Published: 27 January 2019

Citation in BibTeX format

AAAI'19: AAAI Conference on Artificial Intelligence
January 27 - February 1, 2019
Hawaii, Honolulu, USA

RESEARCH-ARTICLE

Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting

ZULONG DIAO, Hunan University, Changsha, Hunan, China

XIN WANG, Stony Brook University, Stony Brook, NY, United States

DAFANG ZHANG, Hunan University, Changsha, Hunan, China

YINGRU LIU, Stony Brook University, Stony Brook, NY, United States

KUN XIE, Hunan University, Changsha, Hunan, China

SHAOYAO HE, Hunan University, Changsha, Hunan, China

Dynamic Spatial-Temporal Graph Convolutional Neural Networks for Traffic Forecasting

Zulong Diao,[1] **Xin Wang,**[2] **Dafang Zhang,**[1*] **Yingru Liu,**[2] **Kun Xie,**[1] **Shaoyao He**[3]
[1]The College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China
[2]The Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, NY, 11794
[3]The College of Architecture, Hunan University, Changsha 410082
{zldiao, dfzhang, xiekun}@hnu.edu.cn, {x.wang, yingru.liu}@stonybrook.edu, syhe829@163.com

Abstract

Graph convolutional neural networks (GCNN) have become an increasingly active field of research. It models the spatial dependencies of nodes in a graph with a pre-defined Laplacian matrix based on node distances. However, in many application scenarios, spatial dependencies change over time, and the use of fixed Laplacian matrix cannot capture the change. To track the spatial dependencies among traffic data, we propose a dynamic spatio-temporal GCNN for accurate traffic forecasting. The core of our deep learning framework is the finding of the change of Laplacian matrix with a dynamic Laplacian matrix estimator. To enable timely learning with a low complexity, we creatively incorporate tensor decomposition into the deep learning framework, where real-time traffic data are decomposed into a global component that is stable and depends on long-term temporal-spatial traffic relationship and a local component that captures the traffic fluctuations. We propose a novel design to estimate the dynamic Laplacian matrix of the graph with above two components based on our theoretical derivation, and introduce our design basis. The forecasting performance is evaluated with two real-time traffic datasets. Experiment results demonstrate that our network can achieve up to 25% accuracy improvement.

Introduction

Traffic forecasting is fundamental to the performance of many components in intelligent transportation systems (ITS). The accurate and reliable traffic forecasting can assist in proactive and dynamic traffic control as well as intelligent route guidance, which will help alleviate the huge congestion problem in the system. In this paper, we are interested in simultaneously predicting the traffic of multiple road segments in a road network.

Great efforts have been devoted to improve the traffic forecasting accuracy (Diao et al. 2018). Some studies apply traditional machine learning methods, such as autoregressive and moving average model (ARMA) (Williams and Hoel 2003) and support vector regression (SVR) (Chen et al. 2015) and etc, to forecast future traffic. As most of these methods are linear and not suitable for handling volatile traffic data, the forecasting accuracy is often low. In recent years, the prediction methods based on deep learning have received considerable attention. Some attempts have been made to apply deep Recurrent Neural Networks (RNN)(Wu and Tan 2016) and Convolution Neural Network (CNN) (Zhang et al. 2016; Zhang, Zheng, and Qi 2016; Ma et al. 2017) to predict traffic flow. However, these methods are not suitable to apply to the data points with irregular graph relationship. As graph structure arises naturally in the traffic network, Graph Convolutional Neural Network (GCNN) is the appealing choice (Yu, Yin, and Zhu 2017a). GCNN with deep architectures have proven to be very efficient in short-term traffic forecasting area (Yao et al. 2018). By incorporating the spectral graph theory (Fan 1997), GCNN is efficient in handling signals that live on irregular or non-Euclidean domains. Nevertheless, existing GCNN frameworks have some limitations that make them less efficient in traffic forecasting.

GCNN heavily depend on the Laplacian matrix of a graph, which is defined as the difference between the diagonal matrix of node degrees and the adjacency matrix. Previous GCNN studies rely on the key assumption that the Laplacian matrix is strictly unchanged and available (i,e. the adjacency matrix of the input graph is constant). However, our previous studies demonstrate that there are huge differences between traffic patterns during different time spans. In addition, traffic accidents may occur every day, which will also affect the relationship between road segments in the road network. These factors will lead to the dynamic changes of adjacency matrix thus the Laplacian matrix. Therefore, the exact Laplacian matrix of the graph could be time-variant and generally intractable.

To address the issues above, we propose a novel spatial-temporal structure to more accurately forecast network-wide traffic speed, and we call it dynamic GCNN (DGCNN). Compared with existing GCNN-based methods, our paper makes the following contributions:

- We creatively incorporate tensor decomposition operations into the deep learning framework to extract the global and local components from traffic samples. From frequency analysis, network-wide traffic samples within a time span are composed of two components: the global

[*]Corresponding author

one determined by the road network structure and the local one determined by specific time-of-day or traffic events. We pre-train our tenor decomposition layer with a particular loss function.

- To learn the Laplacian matrix at a specific time-of-day dynamically according to the global and local data components, we design a deep learning-based Laplacian matrix estimator with detailed theoretical derivation and design basis. The Laplacian matrix estimated in real-time will be sent to the graph convolutional layers for forecasting.

The rest of this paper is organized as follows. We first summarize the related work of GCNN in Section 2, and introduce the background knowledge in Section 3. We then present the technical details of our novel DGCNN model in Section 4. After that, we evaluate the performance of our proposed model through experiments on real-world data sets in Section 5, and conclude our work in Section 6.

