Neurocomputing 453 (2021) 524–537

Contents lists available at ScienceDirect

Neurocomputing

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

M-GWNN: Multi-granularity graph wavelet neural networks for semi-supervised node classification

Wenjie Zheng, Fulan Qian *, Shu Zhao, Yanping Zhang

Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, China
School of Computer Science and Technology, Anhui University, Hefei 230601, China

ARTICLE INFO

Article history:
Received 5 July 2020
Revised 6 October 2020
Accepted 13 October 2020
Available online 22 October 2020
Communicated by Z. Wang

Keywords:
Graph neural networks
Graph wavelet neural network
Multi-granularity network

ABSTRACT

Graph convolutional neural networks (GCNs) based on spectral-domain have achieved impressive performance for semi-supervised node classification task. Recently, graph wavelet neural network (GWNN) has made a significant improvement for this task. However, GWNN is usually shallow based on a one- or two-hop neighborhood structure, making it unable to obtain sufficient global information to make it better. But, if GWNN merely stacks too many convolutional layers, it produces the phenomenon of the wavelet convolutional filters over-smoothing. To stack this challenge, we propose Multi-granularity Graph Wavelet Neural Networks (M-GWNN), a novel spectral GCNs architecture that leverages the proposed Louvain-variant algorithm and the jump connection to improve the ability of node representations for semi-supervised node classification. We first repeatedly apply the proposed Louvain-variant algorithm to aggregate nodes into supernodes to build a hierarchy of successively coarser graph, further refine the coarsened graph symmetrically back to the original by utilizing the jump connection. Moreover, during this process, multiple layers of GWNN are applied to propagate information across entire networks. The proposed M-GWNN efficiently captures node features and graph topological structures of varying granularity to obtain global information. Furthermore, M-GWNN effectively employs the jump connection to connect receptive fields of varying granularity to alleviate the speed of over-smoothing. Experiments on four benchmark datasets demonstrate the effectiveness of the proposed M-GWNN. Particularly, when only a few labeled nodes are provided on the NELL dataset, M-GWNN achieves up to an average 5.7% performance improvement compared with state-of-the-art methods.

1. Introduction

Since many non-Euclidean data that do not have a grid-like structure can only be represented in the form of a graph, extending neural networks to graph structure has attracted considerable attention from researchers. In this paper, we focus on the task of graph-based semi-supervised node classification using graph convolutional neural networks (GCNs) based on spectral-domain. Spectral CNN [1] first attempted to exploit graph Fourier to implement CNN [2] on graphs and maintain the property of CNN, which is the weight sharing parameter used in training. After that, spectral GCNs based on graph Fourier became increasingly popular. Defferrard et al. [3] utilized localized spectral Chebyshev filters to achieve the graph convolution operation for semi-supervised node classification task. Furthermore, GCN [4] demonstrated considerable power and achieved state-of-art performance on this task. Different from GCN [4], graph wavelet neural network (GWNN) [5] is also a spectral graph convolutional neural network that replaces graph Fourier with graph wavelet, which is sparser, more localized, less computationally expensive than Fourier.

However, we found that GWNN is usually shallow and follows a neighborhood aggregation based on one- or two–hop. Owing to the restricted receptive field, GWNN has a difficulty of obtaining sufficient global information to achieve better node representations. But, if GWNN merely stacks too many convolutional layers, it will reduce the performance because of the model produces the phenomenon of over-smoothing. some researches [6,7] proved that spectral GCNs models repeatedly apply Laplacian smoothing, the features of vertices in each connected component [1] of the graph will converge to the same value, and further cause the performance of the model learning node representations to significantly decrease.

* Corresponding author at: Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, China.
E-mail address: qianfulan@hotmail.com (F. Qian).

[1] In graph theory, the connected component of an undirected graph is a subgraph, in which any two vertices are connected by paths.

https://doi.org/10.1016/j.neucom.2020.10.033
0925-2312/

Recently, few embedding methods have proposed hierarchical approaches that capture global topological information of networks to learn node representations. For instance, HARP [8] and MILE [9] repeatedly compresses a network into a series of small networks based on two node collapsing schemes and learn node embeddings for each compressed network by existing network embedding methods. However, because HARP heuristically combines two closed nodes into a new one, the compressed network may not be able to display the global topology of the input network. Other recent approaches, such as graph U-Net [10] and SAGPool [11], which selectively remove some nodes of graph to compress graph and maximize retention of topological information of hierarchical graph.

In this paper, we utilize the concept of community structure existing in real networks [12–16] as an effective strategy to realize hierarchical graph coarsening to capture global information of the graph. Nodes that exist in a single community have similar types and are closely connected [17]. Therefore, placing nodes in the same community closely in the embedding space is benefit for graph mining tasks [16], such as semi-supervised node classification. To be specific, dividing a graph recursively into a series of coarser graphs (communities) helps capture the similarities between nodes at different levels of proximity: recursive partitioning essentially creates a hierarchy of communities in a network, where (a) nodes in the same community at the top level of the hierarchy represent similar clusters of nodes with higher-order structure and (b) the lower communities in the hierarchy maintain a one-hop (neighbor) relationship between connected pairs of nodes. Therefore, generating embedding vectors from the presence of nodes in the community at different granularities can maintain embedding quality and is suitable for the task of semi-supervised node classification. Moreover, the Louvain algorithm [18] is a state-of-the-art community detection algorithm [19] based on hierarchical modularity optimization [20], which is used to measure the quality of the results of the community detection algorithm, and also can characterize the closeness of the communities. However, there are existing several problems that directly applying the Louvain algorithm into hierarchical graph coarsening: 1) the number of nodes in communities may exhibit significant inhomogeneity in each coarsening; 2) the uncontrollability of the aggregation times of the Louvain algorithm; 3) there are maybe existing several nodes that are not assigned to any community during the coarsening. Therefore, to better fuse GWNN to implement a hierarchical approach, we propose the Louvain-variant algorithm building on Louvain.

Formally, we propose Multi-granularity Graph Wavelet Neural Networks (M-GWNN), a novel spectral GCNs architecture that leverages the proposed Louvain-variant algorithm and the jump connection to improve the ability of node representations for semi-supervised node classification. As pictured in Fig. 1, M-GWNN mainly consists of several graph coarsening and refining procedures and one output layer. M-GWNN first repeatedly applies the proposed Louvain-variant algorithm to aggregate nodes into supernodes, further refines the coarsened graph symmetrically back to the original by utilizing the jump connection. Moreover, during this process, multiple layers of GWNN are applied to propagate information across entire networks. The core idea of the M-GWNN model is effectively widening the receptive field of the wavelet convolutional filters by capturing node features and graph topological information of varying granularity, which employs the proposed Louvain-variant algorithm to build a hierarchy of successively coarser graph. Furthermore, the M-GWNN utilizes the jump connection to connect receptive fields of varying granularity to efficiently alleviate the speed of over-smoothing, which combining the graph coarsening and symmetrical refining procedures.

Overall, the main contributions of the proposed M-GWNN for semi-supervised node classification are summarized as follows:

- We study GWNN from a multi-granularity perspective, which, to the best of our knowledge, has not been studied before. Compared to related work, we propose the Louvain-variant algorithm to capture graph information of varying granularity to obtain global information of the graph. Furthermore, we employ the jump connection to connect receptive fields of varying granularity to stack the challenge of wavelet convolutional filters over-smoothing.

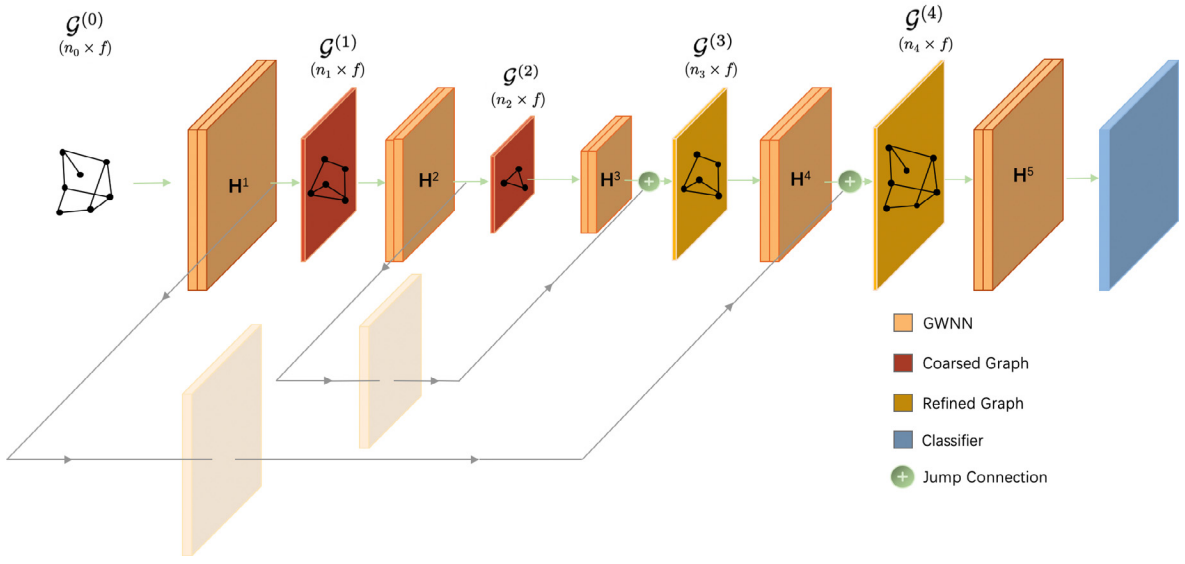

Fig. 1. The architecture of the proposed M-GWNN for semi-supervised node classification. In this illustration, there include two graph coarsening procedures, two graph refining procedures, and one output layer. At layer $\ell$of coarsening procedure, M-GWNN first generates hidden node embeddings $\mathbf{H}^{\ell+1}$, then produces a coarser graph $\mathcal{G}^{(\ell+1)}$ of $n_{\ell+1}$ supernodes with f–dimension, operations in refining procedure is similar.

