SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2022, Vol. 65 152104:1–152104:15
https://doi.org/10.1007/s11432-020-3318-5

Deepwalk-aware graph convolutional networks

Taisong JIN[1], Huaqiang DAI[2], Liujuan CAO[1*], Baochang ZHANG[3], Feiyue HUANG[4], Yue GAO[5] & Rongrong JI[2]

[1]*Media Analytics and Computing Lab, Department of Computer Science and Technology, School of Informatics, Xiamen University, Xiamen* 361005*, China;*
[2]*Media Analytics and Computing Lab, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen* 361005*, China;*
[3]*School of Automation Science and Electrical Engineering, Beihang University, Beijing* 100083*, China;*
[4]*Tencent Youtu Lab, Shanghai* 200233*, China;*
[5]*School of Software, Tsinghua University, Beijing* 100084*, China*

Received 7 January 2021/Revised 7 May 2021/Accepted 12 July 2021/Published online 15 April 2022

Abstract Graph convolutional networks (GCNs) provide a promising way to extract the useful information from graph-structured data. Most of the existing GCNs methods usually focus on local neighborhood information based on specific convolution operations, and ignore the global structure of the input data. To extract the latent representation for the graph-structured data more effectively, we introduce a deepwalk strategy into GCNs to efficiently explore the global graph information. This strategy can complement the local neighborhood information of a graph, resulting in the more robust representation for the graph data. The fusion of the local neighboring and global structured information of a graph can further facilitate deep feature learning at the output layer of GCNs for node classification. Experimental results show that the proposed model has achieved state-of-the-art results on three benchmark datasets including Cora, Citeseer, and Pubmed citation networks.

Keywords graph, convolutional networks, global information, fusion, node classification

Citation Jin T S, Dai H Q, Cao L J, et al. Deepwalk-aware graph convolutional networks. Sci China Inf Sci, 2022, 65(5): 152104, https://doi.org/10.1007/s11432-020-3318-5

1 Introduction

The increasing use of the graph-structured data in real-world applications has attracted much attention in the machine learning community. Due to promising performance, graph convolutional networks (GCNs) [1] have become popular to handle the graph-structured data. The existing GCNs adopt the convolutional operation to propagate the neighboring information of a graph for learning the deep features, which usually focus on capturing the local structure of a graph. The GCNs models have been applied to solving various problems such as image classification [2], object detection [3] and semantic segmentation [4].

The key challenge of graph convolutional networks is to define the suitable graph convolutions to aggregate information for capturing the neighboring structures of a graph. Various GCNs have been proposed, which can typically be divided into the spectral and spatial approaches. Spectral approaches [5] are based on the spectral graph theory to define the convolutional operation via parameterized filters [6]. ChebNet [7] approximates the filters by means of a Chebyshev expansion of the graph Laplacian to enhance the computation efficiency of graph filters. Later, GCN [8] simplifies the previous methods by truncating the Chebyshev polynomial to the first-order neighborhood. GDC [9] uses generalized graph diffusion to alleviate the problem of noisy and meaningless edges in actual graphs. Spatial approaches learn the convolutional operation using the spatial information of a graph via relations between nodes. Graph attention network (GAT) [10] adopts attention mechanisms [11] to learn the relative weights

* Corresponding author (email: caoliujuan@xmu.edu.cn)

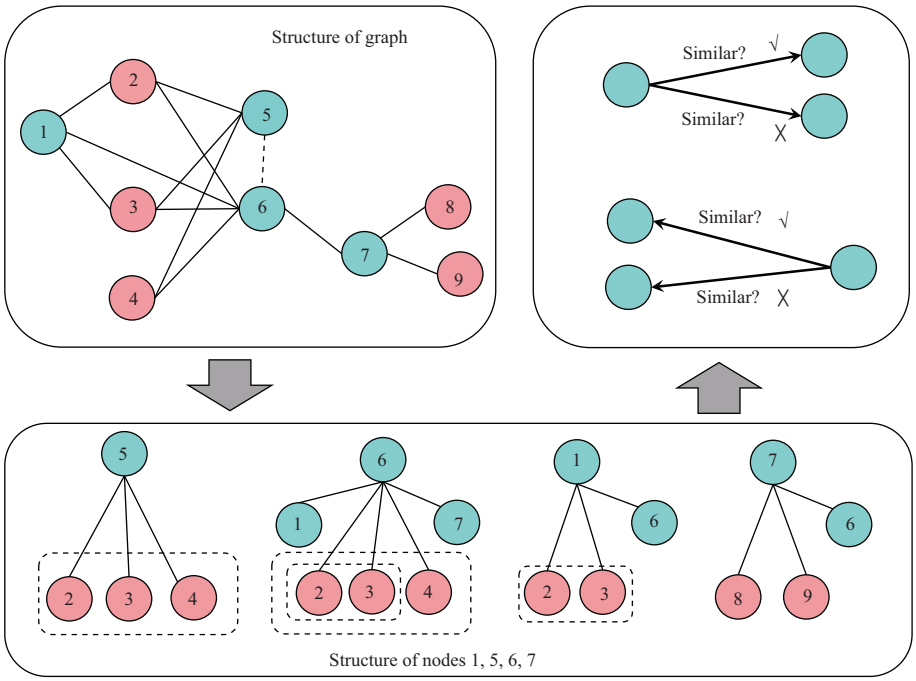

Figure 1 (Color online) The proposed method learns the information about global consistency. Because we focus on nodes 1, 5–7, different colors are used to distinguish them from the other nodes. As shown in the structure of a graph, nodes 6 and 1 are connected, and nodes 6 and 7 are also connected. The existing models only consider the local consistency, where nearby data points may have the same label. But we can see that nodes 5 and 6 have more similar structures (common nodes 2–4) and are more likely to have the same label. However, nodes 5 and 6 are not adjacent. Nodes 5 and 6 show that the structural similarity can better reflect the relationship between the two nodes, which is more convincing and ignored by existing models. Furthermore, we can see that nodes 1 and 7 are connected to node 6, but from a structural point of view, nodes 1 and 6 have more similar neighbors, and their adjacency relationship should be stronger.

between the central node and its neighbors. Mixture model network (MoNet) [12] is a general spatial-based approach, which constructs a convolutional function for a graph. Cluster-GCN [13] is designed to handle a large graph by performing graph convolutions on the subgraphs sampled by a graph clustering algorithm.

Most of the existing studies [5–13] only consider the local neighborhood information of a graph for data embedding. However, the global structured information of a graph that can actually complement the local neighboring information of a graph is ignored. In fact, the local neighborhood information of a graph (local consistency) assumes that adjacent data samples are tended to share the same label. However, the global consistency (structure) of a graph can ensure that data samples that occur in similar contexts tend to have the same label. Specifically, for the nodes that are far away, the global consistency can leverage more detailed structural information of a graph, thereby expanding the receptive field of the nodes and making the better global predictions (see Figure 1). Since the previous GCNs [5–13] usually only capture the nearby node relationships based on a small number of layers, which ignores the global structure of a graph, the limited performance enhancement is derived. To enhance the discriminative ability of deep features for the graph-structured data, it is crucial to leverage the higher-level semantic information such as the global structural similarity to determine whether two nodes in a graph are related.

Note that stacking more GCNs layers could increase the receptive field of the nodes, which is useful to aggregate more information from the global structure of a graph. Some GCNs such as GAT [10] and dual graph convolutional networks (DualGCN) [14] have certain capabilities to capture the global information of a graph. However, the neighborhood aggregation of a graph is essentially a type of Laplacian smoothing and stacking many layers may result in the problem of over-smoothing [15]. Furthermore, the number of GCNs layers is the farthest distance that node features can travel on a graph [8]. Thus, stacking more GCN layers could only capture the limited global structure and the global consistency of a graph is still not fully explored by the existing methods.

