Neural Computing and Applications (2023) 35:13135–13145
https://doi.org/10.1007/s00521-021-06429-1

S.I. : DEEP SOCIAL COMPUTING

Graph convolutional network with multi-similarity attribute matrices fusion for node classification

Youquan Wang[1] · Jie Cao[1] · Haicheng Tao[1]

Received: 5 January 2021 / Accepted: 17 August 2021 / Published online: 17 September 2021

Abstract
Graph convolution networks (GCNs) have become one of the most popular deep neural network-based models in many real-world applications. GCNs can extract features take advantage of both graph structure and node attributes based on convolutional neural networks. Existing GCN models represent nodes by aggregating the graph structure and node attributes from their neighbors which usually disrupt the node similarities in the feature space. In this paper, we propose the MSF-GCN, a graph convolutional network with multi-similarity attributed matrices fusion for node classification. The key idea behind the MSF-GCN is that not only the topology but also the attributes similarities are taken into consideration for node presentation. Specifically, we first apply a GAT-based module to obtain a general representation of the original graph. Next, we construct two k-nearest neighbor graphs based on node attributes with cosine similarity and heat kernel similarity. To balance the disparity between the graph structure and node attributes for each similarity matrix, we develop a self-attention network to integrate the node attributes with topological features. Furthermore, we design a gated-fusion network to merge the cosine similarity vector and heat kernel vector. In our experiments on four real-world datasets, results show that our MSF-GCN model can extract more correlation information from the node attributes and graph structure, and outperform seven state-of-the-art methods.

Keywords Graph convolutional networks · Graph representation learning · Node classification · Multi-similarity attributed matrices fusion

1 Introduction

Recently, graph convolutional networks (GCNs) have become an increasingly powerful tool in various fields such as social media, recommender systems, natural language processing and biology [1–4]. The goal of GCNs is to learn a low dimensional embedding vector of each node for the graph structure data by aggregating the graph structure and node attribute features with convolutional operation. Node representation is realized by collecting the embedding of neighbor nodes, and then performing one or more layers of linear transformation and nonlinear activation. After that, the node vectors can be applied for many important machine learning tasks on graph, such as node classification [5], community detection [6, 7], link prediction [8] and graph anomaly detection [9]. Note that in this paper, we focus on the problem of node classification in attributed graphs: Given a set of labeled nodes with rich node attributed features and topological information, the goal is to predict the classification of unlabeled nodes.

Recent years, GCNs have made remarkable achievements in learning node representation [10, 11]. Generally speaking, GCNs can be categorized into two categories: spectral approaches and spatial approaches. Spectral approaches [12] perform the convolutional operation in the spectral domain. In such way, spectral methods do not need to construct neighbor nodes explicitly for complex graph data. Specifically, spatial approaches [13, 15] conduct convolutions directly on the graph. These methods first construct a fixed size neighborhood for each node in the network and then perform regular convolution on the neighbor nodes.

✉ Jie Cao
jie.cao@nufe.edu.cn

1 Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing, China

Nevertheless, it is reported that node representation learning from the GCNs naturally destroy the node similarity of the attribute feature spaces [16]. Since the node similarity plays an important role for graph representation, the learnable representation can be affected by this phenomenon. In addition, the degree of the node in the graph usually follows the long-tail distribution, which means that many nodes are low-degree nodes with a few of neighbors. In the traditional GCN models, there are many cold-start nodes (many nodes only have few or no neighbors), and it makes the representation of those nodes is not very accuracy. Thus, those nodes can obtain very limited information from their neighbors and GCNs receive poor performance on those nodes. Moreover, GCNs are usually with shallow architectures due to its intrinsic limitation [17], so as to limit the effective propagation of label signal. To solve this problem, Franceschi et al. [18] created a k nearest neighbor (kNN) graph based on some measures of similarity between data points. Although this method can partially improve the performance of GCNs with few labeled data, this method ignores the importance of topological information.

In this work, we aim to design a new Graph Convolutional Network with Multi-Similarity attributed matrices Fusion (MSF-GCN) that can better obtain the balance between the graph structure and node attribute feature during the aggregation. Specifically, MSF-GCN first applies a GAT-based module to obtain the general representation of the original graph. Next, since different similarity metrics can measure different aspect of nodes, we construct two k-nearest-neighbor (kNN) graph based on node features with cosine similarity and heat kernel similarity. Following the constructed graphs, MSF-GCN further conducts the self-attention network to balance the influence between the graph structure and node features for each similarity matrix. By integrating multi-similarity matrices, our model can learn the comprehensive semantics ingrained in the attributed graphs. Furthermore, we develop a gated-fusion network to measure the importance of cosine similarity vector and heat Kernel vector and aggregate them together. Suffice it to say that, although our model only utilizes two similarity matrices, our model can be easily extended to multiple similarity matrices fusion model. Our contributions can be summarized as follows:

– We propose a novel GCN model called MSF-GCN that can effectively balance the information from the graph structure and node attribute feature during the aggregation step. It exhibits start-of-the-art performance on attributed graph with different label rates.
– We take two k-nearest-neighbor (kNN) graphs based on node features with two similarity measures and the topological graph as the input to enhance the MSF-GCN model and improve the node classification task.
– Extensive experimental results show that our proposed model can outperform state-of-the-art baselines on four real-world datasets, which demonstrate the effectiveness of MSF-GCN.

2 Related work

Our work is related to node classification with deep neural network. In this section, we will give the brief reviews about the node classification and graph convolution network, respectively.

2.1 Node classification

In the field of graph structured data learning, there is a very important task about *node classification*. Given the corresponding categories of some nodes in the graph, its task is to predict which categories the unlabeled nodes belong to [19, 20]. This task is also called semi-supervised node classification. Early works on node classification use iterative classification methods with node features [21], label propagation [39], graph regularization [23, 24] to learn a global labeling function over the graph. However, these methods do not usually scale up the large graphs.