- We propose the Louvain-variant algorithm, which is usually used in unsupervised scenarios to solve the semi-supervised problem for the first time. Compared to previous work, a hierarchical clustering method that employs the concept of community structure builds a hierarchy of coarser graph. Moreover, every supernode represents a local topological structure of the original graph, which efficiently captures global information.
- We conduct extensive evaluation over four standard datasets for semi-supervised node classification task, and comprehensive experiments show the effectiveness of the proposed M-GWNN model. Especially, compared to state-of-the-art methods, M-GWNN achieves an average error reduction of up to 5.7% in the NELL dataset.

2. Related works

In this section, we briefly review previous work on graph convolutional neural networks, multi-layer graph representation learning, and algorithms based on graph reduction mechanisms.

2.1. Graph convolutional neural networks based on spectral-domain

Spectral GCNs define convolution by the convolution theorem, and mainly include building upon graph Fourier convolution or graph wavelet convolution.

Spectral graph Fourier. Kipf and Welling [4] utilized first-order spectral Chebyshev filters to achieve graph convolution for semi-supervised node classification task. Zhuang and Ma [21] designed the DGCN by combining the graph adjacency matrix and positive pointwise mutual information matrix to embed both local- and global-consistency knowledge, respectively. Owing to fill the research gap of confidence estimation in the context of GCN, Vashishth et al. [22] proposed ConfGCN that utilizes per node label distribution and confidences in the graph. N-GCN [23] marries semi-supervised and unsupervised learning in the graph to improve the challenging semi-supervised node classification task. Additionally, spectral-domain-based graph convolutional networks have wide landing applications, such as human-object interactions[24,25] and disease prediction [26,27].

Spectral graph wavelet. Hammond et al. [28] presented the spectral graph wavelet transform, which is constructed by localizing the scaled wavelet operator. Moreover, they use the truncated Chebyshev polynomial approximation to avoid the heavy computation of the eigendecomposition of the Laplacian matrix. Owing to solving the design problem of spectral filters in the construction of dictionaries of atoms, Shuman et al. [29] defined spectrum-adapted tight graph wavelet frames in the graph spectral domain. Donnat et al. [30] introduced a method that represents the network neighbor of each node by a low-dimensional embedding by utilizing heat wavelet diffusion patterns and learns structural node embeddings based on structural similarity in graphs. Xu et al. [5] proposed a graph wavelet neural network (GWNN) to perform graph convolution, which employs a graph wavelet as a set of bases taking the place of graph Fourier.

2.2. Multi-layer graph representation learning

Recently, there has been some work in learning multi-layer graph representations. Ying et al. [31] proposed a method for multi-layer graph representation learning through a differentiable pooling procedure, which continually learns subgraphs by learning graph representations of an additional convolution layer and coarsened the graph on this basis. Inspired by this success, Ma and Wang [32] propose a novel pooling operator "eigenPooling", which is based on the Fourier transform of a graph and applies the topological structure and node features of the subgraph to learn supernode representations. However, these methods that use multi-layer graph representation learning based on a graph pooling mechanism cannot generate global node embeddings and cannot be directly considered in the node classification task. HARP [8] introduced a generic framework to scale up graph embedding techniques, which are considered black boxes. The method contains graph coarsening and then utilizes existing unsupervised methods, such as DeepWalk [33] or node2vec [34], to learn node embeddings on the coarsest graph, which are further restored from the coarsened graph to the original to obtain the final graph representation. Unfortunately, it cannot capture node attribute information to the maximum extent possible and is unable to learn node classification in the end-to-end scenario.

2.3. Graph compression

As the quantity of graph data that we generate or collect in research increases continuously, proposed approaches leverage graph compression without losing much graph information to accomplish graph mining tasks such as clustering [35] and community detection [36]. Various graph coarsening techniques have been studied continually [37–39], and we generalize that there are two main methods: graph coarsening and simplification- or sparsification-based graph sampling. The first category is the most popular and is based on grouping or aggregation. GraphSAGE [40] and FastGCN [41] define the importance sampling operation by first clustering the node embeddings and then applying a graph pooling operation. Their design can reduce the computational footprint of GCN but lose key node information during sampling. Graph dedensification [42] is an edge-based grouping method to compress neighborhoods around high-degree nodes. The next category simplifies the input graph by removing fewer "significant" nodes or edges through a variety of techniques, resulting in sparse subgraphs that approximate the attributes of the original graph. For example, graph sampling was fully discussed at the KDD conference [43]. However, these methods cannot learn sufficient information on graph structure and node attribution.

3. Preliminaries

3.1. Notation introduction

Table 1 shows the main notations used in this paper.

3.2. Problem definition

We denote a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A}, \mathbf{X})$, where $|\mathcal{V}| = n = |\mathcal{V}_L| + |\mathcal{V}_U|$ is the vertex set of labeled $(\mathcal{V}_L)$ and unlabeled $(\mathcal{V}_U)$ nodes in a graph. Generally, $|\mathcal{V}_L| \ll |\mathcal{V}|$, the goal of semi-

Table 1
Notations.

Symbol	Description
ζ	The total layers of the M-GWNN network
ℓ	The ℓ-th layer of graph coarsening procedure of the M-GWNN network
m	The maximum coarsening level
v_i	A vertex in the set of vertices $\mathcal{V}$
n_ℓ	The number of nodes for the ℓ-th layer
$\mathcal{G}^{(\ell)}$	A weighted undirected graph for the ℓ-th layer
$\mathbf{A}^\ell$	Adjacency matrix for graph $\mathcal{G}^{(\ell)}$
$\mathbf{X}^\ell$	Graph representations for graph $\mathcal{G}^{(\ell)}$
$\mathbf{H}^{\ell+1}$	Hidden node embeddings of $\mathcal{G}^{(\ell)}$ produced by GWNN model
$\mathbf{M}^{\ell,\ell+1}$	Matching matrix from graph $\mathcal{G}^{(\ell)}$ to $\mathcal{G}^{(\ell+1)}$

supervised node classification is to assign labels for all unlabeled nodes $\mathcal{V}_U$. $\mathcal{E}$ is the edge set, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix of a graph with $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}, \mathbf{X}^{\ell} \in \mathbb{R}^{n_{\ell}\times f_{\ell}}$ is the graph representations of n_{ℓ} data vectors in f_{ℓ} dimension at the ℓlayer (including original, coarse, and refined graph), and each column of $\mathbf{X}$ is a signal x defined over nodes. We set the output dimension $f_{\ell} = f$ for all layers in this paper. The graph Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D} = \mathrm{diag}(d_1, d_2 \ldots, d_n)$ is a diagonal degree matrix, and the complete set of orthonormal eigenvectors $\mathbf{U} = (u_1, u_2, \ldots, u_n)$ of $\mathbf{L}$, ordered by its nonnegative eigenvalues $\{\lambda_i\}_{i=1}^{n}$ where defined as the frequencies of graph. For the proposed M-GWNN framework, $\mathcal{G}^{(\ell)}$ with n_{ℓ} nodes is a graph at layer ℓ. The adjacency matrix and hidden layer representation of $\mathcal{G}^{(\ell)}$ are respectively represented by $\mathbf{A}^{\ell}$ and $\mathbf{H}^{\ell}$. For instance, M-GWNN is illustrated in Fig. 1. Giving a graph $\mathcal{G}^{(0)} = (\mathcal{V}^{(0)}, \mathcal{E}^{(0)}, \mathbf{A}^0, \mathbf{X}^0)$ as the input, $\mathbf{H}^2$ is the graph representation in the second graph $\mathcal{G}^{(1)}$ generated by the GWNN. After a seriesof graph coarsening operations and graph refining operations, correspondingly, $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(3)}$ have the same graph topological structure $\mathbf{A}^1$.

3.3. Graph wavelet neural network (GWNN)

The GWNN [5] proposes a novel spectral graph convolutional neural network that replaces the Fourier basis with a wavelet basis, which is sparser, more localized, with lower computational cost than the Fourier basis, and effectively improves the semi-supervised node classification task. Interestingly, our work is based on GWNN. Therefore, in this section, we briefly review the GWNN model in our work. GWNN defines wavelet basis $\phi_s = (\phi_{s1}, \phi_{s2}, \ldots, \phi_{sn})$ at scale s, where s is a scaling parameter. Mathematically, each wavelet ϕ_{si} can be written as

$$\phi_{si} = \mathbf{U}\mathbf{G}_{si}\mathbf{U}^T \tag{1}$$

where $\mathbf{G}_s = \mathrm{diag}(g(s\lambda 1), \ldots, g(s\lambda_n))$ is a scaling matrix and $g(s\lambda_i) = e^{\lambda_i s}$. Furthermore, the graph wavelet transform is defined as $x^* = \phi_s^{-1}x$, and the inverse wavelet transform is $x = \phi_s x^*$. Note that ϕ_s^{-1} utilizes the $g(-s\lambda_i)$ instead of the $g(s\lambda_i)$ corresponding to [30]. Generally, given graph adjacency matrix $\mathbf{A}^i$ and graph representations $\mathbf{X}^i$ as input, the GWNN message propagation rule at layer ℓ is defined as

$$\textit{feature transformation}: \ \mathbf{H}^{\ell'} = \mathbf{X}^{\ell}\mathbf{W} \tag{2}$$

$$\textit{graph convolution}: \ \mathbf{H}^{\ell+1} = \mathrm{ReLU}(\phi_s \mathbf{F}^{\ell} \phi_s^{-1} \mathbf{H}^{\ell'}) \tag{3}$$

where $\mathbf{W}$ is the trainable weight matrix for feature transformation. $\mathbf{F}^{\ell}$ is the diagonal matrix for the graph convolutional kernel. $\mathbf{H}^{\ell+1} \in \mathbb{R}^{n_{\ell+1}\times f}$ is a hidden node embedding via the resulting GWNN module. Specifically, $\mathbf{X}^0 \in \mathbb{R}^{n_0\times f}$ is the input node feature of the original graph.

