Pattern Recognition 151 (2024) 110364

Contents lists available at ScienceDirect

Pattern Recognition

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

Graph Convolutional Network with elastic topology

Zhihao Wu, Zhaoliang Chen, Shide Du, Sujia Huang, Shiping Wang *

College of Computer and Data Science, Fuzhou University, Fuzhou, 350108, China
Key Laboratory of Intelligent Metro, Fuzhou University, Fuzhou, 350108, China

ARTICLE INFO

Keywords:
Graph convolutional networks
Semi-supervised classification
Learnable topology
Orthogonal constraint

ABSTRACT

Graph Convolutional Network (GCN) has drawn widespread attention in data mining on graphs due to its outstanding performance and rigor theoretical guarantee. However, some recent studies have revealed that GCN-based methods may mine latent information insufficiently owing to the underutilization of the feature space. Besides, the unlearnable topology also significantly imperils the performance of GCN-based methods. In this paper, we conduct experiments to investigate these issues, finding that GCN does not fully consider the potential structure in the feature space, and a fixed topology deteriorates the robustness of GCN. Thus, it is desired to distill node features and establish a learnable graph. Motivated by this goal, we propose a framework dubbed **G**raph **C**onvolutional **N**etwork with **e**lastic **t**opology (GCNet[1]). With the analysis of the optimization for the proposed flexible Laplacian embedding, GCNet is naturally constructed by alternative graph convolutional layers and adaptive topology learning layers. GCNet aims to deeply explore the feature space and employ the mined information to construct a learnable topology, which leads to a more robust graph representation. In addition, a set-level orthogonal loss is utilized to meet the orthogonal constraint required by the flexible Laplacian embedding and promote better class separability. Moreover, comprehensive experiments indicate that GCNet achieves remarkable performance and generalization on several real-world datasets.

1. Introduction

Graph Convolutional Network (GCN) is an effective model designed for graph representation learning. GCN [1] explores previous works and extends convolutional operations to non-Euclidean space based on the spectral theory. Utilizing graph convolutional operations, GCN naturally aggregates information from neighbor nodes via the topological structure. Due to its outstanding performance, GCN has been applied to many domains, such as data mining [2–4], recommendation systems [5–7] and multimedia tasks [8–10], and its various variants have been proposed since the emergence of GCN.

Considerable studies [11–13] pointed out that obtaining a large number of labeled samples is labor-intensive. Consequently, how to extract potential information from unlabeled data is a critical problem. Semi-supervised learning, which only utilizes a small amount of labeled data with substantial unlabeled data, is a training paradigm widely explored nowadays. GCN has shown impressive performance in semi-supervised classification for graph-structural data. To be specific, cross entropy loss is utilized to aggregate label information across nodes via the graph structure, achieving promising performance when applied to GCN. However, recent research [14] has found that it does not lead the model to optimally separate different classes of samples. Moreover, this drawback is exacerbated when dealing with semi-supervised classification tasks due to the scarcity of labeled samples.

Beyond that, although the effectiveness of GCN has been validated, some works have revealed its several shortcomings. For instance, Li et al. [15] indicated that GCN was limited to a shallow model, restricting its exploration of feature information, which is called over-smoothing issue. Wang et al. [16] argued that the fusion ability of GCN was imperfect and proposed a multi-channel model to enhance its capacity for feature aggregation. Sun et al. [17] presented a multi-stage self-training framework for better propagation of label signals. These methods improved the performance of GCN, but there still exist some problems to be investigated, which will be illustrated in Section 2: (1) Although GCN equips with powerful graph convolution, it does not well consider potential structure in the feature space; (2) Predefined and unlearnable topology may impair GCN's performance. These two problems are not independent, and how to establish a unified and effective framework addressing them simultaneously is under-explored.

In order to tackle the aforementioned problems, we propose a framework dubbed **G**raph **C**onvolutional **N**etwork with **e**lastic **t**opology

* Corresponding author at: College of Computer and Data Science, Fuzhou University, Fuzhou, 350108, China.
E-mail addresses: zhihaowu1999@gmail.com (Z. Wu), chenzl23@outlook.com (Z. Chen), dushidems@gmail.com (S. Du), hsujia2021@163.com (S. Huang), shipingwangphd@163.com (S. Wang).

[1] The source code is available at https://github.com/ZhihaoWu99/GCNet.

https://doi.org/10.1016/j.patcog.2024.110364
Received 17 February 2023; Received in revised form 19 November 2023; Accepted 21 February 2024
Available online 28 February 2024

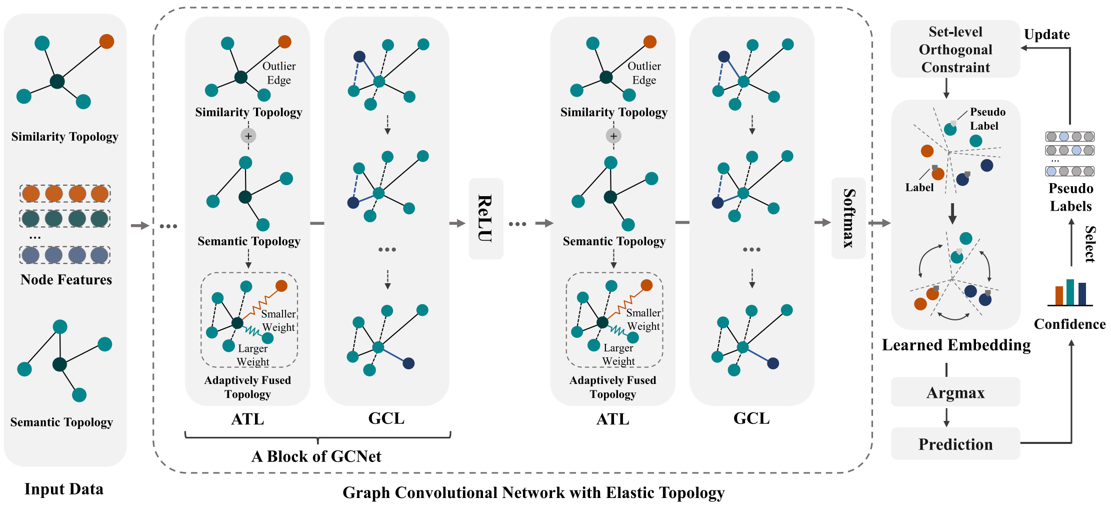

Fig. 1. The overview of the proposed framework GCNet, which consists of several blocks combining Adaptive Topology Learning Layer (ATL) and Graph Convolutional Layer (GCL). ATL dynamically adjusts the weights of edges, providing a more robust topology to the propagation procedure of GCL. Set-level orthogonal constraint with self-supervised mechanism compensates for the lack of cross-entropy and provides theoretical rigor.

(GCNet), which conducts deeper investigation on mining node features and dynamically learns a graph during model training. First, we review the Laplacian embedding and put forward a flexible objective function to learn a more robust graph representation. The flexible Laplacian embedding captures the potential manifold structure in the feature space and fuses it into the original semantic graph, and this exhibits stronger robustness by utilizing $\ell_{2,p}$-norm. Subsequently, we adopt an optimization strategy with alternating updates and naturally establish a connection between the GCN and it. In light of the analysis, we propose a block-wise GCN, where each block consists of a graph convolution layer and an adaptive topology learning layer. In addition, we come up with an orthogonal constraint that takes into account the inter-class separation and the intra-class clustering, which theoretically meets the orthogonal requirement of the original objective function. Fig. 1 briefly illustrates the proposed GCNet. We summarize the major contributions of this paper as follows,

(1). Present a more flexible Laplacian embedding, which considers suppressing the effect of outliers and learning a more robust representation.
(2). Propose a block-based framework with learnable topology inspired by the flexible Laplacian embedding.
(3). A set-level orthogonal loss is put forward, which improves the cross-entropy loss.
(4). The proposed method is applied to semi-supervised node classification tasks, and the experimental results indicate its superiority.

2. Preliminary experiment

To demonstrate the issues of GCN in detail, we design a preliminary experiment. Specifically, we construct a binary adjacency matrix using k-Nearest Neighbor (KNN) algorithm, which is a classical approach to mine structural information from node features, and k is set to 10 in this experiment. KNN-GCN denotes GCN with generated KNN graph. In other words, only node features are fed to the KNN-GCN, and the graph structure is constructed based on the node features with KNN algorithm. Thus, the main difference is that GCN uses more information containing original semantic graph structure and node features while KNN-GCN only gets the latter. Fig. 2 records the performance comparison between GCN and KNN-GCN, The figure indicates that GCN achieves undesired performance, while KNN-GCN performs comparably or even much better than GCN on some datasets. The results are unexpected because GCN can leverage more information than KNN-GCN, so this phenomenon essentially reflects that:

(1). GCN emphasizes the utilization of topological structure, while the potential structure of samples in the feature space is not considered well.
(2). GCN may learn misleading information from some topologies, thus adopting predefined and unlearnable topology impairs the robustness of GCN.