Related Works

In this section, we summarize the literature work and provide the motivations of our work.

GCNN is extended from convolutional neural networks (CNN) with consideration of graph structures. Existing graph convolutional neural networks can be mainly divided into two categories according to the convolutional operator. One is based on concepts from the vertex domain (Niepert, Ahmed, and Kutzkov 2016; Monti et al. 2017; Hechtlinger, Chakravarti, and Qin 2017; Puy, Kitic, and Pérez 2017); The other is based on concepts from the graph spectral domain. As we cannot express a meaningful translation operator in the vertex domain (Defferrard, Bresson, and Vandergheynst 2016), we prefer defining the convolution operator in the spectral domain in this paper.

Spectral graph theory enables us to extend many of the important mathematical ideas and intuitions from classical Fourier analysis to the graph setting. In recent years, GCNN and its variants have been applied to various areas, such as image classification and task forecasting (Bruna et al. 2013; Yu, Yin, and Zhu 2017b; Shuman et al. 2013; Henaff, Bruna, and LeCun 2015; Defferrard, Bresson, and Vandergheynst 2016; Kipf and Welling 2016; Yu, Yin, and Zhu 2017a; Srivastava, Greff, and Schmidhuber 2015).

Previous studies on traffic forecasting with GCNN (Yu, Yin, and Zhu 2017a; Yao et al. 2018; Puy, Kitic, and Pérez 2017) have a fatal drawback. They seldom consider the change of spatial dependencies with the gradual structural evolution of the road network. The Laplacian matrix represents spatial dependencies between road segments. As shown in Section 3, GCNN heavily depends on the Laplacian matrix L. Keeping the same Laplacian matrix all the time may lead to a significant performance decrease. Yao et al. (Yao et al. 2018) apply the traffic flow data of road segments to model dynamic spatial dependencies between road intersections. However, this method increases the burden of data collection and still doesn't track the dynamic change of the Laplacian matrix.

To satisfy the forecasting requirements under a dynamic network structure, we propose a dynamic spatio-temporal graph convolutional neural network. Our framework will automatically modify the Laplacian matrix according to changes of spatial dependencies hidden in the traffic data.

Preliminary

Before presenting our detailed design of the spatial-temporal framework, we provide some background knowledge in this paper. Graph CNN helps us make traffic forecasting with consideration of graph structures in the road network. Based on the Equation 5 in the area of signal processing, we propose a deep learning method to learn the Laplacian matrix. We incorporate tensor operations into our deep learning framework to extract the long-term and short-term traffic tensor from the real-time tensor.

Graph CNNs

Given a directed graph $G = (\nu, \varepsilon, W)$, where the set ν contains $|\nu| = p$ vertices; ε represents a set of edges, W denotes the weight matrix of G. A signal $x : \nu \to R$ defined on the nodes of the graph may be represented as a vector $x \in R^p$, where x_i is the signal value at the i^{th} node. We have the graph Laplacian $L = D - W$ and the eigen decomposition of $L = U\Lambda U^T$, where D is a diagonal matrix with the i_{th} element of the diagonal line being the degree of the node i: $D_{ii} = \sum_j W_{ij}$ and U is an orthogonal matrix.

The Fourier transform for x is defined as $\hat{x} = U^T x$. Hence, the graph convolution of x and y defined in the Fourier domain is

$$x *_G y = U((U^T x) \bigodot (U^T y)), \tag{1}$$

where $\bigodot$ is the element-wise Hadamard product. It follows that a signal x is filtered by g_θ as

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = U g_\theta(\Lambda) U^T x. \tag{2}$$

To make the filter K-localized in space and reduce its computational complexity, Eq. (2) can be further defined as

$$y = U g_\theta(\Lambda) U^T x = U(\sum_{k=0}^{K-1} \theta_k \Lambda^k) U^T x = \sum_{k=0}^{K-1} \theta_k L^k x. \tag{3}$$

where the parameter $\theta \in R^K$ is a vector of polynomial Fourier coefficients. This filter restricts the hop distance $d_g(i,j)$ between two nodes i and j to be within K hops, that is $L^K_{i,j} = 0$ when $d_g(i,j) \geq K$. Consequently, spectral filters represented by K_{th}-order polynomials of the Laplacian are exactly K-localized.

A signal on graph G of p nodes can be described as a matrix X consisting of c_{in} vectors of size p. Consequently, for the signal $X = [x_1, x_2, \cdots, x_{c_{in}}]$, a 1-D graph convolution operation with a kernel tensor θ of size (c_{in}, c_{out}, K) is

$$y_j = \sum_{i \in [1, c_{in}], k \in [1, K]} \theta_{ijk} L^k x_i, j = 1, 2, ..., c_{out} \tag{4}$$

where c_{in} and c_{out} represent the size of the input feature map and the output feature map respectively.

Graph Signal Processing

In the area of signal processing on graphs, signals are modeled as functions on the vertices of a weighted graph. A major challenge in this field is that of learning the graph structure from data. The learned graph must have a meaningful interpretation and be useful for analysis. Given a set of nodes V and a set of corresponding signals on those nodes, the graph structure is learnt by estimating a matrix L whose nonzero patterns define the edge connectivity of the graph.

