Neurocomputing 423 (2021) 639–650

Contents lists available at ScienceDirect

Neurocomputing

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

Graph classification based on structural features of significant nodes and spatial convolutional neural networks

Tinghuai Ma [a,*], Hongmei Wang [a], Lejun Zhang [b,*], Yuan Tian [c], Najla Al-Nabhan [d]

[a] School of Computer & Software, Nanjing University of information Science & Technology, Jiangsu, Nanjing 210-044, China
[b] College of Information Engineering, Yangzhou University, Yangzhou 225127, China
[c] Nanjing Institute of Technology, Jiangsu, Nanjing 211-167, China
[d] College of Computer and Information Sciences, King Saud University, Riyadh 11362, Saudi Arabia

ARTICLE INFO

Article history:
Received 29 February 2020
Revised 16 June 2020
Accepted 14 October 2020
Available online 3 November 2020
Communicated by Wu Jia

Keywords:
Graph classification
Convolutional neural network
Significant vertices
Structural characteristics

ABSTRACT

Many real-world problems can be abstracted into graph classification problems. Recently, graph convolutional networks have achieved great success in the task of node classification and link prediction. However, when using graph convolution network to process the task of graph classification, either global topology information or local information is ignored. Therefore, designing graph convolutional networks to improve the accuracy of graph classification has attracted more and more attention. Inspired by the use of convolutional neural networks to process graph-structured data, we put forward a new spatial convolutional neural network architecture for graph classification. To be specific, we first design a comprehensive weighting method to measure the significance of vertices in the graph based on multiple indicators to choose the central node sequence. Then, the normalization process of the graph is realized by constructing the same size neighborhood graphs for the central vertices. After that, the structural characteristics of the graph are extracted from both local and global aspects. Finally, the tensors obtained after the above steps are respectively input into the following two spatial convolutional neural network architectures to perform classification, one is a simple CNN structure, which has only two convolution layers, one dense layer and one softmax layer. The other is to modify the architecture of CNN, and the channel concatenation layer is introduced to determine the classification result of the entire graph according to the category of the neighborhood graphs. Experimental results on two kinds of real-world datasets, bioinformatics and social network datasets, indicate that our approach obtains competitive results and is superior to some classic kernels and similar deep learning-based algorithms on 6 out of 8 benchmark data sets.

1. Introduction

Many data in the real world, such as social networks [1–4], chemical compounds [5], protein interaction networks [6], etc., can be commonly represented as a graph structure. As an important branch in the field of graph mining, graph classification is applied to numerous practical fields [7–10]. However, since the traditional machine learning algorithm cannot directly process the graph structure data, it is necessary to vectorize such data.

In the past few decades, the graph kernel function is a popular method of vectorizing graph data [11]. The graph classification method based on the kernel uses the graph kernel function to obtain the similarity matrix between the graphs first [5], and then inputs the similarity matrix into the traditional classifier such as Support Vector Machine (SVM) to perform task of the graph classification. There are two main problems of the graph classification methods based on kernel function. First, as the number of samples increases, the cost of storing and calculating the kernel matrix is high. The other is the lack of feature selection process. However, the low discriminative substructure of the graph set will lead to a decrease in classification accuracy, and when mapping the graph to a high-dimensional feature space, it may contain plenty of redundant features, which may also result in increased computational complexity and reduced classification accuracy.

Recently, deep learning approaches like convolutional neural networks (CNNs) have been successful in many fields, such as image classification [12], object recognition [13], and natural language processing [14–16]. The traditional convolutional neural network (CNN) performs convolution operations on pixels on the

* Corresponding authors.
E-mail addresses: thma@nuist.edu.cn (T. Ma), zhanglejun@yzu.edu.cn (L. Zhang).

https://doi.org/10.1016/j.neucom.2020.10.060
0925-2312/

image, while the graph structure lacks an ordered tensor representation. Therefore, the graph convolution network emerges, which is a deep learning method that can process graph-structured data [17–19]. In general, existing graph convolution networks fall into two categories: spectral convolution [20–22] and spatial convolution [23–26]. Spectral convolution exploits the spectral filters using the graph laplacian to redefine the convolution in the Fourier domain [21]. By contrast, spatial convolution is directly convolved on the graph. It usually aggregates the characteristics of neighbor vertices to the central node [24]. However, most existing spatial methods only consider a single indicator to measure the significance of a vertex when selecting a central node sequence, which may lose some nodes that are important to the classification. For example, the PSCN proposed by Niepert et al. [25] used centrality measure, such as betweenness centrality or degree centrality, to map nodes into an ordered sequence to select important nodes, which failed to comprehensively measure the importance of nodes from multiple aspects. At the same time, most of the methods fail to make full use of the structural features and node features of the graph, which will be introduced in detail in related work. In addition, although literatures [27,28] are aimed at image classification and pattern recognition, they use hypergraphs or tensors as tools to achieve local matching to solve classification problems, so it also has great inspiration for processing graph classification task.

In order to solve the issue of considering one-sidedness while choosing the central node and extracting characteristics in the above spatial methods, we put forward a novel spatial convolutional neural network (SCNN) for graph classification, which is based on structural characteristics of significant nodes. In this article, our contributions are as follows:

- While selecting the central node sequence, we put forward a comprehensive weighting method to determine the comprehensive weights of multiple evaluation indicators that measure the significance of the node. The selected evaluation index combines the global and local information of the node. And then the results obtained by multiplying the evaluation index value of the vertex and the corresponding comprehensive weight of each indicator are summed to obtain the significance of the node. Thereby the nodes in the graph can be sorted by the importance of the node.
- For the unity of node characteristics input to the convolutional layer, it is proposed to choose multiple node features and simultaneously introduce the spectral characteristics of nodes from both local and global aspects of the graph, which can retain more structural information of graphs.
- The channel concatenation layer is proposed so that the classification of the whole graph can be determined based on the category of neighborhood graphs.

The rest of the article is organized as follows. We introduce the related work on graph kernel and graph convolution network methods in Section 2. In Section 3, preliminary knowledge involved in this article is introduced. In Section 4, we introduce our novel spatial convolutional neural network for graph classification, including the selection of central node sequences, construction of neighborhood graphs, selection of node features, and architecture of spatial convolutional neural networks. Experimental results are given in Section 5. Finally, conclusion and future work are presented in Section 6.

2. Related work

The kernel methods obtain their similarity by calculating the dot product between two graphs in the high-dimensional feature space, then a traditional classifier such as SVM is subsequently used to carry out graph classification task with the similarity matrix as input [29]. The representative kernel functions used in the current graph classification can be broadly divided into three types. The first type of graph kernel is based on paths and walks, such as the shortest path kernel [30], the random walk kernel [11,31], etc. The second category is based on the graph kernel of finite-scale subgraphs, such as the graphlet kernel [32]. The third category is based on the subtree patterns, such as the Weisfeiler-Lehman subtree kernel [33].

The walk-based kernel method compares the similarity by calculating the matching walks of the two input graphs. Gärtner et al. [31] proposed to calculate the number of matching walks between two graphs by defining the product of the two graphs, so as to compare the similarity of the two graphs. The problem with this method is that the same vertex can be accessed while walking, which results in a large value of the calculated kernel function, so the calculation complexity of processing the product is very high. Kriegel et al. [30] proposed a kernel method based on the shortest path. This method calculates the length of all shortest paths between any two vertices in the graph. If the two shortest paths have the same length, the two paths are similar. This method avoids the problem that the walk-based kernel method will repeatedly traverse the same node, but only focusing on the shortest path may cause some important information in the graph to be lost.

