Pattern Recognition 95 (2019) 308–318

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Learning graph structure via graph convolutional networks

Qi Zhang[a,b], Jianlong Chang[a,b], Gaofeng Meng[a,*], Shibiao Xu[a], Shiming Xiang[a,b], Chunhong Pan[a]

[a] *National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China*
[b] *School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China*

ARTICLE INFO

Article history:
Received 11 May 2018
Revised 5 January 2019
Accepted 15 June 2019
Available online 2 July 2019

Keywords:
Deep learning
Graph convolutional neural networks
Graph structure learning
Changeable kernel sizes

ABSTRACT

Graph convolutional neural networks have aroused more and more attentions on account of the ability to handle the graph-structured data defined on irregular or non-Euclidean domains. Different from the data defined on regular grids, each node in the graph-structured data has different number of neighbors, and the interactions and correlations between nodes vary at different locations, resulting in complex graph structure. However, the existing graph convolutional neural networks generally pay little attention to exploiting the graph structure information. Moreover, most existing graph convolutional neural networks employ the weight sharing strategy which lies on the statistical assumption of stationarity. This assumption is not always verified on the graph-structured data. To address these issues, we propose a method that learns Graph Structure via graph Convolutional Networks (GSCN), which introduces the graph structure parameters measuring the correlation degrees of adjacent nodes. The graph structure parameters are constantly modified the graph structure during the training phase and will help the filters of the proposed method to focus on the relevant nodes in each neighborhood. Meanwhile by combining the graph structure parameters and kernel weights, our method, which relaxes the restriction of weight sharing, is better to handle the graph-structured data of non-stationarity. In addition, the non-linear activation function ReLU and the sparse constraint are employed on the graph structure parameters to promote GSCN to focus on the important links and filter out the insignificant links in each neighborhood. Experiments on various tasks, including text categorization, molecular activity detection, traffic forecasting and skeleton-based action recognition, illustrate the validity of our method.

1. Introduction

Recently, Convolutional Neural Networks (CNN) [1] have been applied extensively and achieved great successes in many fields, ranging from image segmentation and speech recognition to machine translation [2]. Powerful and efficient as they are, however, there is a strong limitation that classical CNNs can only handle the data defined on regular grids.

In the real world, there are numerous types of data that are associated to explicit topological structures. Typically, such data include molecular active data on molecular structures, traffic flow/speed data on traffic road networks, action data on human body skeletons, relation data on social networks, and so on. Unlike the data of images and time sequences that are located on spatially regular grids, obviously these data can be structured better with graphs. In the regular data, the graph is a gird, and each of its nodes has a fixed number of neighbors. Differently, in the graph-structured data, the nodes have different numbers of neighbors (as illustrated in Fig. 1). For examples, each user has a different number of friends in a social network, and each atom links to a different number of atoms in a protein molecular. Due to the complexity and arbitrariness of the structure, the graph structure data cannot be processed by directly employing the classical CNNs with fixed size of rectangular filters.

To remedy the drawback of CNN, graph convolutional neural networks are developed to extend the convolution operator from regular data to irregular data. As a new type of machine learning method, it has received more and more attention and there have been a lot of works on it [3–14]. Roughly, these methods can be divided into two main categories, *i.e.*, spectral and spatial methods. The spatial methods utilize the spatial information of the input graph to construct the neighborhood of each node then execute convolution operation on it [8,9,15]. The spectral methods generalize the convolution by operating on the spectrum of weights via

* Corresponding author.
E-mail addresses: qi.zhang2015@nlpr.ia.ac.cn (Q. Zhang), jianlong.chang@nlpr.ia.ac.cn (J. Chang), gfmeng@nlpr.ia.ac.cn (G. Meng), shibiao.xu@nlpr.ia.ac.cn (S. Xu), smxiang@nlpr.ia.ac.cn (S. Xiang), chpan@nlpr.ia.ac.cn (C. Pan).

https://doi.org/10.1016/j.patcog.2019.06.012
0031-3203/© 2019 Elsevier Ltd. All rights reserved.

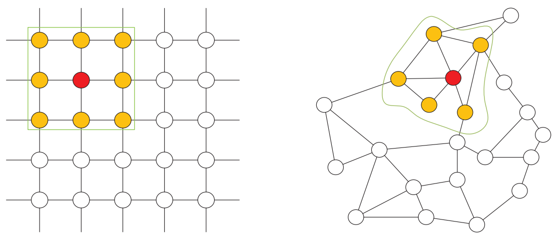
Fig. 1. The left image is the classical CNN on regular data. Each node has a fixed number of neighbors. The right image is the graph CNN on graph-structured data. Each node in irregular data has various number of neighbors. Red node indicates the central node, and yellow nodes are the neighbors of red one.

the spectral graph theory [3–6,12]. These graph CNN methods have been applied in many fields, including molecular feature extraction [15], text categorization [4,5], molecule activity levels prediction [9], traffic forecasting [16], and so on.

Despite that the graph CNN methods are widely used, there are two problems need to be considered.

(a) Current graph CNN methods pay little attention to exploiting the structures of the graph. Generally, they define the graph structures before training and once defined the graph structures will not be changed any more. In some cases, the graph structures are unknown. Hence, we can utilize k-NN [5] or the absolute value of the correlation matrix [9] to obtain the graph structures, but the obtained structures may not be absolutely accurate. In other cases, the graph structures are inherent or can be obtained by some prior knowledge, but they may not be optimal for the current learning tasks. For example, in a social network, a user may have hundreds of friends in its contacts, but only a few friends maintain close ties with it. When extracting data from the social network, it is not suitable to denote this user with hundreds of edges. For another example, in a traffic network, there is a strong correlation between two adjacent trunk roads nodes since most vehicles drive on the trunk roads. However there may be weak correlations between a trunk road and its adjacent alleys for few vehicles will leave the trunk road and drive into an alley. The original topology of traffic graph need to be modified when dealing with the specific tasks. Since both the obtained structures and the inherent structures are not satisfactory, new approaches are needed to learn the underlying graph structure.

(b) Existing graph CNN methods all employ the weight sharing strategy inherited from classical CNN. This strategy lies on the statistical assumption of stationarity which is satisfied on the regular data, such as images, videos and so on [5,9]. However, on graph-structured data, this assumption is not always verified [4]. Taking traffic flow data for an example, some trunk roads are congested almost all the time, however there are always very few cars on some suburban roads. The difference of average daily flow can be dozens even hundreds of times. Moreover, different roads may have different periodic characteristics, namely, some roads have heavier traffic flow on weekdays, some roads have heavier traffic flow on weekends while others have no obvious disparity between weekends and weekdays. In this case, filters with weight sharing are hard to capture the features at different locations of the graph.

To handle such problems, we present a novel model that learns Graph Structure via graph Convolutional Network (GSCN). By introducing the graph structure parameters, GSCN is able to learn the structure information and constantly modify the graph structure during the training phase. Additionally, GSCN can adjust the convolutional kernels at different locations according to the graph structure parameters. In summary, our main contributions are summarized below:

- We proposed a framework of graph convolutional neural network which is able to handle the data defined on irregular or non-Euclidean domains directly.
- A graph structure learning method has been introduced to enable our method to learn the relationship between two adjacent nodes and grasp the graph structure information.
- By combining the graph structure parameters, the proposed method allows the convolution kernel weights to be tuned at different locations, which alleviates the restriction of weight sharing inherited from classical CNN and will be more suitable for the data without statistical stationarity.
- The non-linear activation function ReLU and the sparse constraint are employed on the graph structure parameters to promote GSCN to focus on the important nodes and filter out the insignificant nodes in the neighborhoods. Moreover, the changeable kernel sizes are obtained, which let GSCN deploy different sized kernels at different locations of the graph.
- Extensive experiments have been explored at multiple datasets including text data, molecular structure data, traffic data and human skeleton pose data. The results demonstrate our proposed method significantly outperforms the state-of-the-art ones.