These issues have been partially pointed out by some researchers, and efforts [3,16,18,19] have been made to solve them. Nevertheless, studies on the construction of learnable topology are still limited and lack interpretability. Actually, the observation also suggests that the latent structure extracted from the feature space can improve the topology network fed into GCN, since the only difference between KNN-GCN and GCN is the utilized graph structures. Therefore, we aim to investigate how to solve these two problems through a unified framework.

3. Related work

In this section, we review the well-known GCN and graph Laplacian embedding in two subsections, which are closely related to our work.

3.1. Graph convolutional network

On the basis of the spectral graph convolutions and Chebyshev network, GCN further simplifies the form of graph filter to process network data more efficiently. It is built by multi-layer graph convolution operations, where the layer-wise forward propagation can be formulated as

$$\mathbf{H}^{(l+1)} = \sigma\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{1}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ represents the adjacency matrix with additional self loops, and $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the degree matrix calculated on $\tilde{\mathbf{A}}$. $\mathbf{H}^{(l+1)}$ is the output embedding in the lth layer, equipped with an activation function σ. $\mathbf{W}^{(l)}$ is the learnable weight. GCN was first proposed for the semi-supervised node classification tasks and achieved outstanding performance. Nowadays, numerous studies have explored the mechanism of GCN, exploiting and improving the capacity of aggregating node features to promote the performance of various machine learning tasks. Li et al. [15] indicated that the convolution operation of GCN was indeed a special form of Laplacian smoothing and thus suffered from the over-smoothing problem. Sun et al. [20] employed AdaBoost to design an RNN-like GCN for the purposes of solving the over-smoothing issue and aggregating knowledge from different orders of neighbors.

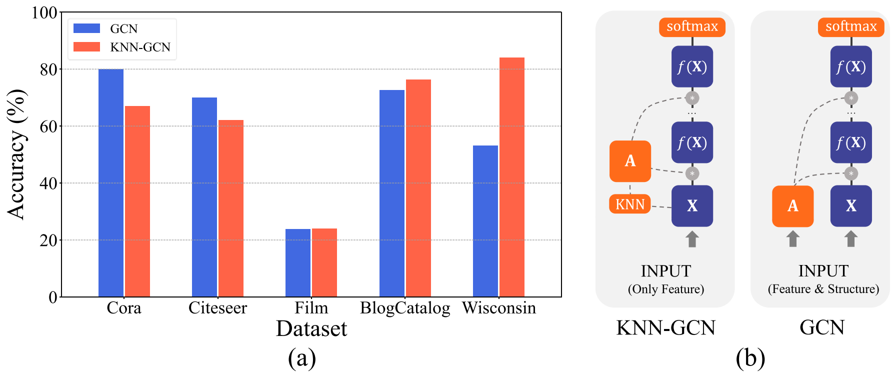

Fig. 2. (a) Classification accuracy of GCN and KNN-GCN on various datasets. (b) The illustrations of two models, where graph structure and node features are fed to GCN while KNN-GCN can only access features.

Zhang et al. [21] presented a hyperbolic GCN, which reconstructed the graph operations from a geometric perspective. Wu et al. [22] hypothesized that the nonlinear activation functions between GCN layers were not critical, and simplified GCN as a linear model. Wang et al. [23] analyzed the vanilla GCN and its variants which simplify GCN by removing the non-linear transformations, and further established decoupled graph convolution for better utilization of the feature propagation. Li et al. [24] incorporated a metric learning method to build adaptive graph structures for different data. These works revealed the deficiencies of GCN and significantly promoted the performance of GCN on different tasks with improved models.

3.2. Graph Laplacian embedding

For given data $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times m}$, graph Laplacian embedding assumes the existence of its potential manifold structure in the feature space, and it aims to map $\mathbf{X}$ into a low-dimensional embedding capable of preserving the local geometry. Graph Laplacian embedding naturally approximates the manifold by constructing a similarity graph. To be specific, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is defined as a symmetric adjacency matrix, where KNN is one of the most common approaches for the construction of $\mathbf{A}$. Considering two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ $(i \neq j)$, which are close in the original feature space, it is natural to maintain such two samples close to each other in the low-dimensional space. Consequently, the objective function on the basis of the manifold assumption can be defined as

$$\min_{\mathbf{Z}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|_2^2 \mathbf{A}_{ij} \quad \text{s.t. } \mathbf{Z}^\top\mathbf{Z} = \mathbf{I}, \tag{2}$$

where $\mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_n]^\top \in \mathbb{R}^{n\times c}$ is the obtained low-dimensional embedding of $\mathbf{X}$, and c is the number of classes. This idea has been widely adopted in many fields, including clustering [25–27], semi-supervised learning [28–30] and multi-view learning [31–33]. By simple algebra, Eq. (2) is equivalent to

$$\min_{\mathbf{Z}} Tr\left(\mathbf{Z}^\top\mathbf{L}\mathbf{Z}\right) \quad \text{s.t. } \mathbf{Z}^\top\mathbf{Z} = \mathbf{I}. \tag{3}$$

Here, $\mathbf{L}$ denotes the Laplacian matrix, which is typically calculated as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ with $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$, and the orthogonal constraint imposed on $\mathbf{Z}$ is to avoid trivial solutions. Some recent studies also employ the symmetric normalized one instead. Graph Laplacian embedding has been widely used, and it is also relevant to GCN. Some studies [34] have revealed the connection between graph Laplacian embedding and GCN. This paper further improves the graph Laplacian embedding and clarifies its relationship with GCN.

4. The proposed method

For better understanding, we begin with introducing some notations used in this paper. A graph, denoted as $\mathcal{G} = (\mathbf{X}, \mathbf{A})$, contains a set of nodes and a set of edges. Namely, $\mathbf{X} \in \mathbb{R}^{n\times m}$ is the feature matrix containing the node information, where n is the number of nodes and m is the feature dimension, and $\mathbf{A} \in \mathbb{R}^{n\times n}$ is the adjacency matrix representing the set of edges with $\mathbf{A}_{ij} = 1$ if node i and node j are connected and $\mathbf{A}_{ij} = 0$ otherwise. The main notations used in this paper and their description are summarized in Table 1 for convenience.

Table 1
Description of the main notations used in this paper.

Notations	Description
$\mathbf{X}$	Node feature matrix.
$\mathbf{Y}$	Node Label matrix.
$\mathbf{A}_k$, $\mathbf{A}_s$	Adjacency matrices constructed by similarities and semantics.
$\mathbf{A}_f$, $\mathbf{L}_f$	Fused adjacency matrix and corresponding Laplacian matrix.
$\mathbf{Z}^{(l)}$	Node embedding in the lth block.
$\mathbf{W}^{(l)}$	Learnable weight matrix in the lth block.
$\sigma(\cdot)$	Activation function.
$\mathbf{I}$	Identity matrix.
$\mathbf{M}_s$	Mask matrix for samples from same classes.
$\mathbf{M}_d$	Mask matrix for samples from different classes.
$\mathcal{L}_{SO}$	Set-level orthogonal loss.
$\mathcal{L}_{CE}$	Cross entropy loss.

4.1. Adaptive Laplacian embedding induced GCN

In light of the widely used manifold assumption, neighborhood samples are usually considered to be closer together in the hidden space, which suggests the following objective function:

$$\min_{\mathbf{Z}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|_2^2 \mathbf{A}_{ij} \quad \text{s.t. } \mathbf{Z}^\top\mathbf{Z} = \mathbf{I}. \tag{4}$$

Typically, the adjacency matrix $\mathbf{A}$ is constructed by employing the similarities among samples, which can be defined by Euclidean distance, cosine similarity or heat kernel, etc. In real-world applications, more and more graph data carrying semantic information are continuously generated, in which the semantic connections among samples are crucial. It is insufficient to capture connections among samples leveraging only sample similarities, as some dissimilar samples may be potentially related and have semantic connections. Therefore, it is natural to incorporate semantic information to obtain a better topology network:

$$\min_{\mathbf{Z}} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left\|\mathbf{z}_i - \mathbf{z}_j\right\|_2^2 \left(\alpha[\mathbf{A}_k]_{ij} + [\mathbf{A}_s]_{ij}\right) \quad \text{s.t. } \mathbf{Z}^\top\mathbf{Z} = \mathbf{I}, \tag{5}$$

where $\mathbf{A}_k$ denotes the adjacency matrix constructed by sample similarities, and $\mathbf{A}_s$ is the adjacency matrix defined by semantic information. α is a hyperparameter balancing the important of similarity adjacency matrix. $\mathbf{A}_s$ is the original graph structure from datasets, defined by semantics like citation, co-authorship and web hyperlink. $\mathbf{A}_k$ is a binary matrix like $\mathbf{A}_s$, and we employ KNN algorithm for the calculation of $\mathbf{A}_k$, which means the elements of $\mathbf{A}_k$ can be specified as

$$[\mathbf{A}_k]_{ij} = \begin{cases} 1, \ \mathbf{x}_i \in \mathrm{KNN}(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in \mathrm{KNN}(\mathbf{x}_i), \\ 0, \text{ otherwise}, \end{cases} \tag{6}$$

where KNN(·) denotes the set of k nearest neighbors. Although Eq. (5) fuses the similarity information from the feature space and additional semantic information, the obtained adjacency matrix is fixed after pre-definition, where the outliers in the adjacency matrix $\mathbf{A}_k$ dominate the objective function. Actually, this effect stems significantly from the ℓ_2-norm in Eq. (5). Several previous works [35,36] have revealed that ℓ_2-norm is sensitive to outliers, while $\ell_{2,1}$-norm is not. However, $\ell_{2,1}$-norm is sensitive to small values. In order to take the advantages of ℓ_2-norm and $\ell_{2,1}$-norm, a more flexible $\ell_{2,p}$-norm is adopted, and Eq. (5) is transformed into

$$\min_{\mathbf{Z}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\alpha \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^p [\mathbf{A}_k]_{ij} + \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^2 [\mathbf{A}_s]_{ij} \right) \text{ s.t. } \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}. \tag{7}$$