In recent years, node embedding for node classification aims to project each node in graph into a low dimension vector with either shallow [25, 26] or deep architectures [27, 28] for classification. DeepWalk [25] constructs the second-order proximity for node embedding with random walks. Node2Vec [26] extends DeepWalk with a truncated random walk on graph and applies the Skip Gram by treating the sampling nodes as sentences. Higher-order of proximity method such as HOPE [29] are based on a high-order proximity preserved embedding. All the above methods can be used in the field of node classification. While a majority of these methods are based on representation before classification and it lacks an end-to-end model for representation and classification together.

2.2 Graph convolution networks

Graph convolution network is evolved from convolution neural network and it generalizes the convolution operation from grid structure data to graph structure data [30]. Overall, GCNs can be categorized into two classes: Spectral convolution and spatial convolution. Spectral convolution defines the spectral representation of a graph in Fourier domain by Laplacian eigen-decomposition [12, 30]. The spatial convolution methods operate directly on the graph to aggregate the multi-order neighbor nodes to the node representation. GraphSAGE obtains the node representations by learning the topological structure and node features in the neighborhood [13]. SGCN [14] explicits heterogeneous information of graph semantics via meta-paths. Besides those methods, GAT [31] and GLCN [32] utilize the self-attention mechanism or feature similarity to reweight the edges of original graphs.

For the attribute graphs, a series of effective deep convolutional network methods have been proposed. Zhuang and Ma [33] advanced the local and global consistency by using the multi-view input and unsupervised temporal loss function. However, the graphs contain many noise or hub nodes which make the graph representation confusing or misleading [34]. Similar to our paper, *k*NN-GCN model [18] constructed a dense similarity matrix with cosine similarity for each node pair. On the basis of this paper, we further propose a graph convolutional network with multi-similarity attributed matrices fusion which considers the topological properties and the node feature similarity matrices.

3 The proposed method

In this section, we introduce the proposed MSF-GCN which apply graph neural network along with the multi-similarity fusion for node classification in the attributed graphs. First, we give an overview of the paper in Sect. 3.1. Then, we describe the core components of MSF-GCN in detail, i.e., *graph attention* component in Sect. 3.2, *multi-similarity attributed graph construction* component in Sect. 3.3 and *fusion* component in Sect. 3.4, respectively. Finally, we provide the model training in Sect. 3.5.

3.1 Overview

Given an attribute graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$ is the set of n nodes, $\mathcal{E}$ is the set of connections between them, $\mathcal{X} \in \mathbb{R}^{n\times d}$ is the set of features for each node, n represents the number of nodes, and v_i denotes the i-th node. In addition, given node v_i, let $x_i = \{x_{i1}, x_{i2}, \cdots, x_{id}\} \in \mathbb{R}^d$ be the attributed feature vector from node i and d is the number of features of each node. The graph $\mathcal{G}$ can also be represented by corresponding adjacency matrix $A \in \{0,1\}^{n\times n}$, where $a_{ij} = 1$ indicates that there is an edge between the node v_i and node v_j, otherwise, $a_{ij} = 0$. The actual label of a node i is denoted by an one-hot vector $y_i \in \mathbb{R}^C$ and C is the number of classes.

The general MSF-GCN framework is illustrated in Fig. 1. The key idea behind the MSF-GCN is that the node representation and node classification should be considered the topological properties but also the node attribute matrices. As shown in Fig. 1, there are one topological graph and two similarity graphs. Although there are only two similarity graphs, our study can be easily extended to multiple similarity. For the topological properties, we use a GAT-based module to obtain a general representation of the original graph in *graph attention* component. Then, to balance the information from graph structure and node features, we construct two k-nearest-neighbor (kNN) graph based on node features $\mathcal{X}$ with two similarity measures in *multi-similarity attributed graphs construction* component. Further, considering that different similarity can construct different similarity matrices, we construct two similarity matrices to get the adjacency matrices. To capture the complex pairwise feature similarity relations, we utilize the self-attention network to balance the influence between the graph structure and node features for each similarity matrix, respectively. Also, we develop a gated-fusion network to measure the importance of cosine similarity vector and heat kernel vector and aggregates them together in *fusion* component. After that, we add them into the training set to update the parameters.

3.2 Graph attention component

In recent years, some baseline methods such as GCN [30] and GAT [31] have achieved remarkable success in different graph learning tasks. In this paper, our graph attention component is a GAT-based module that converts each node to a low-dimensional representation with attention aggregation on topological properties of neighboring nodes. Specifically, we use a neighborhood aggregation strategy to iteratively obtain the vector of node i by fusing vectors of its neighbors $\mathcal{N}(i)$. After L iterations, we can obtain the topological information with L-hop neighborhood. The representation of node i in l-layer can be defined as:

$$h_i^{(l)} = \mathrm{ReLU}^{(l)}\left(h_j^{(l-1)}|j \in \mathcal{N}(i)\right), l \in \{1, \cdots, L\}, \tag{1}$$

where $h_i^{(l)}$ is the representation vector of node i in the l-layer, $h_i^{(0)} = x_i$ is the feature vector of node i. In order to learn the higher order of node i within the graph topological properties, a self-attention mechanism is proposed to iteratively aggregate representation of its neighbor nodes $\mathcal{N}(i)$. Following the idea of graph attention network [31], the attention coefficients can be calculated as:

$$a_{ij}^{(l)} = \mathrm{Atten}\left(W_a^{(l-1)} e_i^{(l-1)}, W_a^{(l-1)} e_j^{(l-1)}\right), \tag{2}$$

where $a_{ij}^{(l)}$ is used to determine the importance of neighbor node j to node i in the l-1 layer, $W_a^{(l-1)}$ is the linear