Given a p-dimensional random graph signal $\boldsymbol{x}$ and its N observations $\boldsymbol{x_1}, \cdots, \boldsymbol{x_N}$, its sample covariance is $Q = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x_i}\boldsymbol{x_i}^T$. It's often formulated as a matrix optimization problem and focus on an efficient algorithmic solution. According to the maximum likelihood estimate of L, we have (Pavez and Ortega 2016)

$$\min_{L\succeq 0; L_{ij}\leqslant 0, \forall i\neq j;} -\log det(L) + tr(QL) \tag{5}$$

Minimizing $tr(QL)$ is equivalent to promoting the average smoothness. The log det function acts as a barrier on the minimum eigenvalue of L, thus enforcing the positive semi-definite constraint. The sign constraints can be handled using Lagrange multipliers.

Tensor Operations

Tensor operations are widely used in various applications (Xie et al. 2018; Xie et al. 2017), such as traffic prediction and anomaly detection.

Tensor Unfolding: For an N-way tensor $\boldsymbol{\chi} \in \mathbb{R}^{I_1\times\cdots\times I_N}$, its mode-n unfolding $\chi_{[n]} \in \mathbb{R}^{I_n\times(I_{n+1}I_{n+2}\cdots I_N I_1 I_2\cdots I_{n-1})}$ contains the tensor element $a_{i_1,i_2,\cdots,i_N}$ at the position in the unfolding matrix with its row index i_n and column index equal to $\sum_{k=1,k\neq n}^{N} i_k \times \prod_{m=k+1,m\neq n}^{N} I_m$.

n-mode product: The n-mode product of an N-way tensor $\boldsymbol{\chi} \in \mathbb{R}^{I_1\times\cdots I_n\cdots\times I_N}$ with matrix $U \in \mathbb{R}^{J\times I_n}$ produces a tensor with size $(I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N)$ and defined as

$$(\boldsymbol{\chi} \times_n U)_{i_1\cdots i_{n-1} j i_{n+1}\cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2\cdots i_N} U_{j i_n} \tag{6}$$

where $\times_n$ is the n-mode product operation.

Tucker decomposition: Given a tensor $\boldsymbol{\chi} \in \mathbb{R}^{I_1\times\cdots\times I_N}$, we can decompose it into a low rank core $\boldsymbol{G} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_N}$ by projecting along each of its modes with projection factors $(U_1, \cdots, U_N)$, with $U_k \in \mathbb{R}^{r_k\times I_k}, k \in (1, \cdots, N)$. We can write as:

$$\boldsymbol{\chi} = \boldsymbol{G} \times_1 U_1 \times_2 U_2 \times \cdots \times_N U_N \tag{7}$$

where r_k represents the selected rank in mode-k of the tensor $\boldsymbol{\chi}$.

Problem and Model

In this section, we introduce the problem of network-wide traffic forecasting, and the basic framework of our proposed system.

We are interested in simultaneously forecasting the traffic speed of multiple road segments. We define a road network as a directed graph $G = (\boldsymbol{\nu}, \varepsilon, W)$, where $\boldsymbol{\nu}$ is a set of vertices each corresponding to a road segment, ε represents a set of edges with each edge connected between two road segments in the network, W denotes the adjacency matrix of G. The average traffic speed of the road segment i in the time interval t is represented by $v_i(t)$.

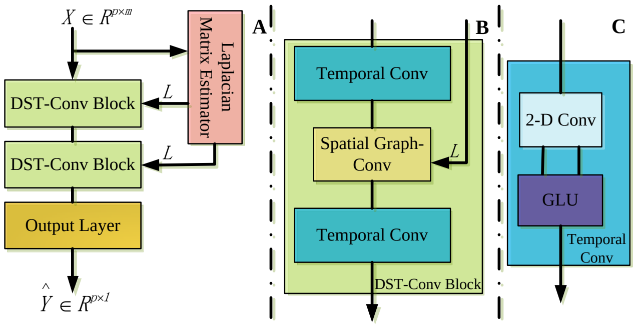

Figure 1: The framework of DGCNN

Given historical speed observations of p road segments $\{v_i(\cdot)\}_{i=1}^{p}$ at time $(t-m+1, \cdots, t-1, t)$, our task is to predict the value of $\{v_i(t+\Delta))\}_{i=1}^{p}$, where Δ denotes the prediction horizon. If we put the historical data together, we can get a data matrix X.

$$X = \begin{bmatrix} v_1(t-m+1) & v_1(t-m+2) & \cdots & v_1(t) \\ v_2(t-m+1) & v_2(t-m+2) & \cdots & v_2(t) \\ \cdots & \cdots & \cdots & \\ v_p(t-m+1) & v_p(t-m+2) & \cdots & v_p(t) \end{bmatrix} \tag{8}$$

As shown in Figure 1 A, our deep learning framework is composed of three modules: a Laplacian matrix estimator, two spatial-temporal convolutional blocks and an output layer. Each spatial-temporal convolutional block is formed as a “sandwich” structure as in Figure 1 B with two gated temporal convolutional layers and a spatial graph convolutional layer in between. So we can take full advantage of spatial dependencies and temporal dependencies hidden in traffic data to assist our forecasting task. The downscaling of channels in the spatial graph convolution layer can also help reduce the number of parameters and the time consumption in training. The output layer contains a temporal convolutional layer and a fully connected layer which transforms the size of the temporal dimension to 1 and generate the final output ($R^{p\times 1}$). As the focus of this paper, the Laplacian matrix estimator will extract a real-time Laplacian matrix L according to the input data and pass it to the latter graph convolutional neural network in spatial-temporal convolutional blocks.

Compared with RNN-based models with an iterative operation over each time slot, CNN-based models process a block of data together and have the superiority of fast training and simple structures. Therefore we employ entire convolutional structures on time axis for a window of data to capture dynamic temporal behavior of traffic flows. Next we will give a detailed description of the Laplacian matrix estimator and the spatial-temporal convolutional block.