3.4. The Louvain algorithm for community detection

In this section, we briefly review the Louvain algorithm, which is a state-of-the-art community detection algorithm. It is well known that modularity [20] is an evaluation index between −1 and 1 that measures the effect of community detection. Formally, modularity Q is defined as

$$Q = \sum_{C}\left[\frac{\sum_{in}}{2p} - \left(\frac{\sum_{tot}}{2p}\right)^2\right] \tag{4}$$

where $\sum_{in}$ is the weight of the edges within community $C, (C \subset \mathcal{G})$. $\sum_{tot}$ represents the weight of the edge connected with the vertexes inside community C, including the edge inside C and the edge outside C. p is the total of the weights of all the edges in $\mathcal{G}$. Moreover, the gain of modularity [44] obtained by moving node i into community C can be easily computed as

$$\Delta Q = \left[\frac{\sum_{in} + k_{i,in}}{2p} - \left(\frac{\sum_{tot} + k_i}{2p}\right)^2\right] - \left[\frac{\sum_{in}}{2p} - \left(\frac{\sum_{tot}}{2p}\right)^2 - \left(\frac{k_i}{2p}\right)^2\right] \tag{5}$$

where K_i is the sum of the weights of the edges associated with node i, and $k_{i,in}$ is the sum of the weights of the edges from i to nodes in the $\mathcal{G}$.

The Louvain algorithm is based on modularity optimization, and the main goal is to continuously divide communities until the entire graph modularity does not increase. The algorithm contains two repeated coarsening steps. Step one is modularity optimization. There are as many vertexes as there are communities in the initial partition. Then, each node is allocated to a community selected based on the gain of modularity ΔQ to maximize the modularity Q. Step two, community aggregation. At its core is the creation of a new graph that consists of vertexes from those communities that were previously discovered.

4. Multi-granularity graph wavelet neural networks (M-GWNN)

In this section, we discuss the proposed multi-granularity graph wavelet neural networks (M-GWNN) in detail.

4.1. Architecture

The goal of the M-GWNN is to learn an optimal graph representation for the GWNN [5] network. Fig. 1 shows the architecture of the M-GWNN. For an M-GWNN network, which contains three procedures, including the graph coarsening procedure, graph refining procedure, and output layer.

During the graph coarsening procedure, at the ℓ layer ($\ell = 0, 1, \ldots, m$. m is the maximum coarsening level), we first employ a GWNN module as formulated in Eqs. (2) and (3), which consists of the feature transformation and the graph convolution. The generated node embeddings $\mathbf{H}^{\ell+1}$ are stored, and we further apply the Louvain-variant algorithm to aggregate nodes to supernodes in the original graph $\mathcal{G}^{(\ell)}$. After the coarsening operation, every supernode represents a local topological structure of the original graph, therefore producing smaller graph $\mathcal{G}^{(\ell+1)}$ and learning novel graph representations $\mathbf{X}^{\ell+1} \in \mathbb{R}^{n_{\ell+1}\times f}$ with fewer nodes than $\mathcal{G}^{(\ell)}$. The $\mathbf{X}^{\ell+1}$ of graph $\mathcal{G}^{(\ell+1)}$ will be propagated into the next layer by repeating the above operation. Details can be seen in the next section.

During the graph refining procedure, symmetrically, at the $m + \ell$ layer, we first utilize GWNN to generate hidden node embeddings $\mathbf{H}^{m+\ell+1}$ and further refine the coarsened graph to restore the original graph. Additionally, to promote training in deeper networks via the model enables adequate information propagation from the coarsened graph to the refined part and alleviates the speed of the wavelet convolutional filters over-smoothing, we use the jump connection by following [45]. See more in Section 4.3.

Combining the graph coarsening operation with the graph refining operation efficiently captures varying granularity of node

features and graph topological information and further enables the acquisition of adequate global information where the M-GWNN alleviates the speed of the wavelet convolutional filters over-smoothing. In the last layer of the graph refining procedure, we obtain graph representations $\mathbf{X}^{\zeta-1}$ with dimensions $n_{\zeta-1} \times f$. Owing to implementing the semi-supervised node classification task, we add a softmax classifier layer on $\mathbf{X}^{\zeta-1}$. We elaborate on them in Section 4.4.

4.2. Graph coarsening procedure

The graph coarsening procedure contains two repeated operations: graph wavelet convolution and graph coarsening. Above all, at the ℓ layer (i.e., graph $\mathcal{G}^{(\ell)}$), we gain the graph topological structure $\mathbf{A}^{\ell}$ and employ a graph wavelet convolution to extract hidden node embeddings $\mathbf{H}^{\ell+1}$, where the information collected is from per node one-hop neighborhoods in the original graph as depicted in Eqs. (2) and (3). Next, we present a coarsening method to assign a target node with its neighbors according to the high modularity gain into a supernode based on the classic Louvain algorithm. Finally, according to the first and second operations, we produce the coarser graph $\mathcal{G}^{(\ell+1)}$ and obtain the graph representations $\mathbf{X}^{\ell+1}$.

Here, we first analyze the Louvain algorithm, which is widely used to solve the problem of community detection. In summary, it has two advantages, which arouse our work to try it to our graph coarsening procedure. Firstly, the Louvain algorithm employ the modularity, which is used to measure the quality of community detection algorithm results, and can characterize the closeness of the discovered community. Moreover, the Louvain algorithm further proposes that modularity can be used as an optimization function, which improves the modularity of the current community structure, further improves the quality of complete hierarchical structure for the network. Secondly, the Louvain algorithm is fast, where time complexity is linear.

However, there are existing several problems that directly applying the Louvain algorithm into hierarchical graph coarsening: 1) The number of nodes in communities may exhibit significant inhomogeneity in each coarsening. In general, the Louvain algorithm provides a powerful tool for the research of community detection as it enables a fast extraction method for the community structure, which doesn't consider the node amount of communities in each coarsening [18]. However, we need each community to be represented as a supernode, which represents a local structure of the original graph and further facilitates capturing global information on the graph. Too many nodes squeeze into a community will not be conducive to this operation. Therefore, we set a threshold manually to control the node amount of each community in each coarsening. 2) The uncontrollability of the aggregation times of the Louvain algorithm. To be specific, the Louvain algorithm only needs one or two aggregations to realize clustering [46]. It aggregates so few times that it runs counter to the multi-granularity scenario we are pursuing. Consequently, one of our improvements manually controls the level of Louvain coarsening. 3) Since the Louvain algorithm is adjusted for the first two problems, there are maybe existing remaining unmarked nodes that are not assigned to any community during each graph coarsening. We follow [47,9] and then employ the normalized heavy edge matching (NHEM) method using the following formula, which punishes the weights of edges connected to high-degree nodes.

$$r(v_i, v_t) = \frac{\mathbf{E}_{it}}{\sqrt{\mathbf{D}(v_i) \cdot \mathbf{D}(v_t)}} \tag{6}$$

where $\mathbf{E}_{i,t}$ is the weight of the edge between node v_i and v_t. $\mathbf{D}(\cdot)$ is the node weight. In Eq. (6), the weight of the edge derives from normalizing the degree of the two nodes associated with the edge.

Formally, we propose the Louvain-variant algorithm to achieve hierarchical graph coarsening. First, every node that is unmarked represents a community in the graph. Next, for each unmarked node v_i, we first consider one-hop neighborhoods j of v_i and then compute the gain of modularity ΔQ. Furthermore, v_i is divided into the community including j, where the gain is a positive maximum. If the gain is negative, node v_i remains in its original community. In addition, if the number of contained nodes of this j achieves the upper bound τ which we manually set, node v_i joins the community with the second peak gain. The process is used sequentially and repeatedly for nodes among the graph until modularity Q is not improved. Thus far, these nodes of the community are assigned as a supernode v_j in the coarser graph, and its edge weight to v_i is the sum of the weights of the nodes contained in v_j. We mark these nodes that are coarsened and keep other nodes unmarked, which avoids repeated grouping operations for all nodes in the graph. After that, we take an unmarked node v_i' and calculate the normalized weights of edges with all its one-hop neighbors, which are the nodes belonging to the community that has been grouped. Then, node v_i' is marked and allocated into the community of neighbors with low weights of edges. Finally, all nodes in the graph are compressed and marked.