2. Related work

CNN is a type of deep neural network, which is widely used and has been extensively studied [17]. The seminal work was published in [18], which proposed the modern framework of CNN. In 2012, Krizhevsky et al. proposed AlexNet [19] that has significant improvements on the image classification task and won the championship of the ImageNet Large Scale Visual Recognition Competition. Since then, many representative CNN frameworks have been developed, including VGGNet [20], ResNet [21] and DenseNet [22]. Recently, the research on CNN has been emerged swiftly and achieved spectacular successes in many fields, including image segmentation [23], video processing [24], speech recognition [25], machine translation [26] and so on.

Since classical CNN has achieved great successes in regular data, it is an appealing and challenging work to extend the successes to new irregular structure data. The first effort to implement CNN analogy on irregular data modeled with graph was made by Bruna et al. [3], which proposed two graph methods based on spatial domain and spectrum domain, respectively. Bruna et al. blazed a new path for graph CNN, but there were still several core issues left for improvement. The spatial method in [3] has too many parameters which make the network hard to train and may cause the problem of over-fitting. The spectrum method in [3] cannot guarantee the strictly localization of the filters and the eigenvalue decomposition is time-consuming especially when dealing with the graph with numerous nodes.

Currently, the existing methods of extending CNN to graph-structured data can be roughly divided into two categories, namely, spectral and spatial methods. In spectral domain, aiming at these shortcomings of [3], Henaff et al. used smoothing kernel to achieve localized filters [4]. Defferrard et al. utilized kth order Chebyshev polynomials parametrization which significantly reduced the computational complexity and achieved strictly localized filters [5]. Kipf et al. simplified the model of [5] further in [6]. It only uses the first order approximation of Chebyshev polynomials requiring less parameters. However, the models in [5,6] allocates weights in a way similar to the concentric zone model. The number of kernel weights depend on the radius of neighborhood k rather than the number of the nodes in the neighborhood. It is inflexible and will finally reduce the capacity of the networks.

The spatial methods utilize the spatial information of the input graph to construct the neighborhood of each node then execute convolution operation on it. Atwood et al. proposed diffusion convolution operation, which simulated diffusion process to construct the neighborhoods by random walk using kth power of the transition matrix [7]. Similar to [4], diffusion convolution allocate kernel weights in a way similar to the concentric zone model. Niepert et al. leveraged graph labeling procedures to construct neighborhoods and ranked the nodes in the neighborhoods [8]. Contrast to [7,15], each node in the neighborhoods has different weights which is similar to classical CNN. GCN introduced by Hechtlinger et al. [9] uses random walk in the graph to select the local neighborhoods. Nevertheless, these methods ignore the structure information of graph when constructing neighborhoods. Typically, two nodes may render very weak ties even they are close to each other. Moreover, filters of these methods employ a fixed kernel size at different locations of input signal. It may be reasonable in regular data, but it is not suitable for graph-structured data, due to the irregularity, complexity and flexibility of graph structure.

3. Method

3.1. Notations

We define some notations in this subsection. Let $G = (V, E)$ be an undirected or directed graph, where V is a finite set of vertices and E is a finite set of edges. The graph G contains N vertices, that is, $|V| = N$. Let $\mathbf{X} \in \mathbb{R}^{N\times J}$ denote the input signal embedded on the input graph where J is the number of the input channels. Similarly, we denote $\mathbf{Y} \in \mathbb{R}^{N\times M}$ as the output signal where M is the number of output channels. Moreover, in this paper, boldface lowercase letters like $\mathbf{q}$ denote vectors, and q_i indicates the ith entry of $\mathbf{q}$. Boldface uppercase letters like $\mathbf{Q}$ imply matrices. Q_{ij} is the entry at the ith row and jth column in $\mathbf{Q}$. Calligraphy uppercase letters like $\mathcal{Q}$ imply tensors. For example, if $\mathcal{Q}$ is a 3-D tensor, $\mathcal{Q}_{jlm}$ denotes an entry of $\mathcal{Q}$, where j, l and m correspond to the first, second and third dimension of $\mathcal{Q}$, respectively.

3.2. Motivations

The graph structure is essential information for the data defined on irregular domains. In some sort of tasks, the graph structures, which determine the relationship of every two nodes of the input graph, are inherent or can be obtained by some prior knowledge. Typically, when handling the traffic forecasting tasks on a specific region, we can obtain the topological structures of traffic road networks. While in other tasks, such as text categorization, the graph structures are not given. Thus, we must construct the graph structures by some means before feeding the data into the networks.

However, in both cases, it is hard to obtain satisfactory graph structures. First, the inherent graph structures may not be the optimal for the current learning tasks. Still taking traffic forecasting as an example, if we denote intersections as nodes and roads as edges, each node pair will have a different correlation degree due to the non-uniform distribution of traffic flow. In extreme cases, some edges can be deleted since there are few cars driven on the corresponding roads. It is not appropriate to use the original topology of traffic networks as the graph structure, namely, the graph structure need to be modified. Second, when handling the tasks without graph structure, some predefine approaches (k-NN [5] or the absolute value of the correlation matrix [9]) are used to get the graph structures. However, these predefined approaches can not guarantee the absolutely accurate of constructed results.

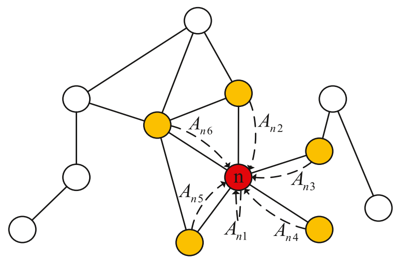

Fig. 2. Red node indicates the central node n, and yellow nodes are the ones in the neighborhood of node n.

Unfortunately, most of existing graph CNN methods, no matter spatial or spectrum methods, pay little attention to exploit and explore the graph structure information. They just feed the inherent graph structures or the constructed ones into the network before training phase. Moreover, once the graph structures are defined, they will not be changed any more. Utilizing the inaccurate graph structure will degrade the performance of the networks.

On the other hand, the weight sharing strategy inherited from classical CNN is employed in the most existing graph CNN methods. This strategy depends on the statistical assumption of stationarity which is satisfied on the regular data [5,9,11]. When handling the data which do not satisfy the assumption, weight sharing strategy will reduce the capacity of the model [4]. To this end, a straightforward solution is to employ Fully-Connected Networks (FCN) or Locally-Connected Networks (LCN) [27,28] whose filter weights change with locations. However the dense training weights increase the risk of over-fitting and make the networks hard to train.

3.3. Learning graph structure via graph convolutional networks

Based on the aforementioned analyses, we propose a method that learns Graph Structure via graph Convolutional Networks (GSCN), which introduces graph structure parameters to capture the graph structure information. Directly learning the relationship of each node pair in graph G has $\mathcal{O}(N^2)$ complexity. Due to the localization of the input data, nodes far from each other generally render weak ties. Thus supposing the nodes far from each other are unrelated, we can only consider the relationship of nodes in the neighborhoods. The complexity can be reduced to $\mathcal{O}(N \cdot K)$ if each node consider its K neighbour nodes. Consequently, the graph structure learning is integrated with the neighborhood constructing.

