Spatio-Temporal Graph Structure Learning for Traffic Forecasting

Qi Zhang,[1,2] **Jianlong Chang,**[1,2] **Gaofeng Meng,**[1] **Shiming Xiang,**[1,2] **Chunhong Pan**[1]
[1]National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{qi.zhang2015, jianlong.chang, gfmeng, smxiang, chpan}@nlpr.ia.ac.cn

Abstract

As an indispensable part in Intelligent Traffic System (ITS), the task of traffic forecasting inherently subjects to the following three challenging aspects. First, traffic data are physically associated with road networks, and thus should be formatted as traffic graphs rather than regular grid-like tensors. Second, traffic data render strong spatial dependence, which implies that the nodes in the traffic graphs usually have complex and dynamic relationships between each other. Third, traffic data demonstrate strong temporal dependence, which is crucial for traffic time series modeling. To address these issues, we propose a novel framework named Structure Learning Convolution (SLC) that enables to extend the traditional convolutional neural network (CNN) to graph domains and learn the graph structure for traffic forecasting. Technically, SLC explicitly models the structure information into the convolutional operation. Under this framework, various non-Euclidean CNN methods can be considered as particular instances of our formulation, yielding a flexible mechanism for learning on the graph. Along this technical line, two SLC modules are proposed to capture the global and local structures respectively and they are integrated to construct an end-to-end network for traffic forecasting. Additionally, in this process, Pseudo three Dimensional convolution (P3D) networks are combined with SLC to capture the temporal dependencies in traffic data. Extensively comparative experiments on six real-world datasets demonstrate our proposed approach significantly outperforms the state-of-the-art ones.

Introduction

Traffic forecasting is regarded as one of the key components of Intelligent Traffic System (ITS). Accurate traffic forecasting is the foundation of many traffic applications, including route planning, taxi order allocation, travel time estimation, and so on. Due to the significance, it has received great attention both in academic and industrial community.

City traffic data are inherently defined on the graph-domains, due to the complicated topological structure of traffic networks. Different from the grid-like data (*e.g.* image in Fig. 1 (a)), the numbers of neighbors in traffic networks are variable at different locations (*e.g.* Fig. 1 (b))

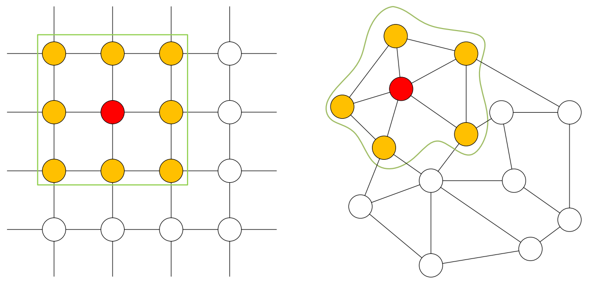

Figure 1: (a) indicates the classical CNN on grid-like data (*e.g.* image) and (b) is the GCNN on traffic data. Red node denotes the central node, and yellow nodes are the neighbors of red one. In the traffic data, each node may stand for a road or junction. It is obvious that the nodes in the traffic data have various numbers of neighbors.

Although some influential works have been developed for traffic forecasting (Ahmed and Cook 1979; Lv et al. 2015; Van Der Voort, Dougherty, and Watson 1996), most of these traditional methods either process each node individually, or simply concatenate the signal of different place to a vector, ignoring the underlying spatial traffic topological information.

More recently, a new type of machine learning method, Graph Convolutional Neural Network (GCNN), is developed to deal with the signals embedded in graph domains (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015; Kipf and Welling 2016; Atwood and Towsley 2016; Niepert, Ahmed, and Kutzkov 2016; Hechtlinger, Chakravarti, and Qin 2017; Chang et al. 2018; 2019). GCNN generalizes the convolution operator from grid-like data to graph-structured data and has achieved superior performance in many fields, including molecular feature extraction (Duvenaud et al. 2015), text categorization (Hechtlinger, Chakravarti, and Qin 2017), point cloud categorization (Simonovsky and Komodakis 2017), and so on. Due to the ability of GCNN to handle graph-structured data, it motivates us to develop a new GCNN method for the traffic forecasting task.

Despite of the great successes of GCNN in many applications, it is a challenging task to applying GCNN on the traffic forecasting task. The main reason is that current GCNN methods pay little attention to exploiting the latent graph structure. They either utilize some predefined methods or depend on the prior knowledge to obtain the graph structures. The obtained structures can not be guaranteed to be accurate for the current learning tasks. Besides, the current GCNN methods usually adopt local graph structures, which ignore the long distant relationship. Moreover, the predefined graph structures are generally fixed. Once defined, the graph structures will not be changed any more. However, the graph structures of traffic data change over time. For example, two specific nodes generally have different relationships on weekdays and weekends, or on morning peak and evening peak. Using a fixed structure is hard to model the temporal dynamic. New approaches are needed to learn the underlying graph structures.

The above concerns motivate us to propose Structure Learning Convolution (SLC), a generic graph convolutional formulation which explicitly models the structure information into the convolutional operation. In our study, we construct two data-driven and time-vary SLC modules, capturing the global and local structures, respectively. These two modules formulate the graph structure as learnable parameters or the output of a learnable function, thus they are able to capture the static and dynamic graph information. In addition, we integrate pseudo three dimensional (P3D) convolution with the SLC to capture the important temporal dependencies in traffic data. In practice, Structure Learning Convolution Neural Network (SLCNN) is finally constituted by switching the traditional convolution in CNNs to SLC. Experiments on real-world traffic datasets demonstrate SLCNN outperforms state-of-the-art traffic forecasting methods by a large margin. The main contributions of our work can be summarized as follows:

- We propose a generic graph convolutional formulation, which defines convolution operation as a combination of structure module and kernel module. Many previously proposed GCNN methods can be regarded as particular instances of our formulation.
- Two data-driven and time-vary SLC modules are proposed to capture the global and local structures, respectively. Each of two proposed modules contains a static structure learning term designed to learn the shared structure of all samples, and a dynamic structure learning term designed to learn the unique structure of each sample.
- P3D ConvNet is incorporated in SLCNN to capture the temporal dependencies in traffic data. To the best of our knowledge, this work is the first to exploit P3D ConvNet in graph-structured traffic data.
- The proposed method is evaluated on six real-world traffic datasets. Extensively comparative experiments demonstrate our proposed approach significantly outperforms the state-of-the-art ones.