Shervashidze et al. [32] proposed a graphlet graph kernel. This method decomposes the graph into small-scale subgraph structures. The size of graphlet is generally set to 3, 4, 5. By counting the number of occurrences of finite-scale sub-graph patterns of different sizes in the two graphs, the graph is converted into a vector. Afterwards, the kernel similarity matrix is obtained by inner product, and finally can be classified by conventional classification methods. This method calculates the time required for graphlet graph kernel and the size of graphlet to grow exponentially, and at the same time, this method fails to utilize node labels. In order to solve the above problem of not using node labels, Shervashidze et al. [33] proposed the Weisfeiler-Lehman subtree kernel. The specific process is to first create and sort multi-set labels for the set labels of each node and all its neighbor nodes. Then, these labels are compressed into new shorter label values according to a certain mapping, and the newly generated label value is used for the next iteration. When comparing the structure of two graphs, only the number of labels co-occurring in the two graphs is calculated, that is, the number of shared subtrees between the two graphs is compared.

With the introduction of Word2Vec and Doc2Vec, there has been a lot of work that combines graph kernels with word embedding models or document embedding models to solve graph classification tasks. Yanardag et al. [29] proposed the deep graph kernel (DGK), which used the Word2Vec model to learn the embedding of atomic substructures, thereby constructing a kernel matrix. These graph classification algorithms based on graph kernel mentioned above can capture the similarity of substructures in different dimensions and achieve higher classification accuracy on many data sets. However, one of the main limitations of these methods is the high computational complexity and lack of scalability, so it is not suitable for large graphs.

Recently, many graph classification approaches based on graph convolutional neural networks have been put forward. The first type is based on spectral convolution. Bruna et al. [20] proposed the first-generation spectral convolution and introduced the knowledge of the map theory to construct the convolution operation. The spectral convolution on the graph can be defined as the product of the signal and the filter in the Fourier domain. This method requires a matrix multiplication for each convolution operation and requires feature decomposition, so the amount of

calculation is large; secondly, each convolution operation must consider all nodes, and there is no spatial locality of the traditional CNN. Later, Defferrard et al. [21] used polynomial approximation to represent the filter, which further simplified the calculation of the graph convolution formula. Using the Chebyshev polynomial with free parameters learned in the neural network model, an approximate smoothing filter was obtained in the spectral domain, which is the second generation spectral convolution. This method only considers K-hop neighbors and has a certain spatial locality. Compared with the first generation spectrum, the computational complexity is reduced.

Kipf & Welling et al. [22] proposed a graph convolution network adapted to graph structure data based on the traditional convolution algorithm. In essence, the graph convolutional network is a local first-order approximation of the spectral convolution, which can be used to encode the local graph structure and node features. The model is a third-generation spectral convolution that iteratively aggregates the neighbor embedding of the node and uses an aggregate function to derive the embedding obtained in the previous iteration and its embedding aggregate to obtain a new embedding. In this method, only 1-hop neighborhoods are considered for each layer convolution, and a larger receptive field is obtained by stacking multiple layers, and the computational complexity is further reduced.

Unlike spectral convolution, spatial convolution operates directly on the topology graph. Atwood and Towsley [24] proposed Diffusion-CNN, which performs random walks on the graph to select adjacent nodes that are close in the convolution space, where the weight assignment is controlled by the hop count between the two nodes. It uses different weights to propagate neighbors of different hops to the center. However, the convolution operation of this approach is related to the power series of the transfer matrix, so the calculation overhead is large. This method performs a simple averaging of the node-level features for the graph classification task. Muhan Zhang et al. [26] proposed a propagation-based spatial graph convolution called DGCNN, which applies graph convolution to extract vertex characteristics, and sorts and selects vertices based on these characteristics in the pooling stage, and then a one-dimensional convolution is performed on the sorted node embeddings. However, the structural characteristics of the graph have not been fully used.

Niepert et al. [25] proposed a spatial graph convolution algorithm called PSCN, which applies the graph labeling algorithms and the external software Nauty [34] to sort and select vertices, and then extracts fixed-size local blocks for each chosen vertex in light of the graph labeling results as the receptive fields of traditional CNN. Finally, the convolution operation is performed to extract characteristics by traversing the linear sequence of vertices. The main disadvantage of this approach is that graph labeling algorithms, such as centrality measures and Weisfeiler-Lehman colors, are generally not injective, and evaluating the significance of vertices based on a single centrality indicator or ranking vertices based on their Weisfeiler-Lehman colors cannot accurately identify vital nodes. At the same time, relying on external software to forcibly break the vertex sequence of the identical tag weakens the sense of sorting and results in the loss of significant vertex information.

The method proposed in this article is based on PSCN. Compared with it, for the graph labeling algorithm used while choosing the central vertex sequence, our method does not use a single index to measure the significance of vertices in the graph, but proposes a combined weighting method and utilizes multiple evaluation indicators to calculate the significance of nodes, which is a method that combines subjective preferences and objective information to evaluate the relative weight of various indicators. The significance of vertices in the graph can be comprehensively measured from multiple aspects, avoiding the use of a single evaluation index to measure the one-sidedness of the importance of nodes. In addition, when extracting characteristics, unlike PSCN and DGCNN, which only extract vertex or edge characteristics without considering the structural characteristics of the graph, multiple characteristics are introduced from local and global aspects to extract more information about the graph in our approach. At the same time, based on the convolution architecture in PSCN, it is proposed to add a channel connection layer after the convolution layer, so that the classification result of the entire graph can be determined according to the category of the neighborhood graph. In Section 4, we will introduce more details of the proposed method.

3. Preliminary knowledge

In this section, the relevant concepts of the graph are introduced firstly. Then several evaluation indicators of node importance that will be used in the comprehensive weighting method proposed in Section 4 are introduced, including kshell, Closeness Centrality (CC), Betweenness Centrality (BC) and Degree Centrality (DC), and the calculation formulas are given.

3.1. Graphs

An undirected and unweighted graph G can be denoted as a triple $G = (V, E, A)$, where $V = \{v_1, v_2, \cdots, v_n\}$is the set of vertices, $E = \{e_1, e_2, \cdots, e_m\}$is the set of edges, and $A \in R^{n\times n}$ is the adjacency matrix of the graph. The adjacency matrix represents the connection relationship between nodes, where $A_{ij} = 0$ means that there is no edge connection between node i and node j, and $A_{ij} = 1$ means that there is an edge from node i to node j. Add each column element of A to get n numbers, then put them on the diagonal (other places are zero) to form an $n \times n$ diagonal matrix, recorded as degree matrix D. Then the Laplacian matrix of the graph can be calculated by the following formula:

$$L = D - A \tag{1}$$

In addition, another more common form of Laplacian matrix is the symmetric normalized Laplacian, which is defined as:

$$L^{sym} = D^{-1/2}LD^{-1/2} = I - D^{-1/2}AD^{-1/2} \tag{2}$$

where I is the identity matrix. The element L_{ij}^{sym} is given as follows:

$$L_{ij}^{sym} = \begin{cases} 1 & \text{if } i = j \text{ and } \deg(v_i) \neq 0 \\ -\frac{1}{\sqrt{\deg(v_i)\deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{is adiacent to} v_j \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

where $\deg(v_i)$ represents the degree of vertex v_i.

3.2. Evaluation indicators of node importance

Kshell considers the global characteristics of a node's position in the network. The larger the kshell value of a node is, the greater the influence of the node is, indicating that the more crucial the position of the node is.

CC is used to describe the position of a vertex in the network. It is an evaluation index based on the global characteristics of the graph. The larger the CC value of a vertex is, the more important it is. The reciprocal of the average shortest distance of the vertex is defined as the CC of the node, which can be defined by the following formula:

$$CC(i) = \frac{n-1}{\sum_{j\neq i} d_{ij}} \tag{4}$$

where d_{ij} represents the shortest path distance from node i to node j.

BC reflects the global attributes of the node. The greater the BC value of a node is, the more vital the node is. The BC value of a vertex can be calculated by the following equation:

$$BC(i) = \sum_{s \neq i \neq t \in V} \frac{d_{st}(i)}{d_{st}} \tag{5}$$

Where d_{st} represents the number of all shortest paths from vertex s to vertex t, and $d_{st}(i)$ is the quantity of shortest paths through vertex i between nodes s and t.