In this paper, we propose a deepwalk-aware graph networks (DGN) method that fuses the local and global structure information of a graph to derive the more effective graph representations. Firstly, Deepwalk [16] uses a random walk to generate long node sequences, which are converted to embedding for

Table 1 List of the important notations

Notation	Description
V	A set of nodes in a graph
v	A node $v \in V$
G	A graph
E	A set of edges in a graph
N	The number of graph nodes
$\boldsymbol{A}$	The graph adjacency matrix
$\overline{\boldsymbol{A}}$	The updated graph adjacency matrix
$\boldsymbol{H}$	The latent feature matrix of GCN_1
$\overline{\boldsymbol{H}}$	The latent feature matrix of GCN_2
d	The dimension of latent representation
u	The output dimension of the GCN layer
t	The walk length
s	The window size
C	The number of the classes
$\boldsymbol{X}$	The original features matrix of the nodes
$\overline{\boldsymbol{V}}$	The deepwalk features matrix of the nodes
$\boldsymbol{M}$	The output matrix of GCN_1
$\boldsymbol{N}$	The output matrix of GCN_2
$\boldsymbol{K}$	The feature matrix of the last softmax layer
γ	The similarity threshold

keeping the long-range relationship between the nodes. Secondly, the node similarity based on deepwalk features is calculated to update the graph adjacency matrix and the global features are further propagated on the updated adjacency matrix. Finally, the local neighboring information and the global structured information are fused to enhance the representation capability of GCNs, which achieve the state-of-the-art results on three benchmark datasets.

The contributions of our paper can be summarized as follows.

(1) The proposed method explores the global structured information of a graph via a deepwalk to enhance the representation capability of deep features, which can be considered as the combination of graph embedding and graph convolutional networks.

(2) The adjacency matrix of a graph is updated by calculating the global information similarity to achieve the richer relationships, and thus global context information of a graph is included, leading to more effective deep features for GCNs. We use cosine similarity to calculate the global structural similarity and use manually set rules to update the adjacency matrix.

(3) The node classification experiments on three benchmark network datasets (84.1% in Cora, 74.1% in Citeseer and 80.2% in Pubmed) show that our model achieves state-of-the-art results.

The rest of this article is arranged as follows. Section 2 introduces the related work. Section 3 presents the proposed method. Section 4 reports the experimental results and gives the discussions. Finally, we conclude the article and give the future work in Section 5.

2 Related work

Graph embedding transfers a graph into a set of low dimensional vectors. Those vectors keep the graph topology and relationships between different nodes. For instance, DeepWalk [16] uses a random walk strategy to transfer a graph into various sequences and leverages the Skipgram model to derive the node embedding. Node2Vec [17] employs the biased-random walks that provide both long-term or short-term relationships among the nodes. LINE [18] leverages the 1st-order and 2nd-order similarity metrics to derive the node embedding, where the 1st-order similarity is used to measure the pairwise similarity between two nodes and the 2nd-order similarity is used to measure the similarity between the neighborhoods of two nodes. SDNE [19] is a deep auto-encoder using a reconstruction loss, which leverages both 1st-order and 2nd-order similarity to supervise the training process. Struc2Vec [20] designs a complicated structural similarity, which uses a hierarchy to measure node similarity at different scales for constructing a multi-layer network. Walklets [21] uses a skipping random walk to generate a corpus of nodes pairs,

which are reachable via the paths of a fixed length. This corpus can be used to learn a series of latent representations in a way that is analytically derivable. DNGR [22] uses random surfing instead of random walk to capture the graph structural information and applies the denoising autoencoder to extract the complex features.

Considering that convolutional neural networks have achieved promising performance for image-based applications such as remote sensing [23–25] and image semantic segmentation [26,27], graph convolutional networks have recently been proposed to deal with the arbitrarily structured graph. Graph convolutional networks define a convolution on a graph and iteratively aggregate the embeddings of local neighbors for a node to derive the new data embedding. The derived data embedding is fed into the softmax layer for a given learning task such as node classification. Aggregating embedding of data locality can make GCNs scale for a large graph and propagation of the features on multiple layers can characterize the global structured information of a graph. For instance, mixture model networks predict the hidden relationship between 2D image and 3D model [28]. Group context graph neural networks (GCGNN) are used to re-identify a group of people across camera systems [29]. To deduce the structural information from a global view, a GCNs architecture is designed to leverage the comprehensive relationships within cross-modal samples to optimize video-text retrieval results [30, 31]. Graph convolutional networks are leveraged to customize an adaptive cross-modal feature learning framework, which is used to solve the RGB-D scene recognition problem [32]. VRD-GCN [33] abstracts the video into a fully connected spatio-temporal graph and uses graph convolutional networks to pass messages and reason in these 3D graphs for the better prediction of the objects and their dynamic relationships.

To enhance the learning performance of GCNs, the researchers have proposed many GCNs models. For instance, a convolution-like operation has been defined based on spectral graph theory. GCNs essentially perform aggregation and transformation on node features without having to learn trainable filters. GraphSAGE [34] samples a fixed number of neighbors and leverages an aggregation function that is invariant to the permutations of node orderings for data representation. GAT [10] leverages an attention scheme to consider different relationships of the adjacent nodes for deriving the learnable filter weights. DualGCN [14] designs two convolutional networks to consider the local consistency and global consistency of the data distributions. Gated attention network (GAAN) [35] computes an additional attention score for each attention head with a self-attention mechanism. GMI [36] leverages mutual information to derive the graph representation via measuring mutual information between two graphs. N-GCN [37] trains multiple GCNs over node pairs with different distances in random walks to derive the results by the combination of the GCNs.

Some studies try to rank a node's neighbors based on a variety of criteria and associate each ranking with a learnable weight to achieve weight sharing across different locations. PATCHY-SAN [38] orders the adjacent nodes according to the graph labelings and then chooses the top q neighbors for convolutional computation to aggregate the neighborhood features. Unfortunately, the processing of PATCHY-SAN requires a large amount of computation. Large-scale graph convolutional network (LGCN) [39] ranks a node's neighbors according to node feature values and assembles a feature matrix, where the first row is fixed to its features and the other rows consist of its neighborhood features. After sorting this feature matrix along each column, it selects the first q rows as the input data for the central node. MvsGCN [40] combines the video summary task with GCN and proposes an important node sampling method. At the same time, MvsGCN proposes two strategies to integrate the task data imbalance into the GCN network to effectively generate a representative summary with good diversity.

More recently, the other work has focused on improving the training efficiency of GCNs. Fast learning with graph convolutional network (Fast-GCN) [41] chooses to sample a fixed number of nodes for each graph convolution layer instead of sampling for each node like GraphSage. Fast-GCN adopts a Monte Carlo approximation and variance reduction technique to facilitate the training process. Adapt GCN [42] leverages an adaptive layer-wise sampling approach and achieves higher accuracy but the sampling scheme is complicated. We can also see the combination of many GCNs and application scenarios. MMGCN [43] adopts a multi-modal strategy to use GCNs to encode the interactive information between users and items in each modality to generate feature representations.

Note that the existing GCNs models usually fail to consider the global structured information of a graph. We introduce deepwalk into GCNs to capture the global consistency of data distribution, leading to the global structure-aware deep features for graph-structured data.