Using simple algebra, Eq. (7) is equivalent to

$$\min_{\mathbf{Z}, \mathbf{A}_f} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^2 [\mathbf{A}_f]_{ij} \ \text{ s.t. } \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}, \tag{8}$$

where $\mathbf{A}_f$ is the fused adjacency matrix computed as

$$[\mathbf{A}_f]_{ij} = [\mathbf{A}_s]_{ij} + \frac{\alpha [\mathbf{A}_k]_{ij}}{\left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^{2-p}}. \tag{9}$$

As mentioned before, we aim to explore the potential manifold structure in the feature space and integrate semantic information to learn a better embedding $\mathbf{Z}$. Nevertheless, the constructed adjacency matrix $\mathbf{A}_k$ usually contains some edges that connect nodes that are outliers, which are called outlier edges in this paper. Motivated by this issue, we propose the objective function to adaptively learn a more robust representation. According to Eq. (9), if nodes i and j are far away but been connected, this edge is regarded as outlier edge, the denominator $\left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^{2-p}$ would be large and the value of $\frac{[\mathbf{A}_k]_{ij}}{\left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^{2-p}}$ would decrease. By this way, we damp the effect from outliers. Further, if there exists a semantic edge between nodes i and j, the connection will be emphasized by adding $[\mathbf{A}_s]_{ij}$ to the fused adjacency matrix even though the value of $[\mathbf{A}_s]_{ij}$ is suppressed.

Then we consider optimizing Problem (8). The goal is to obtain a robust low-dimensional representation $\mathbf{Z}$, and we have two variables $\mathbf{Z}$ and $\mathbf{A}_f$. Therefore, an iterative algorithm is adopted to alternatively update the two variables and minimize the objective function. First, we fix $\mathbf{Z}$, and then $\mathbf{A}_f$ is updated by

$$[\mathbf{A}_f]_{ij}^{(t)} = [\mathbf{A}_s]_{ij} + \frac{\alpha [\mathbf{A}_k]_{ij}}{\left\| \mathbf{z}_i^{(t)} - \mathbf{z}_j^{(t)} \right\|_2^{2-p}}, \tag{10}$$

Next, we shelve the orthogonal constraint for now and propose a relaxing method in Section 4.2. While fixing $\mathbf{A}_f$, the optimal $\mathbf{Z}$ can be sought by solving the following objective function:

$$\min_{\mathbf{Z}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left\| \mathbf{z}_i - \mathbf{z}_j \right\|_2^2 [\mathbf{A}_f]_{ij} = \min_{\mathbf{Z}} Tr\left(\mathbf{Z}^\top \mathbf{L}_f \mathbf{Z} \right), \tag{11}$$

where $\mathbf{L}_f$ is the Laplacian matrix computed on $\mathbf{A}_f$. The derivative of (11) with respect to $\mathbf{Z}$ is

$$\frac{\partial Tr\left(\mathbf{Z}^\top \mathbf{L}_f \mathbf{Z} \right)}{\partial \mathbf{Z}} = 2\mathbf{L}_f \mathbf{Z}. \tag{12}$$

Setting Eq. (12) to zero, we have

$$\mathbf{L}_f \mathbf{Z} = 0 \ \Rightarrow \ \mathbf{Z} = \hat{\mathbf{A}}_f \mathbf{Z}, \tag{13}$$

where $\hat{\mathbf{A}}_f$ is the normalized adjacency matrix $\hat{\mathbf{A}}_f = \tilde{\mathbf{D}}^{-1} \tilde{\mathbf{A}}_f$ with $\tilde{\mathbf{A}}_f = \mathbf{A}_f + \mathbf{I}$ being the adjacency matrix with self-loop. This equation can be explained as a limit distribution where $\mathbf{Z}_{lim} = \hat{\mathbf{A}}_f \mathbf{Z}_{lim}$ following the suggestion of [34,37], then an iterative formula is utilized to update $\mathbf{Z}$:

$$\mathbf{Z}^{(t+1)} = \hat{\mathbf{A}}_f^{(t)} \mathbf{Z}^{(t)}. \tag{14}$$

By initializing the learned embedding $\mathbf{Z}$ as $\mathbf{Z}^{(0)} = \mathbf{XW}$ and repeating Eq. (14), the computation of $\mathbf{Z}^{(t+1)}$ follows

$$\mathbf{Z}^{(t+1)} = \hat{\mathbf{A}}_f^{(t)} \cdots \hat{\mathbf{A}}_f^{(0)} \mathbf{XW}. \tag{15}$$

where $\mathbf{W}$ is a feature mapping applied on $\mathbf{X}$, and we decompose it into multiple mappings as $\mathbf{W} = \mathbf{W}^{(0)} \cdots \mathbf{W}^{(t)}$ and perform one feature mapping in each update, thus we have

$$\mathbf{Z}^{(t+1)} = \hat{\mathbf{A}}_f^{(t)} \mathbf{Z}^{(t)} \mathbf{W}^{(t)}. \tag{16}$$

Note that assuming that convergence is reached in $t+1$ iterations, we decompose $\mathbf{W}$ into $t+1$ projections so that the solution $\mathbf{Z}^{(t+1)}$ is the same while we add a projection operation for each update of $\mathbf{Z}$. By imposing certain constraints on $\mathbf{Z}$, we conduct projection optimization after each update of $\mathbf{Z}$. For example, projecting $\mathbf{Z}$ onto the non-negative plane $\mathcal{S}_+ = \left\{ \mathbf{s} \in \mathbb{R}^d \mid \mathbf{s} \geq \mathbf{0} \right\}$ is equal to adding ReLU function, then Eq. (16) is reformulated as

$$\mathbf{Z}^{(t+1)} = \sigma\left(\hat{\mathbf{A}}_f^{(t)} \mathbf{Z}^{(t)} \mathbf{W}^{(t)} \right), \tag{17}$$

where σ is a projection to a certain set. We find that Eq. (17) naturally meets the forward propagation of GCN. In other words, we propose a flexible Laplacian Embedding in Problem (8) and establish its potential connection with GCN through the optimization of this problem. Further, inspired by the acquired conclusion, we design a block-wise neural network with the following alternating forward computation

$$[\mathbf{A}_f]_{ij}^{(l)} = [\mathbf{A}_s]_{ij} + \frac{\alpha [\mathbf{A}_k]_{ij}}{\left\| \mathbf{z}_i^{(l)} - \mathbf{z}_j^{(l)} \right\|_2^{2-p}}, \tag{18}$$

$$\mathbf{Z}^{(l+1)} = \sigma\left(\hat{\mathbf{A}}_f^{(l)} \mathbf{Z}^{(l)} \mathbf{W}^{(l)} \right), \tag{19}$$

where $\mathbf{Z}^{(l)}$, $\mathbf{A}_f^{(l)}$ and $\mathbf{W}^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l+1)}}$ denote the learned embedding, the adjacency matrix, and the learnable weight in the lth block respectively, where $\mathbf{W}^{(l)}$ is updated by the back propagation. The main difference between Eq. (19) and the form of GCN is that the graph filter constructed by $\hat{\mathbf{A}}_f^{(l)}$ is learnable. To be specific, $\hat{\mathbf{A}}_f^{(l)}$ is calculated with the obtained $\mathbf{Z}^{(l)}$ and the new forward computation defined by Eq. (18) is termed as **A**daptive **T**opology learning **L**ayer (ATL). Each block is composed of the **G**raph **C**onvolutional **L**ayer (GCL) and ATL. By stacking the two types of layer, the form of a 2-block neural network can be simplified as

$$\mathbf{Z}^{(2)} = \hat{\mathbf{A}}_f^{(1)} \operatorname{ReLU}\left(\hat{\mathbf{A}}_f^{(0)} \mathbf{XW}^{(0)} \right) \mathbf{W}^{(1)}, \tag{20}$$

where $\hat{\mathbf{A}}_f^{(0)}$ is the initial graph filter and $\hat{\mathbf{A}}_f^{(1)}$ is adaptively calculated with the learned $\mathbf{Z}^{(1)}$. In particular, we initialize $\hat{\mathbf{A}}_f^{(0)}$ as $\hat{\mathbf{A}}_s$ for $l = 0$, and the input is the feature matrix $\mathbf{X}$.