For ease of understanding, we first focus on a layer of GSCN, and suppose that there is only one input channel and one output channel. That is, the input signal $\mathbf{x} \in \mathbb{R}^N$ and the output signal $\mathbf{y} \in \mathbb{R}^N$ are 1-D vectors and an element of them corresponds to a node of the input graph data or output graph data. The biases of filters are omitted in this paper for notation clarity. Under this circumstance, a layer of GSCN on nth node of input graph-structured data (as illustrated in Fig. 2) can be defined as follows:

$$y_n = f\left(\sum_{l=1}^{K} A_{nl} w_l x^{s(n,l)}\right), \tag{1}$$

where $f(\cdot)$ is the activity function, y_n is the output signal of the layer on nth node, K denotes the neighborhood range of each node, $\mathbf{w}$ denotes the convolutional kernel weights and w_l is an entry of it. $\mathbf{A} \in \mathbb{R}^{N\times K}$ contains the graph structure parameters, specifically, A_{nl} is the correlation degree between the central node n and the lth node in the neighborhood of node n. We denote $s(n, l)$ as the lth node in the neighborhood of node n, thus $x^{s(n,l)}$ indicates the input data embedded on the lth node in the neighborhood of node

n. For any node n in the graph, K nearest nodes, which are less v hops from node n, are considered to construct the neighborhood. Then we order them according to the relative distances from node n. In this paper, v is set to 4 empirically.

Note that due to the complexity of the graph structure, not all signals embedded on the neighboring nodes contribute to the central one on the current task (such as classification or regression). Even though all nodes in the neighborhood relate to the central one, the correlation degree of each node pair is not equal. Some nodes may be inextricably bound. While in extreme cases, some nodes in the neighborhood may be independent to the central one even if they are linked by edges and very close in space. Hence, differing from the existing graph CNN methods, GSCN has two model parameters, **w** and **A**, to be learned during training phase. The convolutional kernel weights in **w** of GSCN play the same role with the ones of classical CNN. The graph structure parameters in **A** are utilized during each convolutional operation. For each entry of **A**, the higher the value is, the higher correlation degree the corresponding node pair has. When $A_{nl} = 0$, it means that node n is independent to the lth node in the neighborhood of node n. With the benefit of **A**, GSCN is able to focus on the nodes in the neighborhood which have high correlation degree with the central one, and pays less attention or ignores the insignificant nodes. In other words, we can capture the underlying graph structure information which is essential for graph-structured data and utilize it to improve the performance of our networks.

In a more general case, suppose that the number of input channels is L and the number of output channels is M, the GSCN can be formalized as follows:

$$Y_{nm} = f\left(\sum_{j=1}^{J}\sum_{l=1}^{K} A_{nl}\mathcal{W}_{jlm}X_j^{s(n,l)}\right), \tag{2}$$

where $\mathbf{Y} \in \mathbb{R}^{N\times M}$, $\mathbf{X} \in \mathbb{R}^{N\times J}$, Y_{nm} is an entry of $\mathbf{Y}$ and $X_j^{s(n,l)}$ is an entry of $\mathbf{X}$. The convolutional kernel $\mathcal{W} \in \mathbb{R}^{J\times K\times M}$ is a 3-D tensor, where J, M are the numbers of the input and output feature maps. $\mathcal{W}_{jlm}$ is an entry of $\mathcal{W}$ and K is the neighborhood range. Note that since each feature map of the input graph shares the same graph structure and each convolutional kernel acts on the same input graph, the graph structure parameters in **A** are scale invariant to J and M.

In addition, Eq. (2) can be rewritten as:

$$Y_{nm} = f\left(\sum_{j=1}^{J}\sum_{l=1}^{K} \mathcal{U}_{njlm}X_j^{s(n,l)}\right), \tag{3}$$

where $\mathcal{U} \in \mathbb{R}^{N\times J\times K\times M}$ and $\mathcal{U}_{njlm} = A_{nl}\mathcal{W}_{jlm}$. Eq. (3) illustrates that, by utilizing the convolutional kernel weights in $\mathcal{W}$ and the graph structure parameters in **A**, we can construct the actual filter weights in $\mathcal{U}$ which are variant at different locations. In fact, GSCN partly inherits the weight sharing strategy. The convolutional kernel weights in $\mathcal{W}$ are location-independent, which is same to classical CNN. While the graph structure parameters in **A** are location-dependent, which corresponds to the actual scenario that the local graph structures are different at different locations. It can be seen that the convolutional kernel weights are adjusted at different locations by the graph structure parameters, and $\mathcal{U}$ is the adjusted result. Hence, GSCN alleviates the limitation of weight sharing and is more suitable for the data which dose not satisfy the statistical assumption of stationarity.

3.4. Multilayer graph neural network

Above analyses focus on one layer of the proposed method. By stacking multiple layers we can obtain a multilayer networks, where the output of the previous layer is fed to the next layer.

It is worthy pointing out that, in GSCN, each layer learns different graph structure parameters, which is different from other graph CNN methods. We denote the graph structure parameters of the ith layer as $\mathbf{A}^i$. It implies that each layer has a different graph structure, which corresponds to the actual scenario because the features embedded on each node have changed. For example, in the traffic networks, each node in the first layer is constructed to record the information of a single road. While in the high layer, the features associated to each node will collect the information from its neighboring nodes. In this way, the node abstracts the information related to a larger area including many roads. Two roads in the first layer may have low correlation degree or even are independent, but in high layers, in the same place, two groups of roads may be closely related. Hence, this explains why the graph structures of different layers should be different from each other.

3.5. Changeable convolutional kernel size

Essentially, the method proposed in this paper is a kind of spatial graph CNN method. The existing spatial graph CNN methods [8,9] usually adopt fixed kernel sizes at different locations. However, each node in graph-structured data may have different number of neighbors, which is different from the nodes in regular data. For example, in protein molecule data, each atom connects to different number of atoms. Moreover, in traffic data, crossroads and T-junctions have different number of neighborhood roads. Due to the complexity and flexibility of the graph structures, some regions in graph-structured data may be very dense while others are sparse. Thus using the kernels with fixed size is not appropriate.

In order to obtain the changeable kernel sizes, we employ nonlinear activation function ReLU [29] on graph structure parameters and then add optional sparse constraint on loss function. ReLU naturally leads to a sparser network [29] and it can guarantee that all entries in $\mathbf{A}^i$ are greater than or equal to zero, since A^i_{nl} is the correlation degree between two nodes of the ith layer and there is no sense when $A^i_{nl} < 0$. The sparse constraint, which is implemented by adding the L_1 norm of $\mathbf{A}^i$ to the loss function, will make the graph structure parameters sparse further. It can promote $\mathbf{A}^i$ to only reserve the nodes with important links and allocate more zero to the weakly related nodes in the neighborhood. The loss function $\mathcal{L}$ can be formulated as follows:

$$\mathcal{L}(\mathbf{X},\mathbf{Y},\theta) = \mathcal{L}_b(\mathbf{X},\mathbf{Y},\theta) + \gamma\sum_{i}^{P}||\mathbf{A}^i||_1, \tag{4}$$

where **X** and **Y** are the input and the output of the network, θ denotes all the parameters in the network, P is the number of layers, $\mathbf{A}^i$ contains the graph structure parameters of the ith layer, $\mathcal{L}_b$ stands for the basic loss function of spacial tasks (such as classification or regression), and γ is the trade-off coefficient which balance the effect of basic loss and the sparse constraint.