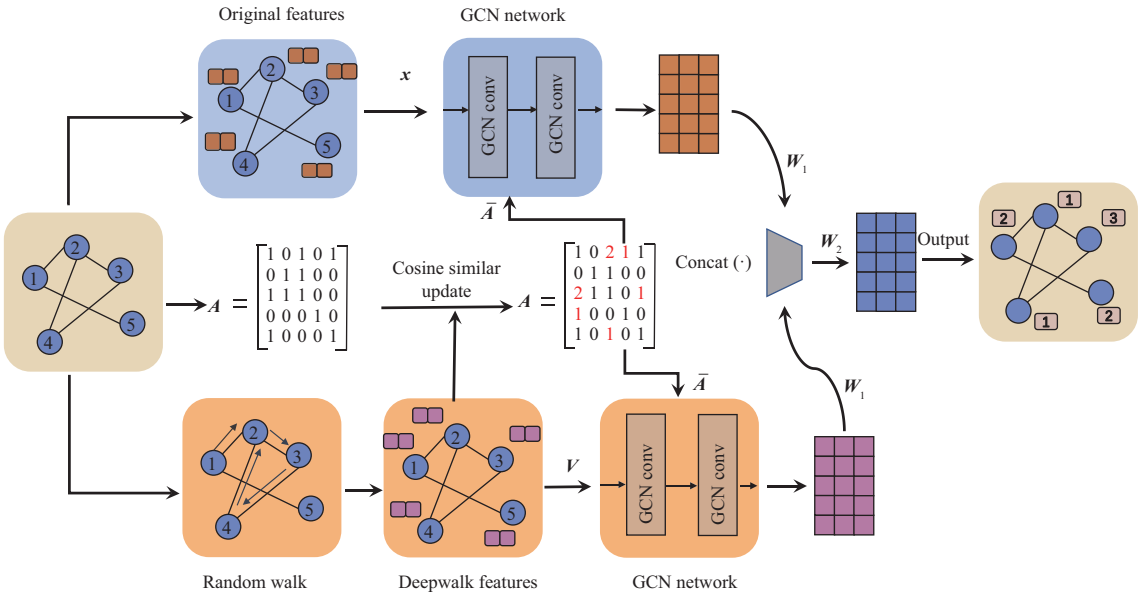

Figure 2 (Color online) The framework of the proposed model. For a graph, we train our model with two GCN networks. The first GCN network uses the original features as input and the second GCN network considers the deepwalk features as the input, which can capture the global structure. Furthermore, the adjacency matrix of a graph is updated based on deepwalk features and the updated adjacency matrix is used as the input of both GCN networks. After training, the features output by the two GCN networks are concatenated after mapping to the same space. Finally, feature dimensionality reduction through MLP is fed into the softmax layer for class prediction.

3 Method

In this section, we present the details of our model. The highlight of the proposed method is to drive the deepwalk-aware GCN features by fusing two graph convolutional networks. We will begin with an introduction to the graph convolutional networks. And then, we will elaborate the deep walk block on extracting the global structure of a graph. Finally, we will present the fuse strategy to fuse two GCNs networks for node classification. For clarification, Table 1 lists the important notations in this article.

Figure 2 shows the framework of the proposed method, which consists of two GCN networks. The training of two networks needs to wait until the adjacency matrix has been updated and the potential features have been obtained. The proposed method performs feature fusion to achieve a combination of local and global consistency in the last network layer.

3.1 Graph convolutional networks

Assuming an undirected graph is represented as $G = (V, E, \boldsymbol{X})$, where $V = \{v_1, \ldots, v_N\}$ is a set of nodes, E is a set of edges that connect the related nodes, and each node $v_i \in V$ is associated with a d-dimensional feature vector $x_i \in \mathbb{R}^d$, resulting in a feature matrix $\boldsymbol{X} = [x_1, x_2, \ldots, x_N] \in \mathbb{R}^{N\times d}$. Based on V and E, the adjacency matrix of a graph is formulated as $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, where $\boldsymbol{A}_{ij} = 1$ if $(v_i, v_j) \in E$, otherwise $\boldsymbol{A}_{ij} = 0$.

In this article, we would like to leverage the graph convolution network (GCN) [8] as the backbone model to conduct the node classification task. Thus, we briefly introduce the network architecture of GCN, where the layer-wise forward-propagation operation of GCN is defined as

$$\boldsymbol{F}^{(k+1)} = \phi\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{F}^{(k)}\boldsymbol{W}^{(k)}\right), \tag{1}$$

where $\boldsymbol{F}^{(k)}$ and $\boldsymbol{F}^{(k+1)}$ are the input and output matrices, respectively, of layer k of the GCN network. $\boldsymbol{A}$ is a graph adjacency matrix, $\mathbf{I}$ is an identity matrix and $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \mathbf{I}$ is aiming to aggregate the features of adjacent nodes. $\tilde{\boldsymbol{D}}$ is a diagonal node degree matrix to normalize $\tilde{\boldsymbol{A}}$, which makes the scale of feature vectors unchanged after aggregation. $\boldsymbol{W}^{(k)}$ is a trainable weight matrix to construct a linear transformation for changing the feature dimension. $\phi(\cdot)$ denotes an activation function.

Considering that $\tilde{A}$ contains only the information of the first-order neighborhood features, we add high-order structural similarity to enrich the information of the adjacency matrix and update the adjacency

matrix to $\overline{A}$. Furthermore, based on the updated adjacency matrix, we design two GCN networks to extract the latent representation of the original data and the global structured data the via deep walk approach (see Subsection 3.2), respectively. After the feature fusion of two GCN networks, the original features can be improved for a better node classification. We define the layer-wise forward-propagation operations of two GCN networks as

$$\boldsymbol{H}^{(k+1)} = \phi\left(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right), \tag{2}$$

$$\overline{\boldsymbol{H}}^{(k+1)} = \phi\left(\hat{\boldsymbol{D}}^{-\frac{1}{2}}\hat{\boldsymbol{A}}\hat{\boldsymbol{D}}^{-\frac{1}{2}}\overline{\boldsymbol{H}}^{(k)}\overline{\boldsymbol{W}}^{(k)}\right), \tag{3}$$

where Eq. (2) is the layer-wise forward-propagation operation of the first GCN network, termed GCN_1, for deriving the local features, and $\boldsymbol{H}^{(k)}$ and $\boldsymbol{H}^{(k+1)}$ are the input and output matrices, respectively, of layer k for the first GCN network. Eq. (3) is the layer-wise forward-propagation operation of the second GCN network, termed GCN_2, for deriving the global structural features, and $\overline{\boldsymbol{H}}^{(k)}$ and $\overline{\boldsymbol{H}}^{(k+1)}$ are the input and output matrices, respectively, of layer k for the second GCN network.

Both GCN networks (GCN_1 and GCN_2) share the same adjacency matrix $\overline{\boldsymbol{A}}$ that is updated according to (6)–(8). The updated adjacency matrix can reflect both the local and global structures of a graph. Furthermore, $\hat{\boldsymbol{A}} = \overline{\boldsymbol{A}} + \mathbf{I}$ is used to aggregate feature vectors of adjacent nodes. $\hat{\boldsymbol{D}}$ is a diagonal node degree matrix for normalizing $\hat{\boldsymbol{A}}$, which makes the scale of feature vectors unchanged after aggregation. For further capturing the local consistency and global consistency of data distribution, except for the updated adjacency matrix, the inputs of two GCN networks are also different. In other words, GCN_1 adopts the original feature matrix $\boldsymbol{X}$ as the input. However, GCN_2 adopts the global structure-aware deepwalk features as the input, where deepwalk features provide a reference for higher-order structural information. Two GCN networks learn the latent representations separately, which are fused as the input of the last softmax layer for classification prediction.

Next, we show that high-order structural similarity information can be obtained through deepwalk, which is used to define two main components of the proposed networks architecture.

3.2 Deepwalk

The original GCN aggregates and propagates the local neighboring information of each node. We introduce the deepwalk approach to extract the global information of a graph. Deepwalk [16] learns the latent representations that encode the relations among the nodes in a continuous vector space, which has been used as a similarity measure for a variety of problems such as community detection [44] and content recommendation [45]. The deepwalk method [16] consists of two parts: random walk generator and the update procedure.