4.2. Set-level orthogonal constraint

As an end-to-end model, the proposed GCNet generates representations from input data and adopts cross entropy loss to make the predicted results as close as possible to the ground truth. It is observed that the one-hot vectors in the learned representation denoting given classes of samples are orthogonal. Therefore, cross entropy loss implicitly requires set-level orthogonality of logits, although it does not explicitly impose constraints on the learned representation $\mathbf{Z}$. Namely, cross entropy loss encourages $\mathbf{Z}$ to satisfy the column-wise orthogonality, which simultaneously meets the aforementioned requirement of the orthogonal constraint $\mathbf{Z}^\top \mathbf{Z} = \mathbf{I}$. Besides, GCNet is designed with capacity to adaptively refine the topology and better propagate label signals, thereby these capacities can be leveraged to enhance the cross entropy loss. Thus, it is crucial to find a way to employ the model

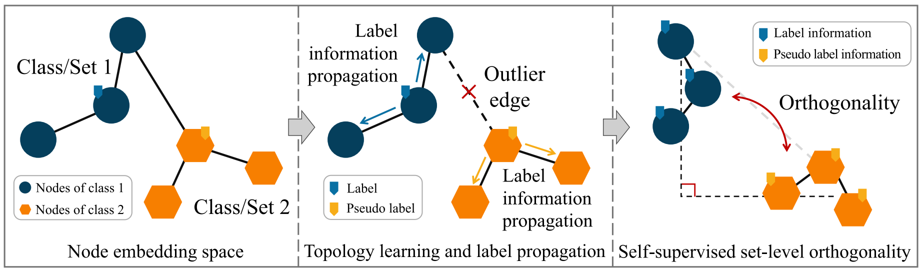

Fig. 3. An illustration about how the label signal propagation with adaptive topology and set-level orthogonal loss work. GCL and ATL eliminate the effect of outlier edges and perform robust label propagation. Under the guidance of labels and pseudo labels, different sets are pushed to be orthogonal.

strengths and achieve the orthogonality of $\mathbf{Z}$. Fig. 3 illustrates our approach.

A common and straightforward way is adding the relaxed constraint to the loss function, but this may lead to trivial solutions and does not fully utilize the model capacities. Therefore, we consider this problem from another perspective. Our target is to learn an optimal representation which partitions nodes into c classes. However, GCN does not clearly consider the intra-class or inter-class distances of nodes, thus the learned embeddings are not able to well separate different classes in the feature space. By performing softmax, the logits are transformed into probabilities. Denoting the column vector in $\mathbf{Z}$ as $\mathbf{z}^{(i)}$, the jth element of $\mathbf{z}^{(i)}$ indicates the probability of the jth node belonging to the ith class. Consequently, we consider the global structure in the feature space. Orthogonality among column vectors can be redescribed as maximizing values of samples belonging to the ith class and minimizing values of samples belonging to various classes in $\mathbf{z}^{(i)}$, which means to preserve inter-class separation and intra-class clustering in the feature space. Further, this target is implemented by the label information and the robust label signal propagation of GCNet. For the semi-supervised node classification with c classes, we construct $\mathbf{M} \in \mathbb{R}^{n\times c}$ encoding the labeled samples, where $\mathbf{M}_{ij} = 1$ if the ith labeled sample belongs to the jth class and otherwise $\mathbf{M}_{ij} = 0$. By denoting the similarity between $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(j)}$ as $\langle \mathbf{z}^{(i)}, \mathbf{z}^{(j)} \rangle$, the intra-class similarity can be calculated by

$$\mathcal{L}_s = \frac{1}{n_s} \sum_{\substack{i,j=1 \\ i\neq j}}^{n} \langle \mathbf{z}^{(i)}, \mathbf{z}^{(j)} \rangle^2 [\mathbf{M}_s]_{ij} = \frac{1}{n_s} \|\mathbf{M}_s \odot \mathbf{S}\|_F^2, \tag{21}$$

where $\odot$ is Hadamard product and $\mathbf{S} \in \mathbb{R}^{n\times n}$ denotes a similarity matrix with $[\mathbf{S}]_{ij} = \langle \mathbf{z}^{(i)}, \mathbf{z}^{(j)} \rangle$. $\mathbf{M}_s \in \mathbb{R}^{n\times n}$ is the mask matrix satisfying $\mathbf{M}_s = \mathbf{M}\mathbf{M}^\top$, that is, $[\mathbf{M}_s]_{ij}$ equals to 1 when samples i and j are both labeled and belong to the same class. $[\mathbf{M}_s]_{ij} = 0$ means that samples i and j have different labels or at least one of them is unlabeled. Besides, $[\mathbf{M}_s]_{ii} = 0$ is set to satisfy the requirement $i \neq j$ in Eq. (21), and $n_s = \sum_{i,j=1}^{n} [\mathbf{M}_s]_{ij}$ is the number of masked samples. Analogously, we calculate the inter-class similarity as

$$\mathcal{L}_d = \frac{1}{n_d} \|\mathbf{M}_d \odot (\mathbf{1} - \mathbf{S})\|_F^2, \tag{22}$$

where $\mathbf{M}_d$ corresponds to the matrix selecting labeled samples from different classes with $n_d = \sum_{i,j=1}^{n} [\mathbf{M}_d]_{ij}$, and $\mathbf{1}$ is a matrix with all elements being 1. Therefore, we can approximate the orthogonality of $\mathbf{Z}$ by minimizing $\mathcal{L}_s$ as well as maximizing $\mathcal{L}_d$. To guarantee that the loss has a lower bound, we specify $\langle \cdot, \cdot \rangle$ as the square of cosine similarity $\cos(\cdot, \cdot)$.

In summary, via the label signal propagation of GCNet, the labeled samples guide the latent representations of all samples to be set-level orthogonal. Nevertheless, when tackling semi-supervised classification tasks, the labeled samples are scarce. So we introduce a confidence-based self-supervised mechanism. Specifically, we leverage the learned embedding $\mathbf{Z}$ to evaluate the confidence of the prediction for each sample, and select the predicted labels with confidence above a preset threshold γ. After every certain number of iterations, these pseudo labels are utilized to update $\hat{\mathbf{M}}_s$ and $\hat{\mathbf{M}}_d$, the mask matrices with pseudo labels. Thus the loss function is defined as

$$\mathcal{L}_{SO} = \hat{\mathcal{L}}_s + \hat{\mathcal{L}}_d, \tag{23}$$

where $\hat{\mathcal{L}}_s$ and $\hat{\mathcal{L}}_d$ denote $\mathcal{L}_s$ and $\mathcal{L}_d$ with additional pseudo labels.

Algorithm 1: Graph Convolutional Network with Elastic Topology (GCNet)

Input: Graph $\mathcal{G} = (\mathbf{X}, \mathbf{A}_s)$, supervised information $\mathbf{Y}$, hidden layers L, layer sizes $\{h_1, h_2, \dots, h_L\}$, number of nearest neighbors k, interval for updating pseudo labels λ, confidence threshold γ, hyperparameters α, β and p.

Output: The learned embedding $\mathbf{Z}$.

```
1:  Construct adjacency matrix $\mathbf{A}_k$ by KNN;
2:  Generate selection matrices $\hat{\mathbf{M}}_s$ and $\hat{\mathbf{M}}_d$;
3:  Initialize all learnable parameter matrices $\{\mathbf{W}^{(l)}\}_{l=0}^{L-1}$ and fused
    adjacency matrix $\mathbf{A}_f$;
4:  for $i \to$ max training epochs do
5:    for $j \to L$ do
6:      Calculate $\mathbf{A}_f^{(l)}$ with Equation (18);
7:      Compute $\mathbf{Z}^{(l+1)}$ with Equation (19);
8:    end for
9:    if $i$ % $\lambda = 0$ then
10:     Select the samples with confidence above the threshold $\gamma$;
11:     Generate pseudo labels for the selected unlabeled samples;
12:     Update $\hat{\mathbf{M}}_s$ and $\hat{\mathbf{M}}_d$ by the generated pseudo labels;
13:   end if
14:   Update $\{\mathbf{W}^{(l)}\}_{l=0}^{L-1}$ by back propagation with loss $\mathcal{L}$;
15: end for
16: return The final output $\mathbf{Z}$.
```

4.3. Training details

We leverage an adaptive topology learning layer and a set-layer orthogonal constraint to capture disparity and correlation between nodes in the feature space, and establish a theoretical connection between these two components. In this subsection, we introduce the training strategy and computational complexity of the proposed framework, and the procedure of GCNet is described in Algorithm 1. For the semi-supervised node classification, it is common to measure the loss function by the cross entropy

$$\mathcal{L}_{CE} = -\sum_{i\in\Omega} \sum_{j=1}^{c} \mathbf{Y}_{ij} log \hat{\mathbf{Y}}_{ij}, \tag{24}$$

where $\mathbf{Y} \in \mathbb{R}^{n\times c}$ denotes the ground-truth label matrix generated from the set Ω which is a tiny part of the entire label space, and $\hat{\mathbf{Y}} \in \mathbb{R}^{n\times c}$ is the predicted label matrix. To build an approximately orthogonal embedding $\mathbf{Z}$, we adopt $\mathcal{L}_{CE}$ and $\mathcal{L}_{SO}$ to form the total loss, i.e.,

$$\mathcal{L} = \mathcal{L}_{CE} + \beta \mathcal{L}_{SO}, \tag{25}$$

Table 2
Detailed statistics of all test datasets.