DC reflects the local significance of the nodes, which only considers the local features but ignores the global features of the graph. The larger the DC value is, the more important this node is in the graph. For an undirected graph with n nodes, the DC value of node i can be expressed by the following formula:

$$DC(i) = \frac{\sum_{j=1}^{n} A_{ij}}{n-1} \tag{6}$$

where A_{ij} represents the element of the $i-th$ row and $j-th$ column in the adjacency matrix A, and $n-1$ denotes the maximum number of connections if possible for node i.

4. Proposed method

In this section, we introduce our novel spatial convolutional neural networks, including the following four parts: the selection of central node sequence, construction and normalization of neighborhood graphs, selection of node features, and architecture of spatial convolutional neural networks. Finally, the time complexity of the proposed algorithm is analyzed briefly.

4.1. Selection of central node sequence

In this paper, the comprehensive weighting method is proposed to calculate the significance of nodes in the graph to select the central node sequence. Specifically, the method selects multiple evaluation indicators that measure the significance of the vertex, and combines the subjective weights with the objective weights to obtain the comprehensive weight of each evaluation index. Then, the obtained comprehensive weight of each indicator is multiplied by the corresponding indicator value of the node. Finally, all the results obtained in the previous step are added to calculate the significance of the vertex.

The proposed comprehensive weighting method is a method that combines subjective preferences and objective information to evaluate the relative weight of various indicators. If only subjective weights are used for evaluation, the results will usually have subjective arbitrariness to a large extent, because the subjective intention of people is reflected in this process. However, using objective weights alone to evaluate, the results has a strong mathematical theoretical basis, but it does not consider the subjective cognition of people. Therefore, both subjective and objective weighting methods have certain flaws. Our proposed comprehensive weighting approach is based on AHP (Analytic hierarchy process) [35] and variation coefficient method, in which AHP calculates the subjective weight of each indicator and the objective weights are determined using the variation coefficient method. Then the multiplication normalization is used to obtain the comprehensive weight of each indicator.

The steps of the AHP are as follows:

a) First, the three scale method is used to compare the four evaluation indicators to build a comparison matrix B.
The construction of the comparison matrix B is based on the following factors: DC describes the local properties of the vertex and conveys less information, so it is least important compared to other evaluation indicators. Compared with the other three evaluation indices, BC can more accurately discover the "bridge" vertices in the network, that is, the necessary way of communication, while the other three indicators do not have this function. Therefore, the importance of giving BC in the construction of matrix B is the highest in this paper. Both kshell and CC describe the global and location attributes of the network, so the same importance evaluation is given in matrix B.
Table 1 lists the values in the comparison matrix B constructed by the three scale method calculated according to Eq. (7).

$$B = (B_{ij}) = \begin{cases} 2, \text{Index}i \text{ is more important than index } j. \\ 1, \text{Index}i \text{ is as important as index } j. \\ 0, \text{Index}j \text{ is more important than index } i. \end{cases} \tag{7}$$

b) The range method is used to construct a judgment matrix C. Let $C = (C_{ij}) = C_b^{(r_i - r_j)/R}$, where C_b is the relative importance of a given pair of rang elements, which is a constant and is generally set to 9, $r_i = \sum_{j=1}^{4} B_{ij}$ and $R = \max(r_1, r_2, r_3, r_4) - \min(r_1, r_2, r_3, r_4)$. Let $M_i = \prod_{j=1}^{4} C_{ij}$, $W_i = \sqrt[4]{M_i}$ and $\tilde{W}_i = \frac{W_i}{\sum_{i=1}^{4} W_i}$. Then, the obtained judgment matrix is as follows:

$$C = \begin{bmatrix} C & kshell & CC & BC & DC & M_i & W_i & \tilde{W}_i \\ kshell & 1 & 1 & 1/3 & 3 & 1 & 1 & 0.1875 \\ CC & 1 & 1 & 1/3 & 3 & 1 & 1 & 0.1875 \\ BC & 3 & 3 & 1 & 9 & 81 & 3 & 0.5625 \\ DC & 1/3 & 1/3 & 1/9 & 1 & 1/81 & 1/3 & 0.0625 \end{bmatrix}$$

c) To determine whether the $\tilde{W}_i$ obtained by the judgment matrix can be used as the weight, the consistency test is as follows: Let $C = (C_{ij})_{4\times4} = \begin{bmatrix} 1 & 1 & 1/3 & 3 \\ 1 & 1 & 1/3 & 3 \\ 3 & 3 & 1 & 9 \\ 1/3 & 1/3 & 1/9 & 1 \end{bmatrix}$ and $H = (H_i)_{4\times1} = C \cdot \tilde{W}_i = \begin{bmatrix} 0.75 \\ 0.75 \\ 2.25 \\ 0.25 \end{bmatrix}$, then the maximum eigenvalue

Table 1
Comparison matrix B constructed by the three scale method, where kshell, Closeness Centrality (CC), Betweenness Centrality (BC) and Degree Centrality (DC) represent indicators to evaluate the importance of nodes.

B	kshell	CC	BC	DC
kshell	1	1	0	2
CC	1	1	0	2
BC	2	2	1	2
DC	0	0	0	1

$\lambda_{\max}$ of the judgment matrix can be calculated, i.e. $\lambda_{\max} = \sum_{i=1}^{4} \frac{H_i}{n\widetilde{W}_i} = \frac{1}{4}\left(\frac{0.75}{0.1875} + \frac{0.75}{0.1875} + \frac{2.25}{0.5625} + \frac{0.25}{0.0625}\right) = 4$. Finally, calculate the consistency index *CI* and the consistency ratio *CR* of the matrix, where $CI = \frac{\lambda_{\max} - n}{n-1}$ and $CR = \frac{CI}{RI}$. *RI* is a random consistency indicator.

Since the calculated values of *CI* and *CR* are both 0, the judgment matrix satisfies the consistency. In general, the consistency of the judgment matrix is thought to be good when $CR < 0.1$; otherwise, the judgment matrix should be re-adjusted to have satisfactory consistency. Through the consistency test, the subjective weights of each indicator are $W_1^s = 0.1875, W_2^s = 0.1875, W_3^s = 0.5625$ and $W_4^s = 0.0625$.

Then, according to the comprehensive weight of each indicator, the significance of the node v_i in this paper is defined as follows:

$$I(v_i) = W_1^c kshell(v_i) + W_2^c CC(v_i) + W_3^c BC(v_i) + W_4^c DC(v_i) \tag{11}$$

To eliminate the influence of the scale of the graph on the value, the importance of the node is normalized to be unified in the $[0, 1]$ interval. Therefore $I(v_i)$ is redefined as follows:

$$I(vn_i) = \frac{I(v_i)}{\sum_{j \in N} I(v_j)} \tag{12}$$

where N is the number of vertices in the graph and $\sum_{i \in N} I(vn_i) = 1$.

The whole selection process of the central node sequence is as shown in Algorithm 1. For a graph G with n vertices, first,

Algorithm 1 Select the Central Node Sequence

Input: A graph $G = (V, E)$ of n nodes, the number of central nodes w.
Output: An ordered set V_{sort} of central nodes.
1 Determine the subjective weight W_i^s of each evaluation indicator by AHP;
2 Calculate the values of the four indicators for all nodes in the graph;
3 Determine the objective weight W_i^o of each evaluation indicator using the variation coefficient method;
4 According to formula (10), multiplication normalization is used to obtain the comprehensive weight W_i^c of each indicator;
5 Calculate the importance of each node v_i in graph G according to formula (11);
6 Normalize node importance according to Equation (12);
7 **if** $w < n$ **then**
8 | V_{sort}= the first w nodes of V according to normalize node importance;
9 **else**
10 | V_{sort}= n ordered nodes + $w - n$ zeros;
11 **end**
12 **return** V_{sort};