One zeros entry in $\mathbf{A}^i$ means that in the ith layer the correspond node is independent with the central node in a neighborhood, and GSCN will ignore the node in the convolutional operation. At some locations of the input graph, all links in the neighborhoods are retained (the actual kernel size equal to K), while elsewhere, only one or two links may be retained (the actual kernel size equal to 1 or 2). These extreme cases are illustrated in the experiments results below (Section 4.7). Thus each actual kernel size is changeable in the proposed method and the aforementioned neighborhood range K is the maximum actual kernel size. In other words, the kernel sizes vary from different locations and adjust themselves at different locations to better fit the graph structure signal, which is similar to Deformable Convolutional Networks (DCN) [30]. Note that the sparse of $\mathbf{A}^i$ makes the actual kernel sizes smaller, but too small kernel sizes, which make the convolutional kernels concentrate on the central node and ignore the context information, will degrade the capacity of the networks. Hence, the sparse constraint is optional and in some tasks (Section 4.3), the γ is set to zero.

3.6. Connection between existing formulations and GSCN

In this subsection, we analyse the relationship between GSCN and existing formulations, including weight-sharing methods and LCN.

3.6.1. Connection between weight-sharing methods and GSCN

Technically, the output of pth sample of weight-sharing CNN methods can be formalized as follows:

$$Y_{nm}^{p} = f\left(\sum_{j=1}^{J}\sum_{l=1}^{K}\mathcal{F}_{jlm}X_{j}^{s(n,l)}\right), \tag{5}$$

where $\mathbf{Y} \in \mathbb{R}^{P\times N\times M}$ is output and $\mathcal{F} \in \mathbb{R}^{J\times K\times M}$ represents the convolutional kernel. In fact, when all entries in $\mathbf{A}$ of GSCN are equal to 1, Eq. (2) is equivalent to Eq. (5). Namely, under this circumstance, GSCN degenerates into a weight-sharing method.

Intuitively,weight-sharing methods treat all nodes in each neighborhood equally, ignoring the relationship of each node pair and graph structure. Hence they fix all entries in $\mathbf{A}^i$ with a constant and only learn the convolutional kernel weights. GSCN utilizes $\mathbf{A}^i$ to learn graph structure of each layer, retaining significance connections and eliminate weak connections. For each individual node, filters focus on the nodes which are more closely related to the central one. Furthermore, GSCN treats reserved nodes distinctly since the correlation degree of each node pair is not equal.

In addition, from the viewpoint of mathematics, we have theoretically analyze the infimum of GSCN and weight sharing methods, which can be used to explain the success of GSCN.

Lemma-1 *When the objective function has infimum, the infimum of GSCN will be smaller or equal than the infimum of weight-sharing graph CNN methods.*

Proof: Note that the learnable parameters in GSCN consist of graph structure parameters $\mathbf{A}$ and kernel weights $\mathcal{W}$. In the weight sharing methods, the kernel weights $\mathcal{F}$ are the only learnable parameters. Without loss of generality and for simplicity, we ignore the input and ground truth. Thus the loss of GSCN and weight-sharing graph CNN methods can be defined as $L_1(\mathbf{A}, \mathcal{W})$ and $L_2(\mathcal{F})$.

Suppose $\mathcal{F}^*$ is a global optimal solution of $\min_{\mathcal{F}} L_2(\mathcal{F})$, $\bar{\mathcal{A}} \in \mathbb{R}^{N\times K}$ is a constant matrix and each entry in $\bar{\mathcal{A}}$ is equal to 1.

1) Assuming $\mathbf{A} = \bar{\mathcal{A}}$ is the global optimal solution of $\min_{\mathbf{A},\mathcal{W}} L_1(\mathbf{A}, \mathcal{W})$. In this case, $\min_{\mathbf{A},\mathcal{W}} L_1(\mathbf{A}, \mathcal{W}) = \min_{\mathcal{W}} L_1(\bar{\mathcal{A}}, \mathcal{W})$. When $\mathbf{A} = \bar{\mathcal{A}}$, GSCN will degenerate into a weight sharing method. Thus, $L_1(\bar{\mathcal{A}}, \mathcal{W})$ is equivalent to $L_2(\mathcal{W})$. We can get that $\min_{\mathbf{A},\mathcal{W}} L_1(\mathbf{A}, \mathcal{W}) = \min_{\mathcal{W}} L_1(\bar{\mathcal{A}}, \mathcal{W}) = \min_{\mathcal{W}} L_2(\mathcal{W}) = L_2(\mathcal{F}^*)$. Namely, GSCN and weight-sharing graph CNN methods have the same infimum.

2) Assuming $\mathbf{A} = \bar{\mathcal{A}}$ is not the global optimal solution of $\min_{\mathbf{A},\mathcal{W}} L_1(\mathbf{A}, \mathcal{W})$. In this case, there must be a global optimal solution $\mathbf{A^*}$ $(\mathbf{A}^* \neq \bar{\mathcal{A}})$, which makes $\min_{\mathbf{A},\mathcal{W}} L_1(\mathbf{A}, \mathcal{W}) = \min_{\mathcal{W}} L_1(\mathbf{A}^*, \mathcal{W}) < \min_{\mathcal{W}} L_1(\bar{\mathbf{A}}, \mathcal{W}) = \min_{\mathcal{W}} L_2(\mathcal{W}) = L_2(\mathcal{F}^*)$. Namely, GSCN has a smaller infimum than weight-sharing methods.

Synthetically Lemma-1 is proved.

Analysis: The common loss functions (MAE, MSE for regression; cross entropy for classification) are large or equal to zero, thus they have infimum. According to Lemma-1, GSCN has smaller infimum than weight-sharing methods in thoes common loss functions, which indicates that GSCN has more potential to achieve better results in common regression and classification tasks.

3.6.2. Connection between LCN and GSCN

In this subsection, we analyse the relationship between GSCN and LCN. Based on the aforementioned analysis, the statistical assumptions of stationarity is not always verified in graph-structured data. When handling the data which dose not satisfy this assumption, weight sharing is not appropriate and may reduce the capacity of model. A direct solution to address this issue is to employ LCN whose filter weights are various with locations. According to [27,28], LCN can be formulated as follows:

$$Y_{nm} = f\left(\sum_{j=1}^{J}\sum_{l=1}^{K}\mathcal{Z}_{njlm}X_{j}^{s(n,l)}\right), \tag{6}$$

where $\mathcal{Z} \in \mathbb{R}^{N\times J\times K\times M}$ contains the filter weights which are related to the location in the graph, namely, different nodes have different filter weights. There are numerous parameters need to be learned in LCN, which makes the network hard to train and restricts it to small-scale graph. Note that Eq. (6) is equivalent to Eq. (3), and the difference is that the filter weights in Eq. (3) are combined by $\mathbf{A}^i$ and $\mathcal{W}$. In fact, for each layer, GSCN can be regarded as a special LCN, which decomposed the filter weights tensor $\mathcal{Z}$ into a matrix $\mathbf{A}^i$ and a 3-D tensor $\mathcal{W}$. Consequently, GSCN has the ability of LCN to handle the data of non-stationarity and reduces the parameters significantly by the decomposition.

3.7. Parameters complexity

In this subsection, we analyze the parameters complexity of a layer of GSCN and compare it with LCN and GCN [9]. GCN can be seen as a representative of the spatial graph CNN methods. Without loss of generality, we assume the layer has J input channels and M output channels. In addition, biases of filters are omitted for notation clarity.

For GCN, the number of parameters is $P_C = J\cdot K\cdot M$ where K is the maximum actual kernel size. For LCN, the number of parameters is $P_L = N\cdot J\cdot K\cdot M$, where N represents the nodes number of input data. As for GSCN, the number of parameters is $P_G = J\cdot K\cdot M + N\cdot K$.

We can see that the parameters complexity of GSCN falls somewhere between GCN and LCN. Compared with GCN, GSCN needs to offer extra $N\cdot K$ parameters to learn graph structure. It may be inevitable to introduce more parameters, if we want to learn graph structure information and alleviate the restriction of weight sharing. On the other hand, compared with LCN, the parameters of GSCN are substantially reduced. Generally speaking, N is considerably larger than M and L, thus the parameters of GSCN are approximately $L\cdot M$ times less than those of LCN.