Datasets	# Nodes	# Edges	# Classes	# Features	Data types
Cora	2,708	5,429	7	1,433	Citation network
Citeseer	3,327	4,732	6	3,703	Citation network
ACM	3,025	13,128	3	1,870	Paper network
BlogCatalog	5,196	171,743	6	8,189	Social network
Flickr	7,575	239,738	9	1,2047	Social network
UAI	3,067	28,311	19	4,973	Webpage network

where β is a hyperparameter that balances the two losses. Next we analyze the computational complexity of the proposed GCNet. For a single block of GCNet, we first consider ATL. The computation in ATL can be considered as a Hadamard product of matrices and one matrix addition with a computational complexity of $\mathcal{O}(n^2)$. And the computational complexity of the lth GCL is $\mathcal{O}(n^2 d^{(l)} + nd^{(l)}d^{(l+1)})$. Denoting the maximum value of set $\{d^{(1)}, d^{(2)}, \dots, d^{(L)}\}$ as d, it consumes $\mathcal{O}(n^2 d + nd^2)$ for each GCL. In particular, the first GCL requires $\mathcal{O}(n^2 m + nmd)$, where m is the feature dimension of the original input. Owing to $d \ll min\{m, n\}$, the total computation requires $\mathcal{O}(n(n+d)(m+Ld))$ when we build an L-layer network.

5. Experiment

5.1. Experimental setting

Six public benchmark datasets are utilized to perform comparative experiments: Cora, Citeseer, BlogCatalog, ACM, Flickr and UAI, and their statistics are shown in Table 2. The detailed descriptions are given as follows.

Cora is a well-known citation network. For details, each node represents a paper, where each paper is classified by its domain, and the edges represent the citation relationships between papers.

Citeseer is a benchmark dataset. As a paper citation network, the nodes and edges have the same meaning as Cora, and the node features are bag-of-words.

BlogCatalog is a social network formed by bloggers and their relationships, where bloggers are divided into 6 types. Node features are extracted from the keywords of user information.

ACM is a paper network, where nodes denote papers. Differing from citation networks, an edge exists between two papers when they have the same authors.

Flickr is a social network, where nodes represent users and edges represent that a user has added another user as a contact. All users are grouped into 9 categories according to their interests.

UAI is a dataset that has been tested with GCN for community detection, which is a webpage citation network. Nodes represent web pages and each edge represents a citation/

To validate the proposed framework, we focus on the performance of semi-supervised node classification and compare our method with several state-of-the-art methods, including Chebyshev [38], GCN [1], GAT [39], ScGCN [40], GNN-LF/GNN-HF [34], AdaGCN [20], AMGCN [16], SSGC [41] and DefGCN [42]. The descriptions of these methods and some detailed settings are given below.

Chebyshev [38] establishes a fast convolution filter on graphs based on spectral theory, which is approximated by Chebyshev polynomials.

GCN [1] explores previous works and further proposes a simplified graph filter, which is a first-order approximation of the truncated Chebyshev polynomial.

GraphSAGE [43] trains a graph neural network, which learns embeddings by incorporating features from local neighborhoods of nodes.

GAT [39] employs masked self-attentional layers to build a graph neural network, enabling nodes to focus on the features of their neighborhoods.

SGC [22] assumes that the nonlinearity is not critical, and simplifies GCN by removing the nonlinearity and collapsing the weight matrix between consecutive layers.

APPNP [44] investigates the relationship between GCN and personalized Pagerank, and puts forward an improved propagation scheme.

ScGCN [40] combines classical GCN filters and filters that are defined by geometric scattering transform to form a hybrid GCN framework.

GNN-LF/GNN-HF [34] is designed via two objective functions with graph kernels considering low-pass or high-pass graph filters respectively.

AdaGCN [20] improves GCN by leveraging Adaboost strategy. It updates layer weights iteratively and allows information to be shared between distinct layers.

AMGCN [16] constructs a multi-channel graph neural network to adaptively fuse topology and node features, utilizing the attention mechanism and the weight sharing mechanism.

SSGC [41] derives an enhanced variant of GCN by adopting a modified Markov diffusion kernel, aiming to aggregate information over larger neighborhoods.

DefGCN [42] designs deformable graph convolutional kernels, which can be performed in different latent spaces to capture long-range relations and adapt to different neighbors.

For a fair comparison, all the configurations are set to default values in the original papers. Then we elaborate the setting of the proposed GCNet.

In the semi-supervised node classification task, only a small fraction of samples is labeled. Following the settings of vanilla GCN, we randomly split samples into a small set of 20 labeled samples per class for training, a set of 500 samples for validating and a set of 1,000 samples for testing. Note that, all the three sets are randomly selected, and are adopted for all algorithms. The learning rate is set to 1×10^{-2}, the weight decay is 5×10^{-4} and the size of hidden units is chosen as 16. The hyperparameter α is set between 1×10^{-2} to 1×10^{2} according to the datasets. All these hyperparameters are selected on the validation set. And p is fixed as 1.5 during all the experiments. Values of pseudo labels depend on the confidence threshold γ, which is fixed as 0.95. In this paper, the proposed framework is implemented by PyTorch platform and run on the computer with AMD R9-5900X CPU, Nvidia RTX 3060 GPU and 72 GB of RAM.

5.2. Node classification

Performance Comparison: The accuracy and F1-score of all methods in the semi-supervised node classification task are reported in Table 3, which demonstrates that the proposed method achieves impressive performance and ranks at the top for both metrics. To facilitate observation, we mark the highest value in red and the second-highest value in blue. As can be seen, our method takes considerable advantage over state-of-the-art GCN-based methods, especially on BlogCatalog and Flickr datasets, outperforming the second-place algorithm by almost 10%. GCNet captures a potential geometric structure in the feature space and dynamically adjusts the graph filters, thus obtaining remarkable performance and robustness. In particular, AMGCN also achieves competitive rank on most datasets only after GCNet, that also devoted to fusing node features and topologies and thus deriving richer information. However, AMGCN employs two fixed topologies to form three channels for propagation, which indicates that the proposed ATL is effective. The conclusion can also be verified in the ablation experiments. In addition, Fig. 4 demonstrates the curves of GCNet in terms of loss value, accuracy and F1-score. It can be observed that the loss values drop rapidly and converge within 300 epochs, while the accuracy and F1-score increase fast with the decrease of loss values and maintain stable after convergence.

Visualization: The t-SNE visualizations of node embeddings on BlogCatalog dataset are plotted in Fig. 5, which are generated by 12

Table 3
Comparative results (mean% and standard deviation%) of all compared methods with accuracy (ACC) and F1-score (F1), where the best results are highlighted in red and the second-best results are highlighted in blue. GraphSAGE fails to work on the ACM dataset, where the relevant results are marked with "–" in the table.