Employing the above operation, we start to build the coarser graph $\mathcal{G}^{(\ell+1)}$. Firstly, we define a matching matrix that stores the matching information from graph $\mathcal{G}^{(\ell)}$ to $\mathcal{G}^{(\ell+1)}$, which is a matrix $\mathbf{M}^{\ell,\ell+1} \in \mathbb{R}^{|\mathcal{V}_{n_\ell}| \times |\mathcal{V}_{n_{\ell+1}}|}$. At the ℓ layer, the value m_{rj} of the r-th row and j-th column in the $\mathbf{M}^{\ell,\ell+1}$ is set to 1 if node v_r in $\mathcal{G}^{(\ell)}$ is grouped to supernode v_j in $\mathcal{G}^{(\ell+1)}$. Following these formulations, the adjacency matrix $\mathbf{A}^{\ell+1}$ and graph representations $\mathbf{X}^{\ell+1}$ of the coarser graph $\mathcal{G}^{(\ell+1)}$ are obtained by using:

$$\mathbf{A}^{\ell+1} = (\mathbf{M}^{\ell,\ell+1})^{\mathrm{T}} \mathbf{A}^{\ell} \mathbf{M}^{\ell,\ell+1} \tag{7}$$

$$\mathbf{X}^{\ell+1} = (\mathbf{M}^{\ell,\ell+1})^{\mathrm{T}} \mathbf{H}^{\ell+1} \tag{8}$$

The built graph $\mathcal{G}^{(\ell+1)}$ and the graph representations matrix $\mathbf{X}^{\ell+1}$ are propagated into the next layer, then as the input of the next layer. We know that the hidden node embeddings $\mathbf{H}^{\ell+1}$ of the $\mathcal{G}^{(\ell)}$ generated by GWNN in the first work of each graph coarsening procedure, and $\mathbf{H}^{\ell+1}$ is stored and further applied to the graph refining procedure. Fig. 2 provides a toy example for illustrating the proposed Louvain-variant algorithm. Algorithm 1 also summarizes the whole graph coarsening procedure. For each iteration of coarsening, the Louvain-variant algorithm is generated in line 6–20, where Fig. 2 shows three phases illustrated in line 6–12, line 13–17, and line 18–20, respectively. A key part of NHEM is to visit the nodes following the ascending order of the number of neighbors (line 14). This is vital for ensuring that each node can be matched as well as the graph that can be coarsened significantly.

Algorithm 1. Graph coarsening procedure

Input: An input graph $\mathcal{G}^{(\ell)}$, graph representations matrix $\mathbf{X}^{\ell}$, and the levels for coarsening m.

Output: Coarse graph $\mathcal{G}^{(\ell+1)}$, GWNN hidden node embeddings $\mathbf{H}^{\ell+1}$, and graph representations $\mathbf{X}^{\ell+1}$.

Initialize $\forall \ell \in [0, m]$, threshold τ;
for *ℓ=0,1,...m* **do**
  /* *Calculate node embeddings of the input graph via GWNN* */ ;
  Compute and store the GWNN output $\mathbf{H}^{\ell+1}$ using Equation (2)(3);
  /* *Coarsen the input graph via Louvain-variant algorithm* */ ;
  Define every node as a community and all nodes among the graph are unmarked;
  Compute initialized modularity Q using Equation (4);
  repeat
    for *each unmarked node* v_i **do**
      if *the total number of nodes in a community* $< \tau$ *and the gain of modularity* $\Delta Q > 0$ *according to Equation (5)* **then**
        Mark v_i and group to a supernode by modularity optimization;
  until *Q does not improve*;
  repeat
    for *remaining unmarked node* $v_i^{'}$ *adjacent to the one-hop neighbor* v_t **do**
      Compute $r(v_i^{'}, v_t)$ according to Equation (6);
    Mark $v_i^{'}$ and group to supernode v_j having the largest $r(v_i^{'}, v_t)$;
  until *all nodes are marked in the graph*;
  Update the weights of nodes and edges in graph;
  Build matching matrix $\mathbf{M}^{\ell,\ell+1}$;
  Derive the adjacency matrix $\mathbf{A}^{\ell+1}$ for building the coarser graph $\mathcal{G}^{(\ell+1)}$ according to Equation (7);
  /* *Building the feature of the coarser graph via GWNN and matching matrix* */ ;
  Compute graph representations $\mathbf{X}^{\ell+1}$ according to Equation (8);
return $\mathbf{A}^{\ell+1}$, $\mathbf{H}^{\ell+1}$, $\mathbf{X}^{\ell+1}$.

4.3. Graph refining procedure

To restore from the coarsened graph to the original, further obtain the final graph representations, set the refining procedure similar to the coarsening procedure, which consists of two repeated operations, generate hidden node embedding vectors and refine the graph.

First, we use a GWNN to generate node embeddings on graph $\mathcal{G}^{(\zeta-\ell-1)}$. Then, during the graph coarsening procedure, we obtain the grouping relationship between nodes of $\mathcal{G}^{(\ell)}$ and their corresponding supernodes of $\mathcal{G}^{(\ell+1)}$ in a matching matrix $\mathbf{M}^{\ell,\ell+1}$ and store the GWNN output $\mathbf{H}^{\ell+1}$. Therefore, we apply the matching matrix $\mathbf{M}^{\ell,\ell+1}$ to restore the refined graph representations, in which the graph coarsening procedure and graph refining procedure are symmetrical, $\mathcal{G}^{(\ell)}$ and $\mathcal{G}^{(\zeta-\ell-1)}$ share the same graph topological structure, so $\mathbf{M}^{\zeta-\ell-1,\zeta-\ell} = \mathbf{M}^{\ell,\ell+1}$ (e.g., $\mathbf{M}^{1,2}$ of $\mathcal{G}^{(1)}$ is equal to $\mathbf{M}^{4,5}$ of $\mathcal{G}^{(5)}$ as Fig. 3). Moreover, we utilize jump connection to combine $\mathbf{H}^{\ell+1}$ of the ℓ layer during the coarsening procedure and the corresponding hidden node embeddings $\mathbf{H}^{\zeta-\ell-1}$ generated during the refining procedure. For example, in Fig. 1, combining $\mathbf{H}^{1}, \mathbf{H}^{4}$ and $\mathbf{M}^{3,4}$ obtains graph representations $\mathbf{X}^{4}$ of the refined graph $\mathcal{G}^{(4)}$. Specifically, graph representations are calculated in the graph refining procedure as follows:

$$\mathbf{X}^{\zeta-\ell} = \mathbf{M}^{\zeta-\ell-1,\zeta-\ell}\mathbf{H}^{\ell} + \mathbf{H}^{\zeta-\ell} \tag{9}$$

4.4. Output layer

Finally, in the ζ^{th} output layer of the M-GWNN network, we employ a GWNN with a softmax classifier on $\mathbf{X}^{\zeta-1}$ to output probabilities of unlabeled nodes:

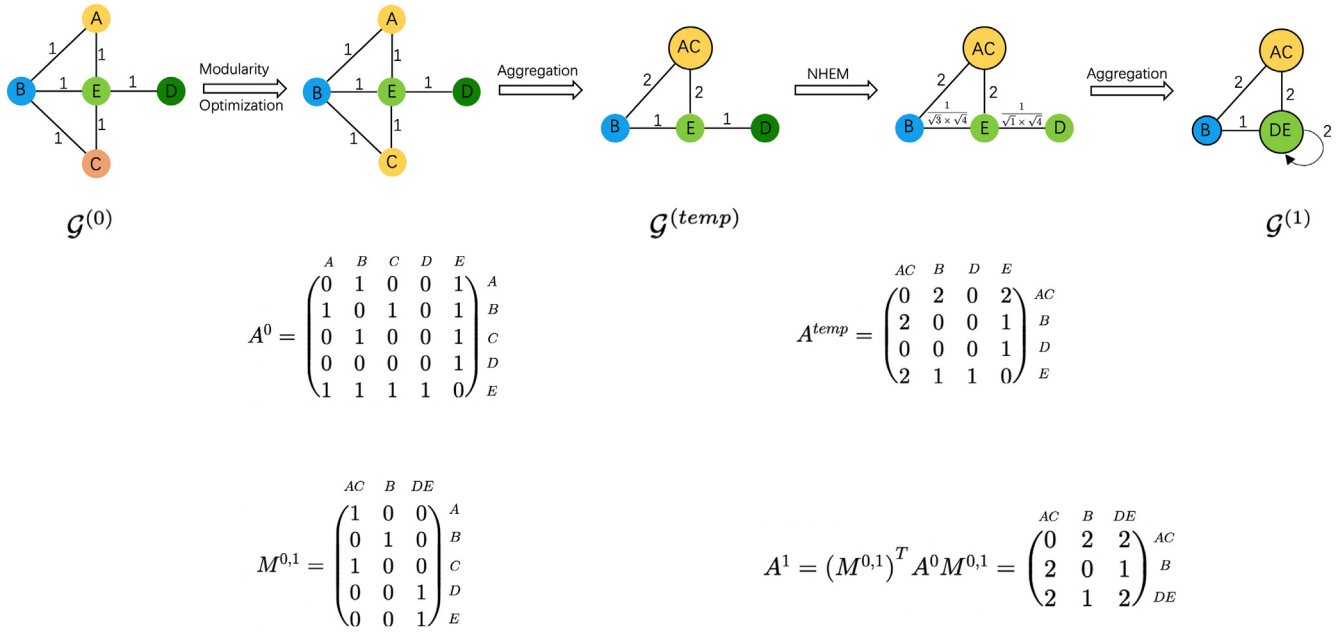

Fig. 2. Toy example for illustrating Louvain-variant algorithm. Firstly, the Louvain-variant algorithm is made of three phases: 1) most nodes in the original graph are marked and aggregated by applying local changes of communities to optimize modularity, the pass is repeated until no increase in modularity is possible; 2) remaining unmarked nodes are aggregated by using the NHEM algorithm in the graph; 3) constructing the coarser graph via the matching matrix. Moreover, in this illustration, the adjacency matrix $\mathbf{A}^0$ as the input, $\mathcal{G}^{(temp)}$ represents the coarser graph obtained through the modularity optimization. The matching matrix $\mathbf{M}^{0,1}$ corresponding to the modularity optimization and NHEM algorithm, and the derivation of the adjacency matrix $\mathbf{A}^1$ of the coarsened graph using Eq. (8).