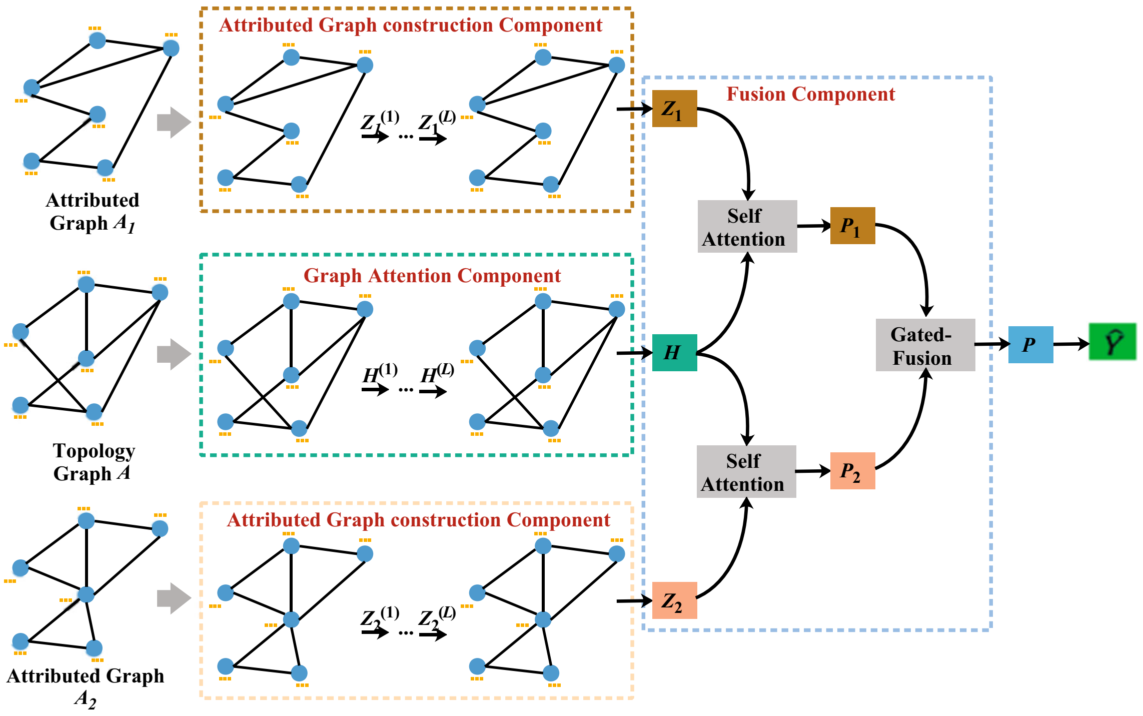

Fig. 1 Overview framework of the proposed MSF-GCN model for node classification

transformation of the weight matrix. For different nodes, the informativeness of neighbor nodes of each node could vary a lot in different layers. To catch the importance of different neighbor nodes, a softmax function is applied to normalize neighbor node j of the node i:

$$\alpha_{ij}^{(l)} = \mathrm{softmax}\left(a_{ij}^{(l)}\right) = \frac{\exp\left(a_{ij}^{(l)}\right)}{\sum_{k\in\mathcal{N}(i)} \exp\left(a_{ik}^{(l)}\right)}. \tag{3}$$

In this paper, we use the MLP with the learnable projection matrix W_{atten} to parameterize the weight vector. Followed by a LeakyReLU nonlinearity (with negative input slop $\beta = 0.2$) is utilized to compute the coefficients by the attention mechanism:

$$\alpha_{ij}^{(l)} = \frac{\exp\left(\mathrm{LeakyReLU}(W_{atten}^{(l-1)}\left[W_a^{(l-1)}e_i^{(l-1)} \| W_a^{(l-1)}h_j^{(l-1)}\right]\right)}{\sum_{k\in\mathcal{N}(i)} \exp\left(\mathrm{LeakyReLU}\left(W_{atten}^{(l-1)}\left[W_a^{(l-1)}e_i^{(l-1)} \| W_a^{(l-1)}e_k^{(l-1)}\right]\right)\right)}, \tag{4}$$

where $\|$ is the concatenation operation.

Then, we use all attention coefficients of the neighbors $\mathcal{N}(i)$ for node i to update the representation vector of node i:

$$h_i^{(l)} = \sigma\left(\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l-1)} W_h^{(l-1)} e_j^{(l-1)}\right), \tag{5}$$

where $\sigma(\cdot)$ is a nonlinearity ReLU function, W_h is a learnable projection matrix. Similar to most of graph self-attention based models [31, 35], we utilize the multi-head attention to stabilize the training of the graph attention component.

$$h_i^{(l)} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in\mathcal{N}(i)} \alpha_{ij}^{(l-1)} W_k^{(l-1)} e_j^{(l-1)}\right), \tag{6}$$

where K is the number of heads. Meanwhile, for each head, we use different project matrices, respectively. The graph attention component is illustrated in Fig. 2. Every node embedding is aggregated by its neighbor nodes. Since concatenation is no longer sensible, we utilize the mean of all heads to aggregate the multi-head graph attention results. Once the forward propagation of graph attention component has finished, we obtain the final representation matrix $H^{(l)} = \{h_1^{(l)}, h_2^{(l)}, \cdots, h_n^{(l)}\}$.

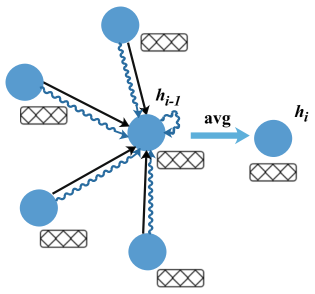

Fig. 2 Illustration of graph attention component

3.3 Multi-similarity attributed graph construction component

In this section, based on the different similarity and the topological representation of each node, we propose a multi-similarity attributed graphs construction component to integrate the node attribute features with topological features.

In order to catch the feature similarity, we convert the matrix $X = \{x_1, x_2, \cdots, x_n\}$ into two different similarity metrics with kNN graphs, respectively. Actually, there are many ways to construct different similarity matrices. Moreover, different semantic similarity matrix can be constructed based on different similarities. In this paper we choose two popular ones, in which the similarities between node i and j can be defined as:

For a given node feature pair (x_i, x_j), the cosine similarity can be calculated as follows:

$$S_1(x_i, x_j) = \frac{x_i^T x_j}{\|x_i\|_2 \|x_j\|_2}. \tag{7}$$

Similar with the cosine similarity, heat kernel similarity [36] can be calculated as follows:

$$S_2(x_i, x_j) = \exp\left(\frac{\|x_i - x_j\|^2}{t}\right), \tag{8}$$

where t is the time parameter in heat similarity and we set $t = 2$.