Metrics	Methods/Datasets	Cora	Citeseer	ACM	BlogCatalog	Flickr	UAI
ACC	Chebyshev [38]	76.2 (0.7)	69.3 (0.4)	82.8 (1.4)	62.5 (1.6)	38.5 (1.6)	49.7 (0.4)
	GCN [1]	79.3 (0.9)	70.1 (0.5)	88.5 (0.7)	84.6 (1.1)	55.5 (0.6)	53.6 (2.1)
	GraphSAGE [43]	76.7 (0.6)	64.4 (0.9)	–	54.9 (0.7)	32.7 (1.0)	41.7 (1.4)
	GAT [39]	79.1 (0.8)	68.3 (0.5)	84.6 (0.5)	65.3 (1.7)	40.4 (0.9)	49.7 (3.0)
	SGC [22]	77.1 (0.0)	67.0 (0.0)	80.4 (0.1)	73.5 (0.2)	51.0 (0.1)	56.5 (3.5)
	APPNP [44]	77.9 (0.1)	66.8 (0.0)	83.2 (0.1)	81.7 (0.1)	49.6 (0.1)	60.2 (0.1)
	ScGCN [40]	77.8 (0.9)	67.9 (0.2)	87.5 (0.7)	68.5 (1.4)	47.5 (1.3)	37.7 (3.6)
	GNN-LF [34]	**81.1 (0.5)**	72.3 (0.9)	90.8 (0.5)	**86.7 (0.6)**	56.6 (0.6)	36.6 (19.8)
	GNN-HF [34]	80.7 (0.2)	68.8 (1.3)	**91.2 (0.5)**	84.5 (0.4)	60.7 (0.4)	54.8 (1.4)
	AdaGCN [20]	75.4 (0.0)	66.7 (0.2)	85.1 (0.9)	80.7 (0.7)	59.3 (0.6)	45.9 (7.6)
	AMGCN [16]	79.3 (0.6)	**72.9 (0.7)**	89.9 (0.4)	85.8 (0.9)	**75.6 (0.9)**	**64.3 (0.9)**
	SSGC [41]	80.2 (0.8)	72.0 (0.5)	85.4 (0.2)	82.3 (0.8)	47.4 (0.2)	59.7 (1.0)
	DefGCN [42]	77.8 (1.0)	67.5 (1.7)	87.8 (0.3)	82.8 (3.4)	48.2 (2.5)	57.8 (1.8)
	GCNet	**81.5 (0.5)**	**73.2 (1.0)**	**92.6 (0.1)**	**95.0 (0.1)**	**83.3 (0.4)**	**65.8 (1.1)**
F1	Chebyshev [38]	76.3 (0.7)	65.4 (0.8)	82.5 (1.4)	62.0 (1.6)	38.4 (1.5)	39.1 (0.2)
	GCN [1]	77.7 (0.9)	66.5 (0.4)	88.8 (0.7)	82.9 (1.1)	53.7 (0.6)	44.3 (1.5)
	GraphSAGE [43]	76.7 (0.5)	60.7 (0.5)	–	54.7 (0.6)	31.0 (1.1)	35.3 (1.0)
	GAT [39]	77.1 (0.7)	64.6 (0.5)	84.8 (0.5)	63.6 (1.9)	38.1 (1.1)	40.8 (1.3)
	SGC [22]	75.1 (0.0)	62.5 (0.0)	80.6 (0.1)	65.8 (0.3)	44.2 (0.2)	46.7 (1.7)
	APPNP [44]	77.5 (0.1)	63.9 (0.0)	83.0 (0.1)	81.0 (0.2)	52.8 (0.2)	45.3 (0.5)
	ScGCN [40]	77.3 (0.9)	64.2 (1.5)	85.1 (0.8)	68.3 (1.5)	46.3 (1.2)	30.1 (3.7)
	GNN-LF [34]	79.1 (0.7)	66.7 (0.4)	90.8 (0.5)	**85.9 (0.6)**	54.3 (1.0)	29.7 (15.1)
	GNN-HF [34]	78.6 (0.3)	64.3 (1.7)	**91.3 (0.5)**	83.8 (0.4)	58.6 (0.6)	44.9 (0.8)
	AdaGCN [20]	75.4 (0.4)	66.7 (0.2)	91.3 (0.5)	80.7 (0.7)	59.3 (0.6)	45.9 (7.9)
	AMGCN [16]	78.2 (1.1)	**68.7 (0.4)**	90.0 (0.4)	85.3 (1.0)	**75.8 (1.0)**	**47.8 (0.6)**
	SSGC [41]	**79.2 (0.5)**	67.1 (0.5)	85.5 (0.2)	82.0 (0.8)	50.5 (0.3)	43.9 (1.0)
	DefGCN [42]	75.8 (0.8)	63.6 (1.6)	87.9 (0.2)	79.4 (4.1)	47.3 (2.5)	42.8 (1.5)
	GCNet	**79.7 (0.2)**	**69.4 (1.2)**	**92.8 (0.1)**	**93.9 (0.2)**	**82.9 (0.4)**	**49.3 (0.6)**

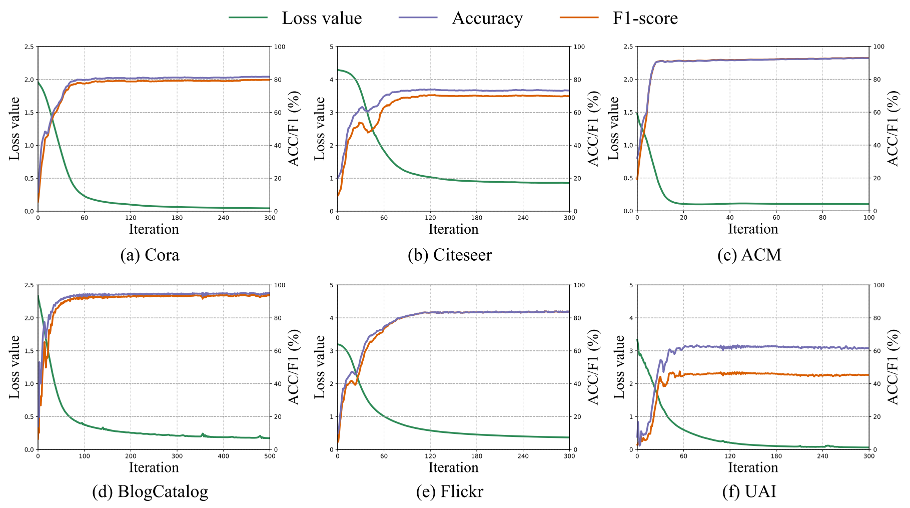

Fig. 4. Curves of loss values (green), accuracy (purple) and F1-score (orange) with on all datasets. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

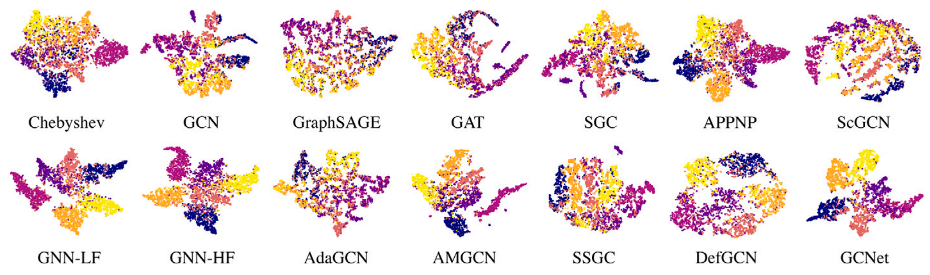

Fig. 5. t-SNE visualizations of representations generated by different methods.

Table 4
Ablation study (mean% and standard deviation%) on all datasets, where SO is set-level orthogonal constraint and SS is the self-supervised mechanism.

Datasets	ATL	SO	SS	ACC	F1	Datasets	ATL	SO	SS	ACC	F1
Cora				79.3 (0.9)	77.7 (0.9)	BlogCatalog				84.6 (1.1)	82.9 (1.1)
		✓		80.6 (0.4)	78.7 (0.7)			✓		84.8 (0.7)	83.0 (0.8)
		✓	✓	80.7 (0.7)	78.9 (1.0)			✓	✓	84.9 (0.6)	83.0 (0.8)
	✓			81.1 (0.6)	79.5 (0.1)		✓			94.4 (0.2)	93.0 (0.3)
	✓	✓		81.3 (0.5)	79.6 (0.4)		✓	✓		94.9 (0.3)	93.5 (0.6)
	✓	✓	✓	**81.5 (0.5)**	**79.7 (0.2)**		✓	✓	✓	**95.0 (0.1)**	**93.9 (0.2)**
Citeseer				70.1 (0.5)	66.5 (0.4)	Flickr				55.5 (0.6)	53.7 (0.6)
		✓		71.7 (0.6)	67.9 (0.7)			✓		56.5 (0.3)	54.6 (0.4)
		✓	✓	71.9 (0.5)	67.8 (0.5)			✓	✓	56.9 (0.8)	55.0 (1.0)
	✓			71.9 (0.4)	68.2 (0.3)		✓			79.5 (0.4)	79.0 (0.4)
	✓	✓		72.4 (0.5)	68.7 (0.6)		✓	✓		81.8 (0.4)	81.4 (0.3)
	✓	✓	✓	**73.2 (1.0)**	**69.4 (1.2)**		✓	✓	✓	**83.3 (0.4)**	**82.9 (0.4)**
ACM				88.5 (0.7)	88.8 (0.7)	UAI				53.6 (2.1)	44.3 (1.5)
		✓		89.6 (0.3)	89.9 (0.4)			✓		55.7 (1.4)	45.0 (0.4)
		✓	✓	89.9 (0.7)	90.1 (0.7)			✓	✓	55.6 (0.8)	45.2 (0.8)
	✓			91.2 (0.3)	91.3 (0.3)		✓			64.7 (1.6)	49.1 (1.3)
	✓	✓		91.7 (0.3)	91.8 (0.3)		✓	✓		65.0 (1.7)	49.2 (1.5)
	✓	✓	✓	**92.6 (0.1)**	**92.8 (0.1)**		✓	✓	✓	**65.8 (1.1)**	**49.3 (0.6)**