4. Experiments

In this section, the proposed method is evaluated on various machine learning tasks to demonstrate the its ability. These tasks contain graph-structured data in multiple domains, including text documents, molecule data, traffic flow data and human pose data on skeleton. After that, two ablation experiments are conducted to analyze the impact of hyperparemeters. Furthermore, we visualize and analyse the graph structure parameters of each layer learned by GSCN.

4.1. Compared methods and experimental settings

We compare GSCN with two spatial graph methods, including Locally-Connected Networks (LCN) and Graph Convolutional Networks (GCN) [9]. For the spectral methods, Chebyshev spectral networks (ChebNets) [5] and Graph Neural Networks (GNN)

Table 1
The squared correlation R^2 of different methods on DPP4.

Method	Architecture	R^2	total parameters
OLS Regression	-	0.135	-
Random Forest	-	0.232	-
Merck Winner DNN	-	0.224	-
Classical NN	FC300-FC100	0.195	676,401
Spectral Networks [4]	CC16-P4-GC16-P4-FC1000-FC1000	0.277	$7.7 \cdot 10^6$
Spectral Networks [4]	CC64-P8-GC64-P8-FC1000-FC1000	0.258	$7.7 \cdot 10^6$
LCN	C10	0.229	366,011
LCN	C10-C20	0.225	7,277,141
ChebNet [5]	C10	0.225	21,580
ChebNet [5]	C10-C20	0.235	44,140
ChebNet [5]	C10-C20-FC300	0.265	12,919,050
GCN [9]	C10	0.246	21,701
GCN [9]	C10-C20	0.266	46,451
GCN [9]	C10-C20-FC300	0.264	12,921,991
GNN [6]	C10	0.232	21,551
GNN [6]	C10-C20	0.216	43,301
GNN [6]	C10-C20-FC300	0.210	12,918,841
GSCN-NS	C10	0.246	21,701
GSCN-NS	C10-C20	0.241	46,451
GSCN-NS	C10-C20-FC300	0.227	12,921,991
GSCN	**C10**	**0.282**	56,149
GSCN	C10-C20	0.275	115,347
GSCN	C10-C20-FC300	0.242	12,990,887

[6] are adopted to compare with the proposed model. To verify the effect of structure learning of GSCN, we add the contrast method GSCN-NS which removes the structure learning module from GSCN. Among these graph methods, all except GSCN and LCN are weight-sharing methods. In addition, multi-layer perceptron (Classical NN) is adopt as a comparative method. Some specific traditional methods (e.g. Linear SVM and Random Forest) in each task are also adopted for more comprehensive comparisons. In the experiments, Hyper parameters are chosen by conducting the experiments on the validation set which consists of one-third of the training data. Adam optimizer [31] is employed to optimize the objective loss functions and the learning rate is set to 0.001.

4.2. Merck molecular activity challenge

The Merck molecular activity is a challenge in Kaggle.[1] whose target is to obtain biological activities of different molecules, given numerical descriptors generated from their chemical structures. This task can be regarded as a regression problem on graph-structured data.

Following [4,9], we apply our method on the DPP4 dataset, which contains 6148 training and 2045 test molecules. Some features in DPP4 dataset are very sparse and equal to zero in the vast majority of the molecules. Thus, we omit the features that are active in less than 20 molecules, resulting in 2153 features. According to the standard set by the Kaggle challenge, the correlation coefficient R^2 is used to evaluate the predictions. In this task, mean square error loss is utilized for this regression task. γ is 1×10^{-7}. In GSCN, LCN and GCN, K is set to 16.

The results of various models are listed in Table 1. First, we can find that the graph methods are better than the traditional methods (*e.g.*, OLS regression, random forest). It demonstrates the graph methods, which are able to utilize the topology of input data, are more suitable to process graph-structured data compared with the traditional methods. Second, our GSCN outperforms the other regression methods. The significant improvements are mainly due to the graph structure learning of GSCN, which greatly improves the capacity of expression of the network. Notice that our method can achieve good effect even in the shallow network. The best result 0.282 is achieved by a one-layer convolutional network without fully connected layer. Other method must use deeper networks to achieve similar results. Deeper GSCN networks on DPP4 dataset degrade the regression result. A possible reason is that the overfitting problem is occurred since the one-layer network has relatively strong expressive ability in this task. Because of introducing graph structure parameter $\mathbf{A}^i$, each GSCN layer has more training parameters. But in this task, compared other graph methods, GSCN can use less layers to achieve better effect, resulting in reducing the number of parameters.

4.3. Text categorization

To demonstrate the generalization of our model, we also apply our method on the classification tasks. We choose two text categorization dataset: 20NEWS and Reuters-21578.

20NEWS contains 20 classes and 17,236 samples (10,167 for training and 7069 for test) [32]. Following the work in [5], we extract the 10,000 most frequently used words in the corpus. The bag-of-words model is employed to represent each document. In addition, we adopt word2vec [33] to represent each word as a 100-D word vector, and leverage cosine similarities of the word vectors to computer the connected weights between nodes.

Reuters-21578[2] is one of the most commonly used collections for text categorization task. The dataset consists of 10,788 text documents associated with 90 classes. Some documents are assigned to several classes. For simplifying the this problem, we deleted the multi-label documents, making it a one-label problem. Finally, we only focus on 9160 text documents (6577 for training and 2583 for testing). Similar to 20NEWS, we extract 1500 most frequently used words in Reuters–21578 and use bag-of-words model to represent each document. Cosine similarities of 100-D word vectors are leveraged to computer connected weights between nodes.

Cross-entropy loss is used in these two text categorization tasks. γ is set to zero. In GSCN, GCN and LCN, K is set to 25. Besides the graph methods, several traditional methods (Linear SVM, Multinomial Naive Bayes (MNB) and Softmax) are employed as the comparison methods.

[1] https://www.kaggle.com/c/MerckActivity.

[2] http://www.research.att.com/~lewis.

Table 2
Classification accuracies on 20NEWS and Reuters-21578.

Model	Architecture	20NEWS	Reuters-21578		
		Accuracy	Precision	Recall	F1-measure
Linear SVM	-	65.90	68.22	59.56	61.27
MNB	-	68.51	64.11	57.79	58.49
Softmax	-	66.28	57.15	38.09	42.43
Classical NN	FC2500	64.64	74.30	65.08	67.64
Classical NN	FC2500-FC2500	65.76	71.64	64.60	66.60
ChebNet [5]	C16-FC100	68.26	72.97	59.20	62.31
GCN [9]	C16-FC100	63.40	74.86	63.82	66.79
GNN [6]	C16-FC100	68.65	74.92	65.69	68.69
LCN	C16-FC100	64.91	75.54	67.17	69.20
GSCN-NS	C16-FC100	64.63	74.66	63.59	66.53
GSCN	**C16-FC100**	**68.83**	**77.93**	**71.31**	**73.25**

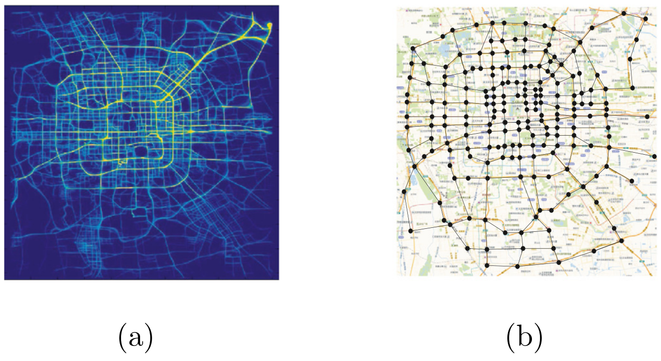

Fig. 3. The traffic network of Beijing. (a) A heatmap of taxis flow. (b) The graph representation of Beijing-198.