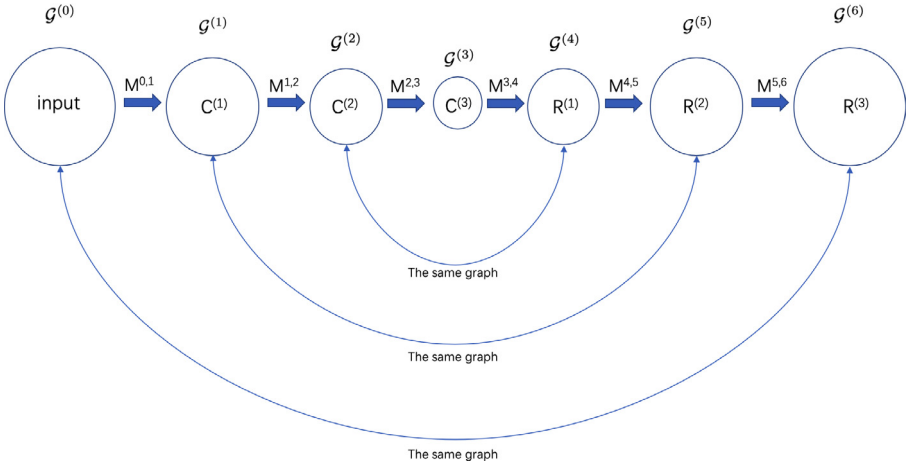

Fig. 3. The illustration of the proposed M-GWNN (graph coarsening and refinement except for GWNN module). In this illustration, there are seven differentiable graphs with the input as the original graph, three coarsened graphs, and three refined graphs. $C^{(i)}$ be denoted the i^{th} coarsened graph where $C^{(3)}$ is the coarsest graph. $R^{(i)}$ be denoted the i^{th} refined graph where $R^{(3)}$ is the refinest graph (the input). We can see that the coarsened graphs and the refined graphs are the absolutely same in the symmetrical position except for the coarsest graph, such as $C^{(1)}$ with $R^{(2)}$, $C^{(2)}$ with $R^{(1)}$. Moreover, matching matrix $\mathbf{M}^{i,i+1}$ are also the same in the symmetrical position, such as $\mathbf{M}^{1,2}$ is equal to $\mathbf{M}^{4,5}$, $\mathbf{M}^{2,3}$ is equal to $\mathbf{M}^{3,4}$. But they are used for coarsening and refinement, just like the two faces of a mirror. See Eqs. (7), (9) for more details about $\mathbf{M}$.

$$\mathbf{H}^{\zeta'} = \mathbf{X}^{\zeta-1}\mathbf{W}^{\zeta} \tag{10}$$

$$\mathbf{H}^{\zeta} = \mathrm{ReLU}(\phi_s \mathbf{F}^{\zeta} \phi_s^{-1} \mathbf{H}^{\zeta'}) \tag{11}$$

$$Z = \sigma(\mathbf{H}^{\zeta}) \tag{12}$$

where $\mathbf{W}^{\zeta} \in \mathbb{R}^{f \times |C|}$ is a trained weight matrix with C classes. $Z \in \mathbb{R}^{n_1 \times |C|}$ is the predicted likelihood on the $|C|$ dimensional probability simplex. The activation function σ is defined as $\mathrm{softmax}(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$ and applied row-wise. Then, we define the loss function $\mathcal{L}$ that minimizes the cross-entropy error on all labeled nodes $\mathcal{V}_L$ in the graph.

$$\mathcal{L} = -\sum_{i \in \mathcal{V}_L} \sum_{c=1}^{|C|} Y_{ic} \ln Z_{ic} \tag{13}$$

where $Y_{ic} = 1$ is the indicator function in which $Y_{ic} = 1$ if the label of node i is c. $\ln Z_{ic}$ represents the predicted probability that node i belongs to class c.

Fig. 4 shows t-SNE [48] visualizations of two-dimensional feature map representations by the pre-softmax activations after the last hidden convolutional layer of M-GWNN. Different colors mark different categories. Specifically, the data distribution of different

categories is obvious and clear, which proves the desired discriminative ability of M-GWNN in generating graph representations and challenging semi-supervised node classification task.

With these different components, describe the entire algorithm of our M-GWNN method in Algorithm 2.

Algorithm 2. M-GWNN for semi-supervised node classification

Input: The set of labeled nodes $\mathcal{V}_L = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$; the set of unlabeled nodes $\mathcal{V}_U = \{(x_{l+1}, y_{l+1}), (x_{l+2}, y_{l+2}), ..., (x_{l+u}, y_{l+u})\}$; the levels for coarsening m; the total layers ζ of the M-GWNN network.

Output: The predicted results of unlabeled nodes: $\hat{y} = (\hat{y}_{l+1}, \hat{y}_{l+2}, ..., \hat{y}_{l+u})$.

Initialize $\forall \ell \in [0, m]$, $\forall \iota \in [m, \zeta - 1]$;
repeat
  /* *Graph coarsening procedure* */ ;
  for ℓ = *0, 1,...* m **do**
    Use Algorithm 1 to obtain the GWNN output $\mathbf{H}^{\ell+1}$, the coarser graph $\mathcal{G}^{(\ell+1)}$ and graph representations $\mathbf{X}^{\ell+1}$;
  /* *Graph refining procedure* */ ;
  for ι = *m+1, m+2,...,* ζ*-1* **do**
    Compute and store the GWNN output $\mathbf{H}^{\iota+1}$ according to Equations (2)(3);
    Use the jump connection to obtain graph representations $\mathbf{X}^{\iota+1}$ according to Equation (9);
  /* *Output* */ ;
  Compute the predicted likelihoods on $\mathbf{H}^{\zeta}$ with a softmax classifier according to Equations (10)(11)(12);
until *the classification loss is minimized according to Equation (13)*;
return $\hat{y}$.

4.5. Time complexity analysis of the Louvain-variant algorithm

Overall, the time complexity of the proposed Louvain-variant algorithm approximates with the Louvain algorithm. Different from the Louvain algorithm, there are three step operations, including modularity optimization, the normalized heavy edge matching (NHEM), and aggregation. Firstly, we control the number of nodes that are coarsened of each community in each iterate, so the time complexity of each iteration of step one is $O(|\mathcal{E}|)$, and $|\mathcal{E}|$ is the number of edges of each input node. Then, the complexity of the NHEM is $O(|\mathcal{E}|'), |\mathcal{E}|'$ is the small number of edges of each unmarked node in each iteration. Also, we control the level m of graph coarsening, so the time complexity of aggregation is $O(|\mathcal{V}| + |\mathcal{E}|), |\mathcal{V}|$ is the number of nodes in each aggregation. According to the above analysis, we know that the proposed Louvain-variant algorithm has a near linear time complexity and thus can be efficiently applied to large-scale networks, such as the NELL dataset.

5. Experiments

5.1. Datasets

To demonstrate the benefit and effectiveness of the proposed M-GWNN on semi-supervised node classification task, we tested it on four standard benchmark datasets, including three Citation Network datasets: Cora, Citeseer, Pubmed (Sen et al. [49]) and Knowledge Graph dataset; the Never-Ending Language Learning

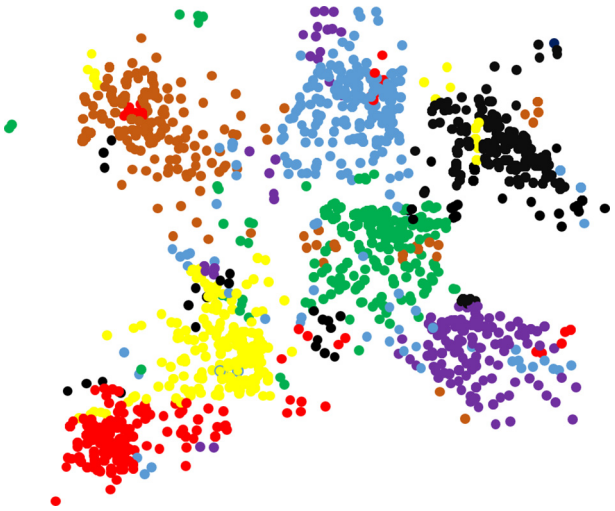

Fig. 4. 2D t-SNE visualization of the last hidden convolutional layer of M-GWNN when trained on Cora graph.

system (NELL) (Carlson et al. [50]; and Yang et al. [51]). Table 2 gives an overview of the four datasets, which are described in detail. Following the experimental setup in GCN [4], we fetched 20 labeled nodes per class in each dataset to train the model and all feature vectors.