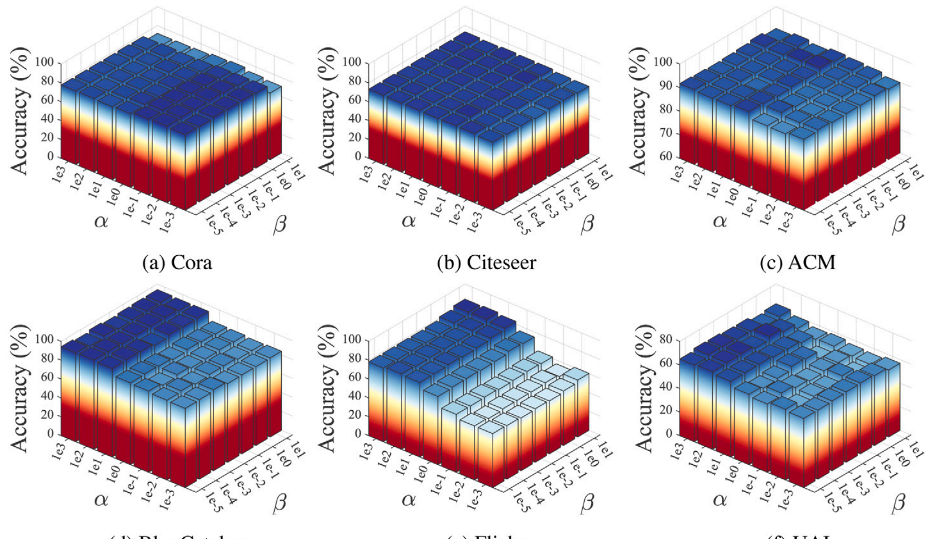

Fig. 6. Parameter sensitivity demonstration (accuracy) with respect to hyperparameters α and β on all datasets.

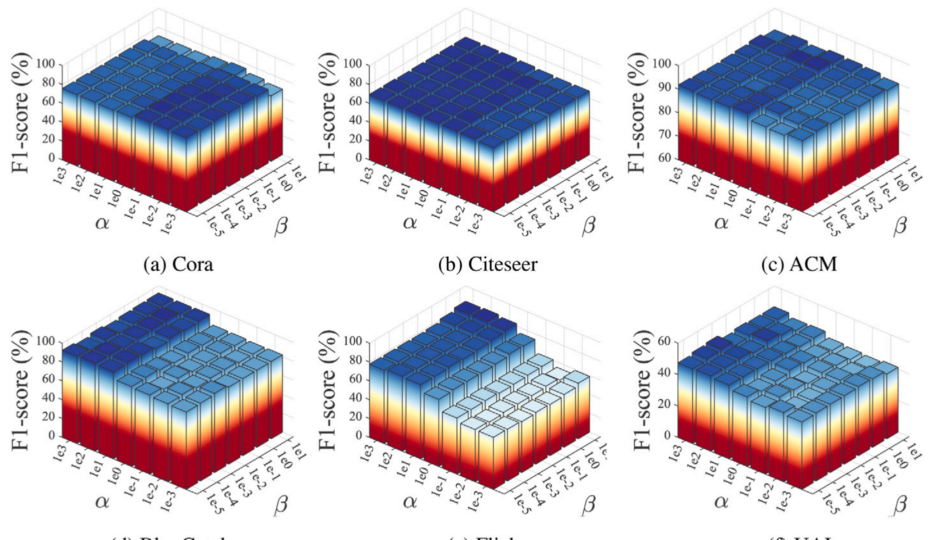

Fig. 7. Parameter sensitivity demonstration (F1-score) with respect to hyperparameters α and β on all datasets.

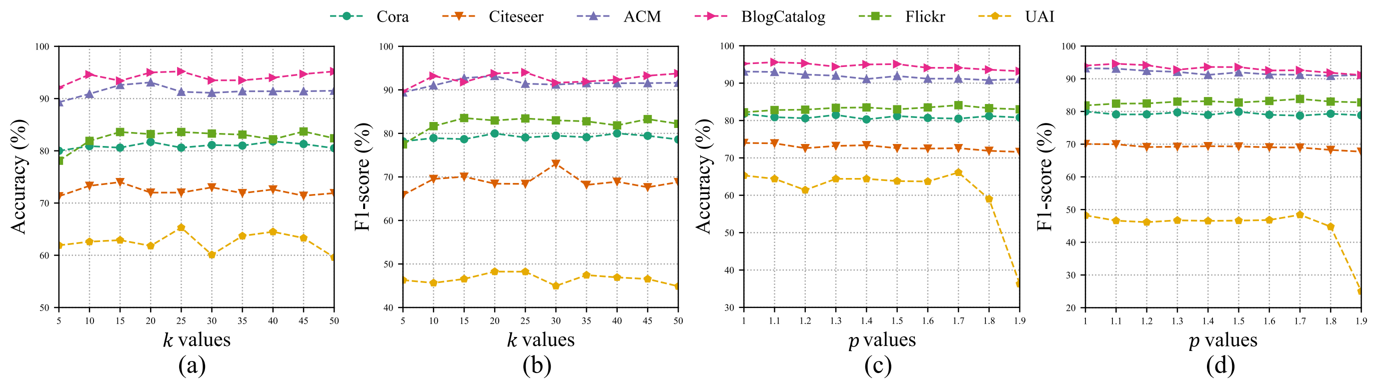

Fig. 8. Performance of GCNet with the varied hyperparameter k.

GCN-based models. It is observed that GCNet, GNN-LF and GNN-HF achieve decent intra-class clustering, but for GNN-LF and GNN-HF, different classes are more difficult to distinguish in the center part while the embedding of GCNet is with better inter-class separability. An interesting phenomenon is that AMGCN performs well with the classification accuracy, however its embedding shows relatively poor discriminating capacity. This may result from that topology constructed by similarity introduces undesired noise, which can be better filtered by GCNet.

Ablation Study: We conduct experiments to verify the impact of different components of the proposed framework, and the results are recorded in Table 4. Both adaptive topology learning layer and set-level orthogonal constraint provide considerable performance gains to the GCNet, and the framework obtains superior performance when combining the two. Note that the self-supervised mechanism works in conjunction with set-level orthogonal constraint and cannot be ablated alone. Observations reveal that it brings only small benefits without ATL, and even leads to a small decrease of accuracy on the UAI dataset. Nonetheless, when ATL is employed, the self-supervised mechanism yields a more positive effect on the framework. This phenomenon proves that ATL optimizes the embedding generated by GCNet, resulting in more reliable pseudo labels provided to self-supervised mechanism.

Parameter Sensitivity: We analyze the effects of hyperparameters α and β, and the experimental results are illustrated in Figs. 6 and 7. According to the observation, it can be concluded that the selection of these two hyperparameters has a remarkable influence on classification performance. More specifically, when the parameter α is too small, both accuracy and F1-score are at the plow. Also, too large α can lead to poor performance, so it is recommended that the value of α is between 1×10^{-2} and 1×10^{2}. In addition, the performance of GCNet varies with the parameter β in a similar trend to α, but is more stable. The performance achieves its peak when β ranges in $[1 \times 10^{-2}, 10]$. Additionally, Fig. 8 illustrate the variation of performance with parameters k and p. It can be seen that the middle part of the curve about k is higher than the two ends, but too large k values do not bring catastrophic results, owing to that ATL can effectively suppress the effect of the edges considered as outliers. And the curve about p generally shows a decreasing trend, that is, the larger the p is, the more susceptible the model is to outliers, which matches our theory.

6. Conclusion

In this paper, we proposed a block-wise framework, which mined deeper information in the feature space to adaptively adjust the topology, and provided a set-level orthogonal constraint approach to ensure theoretical rigor. We first reviewed the Laplacian embedding and put forward a flexible one. Then we analyzed the optimization of the formulated problem. Through a series of analyses, we established a connection between the flexible Laplacian embedding and GCN. Inspired by this, we constructed the basic block of GCNet, which consisted of an adaptive topology learning layer and a graph convolution layer. Moreover, we reconsidered the orthogonal constraint required by the flexible Laplacian embedding and introduced the set-level orthogonal constraint with the self-supervised mechanism. Experiments validated the effectiveness of our theory and revealed the encouraging performance of the proposed model. One possible limitation of this work is the loss depends on available labels, thus may has challenge to extend to unsupervised scenarios. In future work, we will pursue our efforts to further explore the application of the proposed framework to unsupervised learning.

CRediT authorship contribution statement

Zhihao Wu: Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Visualization, Writing – original draft, Writing – review & editing. **Zhaoliang Chen:** Conceptualization, Data curation, Investigation, Methodology, Writing – review & editing. **Shide Du:** Conceptualization, Investigation, Writing – review & editing. **Sujia Huang:** Conceptualization, Data curation, Investigation, Writing – review & editing. **Shiping Wang:** Funding acquisition, Supervision, Writing – review & editing.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (Nos. U21A20472 and 61672159).