Table 2 reports the classification accuracy of various methods on 20NEWS and Reuters-21578. It shows that GSCN performs more favorably against other classical methods. First, as a kind of graph neural method, GSCN is more suitable to manage the graph-structured data compared with the traditional methods (e.g., Linear SVM). Second, different from other graph methods, GSCN has the ability to learn the graph structure rather than using the fixed and predefined connected weights, which can help GSCN to learn more beneficial feature representation for classification. Moreover, benefit from graph structure learning, GSCN alleviates the restriction of weight sharing. Note that although LCN and GSCN have the ability to alleviates the weight-sharing restriction, compared to LCN, a obvious enhance is achieved in GSCN. The possible reason is the overfitting risk since LCN has much more parameters as mentioned in Section 3.7. Moreover compared to GSCN-NS, GSCN has shown a significant performance improvement, which verifies the effectiveness of the structure learning module. The results of Table 2 indicate that the proposed method is effective to handle the classification tasks of graph-structure data.

4.4. Taxis flow forecasting

In this subsection, we apply our method on taxis flow forecasting task to verify the performance of GSCN on the concrete application problem. Two different scale datasets, Beijing-198 and Beijing-1586, are adopted to evaluate the performance of GSCN. For each node in the datasets, the history observations of six intervals are acted as the input signals and the observations of contiguous time interval are employed as forecasting ground-truth.

Beijing-198. Beijing-198 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 Beijing-198 (as illustrated in Fig. 3 (b)). Two nodes are connected by an undirected edge if corresponding traffic intersections are connected by an urban arterial road. We extract traffic flows of 198 traffic intersections at every twenty minutes. There are 12,549 samples in Beijing-198.

Beijing-1586. Different from Beijing-198, the roads of Beijing are represented as nodes in the graph construction. 1586 representative roads are chosen in this dataset (as illustrated in Fig. 4). Moreover, the taxi flow of each road is collected in every 10 min. In this dataset, 29,981 samples form November 2015 to June 2016 are collected.

In this task, the root mean-squared error loss (RMSE) is used. K is set to 8 and the γ is 1×10^{-7}. We compare our method to some traditional methods, including historical average (HA), stacked autoencode (SAE) [34] and autoregressive integrated moving average (ARIMA).

Table 3 shows the comparison of different traffic forecasting methods on two datasets, respectively. Note that the tradition methods (e.g. HA, SAE and ARIMA) are insufficient to handle this complex problem. As a kind of deep learning method, Classical NN can achieve relative good results. But Classical NN just concatenate the input data collected at different locations to be long vectors as inputs, without considering the spatial information in traffic network. Compared with Classical NN, the graph methods (e.g. ChebNet and GCN) can improve the performance, which illustrates that graph methods are more suitable for graph-structured data. The highest promotion is achieved by the proposed method. This indicates that GSCN is effective to deal with non-regular structure signals.

4.5. Skeleton-based action recognition

In this subsection, we apply our method on the skeleton-based action recognition task. Similar to the nodes in the roads network, each node representing a part of body that has close correlations with other nodes. Moreover, the structure of human body can be considered as a kind of graph naturally.

We perform our experiments on the RGB+D dataset of NTU [35] which is one of the largest action recognition datasets with 3D skeleton data. It has 56,880 action samples which consist of almost 4 million frames. Each frame in the skeletal data sample has three dimensional locations of 25 major body joints. We perform down-sampling or repeat sampling on different sequences to warp them into a designated number of frames. In order to keep the number of nodes fixed, we only adopt the samples with one person. Finally, we use 44,837 samples (33,628 for training and 11,209 for testing) associated with 60 classes. In this task, the K is set to 8 and the γ is 1×10^{-7}.

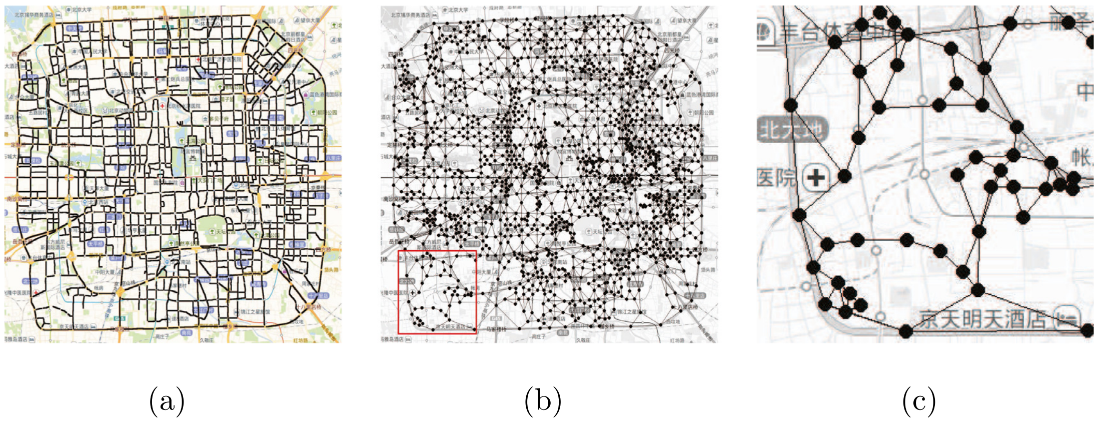

Fig. 4. (a)The choice of roads in Beijing-1586. (b) The graph representation of Beijing-1586. (c) A local renderings of (b).

Table 3
RMSE among various methods on Beijing-198 and Beijing-1586.

	Beijing-198		Beijing-1586	
Method	Architecture	RMSE	Architecture	RMSE
HA	-	108.99	-	12.56
SAE [34]	-	82.57	-	10.34
ARIMA	-	81.64	-	10.03
Classical NN	FC512-FC198	59.84	FC512-FC1586	9.76
Classical NN	FC512-FC512-FC198	59.74	FC512-FC512-FC1586	9.72
ChebNet [5]	C32 - C32 - C1	59.18	C32 - C32 - C1	8.51
GCN [9]	C32 - C32 - C1	58.05	C32 - C32 - C1	7.79
LCN	C32 - C32 - C1	58.83	C32 - C32 - C1	10.09
GNN [6]	C32 - C32 - C1	62.44	C32 - C32 - C1	7.92
GSCN-NL	C32 - C32 - C1	58.35	C32 - C32 - C1	7.85
GSCN	C32 - C32 - C1	**55.47**	C32 - C32 - C1	**7.58**

Table 4
Classification accuracies on NTU.

Method	Architecture	Accuracy
Classical NN	FC1024-FC1024-FC512-FC512	48.63
ChebNet [5]	C48-C48	63.34
GCN [9]	C48-C48	62.68
LCN	C48-C48	54.57
GNN [6]	C48-C48	55.29
GSCN-NS	C48-C48	62.27
GSCN	C48-C48	**64.16**

The classification result of various models are listed in Table 4. There are huge performance differences between the non-graph methods and graph methods, which shows the advantages of graph methods when dealing with graph-structured data. Moreover, the results accord with our observations in previous experiments that that GSCN performs more favorably against other graph methods. This suggests the effectiveness of graph structure learning. It is worth noting the graph in this dataset has only 25 nodes representing the corresponding 25 major body joint. GSCN also outperforms other graph methods on this dataset. Hence, the results illustrate the validity of our method in the small-scale graph.