5.2. Baselines

To evaluate the M-GWNN, we compared our approach to the following baselines:

- **DeepWalk** learns embedding by predicting the local neighborhood of the random walk sampling node on the graph [4].
- **Planetoid** adds a semi-supervised component using both transductive and inductive learning, which utilizes partially labeled data to collectively train the model to predict the class of the labeled nodes and the context of all nodes [51].
- **Spectral CNN** employs graph Fourier transform to implement convolution operator with non-Euclidean data first [1].
- **GCN** learns vector representations of nodes by learning first-order nearest neighbors in spectral-domain graph convolution [4].
- **GAT** obtains vector representations of nodes by introducing a multi-terminal attention mechanism into the one-hop neighborhoods of nodes [52]. The source code for GAT is publicly available.[2]
- **DGCN** takes a positive pointwise mutual information (PPMI) matrix as a supplement to the adjacency matrix to jointly encode both the local and global consistencies [21]. The source code for DGCN is publicly available.[3]
- **GWNN** utilizes graph wavelet as a set of bases taking the place of the eigenvectors of graph Laplacian [5]. The source code for the GWNN is publicly available.[4]
- **ConfGCN** acquires *anisotropic*[5]capabilities by utilizing label distribution and confidences to estimate the influence of one node on another [22]. The source code for ConfGCN is publicly available.[6]
- **N-GCN** learns multiple representations of the GCN via random work at neighborhoods of the different locality [23]. The source code for N-GCN is publicly available.[7]
- **graph U-Net** proposes a multi-layer graph convolutional method based on selecting some nodes to form a smaller graph according to their scalar projection values on a trainable projection vector. The source code for g-U-Nets is publicly available.[8]
- **H-GCN** introduces a hierarchical graph convolutional method based on graph Fourier convolution. The method proposed applying a structural similarity to aggregate nodes to supernodes [53]. The source code for H-GCN is publicly available.[9]

5.3. Experimental setting

We used TensorFlow [54] to implement our method and employed a GPU to speed up our experiment. Weights were initialized following Glorot & Bengio [55]. We trained M-GWNN for a maximum of 1,000 epochs (training iterations) with a *learning rate* of 0.01 using the Adam [56] optimizer and stopped training if the validation loss did not decrease for 100 consecutive epochs. The prediction accuracy was evaluated on a test set of 1,000 labeled samples. We save the model with best hyperparameters picked on the validation sets, and then use the optimal model to run multiple epochs on the testing sets to get the best result. The final results presented are the average of running the test sets with 10 times. Table 3 presents details of partial optimal hyperparameters setting about implementing the proposed method for different datasets, including (1) *number of hidden layer units*; (2) *dropout rate* (1-keep probability); and (3) *wavelet scale*.

5.4. Results and analysis

In this section, we attempt to answer the following questions:

- Q1: How does M-GWNN compare with existing representative methods for semi-supervised node classification task? (Section 5.4.1)
- Q2: How does increasing the number of layers affect performance for comparing M-GWNN with GWNN? (Section 5.4.2)
- Q3: How does the performance of M-GWNN vary with employing different coarsening methods? (Section 5.4.3)
- Q4: How do the hyperparameters influence M-GWNN? (Section 5.4.4)

5.4.1. Node classification

To show the effectiveness in the semi-supervised node classification task, we compared the proposed M-GWNN with other representative methods on four benchmark datasets. The results are summarized in Table 4, with the highest accuracy highlighted in bold for each column. Here, we report mean and standard error of prediction accuracy on the test set split in percent. We intentionally used the experimental results provided by other author's papers in the table for fairness, as opposed to running those experiments ourselves on untuned hyperparameters. For the NELL dataset, some baselines did not accomplish the corresponding experiments. Therefore, we restored their experiments on the NELL dataset according to the source code authors provided. The M-GWNN source code is available at https://github.com/wjzheng96/M-GWNN.

The experiments demonstrate that the performance of M-GWNN is better than state-of-the-art technologies on the Cora, Pubmed and NELL datasets, which demonstrates the effectiveness

[2] https://github.com/PetarV-/GAT.
[3] https://github.com/ZhuangCY/Coding-NN.
[4] https://github.com/benedekrozemberczki/GraphWaveletNeuralNetwork.
[5] anisotropic (adjective): varying in magnitude according to the direction of measurement (Oxford English Dictionary).
[6] http://github.com/malllabiisc/ConfGCN.
[7] https://github.com/samihaija/mixhop.
[8] https://github.com/HongyangGao/Graph-U-Nets.
[9] https://github.com/CRIPAC-DIG/H-GCN.

Table 2
Overview of the Datasets.

Dataset	Type	Nodes	Edges	Classes	Features	Training/Validation/Test
Cora	Citation network	2,708	5,429	7	1,433	140/500/1,000
Citeseer	Citation network	3,327	4,732	6	3,703	120/500/1,000
Pubmed	Citation network	19,717	44,338	3	500	60/500/1,000
NELL	Knowledge graph	65,755	266,144	210	5,414	4,200/500/1,000

Table 3
The setting of the hyperparameters in M-GWNN.

Dataset	#n_h	#dropout	#wavelet_s
Cora	32	0.8	0.6
Citeseer	32	0.5	0.1
Pubmed	32	0.4	0.4
NELL	64	0.2	0.8

of enlarging the receptive field of the spectral wavelet convolutional filters by combining the proposed Louvain-variant algorithm and the jump connection. Specifically, M-GWNN exceeded GWNN by a large margin among all four datasets. It can be noted that compared with citation networks, M-GWNN achieved an average error reduction in 5.7% compared to the state-of-the-art on the NELL dataset. It is noteworthy that for the ERR RED index in Table 4, we used state-of-the-art graph convolutional networks methods as references, which chose from GCN to GWNN.

Overall, we note the following:

- The random-work-based methods (i.e., DeepWalk, Planetoid) did not perform well and were unable to capture the attribute information. The planetoid collectively learns graph embeddings and predicts labels of unlabeled nodes by combining the graph topological structure and attribute information. However, due to taking the random sampling strategy, there is information loss when building the graph structure.
- To avoid the above problem, GCN proposed that aggregating information in the graph from one- or two–hop neighborhoods improves performance. Based on GCN, GAT employs a multi-terminal attention mechanism into one-hop, and ConfGCN uses label confidences to alleviate the effect of noisy nodes during neighborhood aggregation. However, these operations are not helpful in collecting adequate global information.
- DGCN further jointly considers both local and global consistency for each node in the graph; unfortunately, it captures global consistency through random walks. N-GCN learns multiple representations of GCN combining neighborhoods of different localities, and similar to DGCN, it takes random walks to obtain global information. Therefore, the information in the graph structure may also be lost in DGCN and N-GCN. In contrast, M-GWNN captures global information via convolution in varying granularity graphs and visibly outperforms all baselines in three out of four datasets. The experiments further indicate the desired advantage of graph construction and information propagation in M-GWNN to achieve graph representations and learning.
- Compared with graph U-Net, which selectively remove some nodes of graph to compress graph based on their scalar projection values on a trainable projection vector, our method proposed a Louvain-variant algorithm that retains all nodes of varying granularity graph to capture sufficient global structural information of the graph. Moreover, compared with the graph pooling method of the proposed g-U-Nets, the proposed Louvain-variant algorithm doesn't need to introduce any parameters that need to be learned.
- Compared with graph Fourier transform-based H-GCN, our method is based on graph wavelet transform, which can be captured through a fast algorithm without matrix eigenvalue decomposition, low computational cost. The graph wavelet is sparse and limited to the vertex domain, which provides more efficient and more suitable for sparse graphs than graph Fourier. Moreover, different from heuristically combining two closed nodes into a new one, we propose Louvain-variant algorithm, which characterizes the closeness of the discovered community based on modularity optimization, efficiently improving the quality of the graph coarsening.

From the comparison of experimental results, our algorithm performance can reach the average of state-of-the-art methods in the Citeseer dataset. But, compared with H-GCN and graph U-Net, M-GWNN did not achieve the best result. We know that the modularity Q can evaluate the quality of community detection in a network. In the experimental process of the proposed Louvain-variant algorithm, we find that the modularity Q in the Citeseer is lower than in the other three networks. Therefore, using the GWNN algorithm to implement such a network to do neighbor aggregation and then hierarchical propagation will definitely have

Table 4
Results of semi-supervised node classification (% accuracy).

Method	Cora	Citeseer	Pubmed	NELL
Deepwalk [33]	67.2%	43.2%	65.3%	58.1%
Planetoid [51]	75.7%	64.7%	77.2%	61.9%
Spectral CNN [1]	73.2%	58.9%	73.9%	–
GCN [4]	81.5%	70.3%	79.0%	73.0%
GAT [52]	83.0% ± 0.7%	72.3% ± 0.5%	79.0% ± 0.3%	74.1% ± 0.4%
DGCN [21]	83.5%	72.4%	79.3%	74.2%
ConfGCN [22]	82.1%	72.5%	79.5%	74.6% ± 0.5%
N-GCN [23]	83.0%	72.2%	79.5%	74.8% ± 0.2%
g-U-Nets [10]	84.4% ± 0.6%	**73.2% ± 0.5%**	79.6% ± 0.2%	79.4% ± 0.3%
H-GCN [53]	84.1% ± 0.3%	72.9% ± 0.2%	79.8% ± 0.4%	79.9% ± 0.3%
GWNN [5]	82.8%	71.4% ± 0.2%	79.1%	73.8% ± 0.4%
M-GWNN	**84.6% ± 0.5%**	72.6% ± 0.1%	**80.4% ± 0.2%**	**81.2% ± 0.2%**
(%) ERR RED	-1.5%	−0.5%	−1.1%	−5.7%

a certain impact. However, compared with GWNN, our approach with deeper layers considerably improved performance in the Citeseer dataset, which efficiently increased the receptive field but did not accelerate the speed of the model over-smoothing.

5.4.2. Effect of increasing convolutional layers

Recently, Xu and Shen [5] made the GWNN source code available, and we conducted the experiment by adding convolutional layers. For comparison, we evaluated the performance of M-GWNN and GCN and GWNN on the Cora dataset with an increasing number of convolutional layers. The results are summarized in Fig. 5. Similar to GCN but expectedly, we found that the performance of GWNN decreased significantly with an increasing number of layers. This is consistent with our view that GCNs models produce over-smoothing via merely stacking many convolutional layers. However, compared with GCN, we observed that GWNN performance decreased more gradually with increasing the number of layers rather than degrading drastically as GCN, which shows the importance of GWNN detaching the feature transformation from convolution operation, further facilitating our model to make it deeper. Compared with them, M-GWNN achieved better performance with different numbers of convolutional layers, which shows the insensitivity of the M-GWNN model with the deep layers and further indicated the advantage and better performance of M-GWNN in challenging the problem of GCNs models over-smoothing. Notably, the M-GWNN obtained the best performance with five convolutional layers. In other words, we can conclude that the performance of M-GWNN achieved the best performance when there were two graph coarsening operations and two symmetric refining operations on the Cora dataset.

5.4.3. Effect of employing different coarsening methods

In this section, we designed several alternative variants in terms of graph coarsening methods and comprehensively analyzed their performance for semi-supervised node classification among all four datasets. Thus, the alternatives we considered were as follows:

- **Without coarsening and refining operations (M-GWNN_WCR)**. We removed both coarsening and refining operations of the M-GWNN. Different from simply stacking too many GWNN layers, we employed the residual connection to combine every GWNN layer and obtain the final graph representations.
- **Structural equivalence matching (M-GWNN_SEM)**. We replace the Louvain-variant algorithm with a structural equivalence matching approach for graph coarsening. For each iteration of the coarsening operation, we first match a pair of structurally equivalent nodes, which were considered to be structurally equivalent if two nodes have the same neighbors, after aggregating them into a supernode. The rest of the algorithm is the same as our M-GWNN.
- **Louvain matching (M-GWNN_LM)**. For this baseline method, we analyzed its disadvantages in detail theoretically in Section 4.2. Therefore, in this section, we demonstrate that our graph coarsening method is better than LM from an experimental point of view.

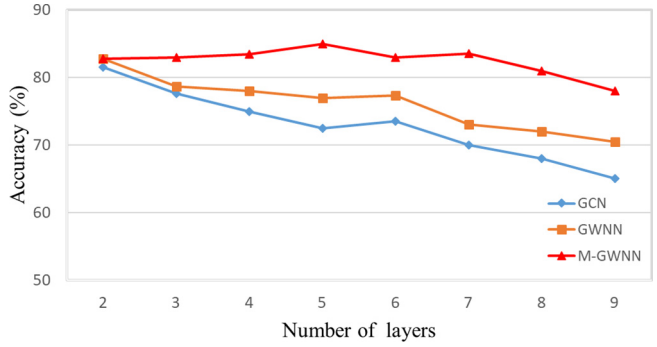

Fig. 5. Evaluation of M-GWNN with different convolutional layers on the Cora dataset. Overall, M-GWNN outperformed baselines, and while the performance of all methods decreased with increasing layers, M-GWNN still maintained a reasonable result.

Fig. 6 shows the comparison of performance on these methods across the four datasets. We execute five convolutional layers for the M-GWNN and its three variants on the Cora and Citeseer dataset. Moreover, we execute seven convolutional layers and nine convolutional layers for the M-GWNN and other variants on the Pubmed and NELL datasets, respectively. First, we obviously note that the M-GWNN consistently performs better than the alternatives. Then, we summarize the key information derived from Fig. 6 as follows:

- The coarsening and refining operations used within the M-GWNN guarantee that the model can obtain node features and graph information of varying granularity, further increasing the receptive field and enabling adequate information propagation. In addition, during combining coarsening with refining operations, M-GWNN applies the jump connection, which offers a reliable solution for the problem of over-smoothing in GCNs models. This further demonstrates the advantages of M-GWNN by combining the proposed Louvain-variant algorithm and the jump connection cooperatively and thus can boost the performance of M-GWNN.
- The matching methods used within M-GWNN offer a better benefit in terms of accuracy for the node classification task than M-GWNN_SEM. We found that although M-GWNN_SEM offered a simpler coarsening method than our approach, the proposed M-GWNN provides better interpretability by utilizing the modularity index [20], which is a self-consistent evaluation system able to point out and define what is a good community division, further improving the quality of graph coarsening.
- The proposed Louvain-variant algorithm as our graph coarsening method qualitatively improves the deficiency of the Louvain algorithm usually used in unsupervised scenarios, which successfully challenges the problem of semi-supervision and acquires an optimistic promotion for among four datasets. One can note from Fig. 6(c) and (d) that M-GWNN effectiveness outperforms M-GWNN_LM by larger margins on the Pubmed and NELL datasets because the M-GWNN_LM aggregated speed of nodes coarsening to supernodes is too fast, so it fails to obtain much more graph topological information of varying granularity.

5.4.4. Effect of hyperparameter

In this section, we focus on discussing the influence of wavelet scale s on the model M-GWNN. In our framework design, M-GWNN applies graph wavelets to implement graph convolutional operation, in which s is the range of neighborhoods. Fig. 7 shows the performances of the model with different s values used for semi-supervised node classification on the Cora dataset. In Fig. 6, with the increase in s, the number of neighborhoods of nodes increased, and the result of the model improved. Nevertheless, we found that when s was too large, some irrelevant nodes were included, resulting in a decrease in the accuracy of the model. One can note that when s is in the range of [0.4,0.7], M-GWNN achieved comparatively optimistic performance, which not only

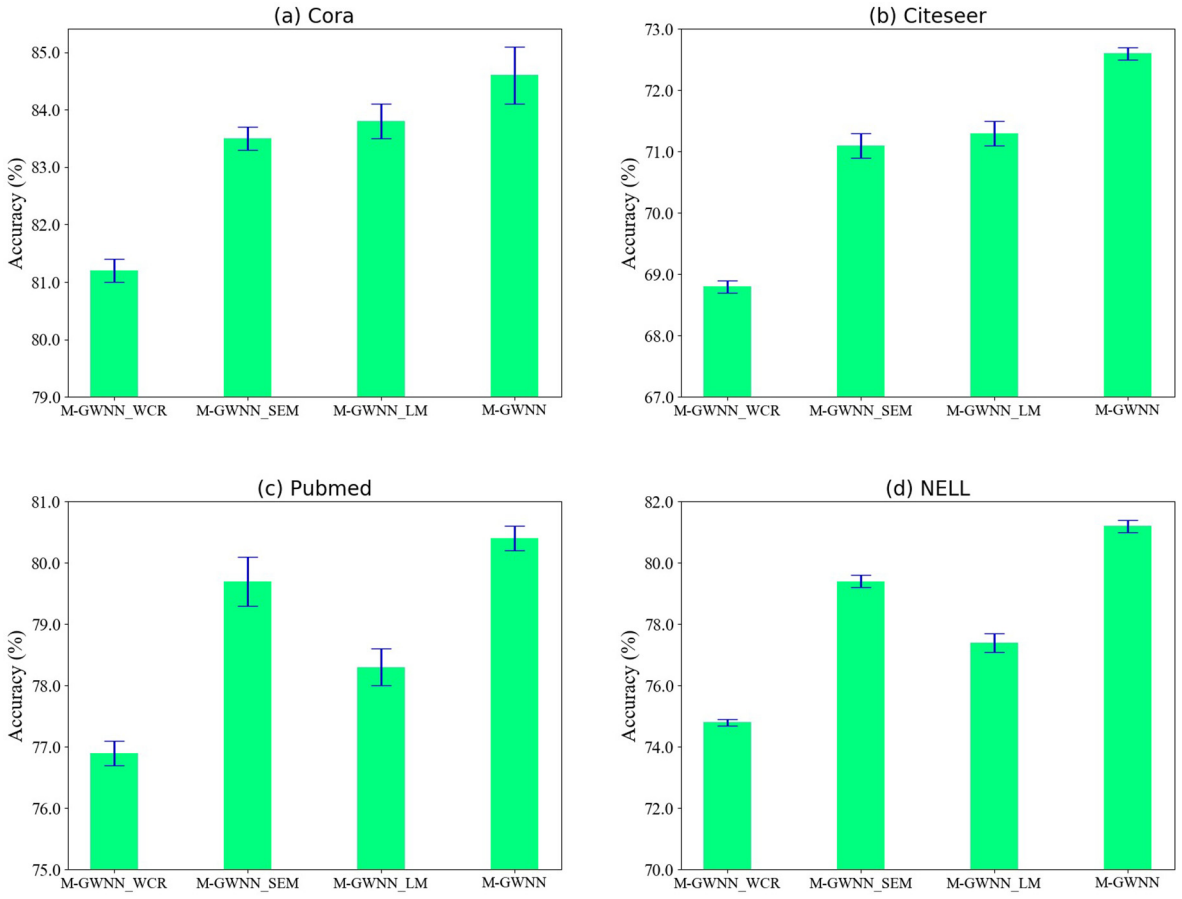

Fig. 6. Performance comparison of M-GWNN and its three variants among all four datasets. The symbol **"I"** represents the error bar, as a standard error of mean accuracy of the model, whose length is the size of the error.

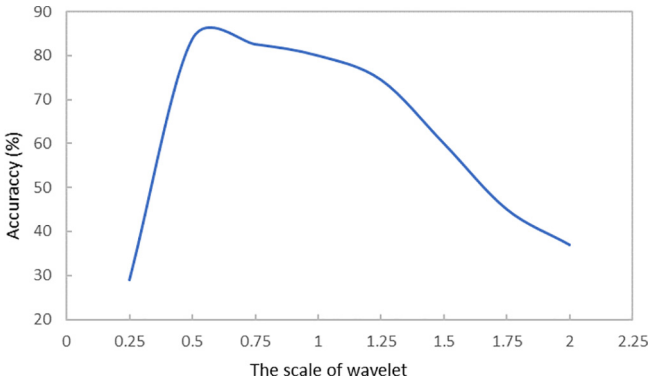

Fig. 7. Influence of wavelet scale s over the Cora graph.

guarantees the locality of graph convolution but also captures the graph topological structure.

6. Conclusion

Throughout this paper, to widen the receptive field and alleviate the speed of the wavelet convolutional filters over-smoothing, we propose multi-granularity graph wavelet neural networks (M-GWNN) for semi-supervised node classification. We first aggregate nodes into supernodes utilizing the proposed Louvain-variant algorithm and then symmetrically refine the coarsened graph to the original by the jump connection. M-GWNN obtains node features and graph topological information of varying granularity, effectively captures a larger receptive field, and enables adequate information propagation. Furthermore, M-GWNN leverages the jump connection to connect receptive fields of varying granularity to alleviate the speed of over-smoothing. Several experiments comprehensively demonstrate that the proposed method outperforms state-of-the-art methods for semi-supervised node classification task. In the future, we will generalize our method for information-rich graphs to challenge the problem of semi-supervised text classification or heterogeneous information network node classification.

CRediT authorship contribution statement

Wenjie Zheng: Conceptualization, Methodology, Software, Data curation, Formal analysis, Writing - original draft, Writing - review & editing. **Fulan Qian:** Investigation, Writing - review & editing, Formal analysis, Data curation, Visualization, Funding acquisition. **Shu Zhao:** Validation, Writing - review & editing, Supervision, Funding acquisition. **Yanping Zhang:** Validation, Writing - review & editing, Supervision, Funding acquisition.

Declaration of Competing Interest

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

Acknowledgements

This work is supported and funded by the National Natural Science Foundation of China (Grant Nos. 61702003, 61673020, 61876001) and Natural Science Foundation of Anhui Province (Grant No. 1808085MF175). Moreover, we would like to thank the reviewers for their constructive comments and suggestions.

References

[1] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representations (ICLR), 2014.
[2] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems (NeurIPS), 2012, pp. 1097–1105.
[3] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems (NeurIPS), 2016, pp. 3844–3852.
[4] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), 2017.
[5] B. Xu, H. Shen, Q. Cao, Y. Qiu, X. Cheng, Graph wavelet neural network, in: International Conference on Learning Representations (ICLR), 2019.
[6] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[7] G. Li, M. Müller, A. Thabet, B. Ghanem, Deepgcns: Can gcns go as deep as cnns?, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019.
[8] H. Chen, B. Perozzi, Y. Hu, S. Skiena, Harp: Hierarchical representation learning for networks, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[9] J. Liang, S. Gurukar, S. Parthasarathy, Mile: A multi-level framework for scalable graph embedding, arXiv preprint arXiv:1802.09612.
[10] H. Gao, S. Ji, Graph u-nets, in: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019.
[11] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: Proceedings of the 36th International Conference on Machine Learning (ICML), 2019.
[12] D. Meunier, R. Lambiotte, A. Fornito, K. Ersche, E.T. Bullmore, Hierarchical modularity in human brain functional networks, Frontiers in Neuroinformatics 3 (2009) 37.
[13] S. Fortunato, Community detection in graphs, Physics Reports 486 (3–5) (2010) 75–174.
[14] R. Mall, R. Langone, J.A. Suykens, Multilevel hierarchical kernel spectral clustering for real-life large scale complex networks, PloS One 9 (6) (2014), e99966.
[15] G. Fu, C. Hou, X. Yao, Learning topological representation for networks via hierarchical sampling, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.
[16] A.K. Bhowmick, K. Meneni, M. Danisch, J.-L. Guillaume, B. Mitra, Louvainne: Hierarchical louvain method for high quality and scalable network embedding, in: Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM), 2020, pp. 43–51.
[17] M.E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2) (2004), 026113.
[18] V.D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment 2008 (10) (2008) P10008.
[19] A. Lancichinetti, S. Fortunato, Community detection algorithms: a comparative analysis, Physical Review E 80 (5) (2009), 056117.
[20] A. Clauset, M.E. Newman, C. Moore, Finding community structure in very large networks, Physical Review E 70 (6) (2004), 066111.
[21] C. Zhuang, Q. Ma, Dual graph convolutional networks for graph-based semi-supervised classification, in: Proceedings of the 2018 World Wide Web Conference (WWW), 2018, pp. 499–508.
[22] S. Vashishth, P. Yadav, M. Bhandari, P. Talukdar, Confidence-based graph convolutional networks for semi-supervised learning, in: Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS), 2019.
[23] S. Abu-El-Haija, A. Kapoor, B. Perozzi, J. Lee, N-gcn: Multi-scale graph convolution for semi-supervised node classification, in: Proceedings of the 35th Conference on Uncertainty in Artificial Intelligence (UAI), 2018.
[24] S. Qi, W. Wang, B. Jia, J. Shen, S.-C. Zhu, Learning human-object interactions by graph parsing neural networks, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 401–417.
[25] P. Ghosh, Y. Yao, L. Davis, A. Divakaran, Stacked spatio-temporal graph convolutional networks for action segmentation, in: The IEEE Winter Conference on Applications of Computer Vision (WACV), 2020, pp. 576–585.
[26] Y.-A. Huang, P. Hu, K.C. Chan, Z.-H. You, Graph convolution for predicting associations between mirna and drug resistance, Bioinformatics 36 (3) (2020) 851–858.
[27] J. Li, S. Zhang, T. Liu, C. Ning, Z. Zhang, W. Zhou, Neural inductive matrix completion with graph convolutional networks for mirna-disease association prediction, Bioinformatics.
[28] D.K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, Applied and Computational Harmonic Analysis 30 (2) (2011) 129–150.
[29] D.I. Shuman, C. Wiesmeyr, N. Holighaus, P. Vandergheynst, Spectrum-adapted tight graph wavelet and vertex-frequency frames, IEEE Transactions on Signal Processing 63 (16) (2015) 4223–4235.
[30] C. Donnat, M. Zitnik, D. Hallac, J. Leskovec, Learning structural node embeddings via diffusion wavelets, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1320–1329.
[31] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, J. Leskovec, Hierarchical graph representation learning with differentiable pooling, in: Advances in Neural Information Processing Systems (NeurIPS), 2018, pp. 4800–4810.
[32] Y. Ma, S. Wang, C.C. Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 723–731.
[33] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2014, pp. 701–710.
[34] A. Grover, J. Leskovec, node2vec: alable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2016, pp. 855–864.
[35] R. Cilibrasi, P.M. Vitányi, Clustering by compression, IEEE Transactions on Information Theory 51 (4) (2005) 1523–1545.
[36] D. Chakrabarti, S. Papadimitriou, D.S. Modha, C. Faloutsos, Fully automatic cross-associations, in: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2004, pp. 79–88.
[37] W. Sadiq, M.E. Orlowska, Analyzing process models using graph reduction techniques, Information Systems 25 (2) (2000) 117–134.
[38] Y. Tian, R.A. Hankins, J.M. Patel, Efficient aggregation for graph summarization, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 567–580.
[39] Y. Liu, T. Safavi, A. Dighe, D. Koutra, Graph summarization methods and applications: A survey, ACM Computing Surveys (CSUR) 51 (3) (2018) 62.
[40] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems (NeurIPS), 2017, pp. 1024–1034.
[41] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, in: International Conference on Learning Representations (ICLR), 2018.
[42] A. Maccioni, D.J. Abadi, Scalable pattern matching over compressed graphs via dedensification, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2016, pp. 1755–1764.
[43] M. Al Hasan, N. K. Ahmed, J. Neville, 1 network sampling: Methods and applications, Citeseer.
[44] P. De Meo, E. Ferrara, G. Fiumara, A. Provetti, Generalized louvain method for community detection in large networks, in: 2011 11th International Conference on Intelligent Systems Design and Applications, IEEE, 2011, pp. 88–93.
[45] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European Conference on Computer Vision (ECCV), Springer, 2016, pp. 630–645.
[46] V.A. Traag, L. Waltman, N.J. van Eck, From louvain to leiden: guaranteeing well-connected communities, Scientific Reports 9 (1) (2019) 1–12.
[47] G. Karypis, V. Kumar, Multilevelk-way partitioning scheme for irregular graphs, Journal of Parallel and Distributed computing 48 (1) (1998) 96–129.
[48] L.v.d. Maaten, G. Hinton, Visualizing data using t-sne, Journal of machine learning research 9 (Nov) (2008) 2579–2605.
[49] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (3) (2008), 93–93.
[50] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, T.M. Mitchell, Toward an architecture for never-ending language learning, in: Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.
[51] Z. Yang, W.W. Cohen, R. Salakhutdinov, Revisiting semi-supervised learning with graph embeddings, International Conference on Machine Learning (ICML).
[52] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations (ICLR), 2017.
[53] F. Hu, Y. Zhu, S. Wu, L. Wang, T. Tan, Hierarchical graph convolutional networks for semi-supervised node classification, in: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI), 2019, pp. 10–16.
[54] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.

[55] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics (AISTATS), 2010, pp. 249–256.
[56] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2014.

Wenjie Zheng is currently pursuing the master degree with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include data mining, graph neural networks and knowledge graphs.

Fulan Qian received the PhD degree in computer science from Anhui University in 2016. She is now a associate professor in the School of Computer Science and Technology, Anhui University. Her current research interests include granular computing, social network and recommendation system.

Shu Zhao received the PhD degree in computer science from Anhui University in 2007. She is now a professor in the School of Computer Science and Technology, Anhui University. Her current research interests include quotient space theory, granular computing, social network and machine learning.

Yanping Zhang is currently a professor in Anhui University. She received the PhD degree from Anhui University in 2005. Her main research interests include computational intelligence, quotient space theory, artificial neural networks and intelligent information processing, machine learning, and so on. She is the Principal Investigator of some 973 projects and the leader of National Natural Science Foundations of China and Anhui Province.