A generator uniformly samples a random vertex $\boldsymbol{v}_i$ in a given graph G as the root of the random walk $\mathcal{W}_{v_i}$. A walk samples uniformly from the neighbors of the most visited node until the walk length reaches the pre-defined value t. We set the maximum length of the walk in the experiment to t. A node can walk to other nodes in the k-hop neighborhood and perform relationship calculations to obtain the structural information of two nodes over a long distance.

The update procedure relies on SkipGram [46] to obtain the final mapping function. SkipGram is a language model that can maximize the co-occurrence probability between words in sentences appearing in window s to predict the context. Given the walk generated by the random walk generator, SkipGram iterates over all possible collocations in a random walk that appears in the window. After mapping v_i to its current representation vector $\Phi(v_i)$, the probability of the neighbors of v_i in a walk is maximized. Based on the node representation modeling, we summarize the optimization problem as follows:

$$\Pr\left(\{v_{i-w},\ldots,v_{i+w}\}\setminus v_i|\Phi(v_i)\right) = \prod_{\substack{j=i\\ j\neq i}}^{i+w}\Pr\left(v_j|\Phi(v_i)\right), \tag{4}$$

$$\min_{\Phi} -\log\Pr\left(\{v_{i-w},\ldots,v_{i-1},v_{i+1},\ldots,v_{i+w}\}|\Phi(v_i)\right). \tag{5}$$

Given a node v_i, we assume that a sequence of nodes appears in the window s that is consistent with the sequence of random walks via maximizing (4). By solving the optimization problem of deepwalk of (5), the mapping function $\Phi(\cdot)$ of a graph is derived. Based on the mapping function $\Phi(\cdot)$, the deepwalk

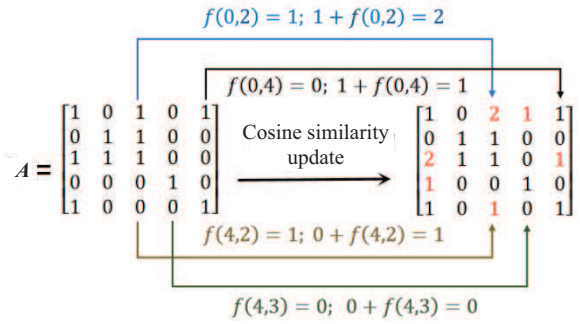

Figure 3 (Color online) An example of updating adjacency matrix. The similarity function in entries $(0, 2)$ and $(4, 2)$ is set to be 1, which represents that the nodes pairs (v_0, v_2) and (v_2, v_4) have stronger neighboring relationships. Then, we update the corresponding entries of the adjacency matrix, which considers the global structured information of a graph. On the contrary, the similarity function in entries $(0, 4)$ and $(4, 3)$ is set to be 0, which represents that the nodes pairs (v_0, v_4) and (v_3, v_4) have the less neighboring relationship and the corresponding entries of the adjacency matrix are not updated.

representations of the nodes are derived as $\overline{\boldsymbol{V}} = [\overline{v}_1, \ldots, \overline{v}_i, \ldots, \overline{v}_N] \in \mathbb{R}^{N\times d}$ for the nodes, where $\overline{v}_i \in \mathbb{R}^d$ is the deepwalk feature of the ith node. These deepwalk features can not only capture the representation of shared similar nodes in a graph, but also preserve the highly non-linear structure of a graph. Based on this investigation, we further propose our main network components.

3.3 Updating adjacency matrix

Considering that the graph information defined by the adjacency matrix $\boldsymbol{A}$ lacks a global structure information, the adjacency matrix is updated with the latent representation of the deepwalk. By introducing the deepwalk approach to generate the embedding that can reflect the similarity of the node structure, we can infer whether there is a connection or a stronger connection between the nodes. Such information can be derived by calculating the cosine similarity of the high-order structural information. The similarity calculation and the updated adjacency matrix are formulated as follows:

$$\cos(\theta|i,j) = \frac{\overline{v}_i \cdot \overline{v}_j}{\|\overline{v}_i\|\|\overline{v}_j\|}, \tag{6}$$

$$f(i,j) = \begin{cases} 1, \text{ if } \cos(\theta|i,j) \geqslant \gamma, \\ 0, \text{ otherwise}, \end{cases} \tag{7}$$

$$\overline{\boldsymbol{A}}_{i,j} = \boldsymbol{A}_{i,j} + f(i,j), \quad i,j = 1,2,\ldots,n, \tag{8}$$

where $\overline{v}_i \in \mathbb{R}^d$ and $\overline{v}_j \in \mathbb{R}^d$ (d is the dimension of latent representation) are the deepwalk representations of nodes i and j in a graph. For one example of updating the adjacency matrix, please refer to Figure 3.

The proposed method calculates the cosine similarity between two nodes and compares the cosine similarity to the predefined threshold. If $\cos(\theta|i,j)$ is greater than the threshold γ, then the corresponding $\boldsymbol{A}_{i,j}$ is set to 1. This means that nodes v_i and v_j have a stronger neighbor relationship and we update the value of the corresponding position of two nodes in the adjacency matrix via the sum of $\boldsymbol{A}_{i,j}$ and $f(i,j)$; otherwise, $\boldsymbol{A}_{i,j}$ is in fact assigned to the large value between $f(i,j)$ and $\boldsymbol{A}_{i,j}$. After calculating the cosine similarity of all pairs of nodes and updating the adjacency matrix, the updated adjacency matrix is input into two GCN networks.

Note that we adopt the threshold scheme instead of the original similarity to update the adjacency matrix of a graph based on the following two reasons: (1) in general, the large cosine similarity (meantime usually smaller than 1) is more likely to reflect the underlying structures of the data. For the large similarities, we enhance the cosine similarity to 1 via the threshold scheme, which makes GCN more effectively capture the global consistency of a graph; (2) a large number of small similarities make the entries of the updated adjacency matrix dense. Thus, we remove the small similarity via the threshold scheme, which facilitates the following step of GCN employment.

3.4 Fusion strategy

To make full use of the global structured information of a graph, we leverage two GCN networks architecture to train the generated global structure information of a graph separately, instead of simply stitching it. Since deepwalk is unsupervised and does not use the real labeled data, the vector of each node only represents its high-order structural information. To make the nodes express the structural better, we use supervised training to constrain and update the final structural features, and fuse the two features in the last layer. We summarize the process as follows:

$$\boldsymbol{K} = \operatorname{softmax}\left(\operatorname{concat}\left(\boldsymbol{W}_1 \times \boldsymbol{M}, \boldsymbol{W}_1 \times \boldsymbol{N}\right) \times \boldsymbol{W}_2\right). \tag{9}$$

To balance the importance of local and global consistency of a graph, we need to specify weight parameters for the outputs of the two GCN networks. In our experiments, we specify $\boldsymbol{W}_1 \in \mathbb{R}^{u\times u}$ (u is the output dimension of the GCN layer) as the weight parameter to map the output matrices $\boldsymbol{M}$ and $\boldsymbol{N}$ of two GCN networks to the same embedding space and $\boldsymbol{W}_2 \in \mathbb{R}^{2u\times C}$ (C is the number of the classes) to reduce the dimension to the number of categories. The matrix $\boldsymbol{K} \in \mathbb{R}^{n\times C}$ is a fusion result of $\boldsymbol{M} \in \mathbb{R}^{n\times u}$ and $\boldsymbol{N} \in \mathbb{R}^{n\times u}$ where $\boldsymbol{M}$ is the output of the local consistency network (GCN_1) and $\boldsymbol{N}$ is the output of the global consistency network (GCN_2). Finally, the labels matrix $\boldsymbol{K}$ is used for multi-class node classification.

Implementation. Algorithm 1 lists the main procedure of the proposed method. As our method consists of two simple GCN neural networks, the conventional parameter update strategies can be applied to train the proposed method.