Then, we set k nearest neighbors with different similarity metrics for each node and obtain two kNN graphs. We denote the adjacency matrix of two similarity graphs as A_1 and A_2, respectively. After building the similarity graph, we feed the graphs into a simple GCN model [30], respectively:

$$Z_f^{(l)} = \sigma\left(\widetilde{D}_f^{-\frac{1}{2}} \widetilde{A}_f \widetilde{D}_f^{-\frac{1}{2}} z_f^{(l-1)} W_f\right), \tag{9}$$

where $\widetilde{A}_f = A_f + I_f$ (f=1, 2 and I_f is an identity matrix), $\widetilde{D}_f$ is the diagonal degree matrix of A_f, W_f is the project weight matrix and σ is the nonlinear activation function ReLU.

3.4 Fusion component

Taking topological representation matrix and two similarity representation vectors as inputs, we resort to the self-attention network [37] to catch the impact of the node features with topological features. The self-attention network has widely used in language model, neural machine translation and abstractive summarization. The core of transformer network is three matrices from input matrices.

The input of self-attention network consists of query, key and value matrices. First, we have to calculate the dot product between query matrix Q and key matrix K. In order to prevent the result from being too large, it is divided by a scale $\sqrt{d_k}$, where d_k is the query of dimension. Then, the results is normalized to probability distribution by softmax operation to obtain the weight one the value matrix V. The scaled dot-product attention can be calculated as:

$$\text{Self\_Atten}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{10}$$

For graph attention component, we simply denote the topological matrix as $H = \{h_1^{(L)}, h_2^{(L)}, \cdots, h_n^{(L)}\}$, while the feature similarity representations $Z_f = \{z_{f1}^{(L)}, z_{f2}^{(L)}, \cdots, z_{fn}^{(L)}\}$ $(f = 1, 2)$ through the output of multi-similarity feature graph construction component. In our context, we utilize the topological embedding to query feature similarity embedding, where the query matrix Q_f are calculated by H, the key matrix K_f and value matrix V_f are calculated by Z_f. Specifically, we transform H and Z_f to the same latent space through nonlinear conversion:

$$Q_f = \text{ReLU}(HW^Q), \tag{11a}$$

$$K_f = \text{ReLU}(Z_f W^K), \tag{11b}$$

$$V_f = \text{ReLU}(Z_f W^V), \tag{11c}$$

where W^Q, W^K and W^V are the project matrices and shared by all nodes. With the learned parameters as attention weights, we can obtain the effect of the topological representation and the feature representation as follows:

$$P_f = \text{SelfAtten}(Q_f, K_f, V_f), \tag{12}$$

To combine the different feature representation of each node, we use gate-fusion network to measure the importance of cosine similarity and heat kernel similarity vector and aggregates them together. Given the cosine similarity matrix P_1 and heat kernel matrix P_2 as input, the gate vector F is used to control the contribution between two different similarities:

$$F = \text{sigmoid}(W_1 P_1 + W_2 P_2 + b_f), \tag{13}$$

where W_1, W_2, b_f are the learnable projection parameters. The final output of the fusion matrix P is computed as:

$$P = (1 - F) \odot P_1 + F \odot P_2, \tag{14}$$

where $\odot$ is element-wise multiplication.

3.5 Model training strategy

In the attribute graph $\mathcal{G}$, given the training and validation node sets $\mathcal{V}_{tr}$ ($\mathcal{V}_{val} \in \mathcal{V}$ whose nodes have been labeled), the purpose of this paper is to predict the categories of unlabeled nodes from the test set $\mathcal{V}_{te} \in \mathcal{V}$. The validation set is used to adjust the model parameters. We apply a softmax function to get the output vector $Y \in \mathbb{R}^{n \times c}$ of our model, i.e., $\widehat{Y} = \text{softmax}(P)$, where $\widehat{y}_i = \{\widehat{y}_{i1}, \widehat{y}_{i2}, \ldots, \widehat{y}_{iC}\} \in \widehat{Y}$ is the probability vector of node i appearing in each category. Then the loss function is defined as the cross-entropy of the probability and the ground-truth on the training set $\mathcal{V}_{tr}$, and it can be calculated as follows:

$$\mathcal{L} = \frac{1}{|\mathcal{V}_{tr}|} \sum_{v_i \in \mathcal{V}_{tr}} (y_{ik} == 1) \ln \widehat{y}_{ik}. \tag{15}$$

This loss function makes nodes that are topological close with multi-feature similarity matrices fusion, which confirms our idea for utilizing the feature similarity for graph representation. All parameters can be randomly initialized and jointly optimized using the back-propagation training processing. We summarize the optimization in Algorithm 1. At each iteration of MSF-GCN algorithm, we training GAT model and two similarity attribute matrices with Eq. (4) and Eq. (9), respectively. Then, a gated-fusion network is used to merge the cosine similarity vector and heat kernel vector. Let $f(\theta) = \mathcal{L}$ and $f_i(\theta)$ is ith out of $f(\theta)$. After that, the classification model is retrained with the full-gradient $\bigtriangledown \mathcal{L}(\theta) = \frac{1}{|\mathcal{V}_{tr}|} \sum_{v_i \in \mathcal{V}_{tr}} (f_i(\theta), y_i)$, where θ is the parameters of MSF-GCN algorithm.

Algorithm 1 MSF-GCN Algorithm

```
Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ and its adjacency matrix $A$, Epochs $T$, training nodes $\mathcal{V}_{tr}$,
  learning rate $\eta$.
Output: Model parameters $\theta$.
  Set $H^{(0)} = \mathcal{X}$
  Random initilize parameters $\theta$
  Obtain the similarity Matrices $A_1$ and $A_2$ using Equation (7) and (8)
  for each training epoch t $\leq$ T do
    for $l = 1, \cdots, L$ do
      Train GAT model on $A$ by Equation (4)
      Calculate $Z_1^{(l)}$ and $Z_2^{(l)}$ on $A_1$ and $A_2$ using Equation (9)
    end for
    Calculate $P_1$ and $P_2$ using Equation (12)
    Calculate $P$ with gated-fusion using Equation (14)
    Update parameters by minimizing $\bigtriangledown \mathcal{L}(\theta)$
  end for
```

4 Experiments

In this section, we first describe the experiments setting from Sect. 4.1, including dataset description, baseline methods, and experimental setups, followed by the main node classification results (Sect. 4.2). Finally, we study the impact of different learning rate, hyper-parameters and different components in Sect. 4.3–4.5, respectively.