Related work

Graph Convolutional Neural Network

Bruna *et al.* first generalized traditional CNN from Euclidean domain to non-Euclidean domain (Bruna et al. 2013), where a spatial method and a spectrum method were proposed. Since then, the graph convolutional neural networks can be divided into two categories, *i.e.* spectral approaches and spatial approaches (Bronstein et al. 2017).

Spatial approaches devote to aggregating neighborhood signals on spatial domains directly. Duvenaud *et al.* proposed a model where all nodes in a neighborhood shared the same kernel weights (Duvenaud et al. 2015). Graph labeling procedure was utilized to rank and select neighborhood nodes, then the classical CNN was used to execute the convolution operation (Niepert, Ahmed, and Kutzkov 2016). Different with (Niepert, Ahmed, and Kutzkov 2016), Hechtlinger *et al.* adopted the random walk to construct neighborhoods (Hechtlinger, Chakravarti, and Qin 2017).

Spectral approaches implement the convolution operation by means of the spectral graph theory. By introducing C order Chebyshev polynomials parametrization, Defferrard *et al.* reduced the computational complexity and achieved strictly localized filters (Defferrard, Bresson, and Vandergheynst 2016) on ChebNet. The computational complexity of ChebNet was further simplified by limiting $C = 2$ (Kipf and Welling 2016).

Traffic Forecasting

As a key component of ITS, traffic forecasting has been studied for decades. As early as 1979, AutoRegressive Integrated Moving Average (ARIMA) was adopted to forecast freeway traffic data (Ahmed and Cook 1979). Subsequently, many variant of ARIMA and nonparametric methods were applied on this task, including KARIMA (Van Der Voort, Dougherty, and Watson 1996), ARIMAX (Williams 2001), k-NN (Davis and Nihan 1991), Bayesian network (Sun, Zhang, and Yu 2006), and so on.

Many scholars pay much attention to deep learning models on account of the extraordinary abilities to solve nonlinear problems. Deep belief network was applied on traffic speed and flow forecasting tasks (Jia, Wu, and Du 2016; Koesdwiady, Soua, and Karray 2016). Lv *et al.* leveraged a stacked autoencoder to learn latent traffic flow features (Lv et al. 2015). Deep Recurrent Neural Networks were applied to traffic forecasting (Yu et al. 2017; Laptev et al. 2017). Zhang *et al.* constructed the traffic network as a regular 2D grid and utilized traditional CNN to forecast the traffic data (Zhang, Zheng, and Qi 2017).

Recently, some GCNN models have been adopted in traffic forecasting tasks. For example, Li *et al.* proposed DCRNN which combined diffusion process and gated recurrent unit (Li et al. 2018). Yu *et al.* proposed STGCN which employed a generalization of Chebnet to capture the spatial correlations of traffic data (Yu, Yin, and Zhu 2018). However, these methods usually do not notice to exploit the graph structures, and they generally only utilize the predefined, local and fixed graph structures.

Structure Learning Convolution

Formulation of SLC

Inherently, convolution can be regarded as an aggregation operation on input signals. However, in graph-structured data, the aggregation operation involves not only the values of the signals but also the graph structures. Therefore, the graph convolutional method should enable to handle the inputs with different topological structures. To this end, we propose SLC that explicitly models the structure information into the convolutional operation.

Given a input signal $\mathbf{x} \in \mathbb{R}^N$ embedded on a graph $\mathcal{G}$ with N nodes, the convolutional operator of SLC can be formulated as:

$$y_i = f\Bigg(\sum_{e_{ij}\in\mathcal{E}} S_{ij} w_j x_j\Bigg), \tag{1}$$

where $f(\cdot)$ is the activation function, y_i is the output signal on the i-th node, x_j indicates the input data embedded on the j-th node. w_j is the j-th element in $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{w}$ denotes the n-dimensional convolutional kernel weights. $e_{ij} \in \mathcal{E}$ means that there is a edge between the j-th node and the i-th node. Note that $\mathcal{E}$ is not necessarily pre-defined, it can be learnable during the training phase. S_{ij} is the entry at the i-th row and j-th column of $\mathbf{S}$ and S_{ij} denotes the correlation degree between the j-th node and the i-th node.

In practice, SLC contains two modules, *i.e.* structure module and convolutional kernel module, which are modeled to learn $\mathbf{S}$ and $\mathbf{w}$, respectively. Suppose C_{in} is the number of input channels, C_{out} is the number of output channels and K_{max} denotes the maximum neighborhood range, $\mathbf{w} \in \mathbb{R}^{C_{in}\times C_{out}\times K_{max}}$ and $\mathbf{S} \in \mathbb{R}^{N\times K_{max}}$. If $K_{max} = N$, $\mathbf{S}$ denotes a global graph structure, else it is a local one. Note that the convolutional kernels $\mathbf{w}$ are weight-sharing and capture the features that generally exist in each node. Graph structure $\mathbf{S}$, which is variable with locations but sharing across input and output channels, is designed to explicitly model the diverse structure information. Profiting from the construction, the SLC enables to aggregate input signals with different topological structures.

Relationship Between SLC and CNN Methods

As the key module in our construction, the definition of $\mathbf{S}$ is very flexible. It can be fixed and predefined, or learnable on the training phase, or obtained by a function. In this subsection, we illustrate that various previously proposed CNN methods can be obtained as particular instances of SLC with suitable definition of $\mathbf{S}$.

Classical CNN. The classical CNN can be formulated as:

$$y_i = f\Bigg(\sum_{e_{ij}\in\mathcal{E}} w_j x_j\Bigg). \tag{2}$$

Actually, Eq. (2) is equivalent to Eq. (1) in the case that all entries of $\mathbf{S}$ are set to 1. This indicates that classical CNN omits the structure learning and treats the neighborhood nodes equally without distinction. Thus, it only pertains to handle signals with the same local structure.

Chebyshev Spectral CNN (ChebNet) Defferrard *et al.* proposed to utilize the Chebyshev polynomial to represent the spectral filters (Defferrard, Bresson, and Vandergheynst 2016). Assuming T_j is the j^{th} term of Chebyshev polynomial and α_j is the corresponding coefficient, then the ChebNet can be defined as:

$$\mathbf{y} = f\Bigg(\sum_{j=1}^{C} \alpha_j T_j(\tilde{\mathbf{L}})\mathbf{x}\Bigg), \tag{3}$$

where $\tilde{\mathbf{L}} = \frac{2}{\lambda_{max}}\mathbf{L} - \mathbf{I}$, $\mathbf{L}$ presents the Laplacian matrix, $\mathbf{I}$ presents the identity matrix, C is the max order of polynomial and λ_{max} is the maximal eigenvalue of $\mathbf{L}$. Eq. (3) can be regraded as a summation of C different SLCs, each of which defines the graph structure as the Chebyshev polynomial of Laplacian matrix. That is, for the j-th SLC, it has $\mathbf{S}^j = T_j(\tilde{\mathbf{L}}) \in \mathbb{R}^{N\times N}$. Namely, although the ChebNet derives from graph spectral theory, it can be regarded as a summation of several particular SLCs.