Algorithm 1 DGN

Require: an adjacency matrix $\boldsymbol{A}$, the updated adjacency matrix $\overline{\boldsymbol{A}}$, the original feature matrix $\boldsymbol{X}$, original graph $G(V, E)$, a set of deepwalk features $\overline{\boldsymbol{V}}$, walk length t, window size s, walks per node r.
Ensure: Node labels matrix $\boldsymbol{K}$.
$\boldsymbol{F}$ = Deepwalk($G(V, E), s, r, t$);
for $(i, j) \in V$ **do**;
Calculate the cosine similarity of V_i and V_j according to (6);
Assign a new value to $\overline{\boldsymbol{A}}_{i,j}$ according to (7) and (8);
end for
$\boldsymbol{M} = \mathbf{GCN}_1(\overline{\boldsymbol{A}}, \boldsymbol{X})$;
$\boldsymbol{N} = \mathbf{GCN}_2(\overline{\boldsymbol{A}}, \overline{\boldsymbol{V}})$;
Calculate the labels matrix $\boldsymbol{K}$ of nodes according to (9);
Return $\boldsymbol{K}$;
Deepwalk($G(V, E), s, r, t$)
Initialization: sample Φ from $\boldsymbol{F}$;
for $i = 0$ to r **do**
O = Shuffle(V);
for $v_i \in O$ **do**
W_{v_i} = RandomWalk(G, v_i, t);
SkipGram(Φ, W_{v_i}, s);
end for
end for
Return $\boldsymbol{F}$

4 Experiments

In this section, we evaluate the proposed method on the node classification task and report the accuracy of classification (ACC), which is the ratio between the number of correctly classified nodes and the total number of nodes.

4.1 Dataset description

We conduct the transductive node classification experiments on the Cora, Citeseer, and Pubmed datasets [39][1)]. These datasets are the benchmark citation networks, where nodes represent documents, features are the bag-of-word representation of a document, and edges are the citations of the documents. We adopt the same experimental setting as LGCN [39]. Table 2 lists the statistic of the used citation networks datasets and their corresponding experimental setting.

1) The source code is available. https://github.com/Paper-code-h/DGN.

Table 2 Summary of datasets used in our node classification experiments

Dataset	Nodes	Edges	Features	Classes	Training nodes	Validation nodes	Test nodes	Degree
Cora	2708	5429	1433	7	140	500	1000	4
Citeseer	3327	4732	3703	6	120	500	1000	5
Pubmed	19717	44338	500	3	60	500	1000	6

• Cora. Cora dataset includes 2708 scientific publications with seven classes. This citation network contains 5429 links, where each node is represented by a 1433-dimensional 0/1-valued vector. About 5.2% of the nodes are labeled, which can be used for training.

• Citeseer. Citeseer dataset includes 3327 scientific publications with six classes. This citation network contains 4732 links, where each publication is represented by a 0/1-valued word vector for indicating the absence/ presence of the word from a dictionary of 3703 unique words. About 3.6% of the nodes are labeled, which can be used for training.

• Pubmed. The Pubmed dataset contains 19717 scientific publications classified into one of three classes. The citation network consists of 44338 links. Each publication is described by a term frequency-inverse document frequency (TF-IDF) vector drawn from a dictionary with 500 terms. About 0.3% of the nodes are labeled, which can be used for training.

4.2 Experimental setup

In our experiments, we set the length of the random walk starting at each node to 40. After each node is part of 10 walking paths, all paths are fed to the skipgram model with a window size of 5. We get a 96-dimensional representation for the Citeseer dataset and a 64-dimensional representation for the Cora and Pubmed datasets. After calculating the cosine similarity of the node representations, we set the threshold of 0.86 for the Cora dataset, 0.9 for the Citeseer dataset and 0.98 for the Pubmed dataset to update the adjacency matrix. We then feed the original features and deepwalk features into two-layer GCN [8]. The final features are derived by a weighted sum of the two networks outputs. Finally, a softmax layer is used as a classifier to make predictions. During training, we employ the Adam optimizer [47] with a learning rate of 0.1 for the Citeseer dataset and 0.01 for the Cora dataset and the Pubmed dataset. Dropout [48] with a rate of 0.5 is applied in each layer of two networks. We initialize weights using the initialization described in [49]. We report the mean classification accuracy with a standard deviation of the 100 runs of the proposed method, where the best classification results are bold faced.

4.3 Results

To demonstrate the learning of the proposed method, we compare the proposed method against DeepWalk [16] and the latest GCNs methods including GCN [8], GAT [10], DualGCN [14] and LGCN [39], StoGCN [50], DGI [10], GMI [36] and N-GCN [37], which are denoted as follows:

DeepWalk [16]. DeepWalk leverages the rand walk scheme to derive the structured information of a graph, which is used to learn latent representations.

GCN [8]. GCN relies on a first-order approximation of Chebyshev polynomials of the diagonal matrix of eigenvalues with less computation cost.

GAT [10]. GAT leverages an attention scheme to consider different relationships of the adjacent nodes for deriving the learnable filter weights.

DualGCN [14]. DualGCN designs two convolutional networks to consider the local consistency and global consistency of the data distributions, where the original features are used as the input of two GCN networks and the weights of two networks are shared.

LGCN [39]. LGCN is the graph convolutional network that order the different adjacent nodes according to the feature values and then choose a fixed number of nodes to transform graph data into grid data for employing the general convolutional computation.

StoGCN [50]. StoGCN leverages historical node representation to reduce the size of the perceptual field of graph convolution for enhancing the computation efficiency of the convolutional operation.

DGI [10]. DGI relies on maximizing mutual information between patch representations and the high-level summaries of graphs, where the derived patch representations summarize subgraphs centered around nodes of interest.

GMI [36]. GMI extends the idea of mutual information to the graph domain by measuring mutual information between two graphs.

Table 3 Classification accuracy on the dataset

Method	Cora (%)	Citeseer (%)	Pubmed (%)
DeepWalk [16]	67.2	43.2	65.3
GCN-2 [8]	81.5	70.3	79.0
GAT [10]	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
DualGCN [14]	83.5	72.6	80.0
LGCN [39]	83.3 ± 0.5	73.0 ±0.6	79.5 ± 0.2
StoGCN [50]	82.0± 0.8	70.9± 0.2	79.0± 0.4
DGI [10]	82.3± 0.6	71.8± 0.7	76.8± 0.6
GMI [36]	83.0± 0.3	73.0± 0.3	80.1± 0.2
N-GCN [37]	83.0	72.2	79.5
DGN (ours)	**84.1 ± 0.2**	**74.1 ± 0.2**	**80.2 ± 0.1**

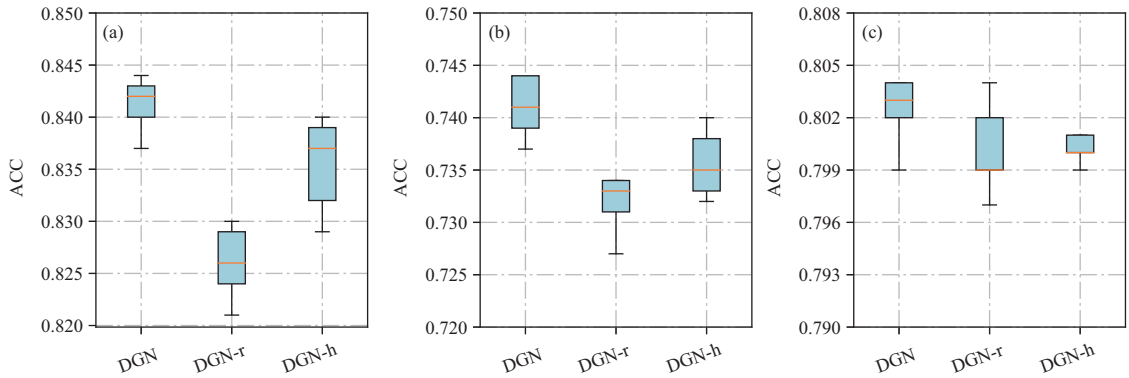

Figure 4 (Color online) Ablation study on the dataset. We compare the complete networks with several variants with parts removed, where GGN-r removes the updated adjacency matrix and uses the original adjacency matrix. DGN-h removes the second GCN network. All models were trained with the same hyper-parameters for 50 epochs on all datasets. The top of the icon in the figure represents the maximum value reached, and the bottom of the icon represents the minimum value reached. (a) Cora; (b) Citeseer; (c) Pubmed.