4.1 Experimental setup

4.1.1 Datasets

The statistic properties of all datasets have been summarized in Table 1. Cora, Citeseer and Pubmed [38] are the citation network datasets, where nodes denote the documents and edges are citation links. For the Cora and Citeseer datasets, the node feature are the bag-of-words representation of each documents. Instead of bag-of-words value, the node features in Pubmed dataset are denoted as the Term Frequency-Inverse Document Frequency (TF-IDF) of each words in the document. BlogCatalog is a blog social network, which contains 5,196 users and 171,743 interactions between the users, and 8,189 attributes denoting the keywords of their blogs.

4.1.2 Baselines

To evaluate the performance of the proposed MSF-GCN framework, we choose five state-of-the-art representative baselines:

- *LP* [39] is a non-GCN model that use belief propagation for supervised node classification on the graphs.
- *SemiEmb* [40] is semi-supervised learning method which combined an embedding-based regularizer with a supervised learner.
- *GCN* [30] is the most representative graph convolutional network-based model for node representation learning. It is widely used as a GNN baseline.
- *GAT* [31] is a self-attention-based GNN method which learn different weights to different neighbor nodes. It is another widely used as a GNN baseline.

Table 1 Summary statistics of attribute graph datasets

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,708	5,429	1433	7
Citeseer	3,327	4,732	3703	6
Pubmed	19,717	44,338	500	3
BlogCatalog	5,196	171,743	8,189	6

- *kNN-GCN* [18] is a node representation learning method which takes the *k*-nearest neighbor graph from the node features as input.
- *DGCN* [33] is a general GNN model which learn the local consistency and global consistency with graph adjacency matrix and positive mutual information matrix.
- *GraphSAGE* [13] is a general GNN model which samples the neighbor nodes randomly and aggregate those neighbors by average function.

4.1.3 Experiment settings

Our model and all neural-based baselines are defined and trained on a Windows server with 3.60 GHz Intel I9-9900k CPU and 11 GB NVIDIA GeForce RTX 2080Ti GPU, and implemented with the PyTorch [42] and PyTorch Geometric [43] libraries. In fact, there are many hyper-parameters in this paper and the baselines, i.e., batching size and learning rate. These parameters are commonly determined by several experimental trails or hard-tuning [44]. Throughout all experiments we use Adam optimizer with initial rate of 0.005 for 300 epochs. We set the number of layers L in all GCN is 2 with hidden layer units 128, dropout rate 0.5. In *k*NN-GCN and our proposed methods, we set the number of nearest neighbors to 10. In GAT and our proposed method, the number of heads is set to 8. For other baselines, we utilize the default parameter setting. We use *Accuracy* to evaluate the performance of baselines and our proposed methods.

4.2 Performance evaluation

For each dataset, we follow the widely used semi-supervised setting in [30, 31] given the node set $\mathcal{V}$, we select 20 nodes in each class for training set, 500 and 1000 nodes as the validation set, test set and training set, respectively. Each experiment is run 10 times. The average accuracy with standard deviation (Std for short) is reported in Table 2. We highlight the best results of all models in boldface. Note that we use the suggested parameters for both GCN, GAT, *k*NN-GCN, DGCN and GraphSAGE as suggested by the authors in their works, respectively.

As shown in Table 2, NSF-GCN achieves state-of-the-art node classification performance on 4 out of 5 benchmarks. NSF-GCN especially outperforms the baselines with a large margin in the *Pubmed* and *BlogCatalog* datasets. To be specific, considering the stability of the all algorithms, MSF-GCN achieves the best performance on all datasets. The results demonstrate the effectiveness of the proposed MSF-GCN method.

Second, NSF-GCN consistently outperforms GAT and *k*NN-GCN on all datasets, indicating that the effectiveness of the fusion mechanism in MSF-GCN with topological properties and node attribute features, because it can extract more useful information than only topological properties GAT and node attribute features *k*NN-GCN, respectively. MSF-GCN further outperformed those structure-only methods GCN, GAT, DGCN and GraphSAGE, which proves the significance of joint learning latent attribute features and structure embedding. This is because when the structure information has many noisy and misleading. In this case, the node feature similarity can provide strong guidance for the training of the model and improve the robustness of the model.

Third, the GCN-based methods (such as GCN, GAT, *k*NN-GCN, DGCN, GraphSAGE, and our MSF-GCN) outperform the non-GCN based methods (such as LP and SemiEmb), which indicates that structure information is not very useful for the attributed graphs.

Moreover, compared with structure-only methods, the improvement of *k*NN-GCN and MSF-GCN are more substantial on the datasets *Pubmed* and *BlogCatalog* with better attributed feature graphs (*k*NN). This implies that *k*NN graph can catch more effective attributed features with TF-IDF and keywords attributes than the bag-of-words values.

4.3 Performance of different learning rates

To better understanding of the advantage of MSF-GCN method, we make an extensive experiment with the baselines under different label rates. We vary the label ratio of changed trainable nodes with 0.1%, 0.5%, 1%, 2% and 3%, respectively. To make better comparison, we include all listed baselines and use the default hyper-parameter setting in the authors implementations. All experiments are repeated 10 times, We test the performance of our method and baselines in distinct label rates in Fig. 3 with the average of *accuracy*.