4.6. Ablation analysis

In this subsection, we employ two experiments to demonstrate the performance of GSCN with the effect of different hyper parameters. Specifically, DPP4 is considered as a basic task in the following experiments, since DPP4 is a common graph-structure dataset and frequently used in the previous work (such as [4,9]).

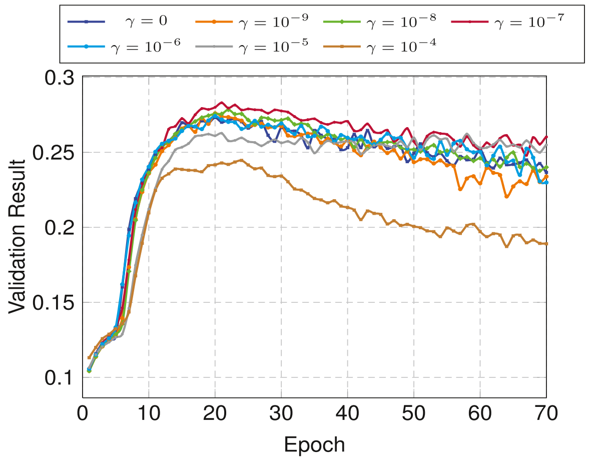

Fig. 5. Result on validation data with different sparse constraints.

4.6.1. Impact of sparse constraint

To investigate the impact of the sparse constraint, we conduct a series of experiments with different sparse constraints on DPP4. Sparse constraint affect sparse degree of graph structure parameters. As illustrated in Fig. 5, too small or too large sparse constraint is not suitable. Small sparse constraint let the graph structure parameters include too many non-zero values. The weekly linked nodes in the neighborhood will involve in the convolution operation. On the other hand, note that the sparse of graph structure parameters makes the actual kernel sizes smaller. Too larger sparse constraint will make the convolutional kernels ignore the context information and eventually degrade the capacity of the networks.

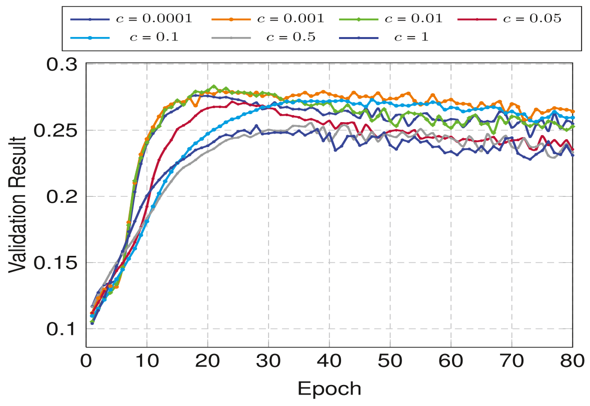

Fig. 6. Result on validation data with diverse initialization values.

4.6.2. *Sensitivity of initialization*

In this subsections, we evaluate the sensitivity of initialization in GSCN. Difference from the usual practice, constant initialization method is adopted to initialize $\mathbf{A}^i$ ($i = 1, 2, \ldots, P$), rather than random initialization. Namely, all entries in $\mathbf{A}^i$ are initialized with a constant c. This is designed to simulate the graph structure learning process: in the beginning, filters in GSCN treat each node equally without discrimination. Then, as the training goes on, $\mathbf{A}^i$ is modified to be more meaningful. Filters allocate higher weights to the closely related nodes and lower weights to the weakly related nodes.

To observe the impact of the initialization values, we vary c from $1e^{-4}$ to 1 on DPP4. The results are reported in Fig. 6. It can be seen that the proposed method is not very sensitive to the initialization. Small values of c (*e.g.* $c \in \{0.001, 0.01, 0.05, 0.1\}$) contribute to relative good results, but if too large values $c \in \{0.5, 1\}$ are selected, the performance will be degraded.

4.7. *Graph structure parameter visualization*

In this subsection, we visualize and analyse the graph structure parameters of each layer ($\mathbf{A}^i$) learned by GSCN. Two experimental visualized results on different datasets (DPP4 and Beijing-198) are presented to better investigate the characters of $\mathbf{A}^i$. The network architectures on DPP4 and Beijing-198 are C10-C20 and C32-C32-C1, respectively.

Despite that both of them are regression problems, two datasets actually correspond to two different situations. The molecular activity prediction task on DPP4 only output a single value. While in taxi flow forecasting task, taxi flows at all locations are require. Specifically, in Beijing-198 dataset, given an input graph of 198 nodes, we need to predict the traffic flow on 198 node of contiguous time interval simultaneously.

For the sake of clarity, we randomly choose 50 nodes to illustrate the $\mathbf{A}^i$. The jth row represents the corresponding graph structure parameters of the jth node. The results of two task are illustrated in Fig. 7. First, it is obviously that not all nodes in the neighborhoods are reserved, a part of entries in $\mathbf{A}^i$ are zero (shown as black). Taking Fig. 7 (a) for an example, although we set kernel size to 16, the actual kernel sizes are usually less than it since not all signals embedded on the neighboring nodes contribute to the central one on the current task. In extreme cases, in a neighborhood, only one or two nodes are regarded to be related nodes.

In addition, It is worthy pointing out that the graph structure parameters assign higher value on the first row in the taxi flow forecasting task (as illustrated in Fig. 7 (c) (d) (e)). As mentioned above, we order the nodes in the neighborhood according to the relative distances from the central one. Hence, the first row of $\mathbf{A}^i$ corresponds to the central nodes of all the neighborhoods. That is, GSCN pays more attention to the central node in the taxi flow forecasting task. It coincides with the actual situation that the taxi flow value in one node of next time interval is more relevant to its own history traffic data. In contrast to Beijing-198, this phenomenon is not observed in DPP4. The graph structure parameters $\mathbf{A}^i$ in DPP4 are sparser and no obvious statistical regularity is noticed. Given an input graph, molecular activity prediction task only output one value, which may urge $\mathbf{A}^i$ to learning more global structure. The visualization results shows that GSCN has the ability to learn different type of graph structure parameters according to different tasks.

5. Conclusion and future work

In this paper, we have proposed a method that learns Graph Structure via graph Convolutional Networks (GSCN). By introducing graph structure parameters, the proposed method is able to leverage the graph structure information and focus on the important nodes in the neighborhood. The parameter complexity of GSCN is reduced to $\mathcal{O}(N \cdot K)$ from $\mathcal{O}(N^2)$, since we only consider the K neighbors of each node. Except for the case of very huge graph, the complexity is acceptable. Moreover, it alleviates the weight sharing limitation, which may degrade performance of the networks when handling graph-structured data. Furthermore, we employ ReLU and the sparse constraint on the graph structure parameters to obtain

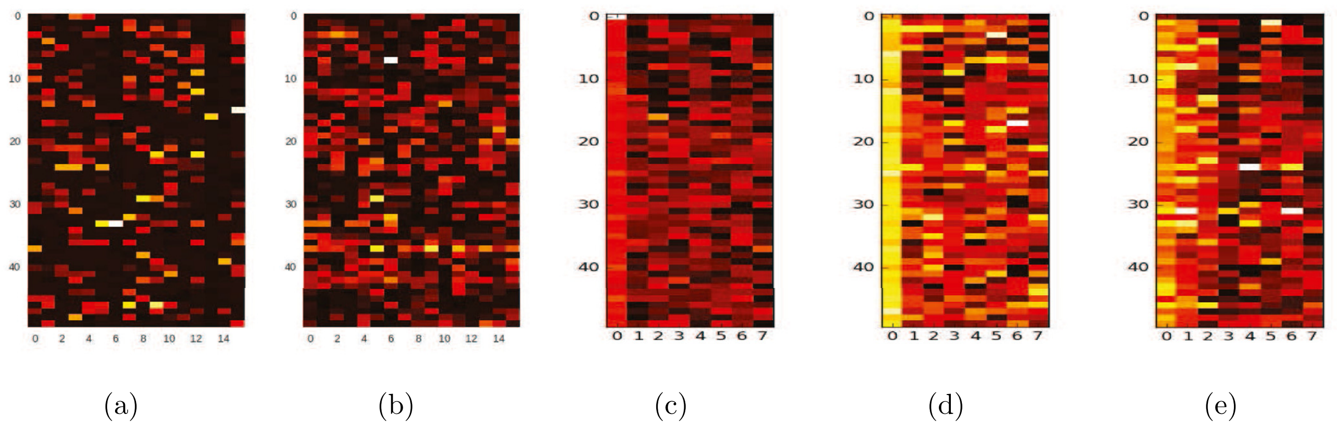

Fig. 7. Graph structure parameters of DPP4 and Beijing-198. (a), (b) are the first and second layers graph structure parameters of DPP4. (c),(d),(e) stand for the first, second and third layer graph structure parameters of Beijing-198, respectively.

changeable kernel sizes, which are more suitable for the graph-structured data. The experiment results of various datasets demonstrate the superiority of the proposed method.

Future works will investigate two directions based on the limitation of GSCN. First, although the parameters complexity has been reduced to $\mathcal{O}(N \cdot K)$, when the number of graph is very large, the parameters complexity will be a vexed problem. We will explore appropriate methods, such as low-rank decomposition, to reduce the parameters complexity of GSCN and apply it on huge graphs. Second, the proposed method dose not contain any pooling operation which is a important part in tradition CNN. We will try to find a way to extend the pooling operation to graph-structured data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 91646207, 61773377, 61573352 and 61620106003.

References

[1] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.

[2] Y. Lecun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[3] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: Proceedings of the International Conference on Learning Representations, 2014.

[4] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, CoRR (2015) abs/1506.05163.

[5] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process Syst. (2016) 3837–3845.

[6] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the International Conference on Learning Representations, 2017.

[7] J. Atwood, D. Towsley, Diffusion-convolutional neural networks, in: Advances in Neural Information Processing Systems, 2016, pp. 1993–2001.

[8] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 2014–2023.

[9] Y. Hechtlinger, P. Chakravarti, J. Qin, A generalization of convolutional neural networks to graph-structured data, CoRR (2017) abs/1704.08165.

[10] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 5425–5434.

[11] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in: Proceedings of the IEEE international Conference on Computer Vision and Pattern Recognition, 2017, pp. 29–38.

[12] R. Li, J. Huang, Learning graph while training: an evolving graph convolutional neural network, in: Proceedings of the International Conference on Learning Representations, 2017.

[13] M.M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Process. Mag. 34 (4) (2017) 18–42.

[14] D. Boscaini, J. Masci, S. Melzi, M.M. Bronstein, U. Castellani, P. Vandergheynst, Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks, Comput. Gr. Forum 34 (5) (2015) 13–23.

[15] D.K. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Advances in Neural Information Processing Systems, 2015, pp. 2224–2232.

[16] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: data-driven traffic forecasting, in: Proceedings of the International Conference on Learning Representations, 2017.

[17] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, et al., Recent advances in convolutional neural networks, Pattern Recognit. 77 (2018) 354–377.

[18] Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in Neural Information Processing Systems, 1989, pp. 396–404.

[19] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1106–1114.

[20] K. Simonyan, A. Zisserman, Very deep convolutional networks for large scale image recognition, in: Proceedings of the International Conference on Learning Representations, 2015.

[21] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[22] G. Huang, Z. Liu, L. van der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2017, pp. 2261–2269.

[23] Y. Duan, F. Liu, L. Jiao, P. Zhao, L. Zhang, SAR Image segmentation based on convolutional-wavelet neural network and Markov random field, Pattern Recognit. 64 (2017) 255–267.

[24] M. Ma, N. Marturi, Y. Li, A. Leonardis, R. Stolkin, Region-sequence based six-stream CNN features for general and fine-grained human action recognition in videos, Pattern Recognit 76 (2018) 506–521.

[25] L. Deng, O. Abdel-Hamid, D. Yu, A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2013, pp. 6669–6673.

[26] J. Gehring, M. Auli, D. Grangier, D. Yarats, Y.N. Dauphin, Convolutional sequence to sequence learning, in: Proceedings of the International Conference on Machine Learning, 2017, pp. 1243–1252.

[27] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: closing the gap to human-level performance in face verification, in: Proceedings of the IEEE Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2014, pp. 1701–1708.

[28] K. Gregor, Y. LeCun, Emergence of complex-like cells in a temporal product network with local receptive fields, CoRR (2010) abs/1006.0448.

[29] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[30] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, Y. Wei, Deformable convolutional networks, in: Proceedings of the International Conference on Computer Vision, 2017, pp. 764–773.

[31] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, CoRR (2014) abs/1412.6980.

[32] T. Joachims, A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, in: Proceedings of the International Conference on Machine Learning, 1997, pp. 143–151.

[33] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, CoRR (2013) abs/1301.3781.

[34] Y. Lv, Y. Duan, W. Kang, Z. Li, F. Wang, Traffic flow prediction with big data: a deep learning approach, IEEE Trans. Intell. Transp. Syst. 16 (2) (2015) 865–873.

[35] A. Shahroudy, J. Liu, T. Ng, G. Wang, NTU RGB+D: a large scale dataset for 3d human activity analysis, in: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 2016, pp. 1010–1019.

Qi Zhang received the B.S. degree in automatic control from Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015. Currently, he is a student in National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include pattern recognition, data mining and machine learning.

Jianlong Chang received the B.S. degree in School of Mathematical Sciences from University of Electronic Science and Technology of China, Chengdu, China, in 2015. Currently, he is a second-year master student in National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include deep unsupervised learning, pattern recognition and machine learning.

Gaofeng Meng (SM'17) received the BS degree in applied mathematics from Northwestern Polytechnical University, in 2002, the MS degree in applied mathematics from Tianjin University, in 2005, and the Ph.D. degree in control science and engineering from Xi'an Jiaotong University, in 2009. He is now an associate professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences. His research interests include document image processing and computer vision. He serves as an associate editor for Neurocomputing since 2014. He is now a senior member of the IEEE.

Shibiao Xu received the BS degree in information engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2009, and the Ph.D. degree in computer science from Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2014. He is currently an associate professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include image based three-dimensional scene reconstruction and scene semantic understanding.

Shiming Xiang received the B.S. degree in mathematics from Chongqing Normal University, Chongqing, China, in 1993, the M.S. degree from Chongqing University, Chongqing, China, in 1996, and the Ph.D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2004. From 1996 to 2001, he was a Lecturer with the Huazhong University of Science and Technology, Wuhan, China. He was a Postdoctorate Candidate with the Department of Automation, Tsinghua University, Beijing, China, until 2006. He is currently a Professor with the National Lab of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include pattern recognition, machine learning, and computer vision.

Chunhong Pan received his B.S. Degree in automatic control from Tsinghua University, Beijing, China, in 1987, his M.S. Degree from Shanghai Institute of Optics and Fine Mechanics, Chinese Academy of Sciences, China, in 1990, and his Ph.D. degree in pattern recognition and intelligent system from Institute of Automation, Chinese Academy of Sciences, Beijing, in 2000. He is currently a professor with the National Laboratory of Pattern Recognition of Institute of Automation, Chinese Academy of Sciences. His research interests include computer vision, image processing, computer graphics, and remote sensing.