N-GCN [37]. N-GCN trains multiple GCNs over node pairs with different distances to derive the results by the combination of the GCNs.

We report node classification accuracies on the datasets as summarized in Table 3. As shown in Table 3, we observe the following.

The proposed method consistently outperforms the other methods on the Citeseer, and Pubmed datasets, respectively. Specifically, our method significantly outperforms LGCN (achieving the second best results), with the increase of 0.9% on Cora as well as 1.4% on Citeseer. Note that DualGCN also has two GCN networks, but DualGCN is still inferior to the proposed method. The performance enhancement benefits from that our method is capable to fuse both the global and local consistencies of a graph, resulting in the more discriminative features. Compared to DualGCN, our method directly leverages the deepwalk features as the input of a GCN network and meantime updates the shared adjacency matrix of two GCN networks. More importantly, two GCN networks of our method are trained separately. However, two networks of DualGCN share the same network weights and deepwalk features are not used as the input of GCN and the adjacency matrix is not updated accordingly. Thus, the network architecture of DualGCN and our method is totally different. The experimental results show that the proposed method is superior to DualGCN by a large margin, which demonstrates that the proposed method can more effectively capture the global structures of a graph.

4.4 Ablation studies

In this subsection, we make ablation analyses for coupling our proposed method and evaluate different update schemes of the adjacency matrix.

4.4.1 *On the update of affinity and latent representation*

To demonstrate the performance of the two components of the proposed framework, we remove each component and test its performance while fixing the other components. We first cancel the update of the

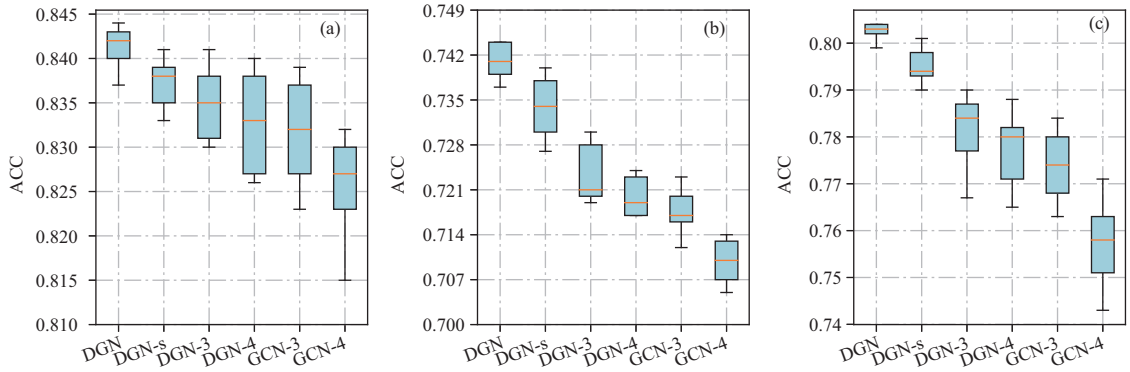

Figure 5 (Color online) Ablation study on the dataset. We compare the proposed method to deeper layer GCN-3, GCN-4, DGN-3, DGN-4 and DGN-s. GCN-3 and GCN-4 are three-layer GCN and four-layer GCN, respectively. DGN-3 and DGN-4 leverage three-layer GCN and four-layer GCN, respectively, as the backbone network within the proposed DGN architecture. DGN-s leverages the original cosine similarity to update the adjacency matrix of a graph complete networks with several variants with parts removed. All models were trained with the same hyper-parameters for 50 epochs on all datasets. The top of the icon in the figure represents the maximum value reached, and the bottom of the icon represents the minimum value reached. (a) Cora; (b) Citeseer; (c) Pubmed.

adjacency matrix from the model (DGN-r) and then remove the input of the latent representation from the model (DGN-h) and show the results in Figure 4.

We observe that both the update of the adjacency matrix and the addition of the latent representation via deepwalk provide the performance enhancements. Removing updates of the adjacency matrix results in large performance degradation, where the performance drops by 2.0%, 1.3% and 0.5%, respectively. These experimental results show that the adjacency matrix is the key component and contains discriminative information on higher-order structural information. A slight deviation of the matrix affects the performance. On the other hand, when removing the latent representation of a deepwalk, the performance drops by 0.5%, 1.0% and 0.3%, respectively.

The experimental results show that the representation only affects the performance when the feature fusion is performed at the end and does not directly affect the networks trained by the original features. Even if this part is missing, the performance degradation is not obvious. These experimental results indicate that our method does capture the global structured information of a graph, which yields the more discriminative features for graph-structured data.

4.4.2 *On the threshold setting and network layer number*

To update the adjacency matrix of a graph, we adopt the threshold strategy to remove the small similarity values and set the large similarity to 1 (see Eq. (7)). To observe how the similarity setting affects the learning performance, we design a novel GCNs model, termed DCN-s, which is the same as the proposed method except that the original cosine similarity is directly used to update the affinity matrix. On the other hand, GCNs can derive the global structures of a graph by enhancing the number of network layer. We compare our method to the deeper layer GCN models: GCN-3 and GCN-4, where GCN-3 and GCN-4 are three-layer GCN and four-layer GCN methods, respectively. Furthermore, we leverage three-layer GCN and four-layer GCN, respectively, as the backbone network within the DGN architecture, where the proposed method is termed DGN-3 and DGN-4, respectively. Figure 5 shows the ablation results.

We observe that the proposed method outperforms DGN-s, uses the original cosine similarity between two nodes to update the adjacency matrix. Compared with the deeper layer GCN model: GCN-3 and GCN-4, the proposed method achieves the performance enhancement by a large margin. Specifically, compared with the proposed method using the two-layer GCN (DGN), the learning performance of the proposed method using the three-layer GCN or four-layer GCN as the backbone network (DGN-3 and DGN-4) is degraded slightly. However, both DGN-3 and DGN-4 are still superior to the original GCN-3 and GCN-4 framework, respectively.

These experimental results indicate that our method (DGN) can more effectively capture the global structure of a graph, which explains most of the performance enhancement of the proposed method. Compared with the proposed method, the deeper layer GCN models (GCN-3 and GCN-4) only capture the limited global structures of a graph. The over-smooth effect of the backbone GCN framework makes learning performance of deeper GCN networks including GCN-3, GCN-4, DGN-3 and DGN-4 degenerate

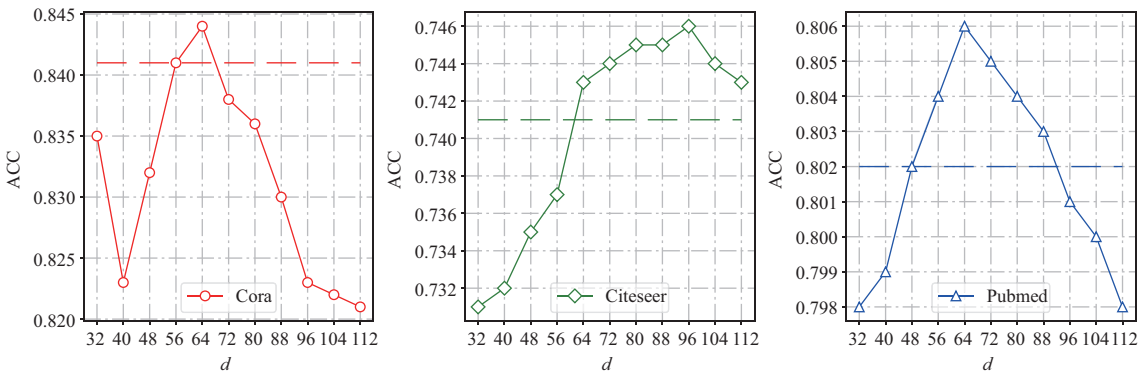

Figure 6 (Color online) Classification accuracy vs. d. d is the dimension of latent representation and ranges from 32 to 112. Our model performs the best when the representation dimension is set to 96 for the Citeseer and 64 for the Cora dataset and Pubmed dataset.