The variation coefficient method. In the evaluation system, the indicator with a large coefficient of variation should be given higher weight. For a graph, we first calculate the values of the four evaluation indicators mentioned in Section 3.1.2 for all vertices in the graph. Then let σ_i and μ_i be the standard deviation and the mean of the $i-th$ evaluation index, respectively. The ratio of the standard deviation to the mean is defined as the coefficient of variation. That is, the coefficient of variation of the $i-th$ evaluation index is calculated by the following form:

$$vc_i = \frac{\sigma_i}{\mu_i} \tag{8}$$

Therefore, the objective weight of the $i-th$ evaluation index in this paper can be written as the following equation:

$$W_i^o = \frac{vc_i}{\sum_{i=1}^{4} vc_i} \tag{9}$$

After that, the multiplication normalization is used to get the comprehensive weight of each factor as follows:

$$W_i^c = \frac{W_i^s \cdot W_i^o}{\sum_{i=1}^{4} W_i^s \cdot W_i^o} \tag{10}$$

the subjective and objective weights of each evaluation indicator are calculated, as in lines 1 to 3 of Algorithm 1. Then the fourth line in Algorithm 1 calculates the comprehensive weight of each evaluation index according to the proposed comprehensive weighting method. Next, lines 5 to 6 in Algorithm 1 calculate the normalized node importance of each node. Finally, lines 7 to 12 in Algorithm 1 rank all nodes according to the calculated normalized vertex importance to select the central node. If the number of vertices in the graph is not less than w, we select the first w nodes to form the central node sequence. Otherwise, $w - n$ zero receptive fields are created, equivalent to padding.

4.2. Construction and normalization of neighborhood graphs

In this section, a neighborhood graph is constructed for each node in the central node sequence obtained in Section 4.1, which is a fixed-size receptive field. Specifically, the neighbor of the central node is regarded as a candidate node of the receptive field, and then the selected neighbor node sequence is normalized to obtain a neighborhood graph. Algorithm 2 lists the steps to construct and normalize the neighborhood graph.

Algorithm 2 Construction and normalization of neighborhood graphs

Input: Ordered central node sequence $V_{sort} = \{v_1, v_2, \cdots, v_w\}$, the size of the neighborhood graph k.
Output: An ordered collection of neighbors N_i^{sort} for each central nodes v_i.

```
1  for i = 1, ··· , w do
2      N_i = [v_i];
3      L_i = [v_i];
4      while |N_i| < k and |L_i| > 0 do
5          L_i = ∪_{v∈L_i} N_1(v);
6          N_i = N_i ∪ L_i;
7      end
8      if |N_i| < k then
9          Add k − |N_i| zeros in the set N_i;
10     end
11     N_i^sort = the first k ordered vertices of N_i, which are sorted in ascending order according to the shortest path
         distance from the root vertex firstly, in descending order according to the value of the degree centrality
         of the vertex secondly, and in descending order according to the closeness centrality value of the vertex
         thirdly;
12     return N_i^sort;
13 end
```

In Algorithm 2, each node of the selected central node sequence is used as the root node, and the breadth-first search (BFS) is used to traverse its neighborhoods corresponding to the node with an increasing distance from the root node and add these nodes to the set N_i. In lines 4 to 7 of Algorithm 2, the first-order neighbors $N_1(v)$ that have not been traversed of the node in each while loop are explored. If the number of the neighbor node set N_i is less than k, the first-order neighbors of these nodes newly added to N_i are collected until not less than k vertices in N_i or all neighbor nodes are traversed. Note that $|L_i|$ denotes the number of vertices that are not traversed in the set L_i and the values of $|N_i|$ and k may not be equal. In lines 8 to 10 of Algorithm 2, if the number of nodes finally selected is less than k, it is supplemented with empty nodes. These neighbor nodes and the root node together constitute the neighborhood set of the central node.

To obtain the normalized neighborhood graph, the total number of nodes in the neighborhood graph containing the central node is limited to k, so the nodes in the above neighborhood set need to be sorted, as shown in line 11 of Algorithm 2. Specifically, in the process of normalizing the neighborhood graph, the nodes in the neighborhood set are sorted in ascending order according to the shortest path distance from the root node firstly, like PSCN. Afterwards, when the length of the shortest path from the nodes in the neighbor set to the root node is the same, unlike the way that PSCN uses Nauty to break the same order, our method sorts the nodes in descending order according to the value of the degree centrality of the vertex. When the degree of the node appears the same again, it is sorted in descending order according to the closeness centrality value of the vertex. Finally, the top k ordered vertices (including the central node) are selected as the neighborhoods of the root node. The neighborhood graphs of all the central nodes are then normalized into a matrix of $k \times w$. For a graph G, the constructed w neighborhood graphs can represent the entire graph.

4.3. Selection of node features

After the constructed neighborhood graph is normalized, the feature extraction of the node considers both local and global aspects in this paper. When selecting the local features for each node of the neighborhood graph, the degree of the node (d), the minimum degree of its neighbor nodes ($dn_{\min}$), the maximum degree of its neighbor nodes ($dn_{\max}$), the average degree of its neighbor nodes (dn_{ave}) and the standard deviation of its neighbor nodes (dn_{std}) are used, which are all calculated in the range of the graph G. In other words, the above five vertex features include the degree information of the vertex itself in the neighborhood graph and its first-order neighbor. Therefore, these local features can well characterize the relationship between a node and its first-order neighborhoods, as well as its first-order neighbors and the second-order neighbors. However, these are only node features in the local scope of the graph G, not in the entire graph.

In order to extract the global characteristics of the nodes, we consider the CC values of the nodes and the spectral features(SF) of the vertices obtained by Laplacian embedding in the scope of the graph G. The reasons for choosing these two features are that the CC is based on the global properties of the graph as described in Section 3.2 and the SF involves global information of the graph for the partition [36]. In detail, the symmetric normalized Laplacian matrix of the graph G is first calculated according to the formula (2) in Section 3.1, and the feature vectors corresponding to the m smallest positive eigenvalues of the matrix L^{sym} in ascending order are selected as the SF of the reduced-dimensional vertices, where each vertex in the neighborhood graph corresponds to an m-dimensional feature. If the number of nodes in the graph is less than m, right zero padding is used to obtain the corresponding eigenvectors.

Finally, combining the local and global features of the vertices, each vertex has $a_v = 6 + m$ attributes, where m denotes the embedding dimension of the spectral characteristics. Therefore, it can be expressed as a tensor of $k \times w \times a_v$ for a graph G, where w denotes the size of the chosen central node sequence, k denotes the number of nodes in the neighbor graph, and a_v represents the number of vertex feature channels.

4.4. Architecture of spatial convolutional neural networks

After the above steps, each graph in the graph set corresponds to a fixed size tensor, so the input tensor of the convolutional neural network is $k \times w \times a_v$. In other words, it can be regarded as a $k \times w$ size picture with a_v feature channels. This section describes the architecture of spatial convolutional neural networks(SCNN) used in detail. A simple CNN structure is used as our baseline,

which has two convolutional layers and a dense layer and a softmax layer, as shown in Fig. 1. To retain more spatial information, no pooling layer is used in Fig. 1. We call this architecture SCNN-baseline. In Fig. 1, the convolution kernel sizes of the first and second convolution layers are $k \times 1$ and 1×10, respectively, and the stride is 1. For the first convolutional layer of SCNN-baseline in Fig. 1, M convolution filters are used to generate $w \times 1 \times M$ multi-channel feature representations. The second convolutional layer of SCNN-baseline in Fig. 1 has N output channels and the output is $(w-10+1) \times 1 \times N$. The dense layer has 64 or 128 units with the ReLU activation function in Fig. 1. Finally, the softmax layer is applied to performing the graph classification task of SCNN-baseline in Fig. 1.

