows Prediction," in *Proceedings of the 2nd International Conference on Compute and Data Analysis*, 2018, pp. 28–35.

[32] 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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[33] C. F. Daganzo, "The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory," *Transp. Res. Part B Methodol.*, vol. 28, no. 4, pp. 269–287, 1994.

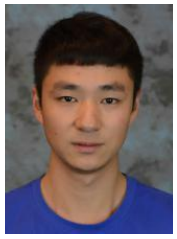
[34] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural networks Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

Zhiyong Cui (S'15) received a B.S. degree in software engineering from Beihang University (2012) and a M.S. degree in software engineering and microelectronics from Peking University (2015). He is currently working toward the Ph.D. degree in civil and environmental engineering at University of Washington, Seattle, WA, USA.

Since 2015, he has been a Research Assistant with the Smart Transportation Applications and Research Laboratory (STAR Lab), University of Washington. His research interests include deep learning, machine learning, traffic data mining and intelligent transportation systems.

Kristian C. Henrickson received a B.S. in Civil Engineering from the University of Idaho in Moscow, ID in 2013, and M.S. in Civil and Environmental Engineering from the University of Washington in Seattle, WA in 2014. He is currently pursuing a Ph.D. in Civil and Environmental Engineering at the University of Washington.

From 20012 to 2013, he was an undergraduate Research Assistant with the National Institute for Advanced Traffic Safety at the University of Idaho. Since 2013, he has worked as a Research Assistant at the University of Washington STAR Laboratory. His research interest includes transportation data quality issues, machine learning, and transportation safety. Mr. Henrickson was selected as the Pacific Northwest Transportation Consortium Student of the year in 2014, and as the University of Idaho Civil Engineering department outstanding senior in 2013.



Ruimin Ke (S'15) received the B.S. degree from the department of automation, Tsinghua University, Beijing, China, in 2014. He is currently working toward the Ph.D. degree in civil and environmental engineering at University of Washington, Seattle, WA, USA.

Since 2014, he has been a Research Assistant with the Smart Transportation Applications and Research Laboratory (STAR Lab), University of Washington. His research interests include traffic video analysis, big data applications in transportation, traffic safety analysis and intelligent transportation systems.

Mr. Ke serves as a member of the Statewide Transportation Data and Information Systems Committee (ABJ20) of Transportation Research Board (TRB). He is currently the Chair of Student Affairs with Chinese Overseas Transportation Association (COTA). He was also a Technical Program Committee Member at the 2015 IEEE International Smart Cities Conference.

Dr. Yinhai Wang is a professor in transportation engineering and the founding director of the Smart Transportation Applications and Research Laboratory (STAR Lab) at the University of Washington (UW). He also serves as director for Pacific Northwest Transportation Consortium (PacTrans), USDOT University Transportation Center for Federal Region 10. He has a Ph.D. in transportation engineering from the University of Tokyo (1998) and a master's degree in computer science from the UW. Dr. Wang's active research fields include traffic sensing, e-science of transportation, transportation safety, etc. He has published over 100 peer reviewed journal articles and delivered more than 110 invited talks and nearly 200 other academic presentations.

Dr. Wang serves as a member of the Transportation Information Systems and Technology Committee and Highway Capacity and Quality of Service Committee of the Transportation Research Board (TRB). He is currently a member of the steering committee for the IEEE Smart Cities and chaired the First IEEE International Smart Cities Conference in 2015. He was an elected member of the Board of Governors for the IEEE ITS Society from 2010 to 2013 and served on the Board of Governors for the ASCE Transportation & Development Institute 2013-2015. Additionally, Dr. Wang is associate editor for three journals: Journal of ITS, Journal of Computing in Civil Engineering, and Journal of Transportation Engineering. He was the winner of the ASCE Journal of Transportation Engineering Best Paper Award for 2003.