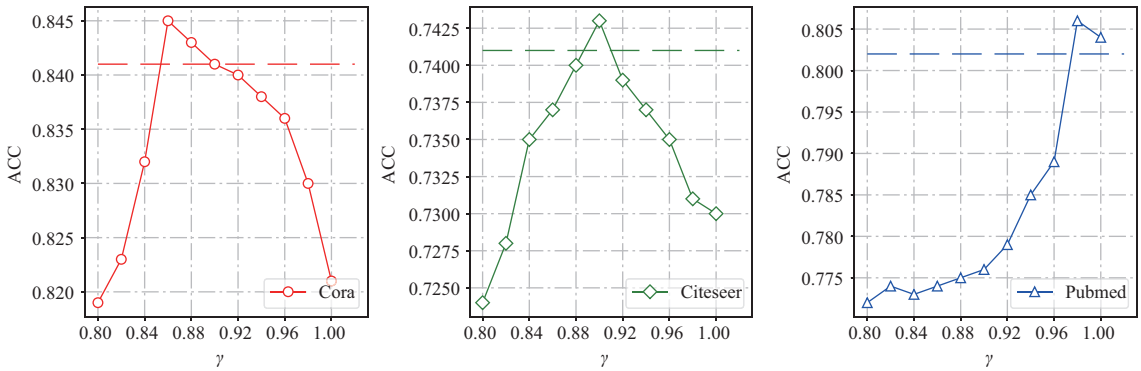

Figure 7 (Color online) Classification accuracy vs. γ. γ is the similarity threshold and ranges from 0.8 to 1.0. Our model performs the best when the similarity threshold is set to 0.86 for the Cora dataset, 0.9 for the Citeseer dataset and 0.98 for the Pubmed dataset.

a little. It is attributed that stacking too many layers may result in over-smoothing [15]. However, the proposed method can significantly enhance the learning performance by fusing graph embedding and GCN models, resulting in the improved classification results.

4.5 Parameter setting

Our method has three key parameters: the dimension of latent representation d, the similarity threshold γ and the output dimension of GCN layer u. We tune these three parameters separately by fixing other parameters to observe how the parameter affects performance.

As shown in Figure 6, our model performs the best when the representation dimension is 96 for the Citeseer dataset and 64 for the Cora and Pubmed datasets. Larger or smaller feature dimensions result in performance degradation. Too small feature dimensions lose graph information, while too large feature dimensions cause information redundancy. Thus, it is crucial to choose a suitable feature dimension.

As shown in Figure 7, our model performs the best when the similarity threshold is set to 0.86 for the Cora dataset, 0.9 for the Citeseer dataset and 0.98 for the Pubmed dataset. The similarity can be considered a correlation between two nodes when using under a small threshold. We believe that two nodes below the threshold have no neighbor relationship. Deepwalk has a varying ability to acquire nodes in different datasets. It has a stronger representation ability for graph networks with denser nodes. For the dataset, if its node degree is higher, the possibility of having a similar structure is greater, so a stronger similarity threshold is needed to determine whether they are adjacent nodes. We see that as the node degree values of Cora, Citeseer, and Pubmed are higher, their similarity threshold is also higher.

As shown in Figure 8, our model performs the best when the output dimension of GCN layer u is set to 128 for the Cora dataset, 128 for Citeseer dataset and 64 for Pubmed dataset. The reason is attributed that too small of a feature dimension loses graph information, while too large feature dimensions cause information redundancy. Appropriate dimensions can reasonably represent local and global information.

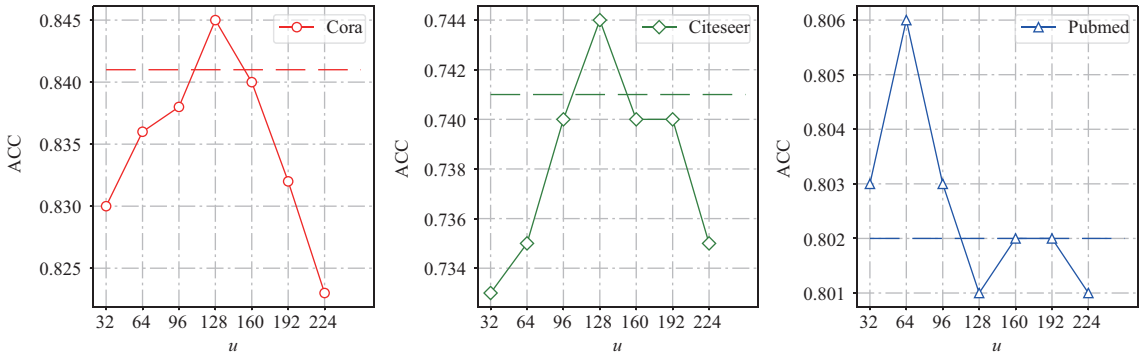

Figure 8 (Color online) Classification accuracy vs. u. u is the output dimension of GCN layer and ranges from 32 to 224. Our model performs the best when the output dimension of GCN layer is set to 128 for the Cora dataset, 128 for Citeseer dataset and 64 for Pubmed dataset.

Table 4 Train time on the dataset

Dataset	Cora (ms)	Citeseer (ms)	Pubmed (ms)
GCN-2 [8]	**15**	**29**	**207**
GCN-3 [8]	56	78	584
GCN-4 [8]	64	96	723
GAT [10]	109	254	268
DualGCN [14]	151	238	1,884
LGCN [39]	86	62	574
GMI [36]	181	239	1,056
N-GCN [37]	82	140	277
DGN-s	54	67	478
DGN (ours)	43	58	420

Table 5 Test time on the dataset

Dataset	Cora (ms)	Citeseer (ms)	Pubmed (ms)
GCN-2 [8]	**9**	**14**	**118**
GCN-3 [8]	27	39	267
GCN-4 [8]	31	48	305
GAT [10]	98	204	178
DualGCN [14]	43	32	241
LGCN [39]	341	381	1,473
GMI [36]	140	176	857
N-GCN [37]	31	61	196
DGN-s	25	32	224
DGN (ours)	22	29	214

4.6 Runtime

Compared with the typical GCNs, our model is not complex, which only adds an additional network to calculate the latent representation. Compared with the simple GCN [8], the number of the network weights of the proposed method only increases by one fold (except for the deepwalk parameters). Besides, the running time of the proposed method increases accordingly. We report training and test time of one epoch implementation of different methods. For the DGI and StoGCN methods, because of no open-source code, we do not report the results of StoGCN and DGI. For a fair comparison, we adopt the same deep learning framework. All tests are conducted on a single Nvidia GPU of GeForce GTX 1080Ti. The running time of the other methods is listed in Tables 4 and 5.

As shown in Tables 4 and 5, the proposed method spends about the double running time of the native GCN method. Although our method involves two networks at the same time, they are very simple and do not require much training time. We also compare the proposed method with other methods based on the same experimental conditions. The results show that our method has the better computational efficiency, which is superior to the other compared methods including GAT, DualGCN, LGCN, GMI and

N-GCN in most cases. From the results, our model shows promising performance in both computational efficiency and node classification results.