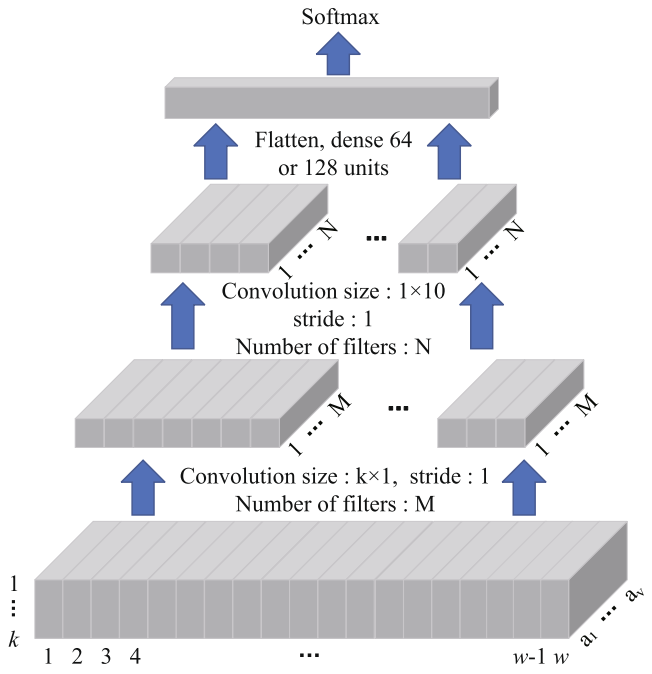

Fig. 1. SCNN-baseline structure, where the input tensor of the structure is $k \times w \times a_v$. The architecture includes two convolutional layers, a dense layer and a softmax layer.

Next, we modify the above structure based on the classification of the neighborhood graph, named SCNN-subgraph. The structure of SCNN-subgraph is shown in Fig. 2. The first layer of the structure is a convolutional layer, the same as the first convolutional layer of the SCNN-baseline in Fig. 1, followed by a channel concatenation layer in Fig. 2. To represent the features of each neighborhood graph, we concatenate different feature channels into a two-dimensional matrix of $w \times (1 \times M)$, where the matrix represents the entire input graph, and each column of the matrix represents the vector representation of the corresponding neighborhood graph in Fig. 2. The softmax function is then used on each column of the two-dimensional matrix to obtain the classification probability of each neighborhood graph in Fig. 2. Finally, we add the output probabilities of the softmax function separately based on the class label, and then take the maximum value as the classification result of the entire graph of SCNN-subgraph in Fig. 2.

4.5. Time complexity of the proposed approach

In the proposed graph classification algorithm, when creating neighborhood graphs, we need to select the central node sequence first, and the time complexity is $O(w)$, where w is the number of central nodes. For a data set with N graphs, the time complexity of this stage is $O(Nw)$. After obtaining the central node sequence, using each node as the root node, the breadth-first search is used to construct a neighborhood graph, and the time complexity of breadth-first search is $O(n)$. Therefore, the time complexity of constructing w neighborhood graphs with node k is $O(wk)$, where k is the size of the receptive field.

5. Experiment

In this section, we conduct experiments on two types of datasets in the real world, i.e., bioinformatics and social network datasets, to evaluate the performance of the proposed approach. Firstly, the proposed method is compared with the baseline methods based on graph kernel and the baseline methods based on deep learning in terms of classification accuracy. Secondly, the impact of different node features on classification accuracy and the different embedding dimensions of spectral features on classification performance are analyzed.

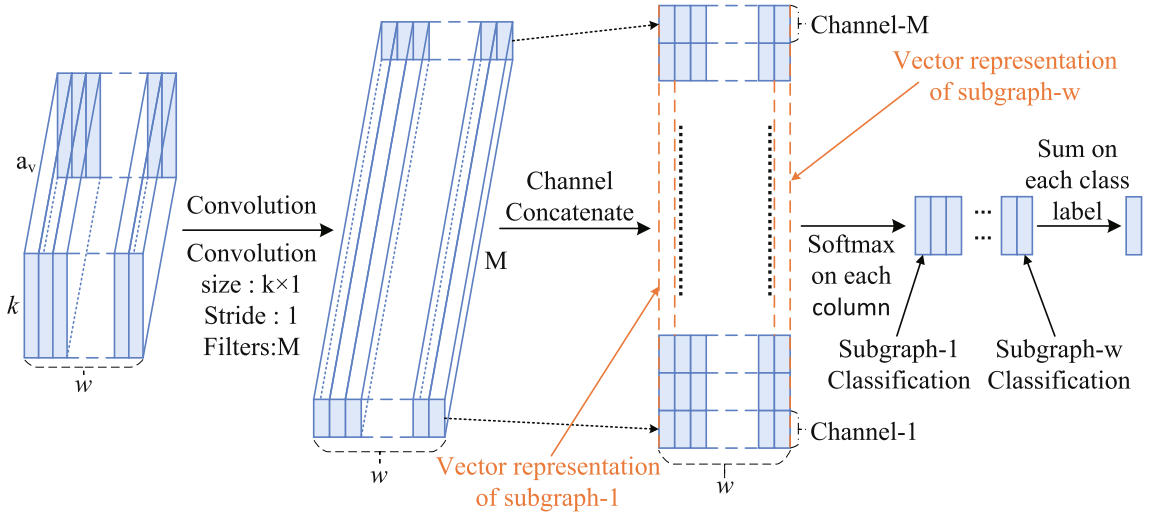

Fig. 2. The structure of SCNN-subgraph, where the input tensor of the structure is $k \times w \times a_v$. A channel concatenation layer is added after a convolutional layer, then each column of the two-dimensional matrix represents the vector representation of the corresponding neighborhood graph.

5.1. Data sets

The data sets used in this article are: MUTAG, PTC, PROTEINS, D&D, ENZYMES, IMDB-B, IMDB-M and COLLAB. The profiles of the above eight data sets are described in Table 2.

Bioinformatics datasets. MUTAG [25] is a data set containing 188 mutagenic aromatic and heteroaromatic nitro compounds. According to whether they have mutagenic effects on Gram-negative bacilli, the compounds can be divided into two categories. PTC [26] is a dataset of 344 organic molecules and is classified into two categories based on carcinogenicity in male and female mice and rats. PROTEINS [32] contains 1,113 proteins, of which the nodes are secondary structural elements (SSE). If two nodes are adjacent in the amino acid sequence or in 3D space, the two nodes are connected by edges. The prediction task is to divide the protein structure into enzyme and non-enzyme. D&D [17] consists of 1,178 protein structures, and each protein structure is represented by a graph structure. The vertices in the graph represent amino acids. If the interval between two nodes is less than 6 angstroms, there is an edge connection between the two nodes. The prediction task is also to divide the protein structure into enzyme and non-enzyme. ENZYMES is a balanced dataset of 600 protein tertiary structures obtained from the BRENDA database, with 3 discrete node labels [29]. ENZYMES comes with the task of classifying the enzymes to one out of 6 EC top-level classes.

Social network datasets. IMDB-B [29] is a movie collaboration dataset. Each graph structure represents the actor's ego-network. The nodes correspond to the actor. The edges indicate that the actor has a cooperative relationship in the same movie. The genre is used to mark the ego-network. The graphs in the dataset are divided into two categories according to the types of movies: action and romance. The goal is to simply determine which genre an ego-network graph belongs to. IMDB-M [29] is similar to IMDB-B. The graphs in this dataset are classified into three categories based on the types of movies: comedy, romance and sci-fi. COLLAB [29] is a scientific collaboration dataset of 5,000 graphs, derived from three public collaboration data sets, namely high-energy physics, condensed matter physics and astrophysics. Each graph in this dataset represents an ego-network of different researchers from each area. The goal is to categorize each graph into the subfield to which the corresponding researcher belongs.

5.2. Baselines and experiment settings

In this section, the baseline methods that are compared with the proposed method are introduced first, including some representative graph kernel methods and several graph classification methods based on deep learning, and then the experimental settings are described.

Baselines. We use the Shortest-Path Kernel (SP) [30], the Random Walk kernel (RW) [31], the Graphlet Kernel (GK) [32], Weisfeiler-Lehman subtree kernel (WL) [33] and Deep Graph Kernels (DGK) [29] as the graph kernel baselines. Due to the large number of graph classification methods based on deep learning, we compare our model with the four recently proposed methods that are closely related to ours. The four deep learning-based baseline approaches include:

(1) PSCN [25]: It is the closest to our approach, which defines the neighborhood of each node and uses a canonical vertex ordering, and then the convolutions are applied on the linear representation of the node embeddings.
(2) DGCNN(Deep Graph Convolution Neural Network) [26]: Extracting vertices characteristics using graph convolution, then performing a soft pooling, and finally using 1D convolution on the ordered nodes.
(3) DCNN(Diffusion-CNN) [24]: Extracting multi-scale substructure characteristics with propagation graph convolution.
(4) ECC(Edge-conditioned convolutional network) [37]: The edge information is merged into the graph convolutional network model and the pooling operation is performed using the graph coarsening method.