As shown in Fig. 3, our proposed MSF-GCN method outperforms other state-of-the-art approaches in *Citeseer*, *Pubmed* and *BlogCatalog* dataset. In the *Cora* dataset, our MSF-GCN method achieves best performance with label

Table 2 Overall performance accuracy comparison (Accuracy±Std)

Dataset	Cora	Citeseer	Pubmed	BlogCatalog
LP	.7542±.0021	.5637±.0018	.6944±.0017	.5223±.0026
SemiEmb	.6374±.0027	.6454±.0022	.7344±.0018	.5531±.0031
GCN	.8136±.0033	.7148±.0038	.7874±.0022	.7443±.0036
GAT	.8286±.0074	.7241±.0068	.7905±.0032	.7506±.0048
*k*NN-GCN	.7908±.0022	.7105±.0020	.8084±.0011	.7736±.0014
DGCN	**.8346±.0026**	.7258±.0043	.7984±.0026	.7473±.0039
GraphSAGE	.8026±.0031	.7032±.0033	.7032±.0033	.7376±.0029
MSF-GCN	.8343±.0022	**.7324±.0019**	**.8228±.0009**	**.7849±.0012**

rate 0.1%. More Specifically, we make three observations from the results:

- It is worth noting that due to the low efficiency of label information dissemination, the performance of all method decreases significantly when label data is scarce. This signifies that the label information plays an important role for node classification.
- Node attributed graph-based methods (such as *k*NN-GCN and our MSF-GCN) tend to be superior to structure-only based methods especially on fewer labeled data with effective graph attribute features, demonstrating the importance of graph attribute features for node classification with few labeled nodes.
- Our MSF-GCN method leverages both the advantage of structure information and node attribute similarities, consistently outperforming other state-of-the-art approaches with different label rates. Moreover, it turns out that the lower label rate the graph has, the more important of effective attribute node similarity graph can produce.

4.4 Hyper-parameters sensitivity analysis

In this section, we mainly illustrate the impact of two important hyper-parameters, namely the number of hidden layers and the number of nearest neighbors. The results over two datasets *Citeseer* and *Pubmed* are shown in Fig. 4a, b in terms of *Average Accuracy*. Note that we kept all other parameters fixed when vary the number of hidden layers and the number of nearest neighbors, coordinately.

4.4.1 The number of hidden layers

As mentioned in Sect. 3, we need generate L layers with L-hop neighborhood. We conduct the experiments by setting different number of layers to obtain the performance variance over *Citeseer* and *Pubmed* datasets. The number of hidden layers varies from 1 to 4. As can be seen in Fig. 4, the results are first increased and then decreased. The best performance among the tested number of the hidden layers was 2 in *Citeseer* dataset and 3 in *Pubmed* dataset.

4.4.2 The number of nearest neighbors

we set k nearest neighbors with different similarity metrics. Thus, we conduct experiments to analyze the influence of different number of nearest neighbors. The number of nearest neighbors k is set to $\{3, 5, 10, 15, 20\}$. In general with the increase of nearest neighbors from 3 to 10 in *Citeseer* dataset and from 3 to 15 in *Pubmed* dataset, the value of accuracy rises steadily. However, when the number of nearest neighbors is more than 15, there is no further improvement in the performance. This result indicates that the number of nearest neighbors influences the classification accuracy.

4.5 Impact of different components

In the proposed MSF-GCN model, there are three core components: cosine similarity matrix fusion, heat kernel similarity matrix fusion and gated-fusion. To evaluate the contribution of three core components, we conduct experiments on different configuration of the MSF-GCN model as follows:

- *MSF-GCN-CS* is our proposed MSF-GCN model which is only aggregated with the cosine similarity matrix.
- *MSF-GCN-HKS* is our proposed MSF-GCN model which is only aggregated with the heat kernel similarity matrix.
- *MSF-GCN-GF* is our proposed MSF-GCN model which removes the gated-fusion component and concatenates different feature representations between P_1 and P_2.
- *MSF-GCN-TS* is our proposed MSF-GCN model which is aggregated with three matrices (cosine similarity, heat kernel and Pearson correlation coefficient).

Table 3 depicts the impact of different components. Note that we kept all other parameters fixed with the parameters setting in Sect. 4.1.3. We highlight the best