5 Conclusion

In this paper, we proposed a global structure-aware graph convolutional network. By incorporating deepwalk into GCNs to calculate a better corresponding embedding, the resulting latent representation can infer the connection relationship between two nodes, and further provide more effective information for the original training by updating the adjacency matrix. Experimental results demonstrate that the proposed model achieves the performance improvement on transductive learning tasks.

As the first step of introducing Deepwalk for graph node classification, we pay more attention to the structural modeling than existing models. For the future work, we plan to leverage more structure information via designing new approaches such as k-hop neighborhood structure calculation to integrate the global information of the graph into a local neighboring structure. Moreover, the proposed network framework has potential to be applied to the other fields such as activity recognition.

Acknowledgements The work was supported by National Key Research and Development Plan Project (Grant Nos. 2018YFC0830105, 2018YFC0830100), in part by National Science Fund for Distinguished Young Scholars (Grant No. 62025603), in part by National Natural Science Foundation of China (Grant Nos. U1705262, 62072386, 62072387, 62076016, 61772443).

References

1 Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry. In: Proceedings of International Conference on Machine Learning, 2017. 1263–1272
2 Deng J, Dong W, Socher R, et al. ImageNet: a large-scale hierarchical image database. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2009. 248–255
3 Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2015. 91–99
4 Chen L C, Papandreou G, Kokkinos I, et al. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In: Proceedings of International Conference of Legal Regulators, 2015
5 Hammond D K, Vandergheynst P, Gribonval R. Wavelets on graphs via spectral graph theory. Appl Comput Harmonic Anal, 2011, 30: 129–150
6 Bruna J, Zaremba W, Szlam A, et al. Spectral networks and locally connected networks on graphs. In: Proceedings of International Conference of Legal Regulators, 2014
7 Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2016. 3844–3852
8 Kipf T, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of International Conference of Legal Regulators, 2017
9 Klicpera J, Weißenberger S, Günnemann S. Diffusion improves graph learning. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2019. 13333–13345
10 Velickovic P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of International Conference of Legal Regulators, 2018
11 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2017. 5998–6008
12 Monti F, Boscaini D, Masci J, et al. Geometric deep learning on graphs and manifolds using mixture model CNNs. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, 2017. 5115–5124
13 Chiang W L, Liu X, Si S, et al. Cluster-GCN: an efficient algorithm for training deep and large graph convolutional networks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019
14 Zhuang C, Ma Q. Dual graph convolutional networks for graph-based semi-supervised classification. In: Proceedings of the World Wide Web Conference, 2018. 499–508
15 Li Q, Han Z, Wu X M. Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the 32nd AAAI Conference on Artificial Intelligence, 2018. 3538–3545
16 Perozzi B, Al-Rfou R, Skiena S. DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014. 701–710
17 Grover A, Leskovec J. Node2vec: scalable feature learning for networks. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016
18 Tang J, Qu M, Wang M, et al. LINE: large-scale information network embedding. In: Proceedings of the World Wide Web Conference, 2015
19 Wang D, Cui P, Zhu W. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016
20 Ribeiro L F R, Saverese P H P, Figueiredo D R. Struc2vec: learning node representations from structural identity. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017
21 Perozzi B, Kulkarni V, Chen H, et al. Don't walk, skip!: online learning of multi-scale network embeddings. In: Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2017. 258–265
22 Cao S, Lu W, Xu Q. Deep neural networks for learning graph representations. In: Proceedings of the Association for the Advance of Artificial Intelligence, 2016
23 Ru L, Du B, Wu C. Multi-temporal scene classification and scene change detection with correlation based fusion. IEEE Trans Image Process, 2021, 30: 1382–1394

24 Zhu D, Du B, Zhang L. Two-stream convolutional networks for hyperspectral target detection. IEEE Trans Geosci Remote Sens, 2021, 59: 6907–6921
25 Xu Y, Du B, Zhang L. Beyond the patchwise classification: spectral-spatial fully convolutional networks for hyperspectral image classification. IEEE Trans Big Data, 2020, 6: 492–506
26 Zhou Q, Yang W, Gao G, et al. Multi-scale deep context convolutional neural networks for semantic segmentation. World Wide Web, 2019, 22: 555–570
27 Zhou Q, Wang Y, Liu J, et al. An open-source project for real-time image semantic segmentation. Sci China Inf Sci, 2019, 62: 227101
28 Nie W Z, Ren M J, Liu A A, et al. M-GCN: multi-branch graph convolution network for 2D image-based on 3D model retrieval. IEEE Trans Multimedia, 2021, 23: 1962–1976
29 Zhu J, Yang H, Lin W, et al. Group re-identification with group context graph neural networks. IEEE Trans Multimedia, 2021, 23: 2614–2626
30 Wang W, Gao J, Yang X, et al. Learning coarse-to-fine graph neural networks for video-text retrieval. IEEE Trans Multimedia, 2021, 23: 2386–2397
31 Mithun N C, Li J, Metze F, et al. Learning joint embedding with multimodal cues for cross-modal video-text retrieval. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval, 2018. 19–27
32 Yuan Y, Xiong Z, Wang Q. ACM: adaptive cross-modal graph convolutional neural networks for RGB-D scene recognition. In: Proceedings of the Association for the Advance of Artificial Intelligence, 2019. 9176–9184
33 Qian X, Zhuang Y, Li Y, et al. Video relation detection with spatio-temporal graph. In: Proceedings of the 27th ACM International Conference on Multimedia, 2019. 84–93
34 Hamilton W L, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the Annual Conference on Neural Information Processing Systems, 2017. 1024–1034
35 Zhang J, Shi X, Xie J, et al. GaAN: gated attention networks for learning on large and spatiotemporal graphs. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2018
36 Peng Z, Huang W, Luo M, et al. Graph representation learning via graphical mutual information maximization. In: Proceedings of the Web Conference, 2020. 259–270
37 Abu-El-Haija S, Kapoor A, Perozzi B, et al. N-GCN: multi-scale graph convolution for semi-supervised node classification. In: Proceedings of the Conference on Uncertainty in Artificial Intelligence, 2019. 841–851
38 Niepert M, Ahmed M O, Kutzkov K. Learning convolutional neural networks for graphs. In: Proceedings of International Conference on Machine Learning, 2016. 2014–2023
39 Gao H, Wang Z, Ji S. Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018. 1416–1424
40 Wu J, Zhong S H, Liu Y. MvsGCN: a novel graph convolutional network for multi-video summarization. In: Proceedings of the 27th ACM International Conference on Multimedia, 2019. 827–835
41 Chen J, Ma T, Xiao C. FastGCN: fast learning with graph convolutional networks via importance sampling. In: Proceedings of the International Conference of Legal Regulators, 2018
42 Huang W, Zhang T, Rong Y, et al. Adaptive sampling towards fast graph representation learning. In: Proceedings of Annual Conference on Neural Information Processing Systems, 2018. 4558–4567
43 Wei Y, Wang X, Nie L, et al. MMGCN: multi-modal graph convolution network for personalized recommendation of micro-video. In: Proceedings of the 27th ACM International Conference on Multimedia, 2019. 1437–1445
44 Andersen R, Chung F, Lang K. Local graph partitioning using pagerank vectors. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), 2006. 475–486
45 Fouss F, Pirotte A, Renders J, et al. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans Knowl Data Eng, 2007, 19: 355–369
46 Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. In: Proceedings of ICLR Workshop, 2013
47 Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the International Conference of Legal Regulators, 2015
48 Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res, 2014, 15: 1929–1958
49 Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. J Mach Learn Res, 2010, 9: 249–256
50 Chen J, Zhu J, Song L. Stochastic training of graph convolutional networks with variance reduction. In: Proceedings of the International Conference on Machine Learning, 2018