Experiment settings. For a fair comparison, the 10-fold cross-validation was used for all experiments. For all graph data sets, the data set is randomly divided into 90% training set and 10% test set. For the RW kernel, we chose the decay parameters from $\left\{10^{-6}, 10^{-5}, \cdots, 10^{-1}\right\}$. The subtree height parameter of WL is set to 10 following the same settings as in [38]. For PSCN, we adopted the average classification accuracy and standard deviation when $k = 10$ from the original paper [25]. For other baselines, we report the best results from previous work [38–40] or original papers [26], since all the experimental settings are the same as ours. Some results are not provided because either the source code is not available or not reported in the previous literature.

For our SCNN-baseline and SCNN-subgraph, we perform a grid search on the PTC dataset to discover the optimal hyper-parameters for the model and use the Adam optimizer. The learning rate is chosen from $\{0.0001, 0.001, 0.005\}$ and the number of epochs is selected from $\{100, 200, 300\}$. When experimenting on other datasets, we use the same hyper-parameters found in the PTC dataset to reduce the computational cost. For each data set, the number of central vertices w and the embedding dimension m of the SF are set to the average number of nodes in the data set. For MUTAG, PTC, PROTEINS, ENZYMES, IMDB-B, and IMDB-M data sets, the count of vertices k in the neighbor graph constructed for each central node is set to 10, and for D&D and COLLAB is set to 20, according to the average number of vertices in the data set, as in [39].

For fairness, the SCNN-baseline uses the same single network structure as described in Section 4.4, as in PSCN. The output channels of the first and the second convolutional layers are 64 and 32, respectively. The dense layer has 128 units with the dropout rate

Table 2
Dataset Description. #graphs represents the total number of samples in the dataset, #classes represents the number of categories in the dataset, Avg # nodes represents the average number of nodes in the dataset, Avg # edges represents the average number of edges in the dataset, and # node labels represents the number of node labels in the dataset.

Data Sets	#graphs	#classes	Avg # nodes	Avg # edges	# node labels
MUTAG	188	2	17.93	19.79	7
PTC	344	2	14.29	14.69	19
PROTEINS	1113	2	39.06	72.82	3
D&D	1178	2	284.32	715.66	82
ENZYMES	600	6	32.63	62.14	3
IMDB-B	1000	2	19.77	96.53	–
IMDB-M	1500	3	13.00	65.94	–
COLLAB	5000	3	74.49	2457.78	–

0.5. The nonlinear activation function used in convolution layers and dense layers is Relu. For the SCNN-subgraph, only a convolutional layer is used with 64 output channels. Note that the label information of the node is not utilized in our method.

5.3. Comparative analysis of classification results

In this section, we compare the proposed method with some representative graph kernels and several deep learning-based methods for graph classification.

The average classification accuracy and standard deviation of the proposed method and the graph kernel baseline methods on eight data sets are shown in Table 3. Compared with the baseline methods based on deep learning, the average classification accuracy and standard deviation on eight data sets are shown in Table 4. Bold in Tables 3 and 4 indicates the method with the best performance on each data set.

Compared with the representative graph kernel baseline methods. It can be seen from Table 3 that the classification results of SCNN-baseline and SCNN-subgraph proposed in this paper on MUTAG, PTC, PROTEINS, D&D, ENZYMES and IMDB-B are better than the graph kernel methods. In addition, compared with the five graph kernel baselines, the classification performance of SCNN-baseline on all data sets has always been in the top 2, and the classification accuracy of SCNN-subgraph is slightly improved over SCNN-baseline on senven data sets. On IMDB-M and COLLAB datasets, the SCNN-subgraph model has better results than the three graph kernel methods compared to the four graph kernel baselines that reported the classification results. In short, compared with the graph kernel-based methods, the method proposed in this paper can use the node information and structure information of the graph more effectively, while the graph kernel methods mainly uses local substructure information and ignores the global information of the graph.

Compared with the deep learning-based approaches for graph classification. From Table 4, it can be seen that SCNN-subgraph shows the highest classification accuracy on MUTAG, PTC, PROTEINS, D&D, ENZYMES, IMDB-B and COLLAB, and the classification accuracy has increased by 1 to 2 percent on most data sets. The SCNN-baseline achieves slightly lower classification accuracy than that obtained by the SCNN-subgraph. Both SCNN-baseline and SCNN-subgraph outperform ECC and DCNN in each case, which indicates that simple node feature summation is invalid and will lead to the loss of many individual nodes and global topology information of the graph, thereby reducing the classification accuracy. In addition, it can be seen from Table 4 that SCNN-baseline shows better classification performance on all data sets compared to PSCN. The reason is that the method proposed in this paper adopts a comprehensive evaluation strategy of multiple indicators to determine the significant nodes (ie, the central node sequence) in the graph. However, PSCN applies the WL algorithm to sort the vertices according to their structural characteristics and relies on external software to forcibly break the order of vertices with the same label. This cannot accurately identify important nodes and leads to the loss of important node information, which affects the subsequent classification results.

Different from the method proposed in this paper, the DGCNN model uses graph convolution to extract vertex characteristics, sorts nodes according to WL color, and then sorts and selects the vertices based on these features in the pooling stage. However, this method only focuses on local vertex features and ignores the global features of the graph such as topology information, resulting in the loss of significant information. In contrast, the advantage of the model proposed in this paper is that it can fully extract the characteristics of the graph from both local and global aspects and make full use of them to normalize the graph, and then input it as a node attribute to the subsequent convolution architecture.

Comparing SCNN-baseline and SCNN-subgraph, we find that the latter is superior to the former on six data sets, which indicates that the classification results of subgraphs constructed by vital nodes will have a great influence on the classification of the entire graph. At the same time, it also indirectly proves the significance of introducing the channel connection layer.

5.4. Analysis of different node features and spectral feature embedding dimensions

In this section, the impact of the different local and global node characteristics selected in Section 4.3 on the classification results is analyzd firstly. For the sake of simplicity, the five local node features ($d, dn_{\min}, dn_{\max}, dn_{\text{ave}}$ and dn_{std}) related to the degree information are classified into one category and then the contributions of

Table 3
Comparison of classification accuracy (± standard deviation) of the Shortest-Path Kernel (SP), the Random Walk kernel (RW), the Graphlet Kernel (GK), Weisfeiler-Lehman subtree kernel (WL) and Deep Graph Kernels (DGK) on bioinformatics and social network datasets.

Methods	MUTAG	PTC	PROTEINS	D&D	ENZYMES	IMDB-B	IMDB-M	COLLAB
SP	85.79 ± 2.51	58.24 ± 2.44	75.07 ± 0.54	–	29.00 ± 0.48	71.26 ± 1.04	–	–
RW	83.72 ± 1.50	57.85 ± 1.30	74.22 ± 0.42	–	22.37 ± 0.35	64.54 ± 1.22	34.54 ± 0.76	69.01 ± 0.09
GK	81.58 ± 2.11	57.26 ± 1.41	71.67 ± 0.55	78.45 ± 0.26	24.87 ± 0.22	65.87 ± 0.98	43.89 ± 0.38	72.84 ± 0.28
WL	82.88 ± 0.57	–	73.52 ± 0.43	79.78 ± 0.36	52.75 ± 0.44	71.88 ± 0.77	**49.50 ± 0.49**	**76.56 ± 0.30**
DGK	82.66 ± 1.45	57.32 ± 1.13	71.68 ± 0.50	–	53.40 ± 0.90	66.96 ± 0.56	44.55 ± 0.52	73.09 ± 0.25
SCNN-baseline	89.46 ± 2.17	62.73 ± 3.09	75.89 ± 1.54	**80.24 ± 0.56**	53.98 ± 0.84	72.40 ± 1.35	45.96 ± 1.29	74.22 ± 1.38
SCNN-subgraph	**90.08 ± 2.01**	**63.47 ± 2.65**	**76.51 ± 1.37**	80.15 ± 0.78	**54.55 ± 0.62**	**73.24 ± 0.96**	46.82 ± 1.83	74.93 ± 0.85

Table 4
Classification results (± standard deviation) on bioinformatics and social network datasets compared with baseline methods based on deep learning, including PSCN, DGCNN, DCNN and ECC.