Graph Neural Network (GNN) Kipf *et al.* simplified the construction of ChebNet with $C = 2$ (Kipf and Welling 2016). Accordingly, GNN can be defined as:

$$\mathbf{y} = f\Bigg(\alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{x}\Bigg), \tag{4}$$

where $\tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I}$, $\tilde{\mathbf{D}} = diag(\sum_{j\neq i}\tilde{w}_{ij})$. In the case we set $\mathbf{S} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}$ and $w_j = \alpha$ for any j, Eq. (4) is equivalent to Eq. (1). That is, GNN only considers the 1-order neighborhood and directly uses the normalized adjacent matrix $\mathbf{W}$ to represent local graph structure (adding $\mathbf{I}$ to capture the central node). In addition, all neighbor nodes share one kernel weight α.

Graph Convolutional Network (GCN) Although GCN (Hechtlinger, Chakravarti, and Qin 2017) does not model the graph structure explicitly, it utilizes the structure information to construct the neighborhood. Namely, GCN converts the graph-structure data to grid-like data by feat of the structure information, then executes the classical 1D CNN on it. After neighborhood construction, the formulation of GCN can be written as Eq. (2), which can be regarded as a particular instance of SLC.

Graph Attention Networks (GAT) Petar *et al.* proposed a spatial GCNN method that execute attention on graph (Veličković et al. 2017), which can be formulated as:

$$\mathbf{y}_i = f\Bigg(\sum_{j\in\mathcal{N}_i} \alpha_{ij} w_j x_j\Bigg), \tag{5}$$

$$\alpha_{ij} = softmax(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}_i} exp(e_{ik})}, \tag{6}$$

where $\mathcal{N}_i$ denotes the neighbor nodes of the i-th node. GAT can be regarded as a particular SLC with the definition of $S_{ij} = \alpha_{ij}$. Essentially, GAT defines the graph structure as the output of a attention function. However, restricted by attention mechanism, it has three limitations: 1) For a node before attention, GAT needs to know which nodes are connected with it and GAT can not add new edges. Namely the graph shape must be predefined. 2) Due to the softmax function, α_{ij} is always larger than zero, which means that GAT can not delete any edge. 3) Generally, the attention range is restricted to the neighborhood of each node. Consequently, GAT needs a predefined graph shape, and it can only learns the edges weights of the fixed graph shape.

SLCNN for Traffic Forecasting

Problem Statement

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with N nodes, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges respectively, the historical traffic data embedded on graph $\mathcal{G}$ with M input channels and P time intervals can be denoted as $\mathbf{D} \in \mathbb{R}^{N \times P \times M}$. The task of traffic forecasting is to learn a mapping function $h(\cdot)$, which takes historical traffic data $\mathbf{D}$ and graph $\mathcal{G}$ as inputs to forecast future Q time intervals traffic data:

$$\tilde{\mathbf{D}} = h\big(\mathbf{D}, \mathcal{G}, \psi\big), \tag{7}$$

where $\tilde{\mathbf{D}} \in \mathbb{R}^{N \times Q \times M}$ is the forecasting result and ψ is the learnable parameters.

Motivation

Although GCNN methods have achieved spectacular successes in many graph-structured data, three fundamental problems need to be tackled before applying GCNN on traffic forecasting tasks. First, formulation of graph structure $\mathbf{S}$ is hard to predefined. In the previous GCNN methods, the formulation of $\mathbf{S}$ usually relies on the precise adjacent matrix $\mathbf{W}$. But generally, $\mathbf{W}$ is unknown. Second, the predefined graph structures generally are local and static. Hence, they ignore the long distance dependencies of some nodes and fail to consider the dynamic property of traffic data. Third, in traffic forecasting tasks, traffic time series have complex temporal dependencies. In the paset few years, scholars proposed many methods to capture the temporal dependencies, such as directly stacking the data of different times (Zhang, Zheng, and Qi 2017), classical CNN-LSTM framework (Yao et al. 2018), Conv-LSTM or GCGRU (Li et al. 2018), ST-Conv Block (Yu, Yin, and Zhu 2018) and so on. There is no widely accepted method and how to optimally model the temporal dependencies of traffic data is still a unresolved problem. New ideas or approaches may be needed.

Based on the above problems, we apply SLC on traffic data, which provides data-driven graph learning mechanisms. Specifically, two SLC modules, named global SLC and local SLC, are developed respectively. The global SLC is designed to capture the global graph structure, *i.e.* the relationship between each node and all other nodes no matter far or near. Besides, although there exist a part of related nodes that are spatially distant, a majority of nodes are only closely related with their neighbors because of the localization of traffic data. To model the local structure better, the local SLC is developed. Each of the proposed SLC modules contains two terms designed to learn the static and dynamic structure information, respectively. Then, these two SLC modules are integrated to construct SLCNN for traffic forecasting. Additionally , P3D ConvNet is incorporated in SLCNN to capture the temporal dependencies in traffic data. The detailed mechanism of each module is described in the following subsections.

Global Structure Learning Convolution

This subsection concentrates on extracting the global graph structure, *i.e.* the relationship between each node and all other nodes. Specifically, under the framework of SLC, we learn a static graph adjacency matrix $\mathbf{W}^s \in \mathbb{R}^{N \times N}$ and a dynamic one $\mathbf{W}^d \in \mathbb{R}^{N \times N}$ from data rather than using predefined approaches. Utilizing ChebNet, the operator of global SLC can be defined as:

$$\mathbf{y}^g = f\Big(\sum_{k=1}^{C_s} \theta_k^s T_k(\mathbf{W}^s)\mathbf{x}\Big) + f\Big(\sum_{k=1}^{C_d} \theta_k^d T_k(\mathbf{W}^d)\mathbf{x}\Big), \tag{8}$$

$$\mathbf{W}^d = \phi(\mathbf{x}, \mathbf{W}_\phi) = \mathbf{x}^T \mathbf{W}_\phi \mathbf{x}, \tag{9}$$

where $\mathbf{x} \in \mathbb{R}^{N \times C_{in}}$ and $\mathbf{y}_g \in \mathbb{R}^{N \times C_{out}}$ are the input and output of global SLC, respectively. $T_k(\cdot) \in \mathbb{R}^{N \times N}$ denotes the k-th order Chebyshev polynomial, $\phi(\mathbf{x}, \mathbf{W}_\phi)$ is a function with $\mathbf{x}$ as its input and $\mathbf{W}_\phi$ as its parameter. $\theta^s \in \mathbb{R}^{C_s \times C_{in} \times C_{out}}$, $\theta^d \in \mathbb{R}^{C_d \times C_{in} \times C_{out}}$, $\mathbf{W}^s \in \mathbb{R}^{N \times N}$ and $\mathbf{W}_\phi \in \mathbb{R}^{C_{in} \times C_{in}}$ are the learnable parameters. Since SLC only captures the spatial structure, the input $\mathbf{x} \in \mathbb{R}^{N \times C_{in}}$ has no time axis. The input $\mathbf{x}$ can be the traffic data on one time interval or the stacked traffic data of multiple intervals.

Two terms in Eq. (8) correspond to global static and global dynamic structure learning, respectively. According to the previous content, ChebNet can be regard as the summation of several particular SLCs which define the graph structure as the Chebyshev polynomial of Laplacian matrix. Specifically, the former term defines $\mathbf{S}^k = T_k(\mathbf{W}^s)$, and directly learns the graph adjacency matrix $\mathbf{W}^s \in \mathbb{R}^{N \times N}$ to capture the global graph structure. After training phase, all samples of one dataset adopt the same $\mathbf{W}^s$, which extracts the static structure information shared by all samples. The latter term defines $\mathbf{S}^k = T_k(\mathbf{W}^d)$. Note that $\mathbf{W}^d \in \mathbb{R}^{N \times N}$ is not the learnable parameter but the output of function $\phi(\mathbf{x})$, which is inspired by non-local network (Wang et al. 2018) that captures the long-range relationship of two nodes by constructing a function with the current sample ($\mathbf{x}$) as input. Hence $\mathbf{W}^d$ is dynamic and depends on the current sample ($\mathbf{x}$). Each sample has a unique graph structure, which is consistent with the dynamic property of traffic data.

Some previous traffic forecasting methods, such as STGCN and DCRNN, also adopt ChebNet as their basic graph CNN operator, however the proposed global SLC differs from theirs in three respects: 1) STGCN and DCRNN need a predefined adjacent matrix $\mathbf{W}$ as input, however in global SLC, $\mathbf{W}$ is learned from data. 2) STGCN and DCRNN adopt thresholded Gaussian kernel [1] to define their adjacent matrix, thus they only capture the local graph structure but fail to consider relationships of distant nodes. In global SLC, $\mathbf{W^s}$ and $\mathbf{W^d} \in \mathbb{R}^{N \times N}$ contain the relationship of each pair nodes, no matter they are far or near, and are able to capture the global graph structure. 3) The previous methods assume all samples in a dataset share a same graph structure. Conversely, since $\mathbf{W}^d$ of the latter term is inherently based on current input, global SLC can obtain different

[1] $W_{ij} = exp(-dist(v_i, v_j)/\sigma)^2$, if dist $(v_i, v_j) < \epsilon$, otherwise $W_{ij} = 0$. $dist(v_i, v_j)$ is the distance between node i and j

structures with different input samples. That is, our formulation enables to learn the dynamic graph structures rather than a static one.

Local Structure Learning Convolution

In this subsection, the local SLC is proposed, which only concentrates on extracting the local graph structure. Specifically, we assume that distant nodes are irrelevant and only consider the relationship of nodes in the neighborhoods to construct the local structure. Relying on such modeling, the graph structure is simplified from $\mathbb{R}^{N\times N}$ to $\mathbb{R}^{N\times K}$, where K is the kernel size and N indicates the number of nodes in the graph. For any node in the graph, K nearest nodes are considered to construct the neighborhood. Under the framework of SLC, a static local graph structure $\mathbf{B}^s \in \mathbb{R}^{N\times K}$ and a dynamic one $\mathbf{B}^d \in \mathbb{R}^{N\times K}$ are learned. Given an input $\mathbf{x} \in \mathbb{R}^{N\times C_{in}}$ embedded on the graph and a activation function $f(\cdot)$, the local SLC is defined as:

$$\mathbf{y}_i^l = f\left(\sum_{j=1}^{K} B_{ij}^s \mathbf{w}_j^s \mathbf{x}_j\right) + f\left(\sum_{j=1}^{K} B_{ij}^d \mathbf{w}_j^d \mathbf{x}_j\right), \tag{10}$$

$$\mathbf{B}^d = g(\mathbf{x}, \vartheta), \tag{11}$$

where $\mathbf{x}_i \in \mathbb{R}^{C_{in}}$ and $\mathbf{y}_i^l \in \mathbb{R}^{C_{out}}$ are the input and output signals on the i-th node respectively, $g(\mathbf{x}, \vartheta)$ is a function with $\mathbf{x}$ as its input and ϑ as its parameter. $\mathbf{w}_j^s \in \mathbb{R}^{C_{out}\times C_{in}}$ and $\mathbf{w}_j^d \in \mathbb{R}^{C_{out}\times C_{in}}$ are the kernel weights on the j-th neighbor node. $\mathbf{w}^s \in \mathbb{R}^{K\times C_{out}\times C_{in}}$, $\mathbf{w}^d \in \mathbb{R}^{K\times C_{out}\times C_{in}}$, $\mathbf{B}^s \in \mathbb{R}^{N\times K}$ and ϑ are the learnable parameters.

Similar with global SLC, two terms in Eq. (10) correspond to local static and dynamic structure learning, respectively. These two term can be regarded as two particular instances of SLC with two different definitions of $\mathbf{S}$. The former term defines $\mathbf{S} = \mathbf{B}^s$, and directly learns the graph adjacency matrix $\mathbf{B}^s \in \mathbb{R}^{N\times K}$ to capture the local graph structure. The latter term defines $\mathbf{S} = \mathbf{B}^d$. Note that $\mathbf{B}^d \in \mathbb{R}^{N\times K}$ is not the learnable parameter but the output of function $g(\mathbf{x}, \vartheta)$. Hence $\mathbf{B}^d$ depends on the current sample ($\mathbf{x}$) and each sample has a unique local graph structure. The function $g(\cdot)$ can be instantiated with many types of networks, including multi-layer perception. In this paper, the p-layer graph CNN is adopted to achieve the function $g(\cdot)$ and the number of output channel of the last convolution layer is K.

Similar with global SLC, local SLC is able to learn local graph structure without given a predefined adjacent matrix and can learn dynamic structure with the benefit of $\mathbf{B}^d = g(\mathbf{x}, \vartheta)$. Besides, compared with ChebNet, local SLC can avoid the parameter allocating problem. That is, for each kernel of ChebNet, the number of weights depends on the max hops of the neighborhood rather than the number of the nodes, *i.e.* the nodes with the same hops share the same weights. It is inflexible and will finally reduce the capacity of the networks (Li and Huang 2017). However, local SLC allocates kernel weights to each neighbor node, which can avoid this problem.

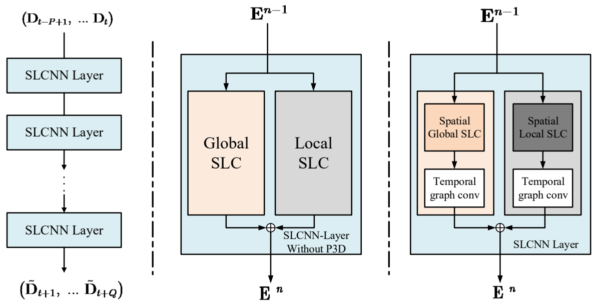

Figure 2: The left image is the architecture of SLCNN, where $\mathbf{D}_t$ is the traffic data at time t and $\tilde{\mathbf{D}}_t$ is the prediction result at time t. The middle image is a SLCNN layer without P3D where $\mathbf{E}^n$ is the output of the n-th layer. The right image is a SLCNN layer with P3D.

Pseudo Three Dimensional SLC

In order to model the temporal correlation that is crucial for traffic time series modeling, Pseudo three Dimensional SLC (P3D-SLC) is proposed to capture the temporal dependence, which combines SLC and P3D together. By modifying C3D (Tran et al. 2015), P3D is not only an economic and effective way for spatio-temporal featuring, but also an efficient way for computing (Qiu, Yao, and Mei 2017). Compared with C3D, the less computational load and parameters of P3D reduce the difficulty of training and the risk of overfitting

Technically, the key component in P3D-SLC is a combination of one structure learning convolutional layer capturing spatial dependencies and one temporal graph convolution layer which employs a 1-D convolution on time axis to model temporal connections. As illustrated in the right image of Fig. 2, a temporal graph convolution is deployed behind the global or local SLC to construct the P3D-SLC. This integrated endows our model the capability of learning spatial and temporal dependencies simultaneously. To the best of our knowledge, this work is the first to exploit P3D ConvNet in graph-structured traffic data.

SLCNN

Structure Learning Convolutional Neural Network (SLCN-N) can be constructed by switching classical convolutions in convolutional neural network to structure learning convolutions. Each layer of SLCNN is defined as summation of global and local P3D-SLC. Intuitively, the architecture of SLCNN is illustrated in the left image of Fig. 2.

In addition, it is worthy pointing out that each layer of SLCNN learns different global and local graph structures, which is different from previous GCNN methods. Since previous GCNN methods usually use a fixed and predefined graph structure, the same structure is uesd in each layer of the networks.

However, the receptive field sizes vary with layers, thus the physical meanings of the features embedded on the n-odes change over layers. In the first layer, each node may correspond to a single road. While in the high layers, fea-

Table 1: Performance of different models on BJF and BRF.

Model	BJF (15/ 30/ 60/ 90 min)		BRF (20/ 40/ 80/ 120 min)	
	MAE	RMSE	MAE	RMSE
HA	80.88	136.5	34.43	86.37
ARIMA	23.86/ 30.07/ 37.62/ 44.25	39.34/ 50.47/ 63.82/ 74.39	15.08/ 18.57/ 22.76/ 26.03	31.74/ 41.77/ 53.46/ 61.51
FNN	29.24/ 29.52 /30.30 /31.25	52.13/ 52.93/ 54.98/ 56.88	15.90/ 16.09/ 16.33/ 16.54	33.91/ 34.84/ 35.89/ 36.50
FC-LSTM	32.38/ 32.20/ 36.27/ 35.48	54.26/ 56.61/ 62.62/ 63.95	18.33/ 20.54/ 19.82/ 19.88	35.34/ 37.48/ 39.31/ 40.04
DCRNN	21.97/ 24.92/ 26.71/ 27.83	36.58/ 42.49/ 46.67/ 49.03	13.56/ 14.92/ 15.86/ 16.51	27.39/ 31.29/ 34.37/ 35.89
STGCN	21.98/ 25.55/ 28.56/ 30.60	36.44/ 43.64/ 49.51/ 53.17	14.06/ 15.87/ 17.62/ 18.90	28.28/ 33.17/ 38.29/ 41.15
GWN	22.01/ 24.91/ 26.74/ 27.97	36.82/ 42.62/ 46.87/ 49.22	13.80/ 15.13/ 16.02/ 16.64	27.94/ 31.66/ 34.69/ 36.35
SLCNN-P	22.02/ 24.51/ 27.89/ 29.74	36.93/ 43.84/ 49.08/ 52.82	14.00/ 15.73/ 17.02/ 18.24	29.29/ 34.92/ 39.21/ 42.72
SLCNN-NP3D	21.65/ 24.63/ 26.42/ 27.59	36.19/ 42.28/ 46.66/ 48.95	13.48/ 14.70/ 15.21/ 15.91	27.74/ 31.28/ 33.09/ 34.77
SLCNN	**21.24/ 23.85/ 25.29/ 26.32**	**35.71/ 41.02/ 44.32/ 46.50**	**13.24/ 14.22/ 14.69/ 15.18**	**26.79/ 29.26/ 30.46/ 31.46**

Table 2: Performance of different models on PeMS-S.

Model	PeMS-S (15/ 30/ 45 min)		
	MAE	RMSE	MAPE (%)
HA	4.01	7.20	10.61
ARIMA	5.55/ 5.86/ 6.27	9.00/ 9.13/ 9.38	12.92/ 13.94/ 15.20
FNN	2.74/ 4.02/ 5.04	4.75/ 6.98/ 8.58	6.38/ 9.72/ 12.38
FC-LSTM	3.57/ 3.94/ 4.16	6.20/ 7.03/ 7.51	8.60/ 9.55/ 10.10
DCRNN	2.37/ 3.31/ 4.01	4.21/ 5.96/ 7.13	5.54/ 8.06/ 9.99
STGCN	2.25/ 3.03/ 3.57	4.04/ 5.70/ 6.77	5.26/ 7.33/ 8.69
GWN	**2.12/ 2.80/ 3.08**	**3.94/ 5.29/ 6.01**	**5.02/ 6.76/ 7.75**
SLCNN-P	2.41/ 3.36/ 4.04	4.34/ 6.08/ 7.30	5.63/ 7.90/ 9.74
SLCNN-NP3D	2.32/ 2.98/ 3.37	4.05/ 6.58/ 6.34	5.32/ 7.35/ 8.31
SLCNN	**2.22/ 2.88/ 3.27**	**4.07/ 5.50/ 6.28**	**5.21/ 7.17/ 8.20**

Table 3: Performance of different models on PeMS-BAY.

Model	PeMS-BAY (15/ 30/ 60 min)		
	MAE	RMSE	MAPE (%)
HA	2.88	5.59	6.84
ARIMA	1.62/ 2.33/ 3.38	3.30/ 4.76/ 6.50	3.5/ 5.4/ 8.3
FNN	2.20/ 2.30/ 2.46	4.42/ 4.63/ 4.98	5.19/ 5.43/ 5.89
FC-LSTM	2.05/ 2.20/ 2.37	4.19/ 4.55/ 4.69	4.8/ 5.2/ 5.7
DCRNN	1.38/ 1.74/ 2.07	2.95/ 3.97/ 4.74	2.9/ 3.9/ 4.9
STGCN	1.46/ 2.00/ 2.67	3.01/ 4.31/ 5.73	2.9/ 4.1/ 5.4
GWN	**1.30/ 2.63/ 1.95**	**2.74/ 3.70/ 4.52**	**2.7/ 3.7/ 4.6**
SLCNN-P	1.54/ 1.96/ 2.52	3.13/ 4.19/ 5.40	3.3/ 4.5/ 6.0
SLCNN-NP3D	1.42/ 1.73/ 2.06	2.94 /3.85/ 4.56	3.0/ 4.1/ 5.2
SLCNN	**1.44/ 1.72/ 2.03**	**2.90/ 3.81/ 4.53**	**3.0/ 3.9/ 4.8**

Table 4: Performance of different models on METR-LA.

Model	METR-LA (15/ 30/ 60 min)		
	MAE	RMSE	MAPE (%)
HA	4.16	7.80	13.0
ARIMA	3.99/ 5.15/ 6.90	8.21/ 10.45/ 13.23	9.6/ 12.7/ 17.4
FNN	3.99/ 4.23/ 4.49	7.94/ 8.17/ 8.69	9.9/ 12.9/ 14.0
FC-LSTM	3.44/ 3.77/ 4.37	6.30/ 7.23/ 8.69	9.6/ 10.9/ 13.2
DCRNN	2.77/ 3.15/ 3.60	5.38/ 6.45/ 7.59	7.3/ 8.8/ 10.5
STGCN	2.87/ 3.48/ 4.45	5.54/ 6.84/ 8.41	7.4/ 9.4/ 11.8
GWN	2.69/ 3.07/ 3.53	5.15/ 6.22/ 7.37	6.9/ 8.4/ 10.0
SLCNN-P	2.75/ 3.22/ 4.04	5.41/ 6.57/ 8.05	7.2/ 9.1/ 11.7
SLCNN-NP3D	2.58/ 2.97/ 3.42	5.26/ 6.25/ 7.45	6.8/ 8.4/ 10.0
SLCNN	**2.53/ 2.88/ 3.30**	**5.18/ 6.15/ 7.20**	**6.7/ 8.0/ 9.7**

tures embedded on a node may collect the information of roads in a region. Hence, the relationships of nodes vary with layers and it is unbefitting to use the graph structure of the first layer at high layers. The proposed SLCNN learns different graph structures at different layers, which is more coincident with actual situations.

Experiments

Dataset Description

Our model is verified on six traffic datasets. Three of them (PeMS-S, PeMS-BAY and METR-LA) are public datasets released by previous work (Li et al. 2018; Yu, Yin, and Zhu 2018; Jagadish et al. 2014) and others (BJF, BRF, BRF-L) are generated by ourselves. Specifically, BJF, BRF, and BRF-L are generated from a real-world GPS trajectory data, in which about 80 million GPS points of 30,000 taxis are recoded per day. The brief introduction is reported as follows:

PeMS-S. PeMS-S is collected from Caltrans Performance Measurement System (PeMS) and the data is collected from California state highway system. The time range of PeMS-S is the weekdays of May and Jun of 2012, the interval is 5 minute and 228 sensors (nodes) are selected.

PeMS-BAY. This dataset is collected from Caltrans PeMS too, but the sensors are selected in Bay Area of California. PeMS-BAY has 6 months of data of 325 sensors (nodes), which ranges form Jan 2017 to May 2017.

METR-LA. This dataset is collected from loop detectors in the highway of Los Angeles County. The data ranges form Mar 2012 to Jun 2012 and there are 207 sensors (nodes).

BJF. Each node in BJF indicates a junction and 190 important junctions in Beijing are selected. The traffic data of each node is recorded in every 15 minutes and the time range is from November 2015 to October 2016.

BRF. BRF has totally 300 nodes and each node indicates a road. The time interval is set to 20 minutes and the time period used of BRF is from November 2015 to May 2016.

BRF-L. Similar to BRF, the traffic flow of each road is recoded in the dataset. BRF-L contains 1586 roads and the time interval is set to 10 minutes. The time period used of BRF-L is from November 2015 to December 2015.

Table 5: Performance of different models on BRF-L.

Model	BRF-L (20/ 40/ 60 min)	
	MAE	RMSE
HA	7.99	20.12
ARIMA	6.87/ 8.35/ 8.90	13.57/ 18.45/ 20.73
FNN	4.60/ 4.62/ 4.62	11.54/ 11.79/ 12.06
FC-LSTM	4.52/ 4.60/ 4.73	11.19/ 11.55/ 12.05
DCRNN	5.95/ 6.45/ 6.77	11.11/ 12.41/ 13.24
STGCN	4.21/ 4.56/ 4.81	9.15/ 10.06/ 10.68
GWN	5.83/ 6.25/ 6.49	11.00/ 12.06/ 12.71
SLCNN-P	4.00/ 4.35/ 4.53	9.10/ 10.25/ 11.05
SLCNN-NP3D	3.89/ 4.22/ 4.45	8.92/ 9.89/ 10.53
SLCNN	**3.79/ 4.02/ 4.16**	**8.73/ 9.45/ 9.94**

Experimental Settings

All experiments are conducted on a 64-bit Linux Server with 2.40 GHz CPU and NVIDIA Titan GPU. Grid search strategy is executed to choose hyper-parameters on validation.

Evaluation Metric. For PeMS-S, PeMS-BAY and METR-LA, Mean Absolute Errors (MAE), Root Mean Squared Errors (RMSE) and Mean Absolute Percentage Error (MAPE) are adopted to evaluate the performance of different methods. Due to the non-uniform distribution of urban taxis in temporal and spatial domains, there are many zeros in BJF, BRF and BRF-L. Hence, we only adopt MAE and RMSE in these three datasets.

Baselines. SLCNN is compared with widely used traffic forecasting methods, which contain the traditional methods, including Historical Average (HA), Auto-Regressive Integrated Moving Average (ARIMA), FC-LSTM (Sutskever, Vinyals, and Le 2014) and Feed-Forward Neural Network (FNN), and GCNN methods, including GWN (Wu et al. 2019), STGCN (Yu, Yin, and Zhu 2018) and DCRNN (Li et al. 2018). Moreover, SLCNN-NP3D, which removes P3D module and stacks multiple intervals data as input, is carried out to verify the effectiveness of P3D. In addition, to verify the effectiveness of structure learning, we carry out SLCNN-P(redefined), which freezes the structure learning ability of SLCNN, only uses the predefined graph structure and removes the P3D module.

Hyper-parameters Setting. The two kernel ranges (C_s, C_d) of global SLC are set to 6 and the kernel size (K) of local SLC is set to 8. The temporal depth of P3D is set to 3. For the sake of fairness, GCNN methods adopt the architecture consisting 3 layers (blocks), where the output channels of three layers are 32, 32, Q, respectively. Q is the number of time intervals of predicted results. We train our models by minimizing MAE and use Adma as optimizer.

Experimental Results

Table 1 to 5 give the results of SLCNN and the compared methods on six datasets. The proposed methods (SLCNN and SLCNN-NP3D) achieve best results on four datasets (BJF, BRF, METR-LA, BRF-L). Although on the other two datasets (PeMS-S and PeMS-bay) the performances of the proposed methods are slightly worse than the performances of GWN, they still outperforms other baselines, including STGCN and DCRNN.

Several observations can be get by further analyses. First, among all the methods, ARIMA has the worst performance. It indicates that comparing with deep learning approaches, traditional statistical method is incapable to handle complex spatio-temporal data. Second, GCNN methods generally achieve better results than non-graph methods, which implies that GCNN methods are more effective to deal with the graph-structured data since they enable to capture the spatial information. Third, the results of SLCNN-NP3D outperform the results of SLCNN-P significantly. It empirically proves that the meaningful or beneficial structures will be learned during training by the graph learning. Fourth, compared with SLCNN-NP3D, more enhances are achieved by SLCNN. It verifies that P3D contributes to capturing the temporal dependencies and it is able to help SLCNN to better model the time series of traffic data.

It is noteworthy that SLCNN-NP3D has achieved amazing performance even when P3D is removed and only spatial features are extracted. In most cases, the results of SLCNN-NP3D outperform the results of STGCN and DCRNN which extract spatio-temporal features simultaneously. This is mainly because these methods ignore exploiting the graph structure, and usually only utilize the fixed, local and predefined graph structures. The improvement of SLCNN-NP3D validates the effectiveness of graph structure learning designed in graph-structured traffic data modeling.

Ablation Study of Local and Global Graph

To verify the effectiveness of the local SLC and global SLC, two variants are compared with SLCNN:

- **SLCNN-L**(ocal): only retaining the local SLC module and removing the global SLC module.
- **SLCNN-G**(lobal): only retaining the global SLC module and removing the local SLC module.

Fig. 3 shows the prediction results of MAE on six datasets. Due to the localization of traffic data, a majority of nodes are only closely related with their near nodes. Compared with global SLC which devotes to modeling the global structure, local SLC can better capture the local structure. However, only adopting local SLC will ignore the relationship of distant nodes. Combination of global and local SLC can capture the local and graph structure simultaneously, and the performance improvement brought by the combination module superior to that of only utilizing one of them.

Ablation Study of Static and Dynamic Graph

To verify the effectiveness of the static term and dynamic term of SLC , two variants are compared with SLCNN:

- **SLCNN-S**(tatic): only retaining the static terms (the first term of Eq. (8) and (10)) and removing the dynamic terms.
- **SLCNN-D**(ynamic): only retaining the dynamic terms (the second term of Eq. (8) and (10)) and removing the static terms.

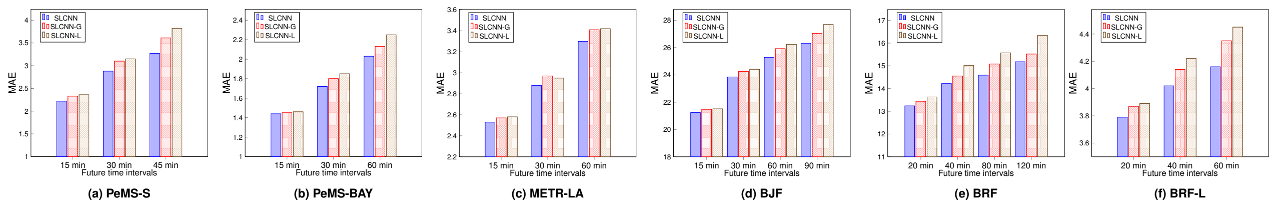

Figure 3: Comparison of SLCNN variants with local graph structure or global graph structure.

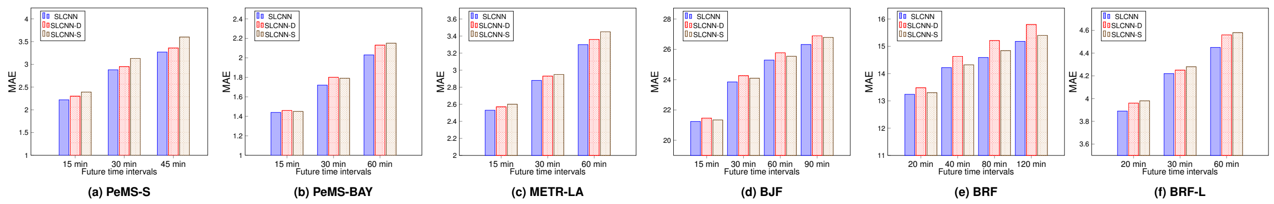

Figure 4: Comparison of SLCNN variants with static graph structure or dynamic graph structure.

Fig. 4 reports the prediction results on six datasets. SLCNN-S captures the graph structure by learnable parameters. After training phase, all samples of one dataset utilize the same parameters. SLCNN-S assums that all samples share a same structure, thus it fail to consider the dynamic property of traffic data. SLCNN-D captures the graph structure by learnable functions. The outputs of the functions are depended on the current sample, and each sample can have a unique graph structure. However SLCNN-D ignore the static structure information shared by all samples. In Fig. 3, compared with other two variants, the proposed model combining the static and dynamic structure learning achieves the best results.

Computation Time

We compare the computation cost of SLCNN with its variants and other methods on the METR-LA dataset in Table 6. Note that the main time consumption is introduced by P3D, while the SLC only consume a little time, since the 3D convolution operation is time consuming. The inference time of SLCNN-NP3D (2.54s) is comparable with GWN, while the inference time of SLCNN (21.53s) is close to DCRNN. As reported in Table 1 - 5, In most cases, the results of SLCNN-NP3D outperform the results of STGCN and DCRNN and the introducing of P3D further improves the results slightly. If users care more about time consumption, SLCNN-NP3D may be a good trade-off, which not only yields superior performances and maintaining high efficiency.

Table 6: The computation cost on META-LA

Model	Computation Time	
	Training(s/epoch)	Inference(s)
DCRNN	249.31	18.73
STGCN	19.10	11.37
GWN	53.68	2.27
SLCNN-NP3D	29.3	2.54
SLCNN	381.25	21.53
SLCNN-S	169.1	9.15
SLCNN-D	273.1	12.40
SLCNN-G	153.5	9.02
SLCNN-L	277.3	11.84
SLCNN-S-NP3D	11.2	1.32
SLCNN-D-NP3D	11.4	1.21
SLCNN-G-NP3D	20.5	1.87
SLCNN-L-NP3D	9.3	1.22

Conslusion

In this paper, we have proposed a generic graph convolutional framework (SLC) for traffic forecasting, which explicitly models the structure information into the convolutional operation. Various previously proposed GCNN methods can be regarded as particular instances of our formulation. Under the framework of SLC, we construct two modules to capture the global and local structures, respectively. Each module formulates the graph structure as learnable parameters or the output of a learnable function to capture the static and dynamic information. Moreover, P3D is incorporated in our model for better capturing the temporal dependence. Experiments on six datasets demonstrate our proposed method outperforms the state-of-the-art ones.

Acknowledgments

This research was supported by the Major Project for New Generation of AI Grant No. 2018AAA0100402 and the National Natural Science Foundation of China under Grants 91646207, 61573352, and 61773377.

References

Ahmed, M. S., and Cook, A. R. 1979. Analysis of freeway traffic time-series data by using box-jenkins techniques.

Transportation Research Record (722):1–9.

Atwood, J., and Towsley, D. 2016. Diffusion-convolutional neural networks. In *NIPS*, 1993–2001.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34(4):18–42.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Chang, J.; Gu, J.; Wang, L.; Meng, G.; Xiang, S.; and Pan, C. 2018. Structure-aware convolutional neural networks. In *NIPS*, 11–20.

Chang, J.; Wang, L.; Meng, G.; Zhang, Q.; Xiang, S.; and Pan, C. 2019. Local-aggregation graph networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Davis, G. A., and Nihan, N. L. 1991. Nonparametric regression and short-term freeway traffic forecasting. *Journal of Transportation Engineering-asce* 117(2):178–188.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 3844–3852.

Duvenaud, D. K.; Maclaurin, D.; Iparraguirre, J.; Bombarell, R.; Hirzel, T.; Aspuru-Guzik, A.; and Adams, R. P. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2224–2232.

Hechtlinger, Y.; Chakravarti, P.; and Qin, J. 2017. A generalization of convolutional neural networks to graph-structured data. *arXiv preprint arXiv:1704.08165*.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXic preprint arXic:1506.05163*.

Jagadish, H.; Gehrke, J.; Labrinidis, A.; Papakonstantinou, Y.; Patel, J. M.; Ramakrishnan, R.; and Shahabi, C. 2014. Big data and its technical challenges. *Communications of the ACM* 57(7):86–94.

Jia, Y.; Wu, J.; and Du, Y. 2016. Traffic speed prediction using deep learning method. In *IEEE International Conference on Intelligent Transportation Systems*, 1217–1222.

Kipf, T. N., and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Koesdwiady, A.; Soua, R.; and Karray, F. 2016. Improving traffic flow prediction with weather information in connected cars: A deep learning approach. *IEEE Transactions on Vehicular Technology* 65(12):9508–9517.

Laptev, N.; Yosinski, J.; Li, L. E.; and Smyl, S. 2017. Time-series extreme event forecasting with neural networks at uber. In *ICML*, number 34, 1–5.

Li, R., and Huang, J. 2017. Learning graph while training: An evolving graph convolutional neural network. *arXiv preprint arXiv:1708.04675*.

Li, Y.; Yu, R.; Shahabi, C.; and Liu, Y. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*.

Lv, Y.; Duan, Y.; Kang, W.; Li, Z.; and Wang, F.-Y. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16(2):865–873.

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

Qiu, Z.; Yao, T.; and Mei, T. 2017. Learning spatio-temporal representation with pseudo-3d residual networks. In *ICCV*, 5533–5541.

Simonovsky, M., and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *arXiv preprint arXiv:1704.02901*.

Sun, S.; Zhang, C.; and Yu, G. 2006. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems* 7(1):124–132.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

Tran, D.; Bourdev, L. D.; Fergus, R.; Torresani, L.; and Paluri, M. 2015. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 4489–4497.

Van Der Voort, M.; Dougherty, M.; and Watson, S. 1996. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies* 4(5):307–318.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *CVPR*, 7794–7803.

Williams, B. M. 2001. Multivariate vehicular traffic flow prediction: Evaluation of arimax modeling. *Transportation Research Record* 1776(1776):194–200.

Wu, Z.; Pan, S.; Long, G.; Jiang, J.; and Zhang, C. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *IJCAI*, 1907–1913.

Yao, H.; Tang, X.; Wei, H.; Zheng, G.; Yu, Y.; and Li, Z. 2018. Modeling spatial-temporal dynamics for traffic prediction. *arXiv preprint arXiv:1803.01254*.

Yu, R.; Li, Y.; Shahabi, C.; Demiryurek, U.; and Liu, Y. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *ICDM*, 777–785. SIAM.

Yu, B.; Yin, H.; and Zhu, Z. 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 3634–3640.

Zhang, J.; Zheng, Y.; and Qi, D. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *AAAI*, 1655–1661.