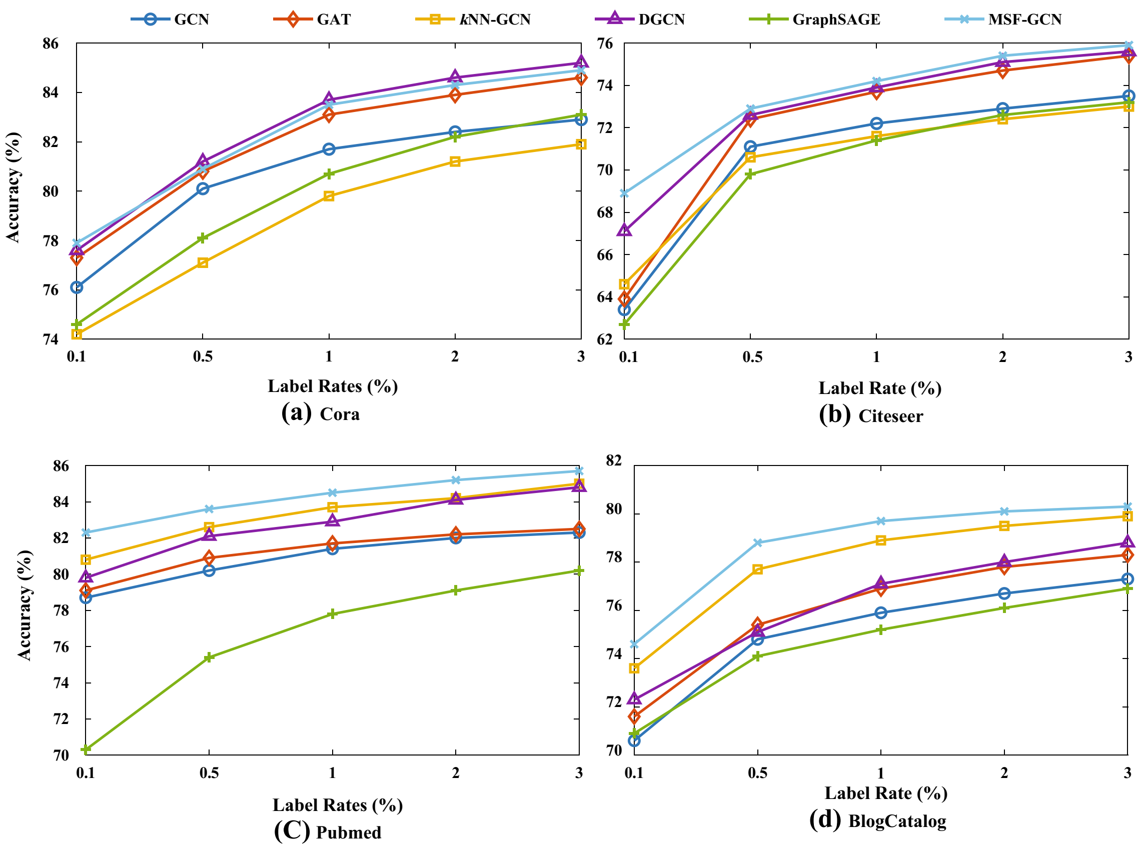

Fig. 3 Performance of different learning rates

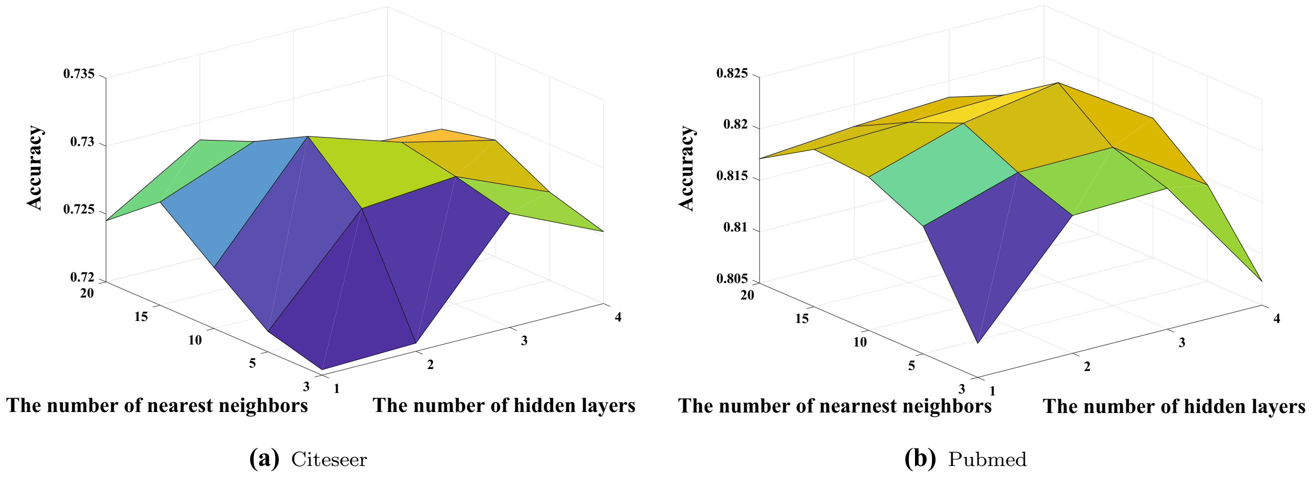

Fig. 4 Performance of different hyperparameters

Table 3 Impact of three core components

Methods	Cora	Citeseer	Pubmed	BlogCatalog
MSF-GCN-CS	.8281±.0024	.7288±.0022	.8163±.0011	.7792±.0014
MSF-GCN-HKS	.8234±.0018	.7236±.0017	.8217±.0008	.7844±.0011
MSF-GCN-GF	.8258±.0022	.7264±.0019	.8192±.0009	.7821±.0012
MSF-GCN	.8343±.0022	.7324±.0019	.8228±.0009	.7849±.0012
MSF-GCN-TS	**.8356±.0022**	**.7337±.0019**	**.8232±.0009**	**.7856±.0012**

results of all models in boldface. From the results in Table 3, we have four major observations as follows. First, without the multiple-similarity attributed graph construction, the performance of MSF-GCN decreases on four datasets. It demonstrates the effectiveness of the multiple-similarity attributed graph construction. It should be noted that node attributed similarity matrices play an important roles for node classification. Second, the different similarity matrices play a different role for different datasets. For example, cosine similarity matrix is more conducive to the application of bag-of-words attributes, while heat-kernel similarity matrix is good at the TF-IDF attributes. It validates the effectiveness of the proposed MSF-GCN architecture. Third, MSF-GCN with gated-fusion would bring more performance improvements the MSF-GCN without gated-fusion, which indicates that gated-fusion might be an important component. Forth,since different node attributed similarity matrices play different roles for node classification, the effect of aggregating more and more attributed similarity matrices shows a downward trend. This is determined by the upper limit of GCN or deep learning-based algorithms.

5 Conclusion

In this paper, we propose a novel graph convolutional network with multisimilarity attribute matrices fusion (i.e., MSF-GCN) for node classification. MSF-GCN is a deep and multiple component fusion framework that consists of three components: graph attention, multi-similarity attributed graph construction, fusion components. In particular, the framework constructs two k-nearest-neighbor (kNN) graphs based on node features X with two similarity measures. In addition, we apply the self-attention network to balance the disparity between the graph structure and node features for each similarity matrix. Our proposed approach outperforms the other state-of-the-art methods with different label rates. However, how to obtain the semantic information with semantic-paths and heterogeneous information [14] is still a challenge in node classification domain. Meanwhile, lack of explanation and suffering from high time complexity are common problems of the classification model. For future work, we intend to develop an explainable and scalable node classification model in social network applications.

Acknowledgements This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 72172057, 92046026, 71701089, in part by the Fundamental Research on Advanced Leading Technology Project of Jiangsu Province under Grant BK20192004C, BK20202011, the Jiangsu Provincial Key Research and Development Program under grant BE2020001-3, the International Innovation Cooperation Project of Jiangsu Province under Grant BZ2020008, and the National Center for International Joint Research on E-Business Information Processing under Grant 2013B01035.