Methods	MUTAG	PTC	PROTEINS	D&D	ENZYMES	IMDB-B	IMDB-M	COLLAB
PSCN	88.95 ± 4.37	62.29 ± 5.68	75.00 ± 2.51	76.27 ± 2.64	–	71.00 ± 2.29	45.23 ± 2.84	72.60 ± 2.15
DGCNN	85.83 ± 1.66	58.59 ± 2.47	75.54 ± 0.94	79.37 ± 0.94	51.00 ± 7.29	70.03 ± 0.86	**47.83 ± 0.85**	73.76 ± 0.49
DCNN	66.98	56.60 ± 2.89	61.29 ± 1.60	58.09 ± 0.53	42.44 ± 1.76	49.06 ± 1.37	33.49 ± 1.42	52.11 ± 0.71
ECC	89.44	–	72.65	74.10	45.67	–	–	67.79
SCNN-baseline	89.46 ± 2.17	62.73 ± 3.09	75.89 ± 1.54	**80.24 ± 0.56**	53.98 ± 0.84	72.40 ± 1.35	45.96 ± 1.29	74.22 ± 1.38
SCNN-subgraph	**90.08 ± 2.01**	**63.47 ± 2.65**	**76.51 ± 1.37**	80.15 ± 0.78	**54.55 ± 0.62**	**73.24 ± 0.96**	46.82 ± 1.83	**74.93 ± 0.85**

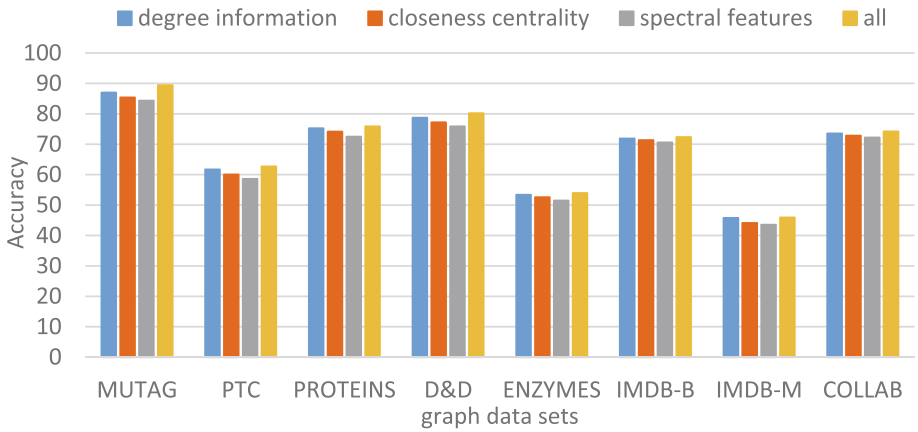

Fig. 3. The classification results obtained by SCNN-baseline model using different node features on different data sets. The selected input node features include degree information, closeness centrality and spectral features. And "all" means to use all three features above.

Table 5
Classification accuracy with different embedding dimension m of spectral features for SCNN-subgraph on bioinformatics and social network datasets.

m	MUTAG	PTC	PROTEINS	D&D	ENZYMES	IMDB-B	IMDB-M	COLLAB
5	88.91	62.08	73.62	77.49	51.78	71.96	45.72	73.25
10	89.62	61.56	75.34	80.19	52.37	72.43	46.68	73.84
20	90.08	63.47	76.05	80.23	53.86	73.24	46.75	74.38
40	90.08	63.47	76.47	79.71	54.21	73.07	46.75	74.93

this type of local features and two global features to the classification result are compared.

We test the SCNN-baseline model with degree information, closeness centrality and spectral features as the only characteristic separately on all datasets. All experiments use the same graph processing steps, including the selection of central node sequences and the construction and normalization of neighborhood graphs. Fig. 3 shows the classification accuracy obtained by SCNN-baseline using different node features on different data sets. It can be seen from Fig. 3 that degree information is crucial among the three features, and the best performance is achieved among the three types of characteristics. In addition, the classification accuracy is further improved when all of these vertex characteristics are utilized, which indicates that the selected local and global characteristics can affect the structural information of the graph from multiple aspects. In summary, all of these features selected in Section 4.3 contribute to improving accuracy of graph classification.

Next, the effects of different embedding dimensions m of spectral features on the experimental results is analyzed. Specifically, we fixed the value of k to 10 or 20 according to the average count of nodes in the data set, and m is chosen from $\{5, 10, 20, 40\}$ to analyze the influence of this parameter on the experimental results.

Table 5 shows the effects of different embedding dimensions m of spectral features on the classification results of the SCNN-subgraph and reports the average classification accuracy of different m on all data sets. As shown in Table 5, competitive results are obtained when $m = 5$, and the results provided when $m = 40$ are approximately equal to the result obtained when m is set to the average number of nodes in the data set. In addition, from the results we can see that when the value of m is close to the average number of nodes in the data set, as m continues to increase, the accuracy tends to be constant or slightly decreased. The reason is that for most graphs, a larger m will bring about more extra zero-padding operations, which will not improve the classification accuracy. At the same time, if m is too large, noise may be introduced to make the accuracy slightly decrease.

6. Conclusion

In this article, a new spatial convolutional neural network model based on the structural characteristics of significant vertices is proposed. Our method selects the central vertices to construct the neighborhood graphs according to the multiple index comprehensive weighting method and chooses multiple vertex characteristics from the local and global aspects to extract more information about the graph. In addition, the channel connection layer is proposed, so that the classification result of the entire graph can be determined according to the classification of neighborhood graphs. The experimental results of SCNN-baseline and SCNN-subgraph on two types of real data sets suggest that our method can improve the performance of graph classification compared with the traditional classic graph kernel approaches and some existing deep learning-based models.

Due to the subjectivity of artificially selecting node importance evaluation indicators, the subsequent research direction can further improve the comprehensive weighting method to more objectively and accurately evaluate vital nodes in the graph. In addition, the graph classification method proposed in this paper is only applicable to the static graph structure where the nodes and edges will not change. However, real-world networks (such as social networks) are constantly changing with time, that is, constantly adding or deleting nodes or edges. Therefore, the follow-up work can consider how to model the dynamic graph, so as to predict the graph after the change. At the same time, it is necessary to improve the scalability of the proposed algorithm on real large-scale data sets (for example, with billions of nodes and edges).

CRediT authorship contribution statement

Tinghuai Ma: Supervision, Conceptualization, Funding acquisition. **Hongmei Wang:** Methodology, Software, Writing - original draft. **Lejun Zhang:** Conceptualization, Visualization, Validation,

Writing - review & editing. **Yuan Tian:** Writing - review & editing. **Najla Al-Nabhan:** Writing - review & editing, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported in part by the National Science Foundation of China (No. U1736105). The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through Research Group No. RG-1441-331.