Dynamic Spatial-Temporal Graph Convolutional Neural Network

The Laplacian matrix is crucial for determining the receptive field of the graph convolutional operation. An inaccurate Laplacian matrix will reduce the forecasting accuracy. For a graph with N nodes, the Laplacian matrix has $N \times N$ entries. Obtaining accurate Laplacian matrix in real-time faces two major challenges. First, the number of parameters to learn for the complete matrix will increase quickly with the network size. Second, the traditional convergent gradient descent algorithm is time consuming and cannot update the Laplacian matrix quickly upon the arrival of new traffic data to meet the need of online traffic forecasting.

We first present our design of a dynamic Laplacian matrix estimator, and then propose a novel spatial-temporal convolutional neural network for efficient traffic prediction.

Dynamic Laplacian Matrix Estimator

Given a global Laplacian matrix that represents the static graph structure of a traffic network, the real-time Laplacian matrix will fluctuate up and down around it. The perturbation is the result of minor changes of the graph structure, which is mainly caused by short-term traffic pattern and accidents. Thus we only need to determine the variation of Laplacian matrix based on the short-term traffic. As a result of spatial and temporal correlation, long-term traffic data form a low rank tensor. Because the traffic dynamics and abnormality are rare in the temporal dimension, short-term data form a sparse tensor in the temporal dimension.

We thus incorporate the tensor operation into the neural network and decompose the traffic tensor into two components, a low-rank tensor that maintains the long-term and highly correlated traffic information, and a sparse tensor that tracks the dynamic changes of traffic. Accordingly, our Laplacian matrix estimator includes two parts, a Tensor Decomposition Layer (TDL) and a unit for dynamic Laplacian matrix learning.

Tensor Decomposition Layer In our Laplacian matrix estimator, we apply TDL to extract the global and local component of the traffic data with two operations, contraction and recovery. On the right part of Figure 2, we show a 3-order traffic tensor $\boldsymbol{\chi}$ with (p, m, c) corresponding to the number of nodes in the spatial domain, number of time slots to consider in the temporal domain and the number of channels thus states to monitor. We set c to 1 in this paper to focus on the monitoring of traffic speed. (r_1, r_2) corresponds to the rank of the spatial and temporal modes, and will be generally lower than p and m due to the traffic correlation in spatial and temporal domains.

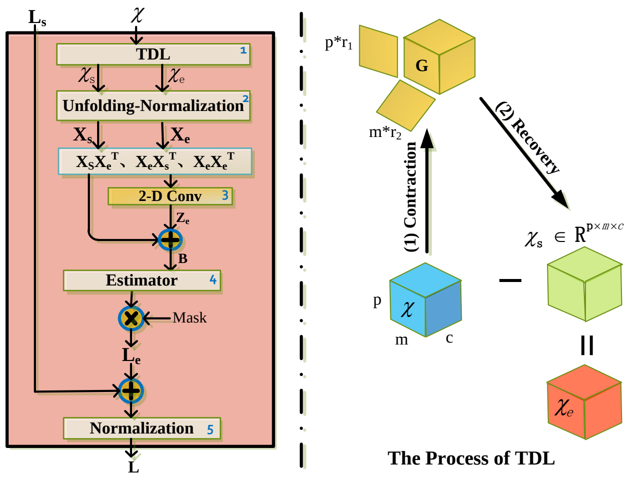

Figure 2: Laplacian Matrix Estimator

In the contraction operation, TDL runs two n-mode products along the spatial and temporal modes with the projection factors $U_1 \in \mathbb{R}^{p\times r_1}$, $U_2 \in \mathbb{R}^{m\times r_2}$ to project $\boldsymbol{\chi}$ into the low-dimensional space:

$$\boldsymbol{\chi} = \boldsymbol{G} \times_1 U_1 \times_2 U_2. \tag{9}$$

Given $\boldsymbol{\chi}$, U_1 and U_2, we can easily calculate the low-dimensional tensor $\boldsymbol{G}$ according to the Equation 9. In the recovery operation, TDL estimates the low-rank tensor $\boldsymbol{\chi_s}$ through the expansion from the low dimensional space:

$$\boldsymbol{\chi_s} = \boldsymbol{G} \times_1 U_1 \times_2 U_2 \times_2 U_2^T \times_1 U_1^T \tag{10}$$

The projection factors U_1 and U_2 are set as the parameters to learn through gradient back-propagation in our deep learning framework. Once the training process is completed, the two projection factors can be applied to estimate the long-term traffic $\boldsymbol{\chi_s}$ and used in each time slot, so we can obtain the short-term dynamic traffic tensor $\boldsymbol{\chi_e}$ as follows:

$$\boldsymbol{\chi_e} = \boldsymbol{\chi} - \boldsymbol{\chi_s}. \tag{11}$$

Learning of Dynamic Laplacian Matrix The approximated low-rank tensor $\boldsymbol{\chi_s}$ is impacted by the long-term and global dependencies in spatial and temporal dimensions. The sparse tensor $\boldsymbol{\chi_e}$ represents the local fluctuation within a specific time span of a day, which is affected by short-term spatial and temporal dependencies as well as external factors such as traffic accidents and weather.

We define the dynamic Laplacian matrix at any time as $L = L_s + L_e$, where L_s and L_e represent the global Laplacian matrix and the local Laplacian matrix respectively. L_e acts as a short-term perturbation on L. We compute the global Laplacian matrix based on the distances among sensors employed on each road segment.

To find the detailed Laplacian matrix L_e, we make the estimation with the low-rank tensor $\boldsymbol{\chi_s}$ and sparse tensor $\boldsymbol{\chi_e}$. To find the Laplacian matrix with the Eq. (5), the Lagrangian is $-\log det(L) + tr(QL) + tr(ZL)$, where Z represents the Lagrange multiplier matrix with the values of diagonal elements being zero. According to the Karush-Kuhn-Tucker (KKT) conditions we have:

$$\begin{aligned} L &= (Q+Z)^{-1} \\ \text{subject to} \quad Z &= Z^T, Z_{ij} \geqslant 0 \end{aligned} \tag{12}$$

The matrix Z acts as a perturbation on the sample covariance matrix Q such that the learned Laplacian matrix Z can represent the spatial dependencies of the road network. Through unfolding and zero-mean normalization, we can transfer the low-rank tensor $\boldsymbol{\chi_s}$ and sparse tensor $\boldsymbol{\chi_e}$ to two matrices $X_s \in \mathbb{R}^{p\times(r_2 * r_3)}$ and $X_e \in \mathbb{R}^{p\times(r_2 * r_3)}$. X_s can be considered as a low-rank matrix that corresponds to the long-term traffic data. Replacing the sample covariance matrix with $X_s {X_s}^T$, based on Eq. 12, we have:

$$L_s = (X_s {X_s}^T + Z_s)^{-1} \tag{13}$$

Similarly, inserting the real-time Laplacian matrix $L = L_s + L_e$ and its corresponding traffic data $X = X_s + X_e$ into the Eq. (12), then:

$$\begin{aligned} L_s + L_e &= [(X_s + X_e)(X_s + X_e)^T + Z_s + Z_e]^{-1} \\ &= [(X_s + X_e)({X_s}^T + {X_e}^T) + Z_s + Z_e]^{-1} \\ &= [(X_s{X_s}^T + Z_s) + (X_s{X_e}^T + X_e{X_s}^T + X_e{X_e}^T + Z_e)]^{-1} \\ &= [{L_s}^{-1} + (X_s{X_e}^T + X_e{X_s}^T + X_e{X_e}^T + Z_e)]^{-1} \end{aligned} \tag{14}$$

where L_s represents the global Laplacian matrix and Z_s represents the corresponding Laplacian multipliers.

Eq. (13) and Eq. (14) give the global Laplacian matrix L_s and the dynamic Laplacian matrix L respectively. Although L_s is given in advance, the complexity of finding the matrix inversion is $O(n^3)$. To simplify the calculation, we would like to get rid of the inverse operations in Eq. (14), and then learn L_e based on deep learning methods.

Based on (Henderson and Searle 1981), for two matrices A and B with A and $A+B$ invertible, we have

$$(A+B)^{-1} = A^{-1} - A^{-1}B(E + A^{-1}B)^{-1}A^{-1} \tag{15}$$

E represent an identity matrix. Let B represent $(X_s{X_e}^T + X_e{X_s}^T + X_e{X_e}^T + Z_e)$, based on Eq. (15), the Eq. (14) can be rewritten as:

$$\begin{aligned} L_s + L_e &= L_s - L_sB(E + L_sB)^{-1}L_s \\ L_e &= -L_sB(E + L_sB)^{-1}L_s \end{aligned} \tag{16}$$

Iteratively applying the Eq. (15) to expand $(E + L_sB)^{-1}$, Eq. (16) can be further expanded as:

$$L_e = \sum_{i=1}^{\propto} (-1)^i L_s (BL_s)^i \tag{17}$$

We can estimate L_e with a finite number of summation operations:

$$L_e = \sum_{i=1}^{I} (-1)^i L_s (BL_s)^i + o(BL_s) \tag{18}$$

where $o(BL_s)$ is the error from Eq. 17.

The equation used to estimate L_e doesn't include the inverse operations, which greatly reduces the time consumption of our deep learning framework. As $X_s{X_e}^T, X_e{X_s}^T, X_e{X_e}^T$ can be computed directly with matrices X_s, X_e, we only need to learn Z_e in B. $Z_e \in \mathbb{R}^{p\times p}$, where p equals to the number of road segments on the road network. To avoid involving too many parameters in training and ensure the scalability of our model with the expansion of the road network, we employ two 2-D convolutional layers for the data fitting.

For the overall dynamic Laplacian matrix estimator shown on the left part of Figure 2, the input data will go through five sub-processes as follows:

1. **Tensor Decomposition**. The Tensor Decomposition Layer (TDL) incorporates the tensor operations into the neural network and splits the traffic tensor $\boldsymbol{\chi}$ into a low rank tensor $\boldsymbol{\chi_s}$ and a sparse tensor $\boldsymbol{\chi_e}$.
2. **Unfolding-Normalization**. After an unfolding operation along the spatial dimension and an zero-mean normalization, the two tensors generated by the first sub-process are further translated to two zero-mean matrices $X_s, X_e \in \mathbb{R}^{p\times(m*c)}$, based on which we can easily find the values of $X_s{X_e}^T$,$X_e{X_s}^T$,$X_e{X_e}^T$.
3. **2-D Conv**. We stack two 2-D convolutional layers to fit the Lagrange multipliers Z_e. The input data of the first convolutional layer is a 3-D tensor $\mathbb{R}^{p\times p\times 3}$, which is formed by stacking $X_s{X_e}^T$,$X_e{X_s}^T$,$X_e{X_e}^T$ together. The size of the output feature maps on the two convolutional layers are set to 3 and 1 respectively.
4. **Estimator**. We find matrices B and L_e according to the Eq. (18).
5. **Normalization**. We find $L = L_s + L_e$ and then normalize L through $D^{-1/2}LD^{-1/2}$, where D is a diagonal matrix with the element on a row equal to the absolute value of the summation of L's off-diagonal elements on that row.

Compared with most graph learning methods in the literature, our deep learning framework has higher execution efficiency. Without the inverse operation, our estimator won't increase the burden of the deep learning framework when performing traffic forecasting.

Pre-training the TDL Before training the whole deep learning framework, we pre-train TDL to initialize its parameters. The corresponding loss function is

$$L(U_1, U_2) = \sum^{\xi} tr(X_sL_sX_s^T) + \beta \|X_e\|_F. \tag{19}$$

where U_1, U_2 are trainable parameters in TDL, L_s is the global Laplacian matrix given in advance. ξ represents the total number of data samples for training. tr is a trace operation. X_s is the matrix extracted from the traffic component χ_s by TDL, and β is a weight coefficient which controls the proportion of two items in Eq. (19). Minimizing $tr(X_sX_s^TL_s)$ is equivalent to promoting the average smoothness with respect to L_s. This allows us to associate

L_s with X_s and guarantees that X_s extracted by TDL can meet the Eq. (13).

Spatial-Temporal Convolutional Block

In our learning framework, we apply a spatio-temporal convolutional block to jointly process graph-structured time series and fuse features from both temporal and spatial domains.

Gated CNNs for Extracting Temporal Features We set a 2-D temporal convolutional layer to capture short-term temporal features of traffic flows. Given the input of temporal convolutional layer $\chi \in \mathbb{R}^{p\times m\times c_{in}}$, where p, m, c_{in} represent the size of the spatial, temporal and channel dimensions respectively, the convolutional kernel $\Gamma \in \mathbb{R}^{1\times K\times c_{in}\times 2c_{out}}$ will map the input to an output element $[Y_1, Y_2] \in \mathbb{R}^{p\times(m-K+1)\times(2c_{out})}$ (Y_1, Y_2 are split into half with the same size of channels). As shown in Figure 1 C, the temporal gated convolution can be defined as:

$$\Gamma *_{\mathcal{T}} \chi = Y_1 \bigodot \sigma(Y_2) \tag{20}$$

where $Y_1, Y_2 \in \mathbb{R}^{p\times(m-K+1)\times c_{out}}$ are the input of gates in gated linear units (GLU) separately, and $\bigodot$ denotes the element-wise Hadamard product. The sigmoid gate $\sigma(Y_2)$ controls which inputs Y_1 of the current status are relevant for discovering compositional structure and dynamic variances in time series.

Graph CNNs for Extracting Spatial Features Given the input of graph convolutional layer as $\chi \in \mathbb{R}^{p\times m\times c_{in}}$, where p, m, c_i represent the size of the spatial, temporal and channel dimensions respectively. We split the input data into m parts along the temporal dimension, with each being a matrix $\mathbb{R}^{p\times c_{in}}$, and then capture spatial features of each part by a 1-D graph convolutional layer. In order to speed up the 1-D graph convolutional operation, we employ the Chebyshev polynomial $T_k(x)$ to approximate kernels (Yu, Yin, and Zhu 2017a). The graph convolution in Eq. (4) can then be rewritten as:

$$y_j = \sum_{i\in[1,c_{in}],k\in[1,K_s]} \theta_{ijk} T_k(\tilde{L}) x_i, j = 1, 2, ..., c_{out} \tag{21}$$

where $y_j \in \mathbb{R}^{p\times c_{out}}$. $T_k(\tilde{L})$ is the Chebyshev polynomial of order k evaluated at the scaled Laplacian $\tilde{L} = 2L/\lambda_{max} - I_n$.

Experiments

Experimental Settings

Data. We evaluate the performance of our proposed model using two real-world large-scale datasets collected from the monitoring of traffic in New York City (NYC) and in California, respectively:

- NYC: This traffic dataset contains traffic information collected from traffic speed detectors deployed on road segments of Manhattan district in New York city. We select 50 sensors and collect 2 months of data ranging from December $1st$2017 to January $30th$2018.

Table 1: Forecasting error given by MAE and RMSE on NYC dataset (15/30/45 min)

	MAE	RMSE
VAR	4.14/ 4.84/ 5.29	6.01/ 6.63/ 7.45
FNN	3.60/ 3.79/ 4.16	5.80/ 5.87/ 6.20
GCGRU	3.20/ 3.36/ 3.49	5.23/ 5.31/ 5.50
STGCN	3.18/ 3.31/ 3.44	5.18/ 5.30/ 5.42
DGCNN	3.06/ 3.14/ 3.29	5.02/ 5.22/ 5.30

- PeMS: This traffic dataset contains real-time speed data from freeways in California. We select $50/142/228$ detectors and collect 6 months of data range from April $1st$2017 to September $30th$2017.

The traffic data are aggregated and output by each detector every 5 minutes. All studies use one hour as the historical time window to forecast the traffic condition in the next 15/30/45 minutes according to 12 observed data points in the window.

Evaluation Metric and Baselines. We adopt Mean Average Error (MAE) and Rooted Mean Square Error (RMSE) to evaluate the performance of different methods. We also implement four baseline schemes for the performance reference: 1. Vector Autoregression (VAR); 2) Feed-Forward Neural Network (FNN); 3) Graph Convolutional GRU (GCGRU, published in ICLR-2018) (Li et al. 2017); and 4) Spatio-Temporal Graph Convolutional Networks (STGCN, published in IJCAI-2018) (Yu, Yin, and Zhu 2017a).

All these deep learning models are trained for 50 epochs with batch size as 50. The initial learning rate is 10^{-3} with a decay rate of 0.7 after every 5 epochs. Both the graph convolution kernel size and temporal convolution kernel size are set to 3. The size of the output feature map in the spatial-temporal convolution block of DGCNN are 64, 16, 64 respectively.

Performance Comparison

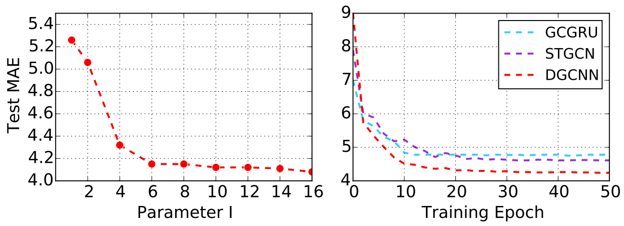

Figure 3: Test MAE versus the parameter I in DGCNN (left); Test MAE versus the number of training epochs (right). (PeMS-50)

Parameter Selection and Training Efficiency In order to select a suitable parameter I for the Eq. (18) in our Laplacian matrix estimator, we test the impact of I on the left of the Figure 3. As expected, the test MAE decreases with the increase of I for the better estimation of the changes of

Laplace matrix. When the parameter I is greater than 6, the rate of decline tends to be slow. Therefore, we set the parameter I to 6 in this paper.

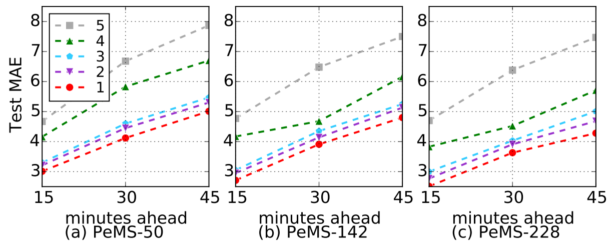

Figure 4: Forecasting accuracy: 1) DGCNN, 2) STGCN, 3) GCGRU, 4) FNN, 5) VAR.

Forecasting Accuracy We run each model 10 times and report the average results. From Table 1 and Figure 4, our proposed DGCNN outperforms all competing baselines by achieving the lowest RMSE and MAE on both datasets for the traffic forecasting.

In brief, the traditional linear prediction method VAR performs the worst due to its incapability of handling volatile traffic data. Compared with other three deep learning models, our DGCNN also perform better with on average $8\% - 10\%$ accuracy improvement. Traffic patterns and spatial dependencies on road network are dynamic. Overlooking the dynamic changes of spatial dependencies on the road network, the forecasting error of reference schemes are higher.

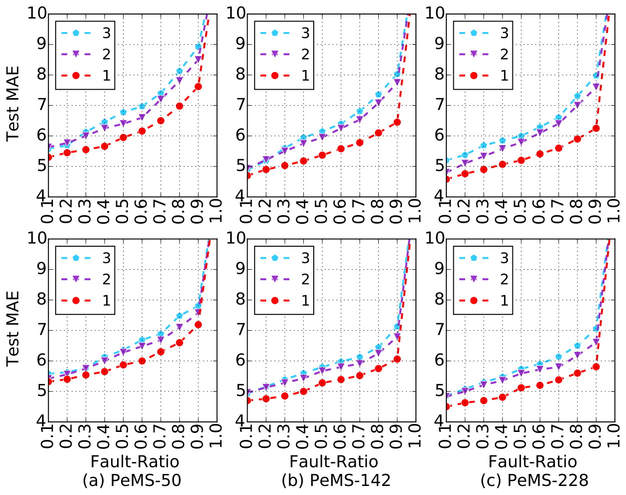

Figure 5: Fault-tolerance comparison with state-of-the-art models based on graph CNNs: 1) DGCNN, 2) STGCN (IJCAI-2018), 3) GCGRU (ICLR-2018).

Fault Tolerance Comparison The real-time traffic samples may be partial abnormal as a result of sensor malfunction or traffic accidents on some road segments. To examine the fault-tolerance ability in extreme environments, we randomly select a fraction (10% to 100%) of road segments to sabotage their 12 historical observations. We carry out two groups of studies and forecast the traffic condition in the next 45 minutes: On the top row of Figure 5, the damaged observations from nodes selected are replaced with zero-mean white Gaussian noise (variances 1.0). On the bottom row, zeros are used to replace the "damaged" observations.

Fusing the spatial-temporal information with dynamic Laplacian matrix, our model is shown to be more fault tolerant with on average $10\% - 25\%$ accuracy improvement compared with two state-of-the-art models based on graph CNNs. Even when the fault ratio reaches 0.9, DGCNN still has a strong forecast capability. With the same amount of noise contamination, other models' performance drops dramatically without exception. Comparing the results over three PeMS datasets, the performance gain of our model will become larger with the increase of road network scale. Our DGCNN model can detect the changes of spatial dependencies hidden in "contaminated" traffic samples and adjust the receptive field of graph convolution operations. The right of Figure 3 shows the learning curves of three models with roughly the same number of parameters. With the increase of training epochs, DGCNN achieves the lowest validation error compared with GCNN and STGCN, which shows its training effectiveness. The intuition is that the dynamic Laplacian matrix estimator gives the model the ability and flexibility to capture the influence from various factors in the road network.

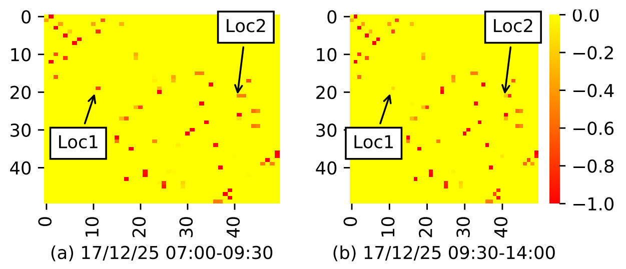

Figure 6: Spatial dependencies learning on two consecutive time spans.

Spatial Dependencies Learning To identify the capability of our novel model in learning the spatial dependencies of a traffic network, we carry out many simulations over two datasets. We first measure the impact of spatial order of the input data on the learning of spatial dependencies, as done in (Cui, Ke, and Wang 2018). We randomly rearrange the spatial dimension of input data and retrain our model. The learned Laplacian matrix and the forecasting accuracy of our model do not have obvious changes, which demonstrates the efficiency of our model. In addition, we measure the influence of different time spans of a day on the learning of spatial dependencies. As shown in Figure 6, the heatmaps of the learned Laplacian matrix on any two consecutive time spans are very close to each other, which means that our novel model is able to learn the long-term structure of the

Laplacian matrix. The two locations marked with "LOC1" and "LOC2" also demonstrate that our model can learn the local changes of the Laplacian matrix.

Conclusion and Future Work

In this paper, we propose a novel dynamic graph convolution neural network (DGCNN) for traffic forecasting. To the best of our knowledge, this is the first graph convolution neural network that can follow the evolution of spatial dependencies. The experiment results demonstrate that our proposed neural network can achieve on average $10\% - 25\%$ higher accuracy compared to other models. The proposed dynamic Laplacian matrix estimator plays an important role in the forecasting process. In the future work, we also plan to combine our DGCNN with other deep learning methods to learn the structured features hidden in the input data.

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant No.61472130 and No.61572184, a Planned Science and Technology Project of Hunan Province in 2017, the NSF grant CNS-1526843 and ECCS-1731238, a Hunan Provincial Natural Science Foundation of China under Grant No.2017JJ1010.

References

Bruna, J.; Zaremba, W.; Szlam, A.; and Lecun, Y. 2013. Spectral networks and locally connected networks on graphs. *Computer Science*.

Chen, R.; Liang, C. Y.; Hong, W. C.; and Gu, D. X. 2015. Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm. *Applied Soft Computing* 26(C):435–443.

Cui, Z.; Ke, R.; and Wang, Y. 2018. Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering.

Diao, Z.; Zhang, D.; Wang, X.; Xie, K.; He, S.; Lu, X.; and Li, Y. 2018. A hybrid model for short-term traffic volume prediction in massive transportation systems. *IEEE Transactions on Intelligent Transportation Systems* (99):1–12.

Fan, R. K. C. 1997. *Spectral graph theory*. Published for the Conference Board of the mathematical sciences by the American Mathematical Society,.

Hechtlinger, Y.; Chakravarti, P.; and Qin, J. 2017. A generalization of convolutional neural networks to graph-structured data.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *CoRR* abs/1506.05163.

Henderson, H. V., and Searle, S. R. 1981. On deriving the inverse of a sum of matrices. *Siam Review* 23(1):53–60.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting.

Ma, X.; Dai, Z.; He, Z.; Ma, J.; Wang, Y.; and Wang, Y. 2017. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 17(4).

Monti, F.; Boscaini, D.; Masci, J.; Rodola, E.; Svoboda, J.; and Bronstein, M. M. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. 5425–5434.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. 2014–2023.

Pavez, E., and Ortega, A. 2016. Generalized laplacian precision matrix estimation for graph signal processing. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Puy, G.; Kitic, S.; and Pérez, P. 2017. Unifying local and non-local signal processing with graph cnns.

Shuman, D. I.; Narang, S. K.; Frossard, P.; Ortega, A.; and Vandergheynst, P. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine* 30(3):83–98.

Srivastava, R. K.; Greff, K.; and Schmidhuber, J. 2015. Highway networks. *Computer Science*.

Williams, B. M., and Hoel, L. A. 2003. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of Transportation Engineering* 129(6):664–672.

Wu, Y., and Tan, H. 2016. Short-term traffic flow forecasting with spatial-temporal correlation in a hybrid deep learning framework.

Xie, K.; Li, X.; Wang, X.; Xie, G.; Wen, J.; Cao, J.; and Zhang, D. 2017. Fast tensor factorization for accurate internet anomaly detection. *IEEE/ACM transactions on networking* 25(6):3794–3807.

Xie, K.; Li, X.; Wang, X.; Xie, G.; Wen, J.; and Zhang, D. 2018. Graph based tensor recovery for accurate internet anomaly detection. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 1502–1510. IEEE.

Yao, H.; Tang, X.; Wei, H.; Zheng, G.; Yu, Y.; and Li, Z. 2018. Modeling spatial-temporal dynamics for traffic prediction.

Yu, B.; Yin, H.; and Zhu, Z. 2017a. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting.

Yu, B.; Yin, H.; and Zhu, Z. 2017b. Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting. *CoRR* abs/1709.04875.

Zhang, J.; Zheng, Y.; Qi, D.; Li, R.; and Yi, X. 2016. Dnn-based prediction model for spatio-temporal data. In *ACM Sigspatial International Conference on Advances in Geographic Information Systems*, 92.

Zhang, J.; Zheng, Y.; and Qi, D. 2016. Deep spatio-temporal residual networks for citywide crowd flows prediction.