Declaration

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Sun K, Zhu Z, Lin Z (2020) Multi-stage self-supervised learning for graph convolutional networks. In: Proceedings of 34th AAAI conference on artificial intelligence (AAAI 2020), AAAI, pp 5892–5899
2. Zhao J, Zhou Z, Guan Z, et al (2019) Intentgc: a scalable graph convolution framework fusing heterogeneous information for recommendation. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD 2019), ACM, pp 2347–2357
3. Huang Q, Wei J, Cai Y et al (2020) Aligned dual channel graph convolutional network for visual question answering. In: Proceedings of the 58th annual meeting of the association for computational linguistics (ACL 2020), ACL, pp 7166–7176
4. Qiao L, Zhao H, Huang X et al (2019) A structure-enriched neural network for network embedding. Exp Syst Appl 117:300–311
5. Gao H, Wang Z, Ji S (2018) Large-scale learnable graph convolutional networks. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD 2018), ACM, pp 1416–1424
6. Liu F, Xue S, Wu J et al (2020) Deep learning for community detection: progress, challenges and opportunities. In: Proceedings of the twenty-ninth international joint conference on artificial intelligence (IJCAI-20), pp 4981–4987
7. Su X, Xue S, Liu F et al (2021) A comprehensive survey on community detection with deep learning. arXiv preprint arXiv:2105.12584
8. Chami I, Ying Z, Re C et al (2019) Hyperbolic graph convolutional neural networks. In: Proceedings of advances in neural information processing systems 2019 (NeurIPS 2019), pp 4868–4879
9. Ma X, Wu J, Xue S et al (2021) A comprehensive survey on graph anomaly detection with deep learning. arXiv preprint arXiv:2106.07178
10. Cai H, Zheng VW, Chang KC (2018) A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans Knowl Data Eng 30(9):1616–1637
11. Zhang D, Yin J, Zhu X, Zhang C (2020) Network representation learning: a survey. IEEE Trans Big Data 6(1):3–28
12. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering.In: Proceedings of Advances in neural information processing systems (NeurIPS 2016), pp 3844–3852
13. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. In: Proceedings of advances in neural information processing systems (NeurIPS 2017), pp 1024–1034
14. Wu L, Li Z, Zhao H, et al (2021) Learning the Implicit Semantic Representation on Graph-Structured Data. arXiv preprint arXiv:2101.06471
15. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: Proceedings of international

conference on machine learning (ICML 2016),JMLR, pp 2014–2023
16. Jin W, Derr T, Wang Y et al (2020) Node similarity preserving graph convolutional networks. arXiv preprint arXiv:2011.09643
17. Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. arXiv preprint arXiv:1801.07606
18. Franceschi L, Niepert M, Pontil M (2019) Learning discrete structures for graph neural networks. In: Proceeding of the 36th international conference on machine learning (ICML 2019), pp 1972–1982
19. Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. In: Aggarwal CC (ed) Social network data analytics. Springer, Boston, pp 115–148
20. Tang J, Aggarwal C, Liu H (2016) Node classification in signed social networks. In: Proceedings of the 2016 SIAM international conference on data mining (SDM 2016), SIAM, pp 54–62
21. Neville J, Jensen D (2000) Iterative classification in relational data. In: Proceedings of AAAI 2000 workshop on learning statistical models from relational data, pp 13–20
22. Zhu X, Ghahramani Z, Lafferty JD (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: Proceedings of the 20th international conference on machine learning (ICML 2003), pp 912–919
23. Zhou D, Bousquet O, Lal T, et al (2004) Learning with local and global consistency. In: Proceedings of advances in neural information processing systems (NeurIPS 2003), pp 321–328
24. Wu J, Hong Z, Pan S et al (2014) Multi-graph-view learning for graph classification. In: 2014 IEEE international conference on data mining (ICDM 2014), IEEE, pp 590–599
25. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 2014), ACM, pp 701–710
26. Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 2016), ACM, pp 855–864
27. Chang S, Han W, Tang J, Aggarwal CC, Huang TS et al (2015) Heterogeneous network embedding via deep architectures. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining (SIGKDD 2015), pp 119–128
28. Dai H, Dai B, Song L (2016) Discriminative embeddings of latent variable models for structured data. In: Proceedings of international conference on machine learning (ICML 2016), JMLR, pp 2702–2711
29. Ou M, Cui P, Pei J, Zhang Z et al (2016) Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD 2016), ACM, pp 1105–1114
30. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: Proceeding of 5th International conference on learning representations (ICLR 2017), pp 1–13
31. Veličković P, Cucurull G, Casanova A et al (2018) Graph attention networks. In: Proceeding of 6th international conference on learning representations (ICLR 2018), pp 1–12 (2018)
32. Jiang B, Zhang Z, Lin D et al (2019) Semi-supervised learning with graph learning-convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR 2019, pp 11313–11320
33. Zhuang C, Ma Q (2018) Dual graph convolutional networks for graph-based semi-supervised classification. In: Proceedings of the 2018 world wide web conference (WWW 2018), ACM, pp 499–508
34. Xu K, Li C, Tian Y et al (2018) Representation learning on graphs with jumping knowledge networks. In: Proceedings of international conference on machine learning (ICML 2018), JMLR, pp 5453–5462
35. Wang X, Ji H, Shi C et al (2019) Heterogeneous graph attention network. In: Proceedings of The 2019 world wide web conference (WWW 2019), pp 2022–2032
36. Xiao B, Hancock ER, Wilson RC (2009) Graph characteristics from the heat kernel trace. Pattern Recogn 42(11):2589–606
37. Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. In: Proceeding of Advances in neural information processing systems (NeurIPS 2017), pp 5998–6008
38. Sen P, Namata G, Bilgic M et al (2008) Collective classification in network data. AI Mag 29(3):93–106
39. Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using gaussian fields and harmonic functions. In Proceedings of the 20th international conference on machine learning (ICML 2003), pp 912–919
40. Weston J, Ratle F, Mobahi H et al (2012) Deep learning via semi-supervised embedding. Neural networks, tricks of the trade. Springer, Berlin Heidelberg, pp 639–655
41. Hu F, Zhu Y, Wu S et al (2018) Hierarchical graph convolutional networks for semi-supervised node classification. In: Proceedings of the twenty-eighth international joint conference on artificial intelligence (IJCAI 2019), pp 4532–4539
42. Paszke A, Gross S, Massa F et al (2019) Pytorch: An imperative style, high-performance deep learning library. In: Proceedings of advances in neural information processing systems (NeurIPS 2019), pp 8026–8037
43. Fey M, Lenssen JE (2019) Fast graph representation learning with PyTorch Geometric. arXiv preprint arXiv:1903.02428
44. Tajbakhsh N, Shin J, Gurudu S et al (2016) Convolutional neural networks for medical image analysis: full training or fine tuning? IEEE Trans Med Imaging 35(5):1299–1312