References

[1] T. Ma, J. Jia, Y. Xue, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Protection of location privacy for moving kNN queries in social networks, Appl. Soft Comput. 66 (2018) 525–532.
[2] J. Kim, M. Hastak, Social network analysis: characteristics of online social networks after a disaster, Int. J. Inf. Manage. 38 (1) (2018) 86–96.
[3] T. Ma, S. Yu, J. Cao, Y. Tian, M. Al-Rodhaan, Infmatch: finding isomorphism subgraph on a big target graph based on the importance of vertex, Physica A 527 (2019) 121278.
[4] T. Ma, Q. Liu, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, LGIEM: global and local node influence based community detection, Future Gener. Comput. Syst. 105 (2020) 533–546.
[5] T. Ma, W. Shao, Y. Hao, J. Cao, Graph classification based on graph set reconstruction and graph kernel feature reduction, Neurocomputing 296 (2018) 33–45.
[6] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, Adv. Neural Inform. Process. Syst. (2015) 2224–2232.
[7] Z. Sun, H. Huo, J. Huan, J.S. Vitter, Feature reduction based on semantic similarity for graph classification, Neurocomputinghttps://doi.org/10.1016/j.neucom.2020.02.047.
[8] N. Jin, C. Young, W. Wang, GAIA: graph classification using evolutionary computation, ACM SIGMOD International Conference on Management of Data (2010) 879–890.
[9] X. Li, Q. Yang, X. Lin, S. Wu, Itrust: interpersonal trust measurements from social interactions, IEEE/ACM Trans. Networking 30 (4) (2016) 54–58.
[10] H.-A. Bay-Ahmed, A.-O. Boudraa, D. Dare-Emzivat, A joint spectral similarity measure for graphs classification, Pattern Recogn. Lett. 120 (2019) 1–7.
[11] S.V.N. Vishwanathan, N.N. Schraudolph, R. Kondor, K.M. Borgwardt, Graph kernels, J. Mach. Learn. Res. 11 (Apr) (2010) 1201–1242.
[12] D. Han, Q. Liu, W. Fan, A new image classification method using CNN transfer learning and web data augmentation, Expert Syst. Appl. 95 (2018) 43–56.
[13] S. Zhi, Y. Liu, X. Li, Y. Guo, Toward real-time 3D object recognition: a lightweight volumetric CNN framework using multitask learning, Comput. Graph. 71 (2018) 199–207.
[14] T. Ma, Y. Zhao, H. Zhou, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Natural disaster topic extraction in sina microblogging based on graph analysis, Expert Syst. Appl. 115 (2019) 346–355.
[15] T. Ma, H. Rong, Y. Hao, J. Cao, Y. Tian, M. Al-Rodhaan, A novel sentiment polarity detection framework for chinese, IEEE Trans. Affective Comput. https://doi.org/10.1109/TAFFC.2019.2932061.
[16] H. Rong, T. Ma, J. Cao, Y. Tian, A. Al-Dhelaan, M. Al-Rodhaan, Deep rolling: a novel emotion prediction model for a multi-participant communication context, Inf. Sci. 488 (2019) 158–180.
[17] Y. Xie, C. Yao, M. Gong, C. Chen, A. Qin, Graph convolutional networks with multi-level coarsening for graph classification, Knowl.-Based Syst. 105578 (2020).
[18] X. Fan, M. Gong, Y. Xie, F. Jiang, H. Li, Structured self-attention architecture for graph-level representation learning, Pattern Recogn. 100 (2020) 107084.
[19] F. Manessi, A. Rozza, M. Manzo, Dynamic graph convolutional networks, Pattern Recogn. 97 (2020) 107000.
[20] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, 2013. arXiv:1312.6203.
[21] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inform. Process. Syst. (2016) 3844–3852.
[22] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016. arXiv:1609.02907.
[23] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inform. Process. Syst. (2017) 1024–1034.
[24] J. Atwood, D. Towsley, Diffusion-convolutional neural networks, Adv. Neural Inform. Process. Syst. (2016) 1993–2001.
[25] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in, in: Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 2014–2023.
[26] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018, pp. 4438–4445.
[27] D. Das, C. George Lee, Sample-to-sample correspondence for unsupervised domain adaptation, Eng. Appl. Artif. Intell. 73 (2018) 80–91.
[28] D. Das, C. George Lee, Unsupervised domain adaptation using regularized hyper-graph matching, in: 25th IEEE International Conference on Image Processing (ICIP), 2018, https://doi.org/10.1109/icip.2018.8451152.
[29] P. Yanardag, S.V.N. Vishwanathan, Deep graph kernels, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2015) 1365–1374.
[30] K.M. Borgwardt, H.-P. Kriegel, Shortest-path kernels on graphs, in: Fifth IEEE international conference on data mining, IEEE, 2005, pp. 74–81.
[31] T. Gärtner, P. Flach, S. Wrobel, On graph kernels: Hardness results and efficient alternatives, in: Learning theory and kernel machines, Springer, 2003, pp. 129–143.
[32] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, K. Borgwardt, Efficient graphlet kernels for large graph comparison, in, in: Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, 2009, pp. 488–495.
[33] N. Shervashidze, P. Schweitzer, E.J.V. Leeuwen, K. Mehlhorn, K.M. Borgwardt, Weisfeiler-lehman graph kernels, J. Mach. Learn. Res. 12 (Sep) (2011) 2539–2561.
[34] B. Knueven, J. Ostrowski, S. Pokutta, Detecting almost symmetries of graphs, Math. Program. Comput. 10 (2) (2018) 143–185.
[35] T. Bian, J. Hu, Y. Deng, Identifying influential nodes in complex networks based on AHP, Physica A 479 (2017) 422–436.
[36] A. Barvinok, P. Soberón, Computing the partition function for graph homomorphisms with multiplicities, J. Combinatorial Theory, Series A 137 (2016) 1–26.
[37] M. Simonovsky, N. Komodakis, Dynamic edge-conditioned filters in convolutional neural networks on graphs, in, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 3693–3702.
[38] L. Bai, L. Cui, Y. Jiao, L. Rossi, E.R. Hancock, Learning backtrackless aligned-spatial graph convolutional networks for graph classification, 2019. arXiv:1904.04238.
[39] J. Jiang, C. Xu, Z. Cui, T. Zhang, W. Zheng, J. Yang, Walk-steered convolution for graph classification, IEEE Trans. Neural Networks Learn. Syst. (2019) 1–14, https://doi.org/10.1109/tnnls.2019.2956095.
[40] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, Adv. Neural Inform. Process. Syst. (2018) 4800–4810.

Tinghuai Ma is a professor in Computer Sciences at Nanjing University of Information Science & Technology, China. He received his Bachelor (HUST, China, 1997), Master (HUST, China, 2000), PhD (Chinese Academy of Science, 2003) and was Post-doctoral associate (AJOU University, 2004). From Nov.2007 to Jul. 2008, he visited Chinese Meteorology Administration. From Feb.2009 to Aug. 2009, he was a visiting professor in Ubiquitous computing Lab, Kyung Hee University. His research interests are data mining, Cloud Computing, ubiquitous computing, privacy preserving etc. He has published more than 100 journal/-conference papers.

Hongmei Wang received her Bachelor degree in Computer Science & Technology from Nanjing University of Information Science & Technology, China in 2017. Currently, she is a candidate of PhD. in Nanjing University of Information Science & Technology. Her research interest is community detection.

Lejun Zhang, received his M.S. degree in computer science and technology in Harbin Institute of Technology and the Ph.D. degrees in computer science and technology at Harbin Engineering University, now he is a professor at Yangzhou University. His research interests include computer network, social network analysis, dynamic network analysis and information security.

Yuan Tian has received her master and Ph.D degree from KyungHee University and she is currently working as Assistant Professor at College of Computer and Information Sciences, King Saud University, Kingdom of Saudi Arabia. She is member of technical committees of several international conferences. In addition, she is an active reviewer of many international journals. Her research interests are broadly divided into privacy and security, which are related to cloud.

Najla Al-Nabhan has received her BS in Computer Applications (Hon) and MS in Computer Science both from The George Washington University on 2005 and 2008 respectively. In 2013, she received her Ph.D. in Computer Science from King Saud University. She is currently working as the Vic Assistant professor at Computer Science Department, College of Computer and Information Sciences(CCIS), King Saud University(KSU), Riyadh, Saudi Arabia. Her current research interest includes: Wireless Sensor Networks, Multimedia Sensor networks, Cognitive Networks, and Network Security.