References

[1] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: Proceedings of the 5th International Conference on Learning Representations, 2017, pp. 1–14.
[2] F. Manessi, A. Rozza, M. Manzo, Dynamic graph convolutional networks, Pattern Recognit. 97 (2020) 107000.
[3] Z. Zhang, C. Chen, Y. Chang, W. Hu, X. Xing, Y. Zhou, Z. Zheng, SHNE: Semantics and homophily preserving network embedding, IEEE Trans. Neural Netw. Learn. Syst. (2021) 1–12.
[4] Z. Wu, Z. Zhang, J. Fan, Graph convolutional kernel machine versus graph convolutional networks, Adv. Neural Inf. Process. Syst. 36 (2023) 1–14.
[5] C. Zhang, S. Xue, J. Li, J. Wu, B. Du, D. Liu, J. Chang, Multi-aspect enhanced graph neural networks for recommendation, Neural Netw. 157 (2023) 90–102.
[6] Q. Dai, X. Wu, L. Fan, Q. Li, H. Liu, X. Zhang, D. Wang, G. Lin, K. Yang, Personalized knowledge-aware recommendation with collaborative and attentive graph convolutional networks, Pattern Recognit. 128 (2022) 108628.
[7] Y. Chen, L. Huang, C. Wang, J. Lai, Hybrid-order gated graph neural network for session-based recommendation, IEEE Trans. Ind. Inform. 18 (3) (2022) 1458–1467.
[8] L.A. Passos, J.P. Papa, J. Del Ser, A. Hussain, A. Adeel, Multimodal audio-visual information fusion using canonical-correlated graph neural network for energy-efficient speech enhancement, Inf. Fusion 90 (2023) 1–11.
[9] Y. Gao, Y. Feng, S. Ji, R. Ji, HGNN$^+$: General hypergraph neural networks, IEEE Trans. Pattern Anal. Mach. Intell. 45 (3) (2023) 3181–3199.
[10] Z. Wu, X. Lin, Z. Lin, Z. Chen, Y. Bai, S. Wang, Interpretable graph convolutional network for multi-view semi-supervised learning, IEEE Trans. Multimed. 25 (2023) 8593–8606.
[11] Y. Chen, X. Xiao, C. Peng, G. Lu, Y. Zhou, Low-rank tensor graph learning for multi-view subspace clustering, IEEE Trans. Circuits Syst. Video Technol. 32 (1) (2022) 92–104.
[12] J. Cai, S. Wang, C. Xu, W. Guo, Unsupervised deep clustering via contractive feature representation and focal loss, Pattern Recognit. 123 (2022) 108386.
[13] J. Yu, A.L. Jia, Mlgal: multi-level label graph adaptive learning for node clustering in the attributed graph, Knowl.-Based Syst. 278 (2023) 110876.
[14] K. Ranasinghe, M. Naseer, M. Hayat, S.H. Khan, F.S. Khan, Orthogonal projection loss, in: Proceedings of IEEE/CVF International Conference on Computer Vision, 2021, pp. 12313–12323.
[15] Q. Li, Z. Han, X. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the 32rd AAAI Conference on Artificial Intelligence, 2018, pp. 3538–3545.
[16] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, J. Pei, AM-GCN: adaptive multi-channel graph convolutional networks, in: Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2020, pp. 1243–1253.
[17] K. Sun, Z. Lin, Z. Zhu, Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020, pp. 5892–5899.
[18] Z. Chen, Z. Wu, S. Wang, W. Guo, Dual low-rank graph autoencoder for semantic and topological networks, in: Proceedings of the 37th AAAI Conference on Artificial Intelligence, 37, (4) 2023, pp. 4191–4198.
[19] Z. Wu, L. Shu, Z. Xu, Y. Chang, C. Chen, Z. Zheng, Robust tensor graph convolutional networks via T-SVD based graph augmentation, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2022, pp. 2090–2099.
[20] K. Sun, Z. Zhu, Z. Lin, AdaGCN: Adaboosting graph convolutional networks into deep models, in: Proceedings of the 9th International Conference on Learning Representations, 2021, pp. 1–15.
[21] Y. Zhang, X. Wang, C. Shi, N. Liu, G. Song, Lorentzian graph convolutional networks, in: Proceedings of International World Wide Web Conference, 2021, pp. 1249–1261.
[22] F. Wu, A.H.S. Jr., T. Zhang, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying graph convolutional networks, in: Proceedings of the 36th International Conference on Machine Learning, Vol. 97, 2019, pp. 6861–6871.
[23] Y. Wang, Y. Wang, J. Yang, Z. Lin, Dissecting the diffusion process in linear graph convolutional networks, Adv. Neural Inf. Process. Syst. (2021) 5758–5769.
[24] R. Li, S. Wang, F. Zhu, J. Huang, Adaptive graph convolutional neural networks, in: Proceedings of the 32rd AAAI Conference on Artificial Intelligence, 2018, pp. 3546–3553.
[25] Z. Chen, C. Chen, Z. Zhang, Z. Zheng, Q. Zou, Variational graph embedding and clustering with Laplacian eigenmaps, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, 2019, pp. 2144–2150.
[26] J. Wang, C. Tang, X. Zheng, X. Liu, W. Zhang, E. Zhu, Graph regularized spatial-spectral subspace clustering for hyperspectral band selection, Neural Netw. 153 (2022) 292–302.
[27] Z. Li, C. Tang, X. Liu, X. Zheng, W. Zhang, E. Zhu, Consensus graph learning for multi-view clustering, IEEE Trans. Multimedia 24 (2022) 2461–2472.
[28] A. Huang, Z. Wang, Y. Zheng, T. Zhao, C. Lin, Embedding regularizer learning for multi-view semi-supervised classification, IEEE Trans. Image Process. 30 (2021) 6997–7011.
[29] S. Wang, Z. Chen, S. Du, Z. Lin, Learning deep sparse regularizers with applications to multi-view clustering and semi-supervised classification, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2022) 5042–5055.
[30] A.D. Col, F. Petronetto, Graph regularization multidimensional projection, Pattern Recognit. 129 (2022) 108690.
[31] Z. Kang, C. Peng, Q. Cheng, X. Liu, X. Peng, Z. Xu, L. Tian, Structured graph learning for clustering and semi-supervised classification, Pattern Recognit. 110 (2021) 107627.
[32] S. Huang, Y. Zhang, L. Fu, S. Wang, Learnable multi-view matrix factorization with graph embedding and flexible loss, IEEE Trans. Multimedia 25 (2023) 3259–3272.
[33] A. Huang, W. Chen, T. Zhao, C.W. Chen, Joint learning of latent similarity and local embedding for multi-view clustering, IEEE Trans. Image Process. 30 (2021) 6772–6784, http://dx.doi.org/10.1109/TIP.2021.3096086.
[34] M. Zhu, X. Wang, C. Shi, H. Ji, P. Cui, Interpreting and unifying graph neural networks with an optimization framework, in: Proceedings of International World Wide Web Conference, 2021, pp. 1215–1226.
[35] F. Nie, H. Wang, H. Huang, C.H.Q. Ding, Adaptive loss minimization for semi-supervised elastic embedding, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, 2013, pp. 1565–1571.
[36] Q. Wang, Q. Gao, X. Gao, F. Nie, $\ell_{2,p}$-Norm based PCA for image recognition, IEEE Trans. Image Process. 27 (3) (2018) 1336–1346.
[37] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: Proceedings of the 35th International Conference on Machine Learning, 2018, pp. 5453–5462.
[38] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016) 3837–3845.
[39] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: Proceedings of the 6th International Conference on Learning Representations, 2018, pp. 1–12.
[40] Y. Min, F. Wenkel, G. Wolf, Scattering GCN: overcoming oversmoothness in graph convolutional networks, Adv. Neural Inf. Process. Syst. 33 (2020) 14498–14508.
[41] H. Zhu, P. Koniusz, Simple spectral graph convolution, in: Proceedings of the 9th International Conference on Learning Representations, 2021, pp. 1–15.
[42] J. Park, S. Yoo, J. Park, H.J. Kim, Deformable graph convolutional networks, in: Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022, pp. 7949–7956.
[43] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Adv. Neural Inf. Process. Syst. 30 (2017) 1024–1034.
[44] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized PageRank, in: Proceedings of the 7th International Conference on Learning Representation, 2019, pp. 1–15.

Zhihao Wu received his B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2021. He is currently pursuing the M.S. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His current research interests include machine learning, deep learning, graph neural networks and multi-view learning.

Zhaoliang Chen received his B.S. degree from the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China in 2019. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. He is also currently visiting Faculty of Computer Science, University of Vienna, Vienna, Austria. His current research interests include machine learning, deep learning, graph neural networks and recommender systems.

Shide Du received his M.S. degree from the College of Computer and Data Science, Fuzhou University, Fuzhou, China in 2022. He is currently pursuing the Ph.D. degree with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His current research interests include machine learning, differentiable programming, and deep learning.

Sujia Huang received the B.S. degree from the College of Computer and Information Engineering, