2018 24th International Conference on Pattern Recognition (ICPR)
Beijing, China, August 20-24, 2018

Kernel-Weighted Graph Convolutional Network: A Deep Learning Approach for Traffic Forecasting

Qi Zhang[1,2], Qizhao Jin[1], Jianlong Chang[1,2], Shiming Xiang[1,2] and 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
Email:{qi.zhang2015, jianlong.chang, smxiang, chpan}@nlpr.ia.ac.cn , jinqizhao2016@ia.ac.cn

***Abstract*—Traffic forecasting is of great significance and has many applications in Intelligent Traffic System (ITS). In spite of many thoughtful attempts in the past decades, this task still remains far from being solved, due to the diversity, complexity and nonlinearity of traffic situations. Technically, it can be cast on the framework of regressions with spatial-template data. Typically, one may consider to employ the Convolutional Neural Network (CNN) to achieve this goal. Unfortunately, the traditional CNN is developed for grid data. By contrast, here we are facing with non-grid traffic data points that are observed spatially at locations of interest. To this end, this paper proposes a novel Kernel-Weighted Graph Convolutional Network (KW-GCN) for traffic forecasting, which learns simultaneously a group of convolutional kernels and their linear combination weights for each of the nodes in the graph. This yields a mechanism that is able to learn the features locally and exploit the structure information of traffic road-network globally. By introducing additional parameters, our KW-GCN can relax the restriction of weight sharing in classical CNN to better handle the traffic data of non-stationarity. Furthermore, it has been illustrated that the proposed linear weighting of kernels can be viewed as the low-rank decomposition of the well-known locally-connected networks, and thus it avoids over-fitting to some degree. We apply our approach to the real-world GPS data set of about 30,000 taxis in seven months in Beijing. Experiments on both taxi-flow forecasting and road-speed forecasting demonstrate that our method significantly outperforms the state-of-the-art ones.**

I. Introduction

With the development of urbanization, traffic congestion has became an increasingly important issue. Traffic forecasting, one of the most important studies of Intelligent Transportation System (ITS), has received more and more attention. Actually, effective and efficient forecasting results can help travelers choose suitable travel routes, provide city managers with s-trategic decision support in traffic construction, alleviate traffic congestions and reduce the waste of resources.

The goal of traffic forecasting is to predict the future traffic conditions, such as speeds, flows, and congestion levels, based on the historical traffic data and the topological structure of traffic network. Technically, the tasks of traffic forecasting have some unique characteristics. First of all, traffic data is originated from the topology of traffic network that renders irregular grids. Typically, each road or intersection has various number of neighbors in different locations. Thus, the data is very different from those like images or videos that are defined on regular grids. What is more, the task of traffic forecasting can not be considered in isolation from the traffic network structure since the traffic conditions on different locations are coupled together in complex spatial dependencies. For example, when a traffic jam or accident occurs, not only the current road but also the adjacent ones will be affected by it. Moreover, generally, the spatial dependencies are localized. In other words, adjacent roads are often highly relevant to each other while those far from them render weak ties. In the literature, although there have been many traffic forecasting approaches [1], [2], [3], [4], [5], most of these traditional methods just concatenate the input data collected at different locations to be long vectors, without considering the spatial information in traffic network. In spite of the great progresses in the past decades, this task still remains far from being solved for real-world applications.

In the fields of pattern recognition, deep learning has been proven to be a popular yet useful methodology in practice. In the deep learning family, Convolutional Neural Networks (CNN) [6] actually acts a fundamental model for many tasks. However, CNN is suitable for data organized on grids. To remedy this drawback, Graph Convolutional Neural Networks (GCNN) is developed to extend the convolution operator from regular data to irregular data [7], [8], [9], [10], [11]. As a new type of machine learning method, GCNN has been applied in many fields, including molecular feature extraction [7], text categorization [8], [9], point cloud categorization [10] and so on. In summary, GCNN is capable of exploiting the graph structure information, extracting locally feature and handling irregular data of arbitrary connectivity directly. This observation motivates us to develop a new GCNN approach for the task of traffic forecasting as the data can be well described in term of graph derived from the traffic road network.

However, existing GCNN methods all inherit the weight sharing strategy from the classical CNN. This strategy is technically guaranteed under the assumption about the statistical stationarity on the regular gird of data, such as images, videos and so on [8], [9]. Unfortunately, it may not be held again typically for traffic data. In other words, the local statistics of traffic data change from district to district, rendering non-invariance to locations. Actually, traffic congestion is more likely to happen at some particular roads, and downtown roads always have higher traffic flow than suburban roads. Moreover, traffic conditions at different locations have different periodic patterns. For example, some roads have higher traffic flow at morning and evening peak time while some other roads have

978-1-5386-3788-3/18/$31.00 ©2018 IEEE

no remarkable difference. Since different regions of traffic roads graph have different local statistics, the spatial stationarity assumption of convolution cannot hold anymore[12] .

Based on the above observation, this paper proposes a novel model of Kernel-Weighted Graph Convolutional Networks (KW-GCN). On this network, a group of convolutional kernels and their linear combination weights are simultaneously learned, which are then combined linearly together to perform the convolutional operations. Such a design gears to the needs for dealing with the traffic data. The main contributions of this work are summarized below:

- By adjusting convolutional kernels on different locations automatically, our KW-GCN can relax the restriction of weight sharing in classical CNN, which is more suitable for traffic data compared with other GCNN methods.
- We illustrate that the operation of KW-GCN can be regard as a low-rank decomposition of Locally-Connected Network (LCN) [12], [13]. Hence compared with LCN, KW-GCN significantly reduces the parameters.
- We have evaluated our method on the real-world Global Position System (GPS) data set generated by about 30,000 taxis in seven months in Beijing. This data set consists of about $210 \times 80,000,000$ GPS points (about 80 million points per day) and totally recorded as sequence data in 2450G (unpressed). Typically, our model has been validated by two types of traffic forecasting: taxi-flow forecasting and road-speed forecasting. Extensively comparative experiments demonstrate our proposed approach significantly outperforms the state-of-the-art ones.

II. Related Work

A. Traffic Forecasting

Traffic forecasting is a classical problem in the field of ITS that has been studied for decades. As early as 1979, Ahmed *et al.* leveraged the autoregressive integrated moving average (ARIMA) model to forecast freeway traffic data [1]. Subsequently, a variety of extensions of ARIMA were proposed to improve the prediction accuracy, including subset ARIMA [10], seasonal ARIMA [11] and so on. In addition, some researchers chose Kalman filtering to predict short-term traffic flow [5], [14]. Generally, no great advances have been achieved within the traditional prediction frameworks.

Due to the great power of solving nonlinear problems, deep learning models for traffic forecasting have attracted the attention of many scholars. Lv *et al.* utilized the stacked autoencoder (SAE) model to learn generic traffic data features for traffic flow prediction [2]. Deep belief network was successfully applied in traffic studies for traffic speed [15] and flow [16], [17] prediction. Ma *et al.* adopted long short-term memory neural network for traffic speed prediction [18]. However, these methods are either univariate which only predicted the traffic condition of one place, or simply concatenated input data at different locations. In both cases, the topological structure of traffic network was ignored.

B. Graph Convolutional Neural Network

The first effort to generalize CNN from regular grid data to graph-structure data was made by Bruna *et al.* [19], which introduced a spatial method and a spectrum method based on spatial and spectrum domain, respectively. Subsequently, there are two main types of graph convolutional neural networks: spectral approaches and spatial approaches [20].

In spectral domain, in order to address the drawback of [19], Defferrard *et al.* used k order Chebyshev polynomials parametrization to reduce the computational complexity and achieved strictly localized filters [9]. Kipf *et al.* further simplified the model of [9] by limiting the $k = 1$ to reduce the computational complexity [21].

In spatial domain, Duvenaud *et al.* proposed a simple model to learn molecular fingerprints [7], where all nodes in a neighborhood shared the same kernel weights. Niepert *et al.* used graph labeling procedures to rank and select nodes to create neighborhoods then the classical 1D CNN was used to execute the convolution operation [22]. Similar to [22], Hechtlinger *et al.* leveraged the expectations of random walk to construct neighborhoods [8].

However, both spectral approaches and spatial approaches inherit the weight sharing strategy from classical CNN which implicitly assumes the statistical properties of input data satisfy stationarity. When handling the data which dose not satisfy the assumption, such as traffic data, new approaches are needed to improve further the performance.

III. Method

A. Notation

In this paper, boldface lowercase letters like $\mathbf{u}$ denote vectors, and u_i indicates the i^{th} entry of $\mathbf{u}$. Boldface uppercase letters like $\mathbf{U}$ denote matrices. $\mathbf{U}_{i*}$ and $\mathbf{U}_{*j}$ are the i^{th} row and the j^{th} column of $\mathbf{U}$, respectively. U_{ij} stands for the entry at the i^{th} row and j^{th} column in $\mathbf{U}$. Calligraphy uppercase letters like $\mathcal{U}$ stand for tensors. If $\mathcal{U}$ is a 3-D tensor, $\mathcal{U}^r_{kj}$ denotes an entry of $\mathcal{U}$, where r, k and j correspond to the first, second and third dimension of $\mathcal{U}$, respectively.

B. Motivation

The problem of traffic forecasting task is to predict future values based on the history observations. More formally, Let the input be $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_J] \in \mathbb{R}^{N \times J}$ which is embedded on the graph G, where J is the number of the input channels and N is the number of nodes in G. Each input channel corresponds to one history observation. Similarly, the output is $\mathbf{Y} \in \mathbb{R}^{N \times Q}$, where Q is the number of output channels.

As mentioned above, existing GCNN methods employ the weight sharing strategy inherited from classical CNN. That is, all the nodes in the graph at different locations share the same weights of convolutional kernel. However, in the traffic data, different nodes corresponding to roads or intersections at different locations have diverse local statistics. Consequently, using the same set of kernels to handle these nodes could largely degrade the capability of the networks. To this end, a straightforward solution is to employ LCN of which filter

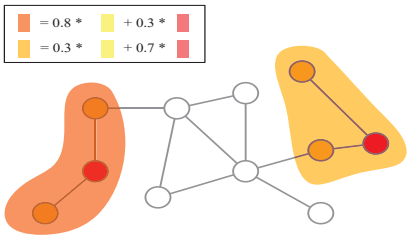


Fig. 1. The operations of KW-GCN on two different nodes of a graph. Red nodes are central node and brown ones are the neighbors. Two candidate convolutional kernels, indicated as pink and yellow are used and the kernel size is three. The number 0.8, 0.3 and 0.7 are the kernel weighted parameters. We can get different filter weights at different location based on the location-dependent kernel weighted parameters.

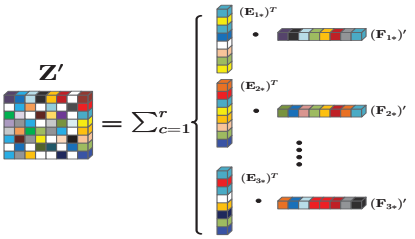


Fig. 2. The low rank approximation of LCN.

weights change with locations. Hence, LCN is able to process the data of non-stationarity. However, compared with the GCNN methods, the training parameters of LCN are greatly increased. The dense training weights make the networks hard to train and may cause the problem of over-fitting which will finally restrict the real-world applications.

Actually, GCNN and LCN can be regarded as two extreme methods. GCNN requires all the nodes have the same local statistics so that it can make the convolution filters shift-invariant [23]. LCN learns totally different filters at diverse locations, whose dense parameters restrict it to small-scale scenes. Both are not appropriate to deal with real-world traffic data. Therefore, we consider to develop a new method to combine their merits while circumvent their drawbacks.

C. Kernel-Weighted Graph Convolutional Network

Based on above analysis, we propose to introduce another type of GCNN method, named Kernel-Weights Graph Convolutional Network (KW-GCN). For easy understanding, we assume first there is only one output channel. Under this case, KW-GCN introduces multiple candidate convolution kernels and kernel weighted parameters to fit an appropriate kernel which changes with locations. Formally, c candidate convolutional kernels are denoted by $\mathcal{W} = [\mathbf{W}^1, \mathbf{W}^2, \ldots, \mathbf{W}^c] \in \mathbb{R}^{c\times K\times J}$, where K is the kernel size and $\mathbf{W}^r \in \mathbb{R}^{K\times J}$ is the r^{th} candidate convolutional kernel. $\mathcal{W}^r_{kj}$ is an entry of $\mathcal{W}$. $\mathbf{A} \in \mathbb{R}^{c\times N}$ contains all the kernel weighted parameters and A_{rn} indicates the kernel weighted parameter of r^{th} candidate convolutional kernel at n^{th} node. The output of KW-GCN at the n^{th} node can be defined as:

$$y_n = f\left(\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{r=1}^{c} A_{rn}\mathcal{W}^r_{kj}X_j^{S(n,k)}\right), \tag{1}$$

where $f(\cdot)$ is the activity function, y_n is the output at the n^{th} node. Recall that $\mathbf{X} \in \mathbb{R}^{N\times J}$ is the input data embedded on the whole graph. We denote $S(n,k)$ as the k^{th} node in the neighborhood of node n, thus $X_j^{S(n,k)}$ indicates the input data of j^{th} channel embedded on the k^{th} node in the neighborhood of node n. A fixed kernel size is used on each node followed by [22], [8]. For any node n in the graph, we choose the K nearest nodes which are less v hops from node n to construct the neighborhood, then order them according to the relative distances from node n. In this paper, v is set to 4 empirically.

For clarity, Eq. (1) can be rewritten as:

$$y_n = f\left(\sum_{j=1}^{J}\sum_{k=1}^{K}\overline{W}^n_{kj}X_j^{S(n,k)}\right), \tag{2}$$

where $\overline{\mathbf{W}}^n = \sum_{r=1}^{c} A_{rn}\mathbf{W}^r \in \mathbb{R}^{K\times J}$. Eq. (2) illustrates that KW-GCN executes a convolution operation on the n^{th} node, and the actually convolutional kernel $\overline{\mathbf{W}}^n$ is linear combined by candidate convolutional kernels $\{\mathbf{W}^r\}_{r=1}^c$ and the kernel weighted parameters $\{A_{rn}\}_{r=1}^c$. Actually, KW-GCN partly inherits the weight sharing strategy: the candidate convolutional kernels in $\mathcal{W}$ are location-independent while the kernel weighted parameters in $\mathbf{A}$ are location-dependent. We can obtain an approximate kernel at different locations by weighting the candidate kernels (as illustrated in Fig. 1). Hence, a kernel which changes with locations can be obtained without the need to learn N different convolutional kernels.

It is worthy pointing out that GCNN and LCN can be considered as two special cases of the proposed method. When $c = 1$ and all entries in $\mathbf{A}$ are equal to each other, only one candidate kernel is used and thus the actual convolutional kernel weights at different locations are identical. In this case, it is equivalent to GCNN. When c equals to N and the kernel weighted parameter on each node is one-hot representation, N candidate kernel weights are needed and one candidate kernel is chosen at each node due to one-hot coding. Thus, in this case, it is equivalent to LCN.

The above analysis on KW-GCN are conducted on the case that there is only one output channel. In a more general case, the output channel is Q. Thus KW-GCN can be generalized as follows:

$$Y_{nq} = f\left(\sum_{j=1}^{J}\sum_{k=1}^{K}\sum_{r=1}^{c} A_{rn}\mathcal{W}^r_{kjq}X_j^{S(n,k)}\right), \tag{3}$$

where $Y \in \mathbb{R}^{N\times Q}$ is the output of KW-GCN, $\mathcal{W} \in \mathbb{R}^{c\times K\times J\times Q}$ and $\mathbf{A} \in \mathbb{R}^{c\times N}$ are the parameters to be learned. $\mathcal{W}^r_{kjq}$ is an entry of $\mathcal{W}$. Eq. (3) means that KW-GCN uses c groups of candidate kernels to fit one group of kernels on each node, where each group contains Q filters.

D. Explanations via Low-Rank Approximation

In this subsection, we illustrate that the KW-GCN can be regarded as a low-rank approximation of LCN. Without loss

of generality, we consider the case $Q = 1$. According to [13], [12], we can formulate LCN as follows:

$$y_n = f\Big(\sum_{j=1}^{J}\sum_{k=1}^{K}\mathcal{V}_{kj}^{n}X_j^{S(n,k)}\Big). \tag{4}$$

$\mathcal{V} = [\mathbf{V}^1, \mathbf{V}^2, \ldots, \mathbf{V}^N] \in \mathbb{R}^{N\times K\times J}$ contains N filters, where $\mathbf{V}^n \in \mathbb{R}^{K\times J}$ ($n \in \{1, 2, \ldots, N\}$) denotes the filter on n^{th} node. $\mathcal{V}_{jk}^{n}$ is an entry of $\mathcal{V}$.

The goal is to find a low-rank tensor $\mathcal{Z} \in \mathbb{R}^{N\times K\times J}$ to approximate $\mathcal{V}$. Note that we focus on exploiting the redundancy that exists in the spatial dimension, namely, the first dimension of $\mathcal{Z}$. Hence, we reshape $\mathcal{Z} \in \mathbb{R}^{N\times K\times J}$ to $\mathbf{Z}' \in \mathbb{R}^{N\times T}$ where $T = K \times J$. According to [24], the low-rank decomposition (as illustrated in Fig. 2) of the matrix $\mathbf{Z}'$ is:

$$\mathbf{Z}' = \sum_{r=1}^{c}(\mathbf{E}_{r*})^T\mathbf{F}'_{r*}, \tag{5}$$

where $\mathbf{E} \in \mathbb{R}^{c\times N}$, $\mathbf{F}' \in \mathbb{R}^{c\times T}$, $\mathbf{E}_{r*} \in \mathbb{R}^{N}$, $\mathbf{F}'_{r*} \in \mathbb{R}^{T}$ and c is a hyper-parameter controlling the rank. Then we reshape $\mathbf{F}'$ and $\mathbf{Z}'$ to $\mathcal{F} \in \mathbb{R}^{c\times K\times J}$ and $\mathcal{Z} \in \mathbb{R}^{N\times K\times J}$, respectively. Eq. (5) can be rewritten as:

$$\mathcal{Z}_{kj}^{n} = \sum_{r=1}^{c} E_{rn}\mathcal{F}_{kj}^{r}. \tag{6}$$

Similar to $\mathcal{V}_{jk}^{n}$, $\mathcal{Z}_{jk}^{n}$ is an entry of $\mathcal{Z}$ and $\mathcal{F}_{kj}^{r}$ is an entry of $\mathcal{F}$. With this form, LCN (Eq. (4)) can be approximated as:

$$\begin{aligned} y_n =& f\Big(\sum_{j=1}^{J}\sum_{k=1}^{K}\mathcal{V}_{kj}^{n}X_j^{S(n,k)}\Big) \approx f\Big(\sum_{j=1}^{J}\sum_{i=1}^{K}\mathcal{Z}_{kj}^{n}X_j^{S(n,k)}\Big) \\ =& f\Big(\sum_{j=1}^{J}\sum_{i=1}^{K}\sum_{r=1}^{c}E_{rn}\mathcal{F}_{kj}^{r}X_j^{S(n,k)}\Big), \end{aligned} \tag{7}$$

which is actually equivalent to the definition of KW-GCN (Eq. (1)). Two parameters of KW-GCN in Eq. (1): kernel weighted parameters ($\mathbf{A}$) and candidate convolutional kernels ($\mathcal{W}$), can be regarded as the results low-rank decomposition. Hyper-parameter c controls the rank of approximation matrix and determines the number of candidate convolutional kernels.

E. Parameters Complexity

In this subsection, we analyze the parameters complexity of KW-GCN and compare it with CNN and LCN. A layer with J input channels and Q output channels is consider and the biases of filters are omitted.

For CNN, the number of parameters is $P_C = J \times Q \times K$. For the LCN, the number of parameters is $P_L = J \times Q \times K \times N$. As for KW-GCN, the number of parameters is $P_K = J \times Q \times K \times c + c \times N$. Thus, we see that P_K is between P_L and P_C. Compared with CNN, KW-GCN needs to learn extra parameters. In order to alleviate the restriction of weight sharing, it may be inevitable to introduce more parameters. On the other hand, P_K is approximately JQK/c times less than P_L (N is generally considerably larger than J, Q, c). Due to the low-rank approximation, we usually set c to a small value. Consequently, the parameters of KW-GCN are dramatically reduced compared with LCN.

IV. Experiments

A. Datasets

In this section, we verify the effectiveness of KW-GCN on four datasets. Specifically, these datasets is complied from a real-world GPS trajectory data generated by about 30,000 taxis from November 2015 to June 2016 in Beijing. For each day, up to 80 million GPS points with time and space information are recoded. Therefore, 16.8 billion (210×80 million) GPS points are totally collected, resulting in a tremendous sequence data of 2450 GB. By adequate analysis of this raw GPS data, we construct BeijingIF, BeijingRF-1, BeijingRF-2 and BeijingRS for multiple tasks including intersection taxi-flow forecasting, road taxi-flow forecasting and road-speed forecasting. For each node in the datasets, the history observation of six interval is acted as the input signals and the observation of contiguous time interval is employed as forecasting ground-truth.

BeijingIF. This dataset is constructed with respect to the traffic intersections. According to the heatmap of Beijing taxi flow (as illustrated in Fig. 3 (a)) extracted from the GPS data, we select the 198 important intersections as the graph nodes of BeijingIF. Then, two nodes are connected by an undirected edge if the corresponding traffic intersections are linked via an urban arterial road (as illustrated in Fig. 3 (b)). Furthermore, the summation of the taxi flow of each node is recoded in every 20 minutes, which implies previous six interval (120 minutes) are utilized to predict the next one (20 minutes). Here, 12,596 samples are collected for experiments.

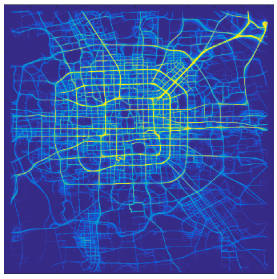
BeijingRF-1. Different from BeijingIF, the roads of Beijing are represented as nodes in the graph construction (as illustrated in Fig. 4 (b)). Specifically, 351 representative roads are chosen in this dataset (as illustrated in Fig. 4 (a)), and the total taxi flow of each road is recoded in every 60 minutes. Hence, We utilize previous six interval (6 hours) to predict the next one (1 hour). Here, 4,997 samples are collected in this dataset.

BeijingRF-2. Similar to BeijingRF-1, the total flow of each road is recoded in BeijingRF-2. While BeijingRF-2 has 1586 nodes (as illustrated in Fig. 5), and the time interval is set to 10 minutes. There are 30,672 samples in this dataset.

BeijingRS. The topology of this dataset is similar to that of BeijingRF, while only 293 nodes are remained by removing the ones without complete speed information. Typically, the average speed of each road is collected in every 10 minutes. In this dataset, 29,981 samples are collected.

B. Implementation Details

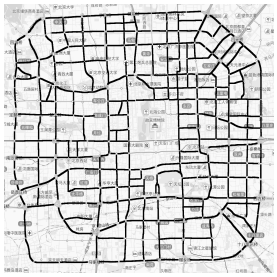
In training stage, all parameters expect the kernel-weighted parameters in $\mathbf{A} \in \mathbb{R}^{c\times N}$ are randomly initialized by a uniform distribution with the interval $[-0.01, 0.01]$. We employ the constant initialization method to initialize $\mathbf{A}$, that is, all the entries in $\mathbf{A}$ are initialized with a constant q. We assume that, in the beginning, each candidate kernel is treated without distinction on each node. Then, as the training continued, $\mathbf{A}$ is



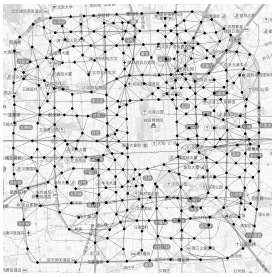
(a)

(b)

Fig. 3. (a) The heatmap of taxis flow. (b) The graph representation of BeijingIF



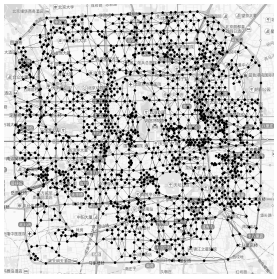
(a)



(b)

Fig. 4. (a) The choice of roads of Beijing. (b) The graph representation of the BeijingRF-1 and BeijingRS.

(a)



(b)

Fig. 5. (a) The choice of roads in Beijing. (b) The graph representation of the BeijingRF-2.

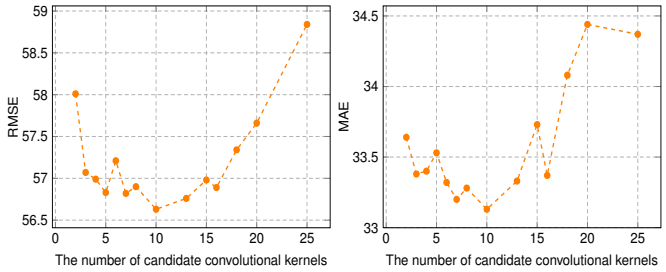


Fig. 6. The validation errors with various number of candidate convolutional kernels.

modified to be more meaningful. In this task, we set q to 0.5. In addition, we use Adma optimizer to train KW-GCN and ReLU [25] is chosen as nonlinear activity function. Mean average error loss is utilized for the regression tasks. The learning rate is 0.001 and the batch size is 64. The number of candidate convolutional kernels c is set to 10, 2, 2 and 3 on the four datasets, respectively.

C. Results Analysis

Totally, seven methods are employed here to compared with our method. We compare KW-GCN with two graph neural network methods: Locally-Connected Netwroks (LCN) and graph convolution networks (GCN) [8]. Moreover several traditional traffic forecasting methods, including fully connected neural networks (Classical NN), stacked autoencoder (SAE) [2], Historical Average (HA), Vector Auto-Regressive (VAR)[1] and AutoRegressive Integrated Moving Average (ARIMA), are employed for comprehensive comparisons.

The architecture C16-C16-C1 means a three layers network, where C(u) denotes a convolutional layer with u output channels followed by an activation. FC(u) is a fully connected layer with u hidden units followed by an activation. For fair comparisons, three graph methods (KW-GCN, LCN and GCNs) are set to the same network architecture.

The results of various models on four datasets are listed in Table I-IV, respectively. First, we can find that deep learning methods generally outperform the traditional methods (HA, ARIMA, VAR), which demonstrate the great power of solving nonlinear problems. Second, compared with SAE and classical NN, graph methods (GCN, LCN and KW-GCN) have better performance. This implies that utilizing the topological structure of traffic network is beneficial to deal with traffic data. Moreover, our method achieves better performance than LCN and GCN. There are two reasons for the improvements. First, compared with LCN, KW-GCN has less training parameters reducing the difficulty of training and the risk of overfitting. Second, compared with GCN, our method is able to adjust filter weights at different locations, which is more suited to handle data of non-stationarity.

Additionally, to observe the impact of the number of the candidate convolutional kernels c, we vary c between 2 and 25 on BeijingIF. Fig. 6 visually shows the value of c plays an important role to the performance of KW-GCN. We see too small or too large value of c degrades the performance. When c is too small, the number of candidate convolutional kernels is not enough to capture diverse local knowledge on the traffic data. Too large c increases the complexity of learning and increase the over-fitting risk. In this experiment, thus we take c = 10 to perform our proposed KW-GCN on BeijingIF.

V. Conclusion

In this paper, we have proposed the Kernel-Weighted Graph Convolutional Networks (KW-GCN). By introducing candidate convolutional kernels and kernel weighted parameters, the proposed approach alleviates the weight sharing restriction which is not appropriate for the data whose statistical properties don't satisfy the stationarity assumption. In addition,

[1]For the data with too many nodes, in order to avoid the out of memory issue, we first use k-means to divided the graph into several subgraphs and then do VAR on each subgraph.

TABLE I
PERFORMANCE OF TAXI-FLOW FORECASTING ON BEIJINGIF

Method	Architecture	RMSE	MAE
HA	-	108.99	60.30
SAE[2]	-	82.57	51.59
VAR	-	64.8	39.00
ARIMA	-	81.64	50.45
Classical NN-1	FC512-FC198	59.87	35.77
Classical NN-2	FC512-FC512-FC198	59.74	35.74
GCN[8]	C16 - C16 - C1	60.46	35.27
LCN	C16 - C16 - C1	58.93	34.48
KW-GCN	C16 - C16 - C1	**56.73**	**33.16**

TABLE II
PERFORMANCE OF TAXI-FLOW FORECASTING ON BEIJINGRF-1

Method	Architecture	RMSE	MAE
HA	-	133.02	48.62
SAE[2]	-	67.64	51.09
VAR	-	75.82	37.5
ARIMA	-	132.83	56.32
Classical NN-1	FC512-F351	59.68	39.83
Classical NN-2	FC512-FC512-FC351	62.48	41.01
GCN[8]	C16 - C16 - C1	26.85	15.87
LCN	C16 - C16 - C1	32.93	19.83
KW-GCN	C16 - C16 - C1	**25.41**	**15.16**

TABLE III
PERFORMANCE OF TAXI-FLOW FORECASTING ON BEIJINGRF-2

Method	Architecture	RMSE	MAE
HA	-	12.56	5.19
SAE[2]	-	10.34	4.26
VAR	-	11.28	4.71
ARIMA	-	10.03	4.53
Classical NN-1	FC512-F1586	9.81	4.28
Classical NN-2	FC512-FC512-FC1586	9.35	4.18
GCN[8]	C16 - C16 - C1	7.87	3.78
LCN	C16 - C16 - C1	8.13	3.74
KW-GCN	C16 - C16 - C1	**7.57**	**3.56**

TABLE IV
PERFORMANCE OF ROAD-SPEED FORECASTING ON BEIJINGRS

Method	Architecture	RMSE	MAE
HA	-	312.46	224.95
SAE[2]	-	309.20	217.48
VAR	-	305.26	217.89
ARIMA	-	295.15	206.81
Classical NN-1	FC512-FC293	308.92	212.71
Classical NN-2	FC512-FC512-FC293	303.90	214.63
GCN[8]	C16 - C16 - C1	283.06	188.79
LCN	C16 - C16 - C1	285.89	193.27
KW-GCN	C16 - C16 - C1	**276.53**	**183.69**

we illustrated that KW-GCN can be regarded as a low rank approximation of LCN. Hence, compared to LCN, KW-GCN can significantly reduce parameters. Experiments conducted on four different datasets demonstrate the superiority of our KW-GCN to the other traffic forecasting methods.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grant 91646207.

REFERENCES

[1] M. S. Ahmed and A. R. Cook, "Analysis of freeway traffic time-series data by using box-jenkins techniques," *Transportation Research Record*, no. 722, pp. 1–9, 1979.

[2] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: a deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.

[3] S. Lee and D. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *Transportation Research Record Journal of the Transportation Research Board*, vol. 1678, no. 1, pp. 179–188, 1999.

[4] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining kohonen maps with arima time series models to forecast traffic flow," *Transportation Research Part C: Emerging Technologies*, vol. 4, no. 5, pp. 307–318, 1996.

[5] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through kalman filtering theory," *Transportation Research Part B: Methodological*, vol. 18, no. 1, pp. 1–11, 1984.

[6] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[7] D. K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, and R. P. Adams, "Convolutional networks on graphs for learning molecular fingerprints," in *Advances in neural information processing systems*, 2015, pp. 2224–2232.

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

[9] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016, pp. 3844–3852.

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

[11] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional neural network: A deep learning framework for traffic forecasting," *arXiv preprint arXiv:1709.04875*, 2017.

[12] Y. Taigman, M. Yang, Marc, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[13] K. Gregor and Y. Lecun, "Emergence of complex-like cells in a temporal product network with local receptive fields," *Computer Science*, 2010.

[14] Y. Xie, Y. Zhang, and Z. Ye, "Short-term traffic volume forecasting using kalman filter with discrete wavelet decomposition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 22, no. 5, pp. 326–334, 2007.

[15] Y. Jia, J. Wu, and Y. Du, "Traffic speed prediction using deep learning method," in *IEEE International Conference on Intelligent Transportation Systems*, 2016, pp. 1217–1222.

[16] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: deep belief networks with multitask learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2191–2201, 2014.

[17] A. Koesdwiady, R. Soua, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: A deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9508–9517, 2016.

[18] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

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

[20] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

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

[22] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, 2016, pp. 2014–2023.

[23] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[24] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[25] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.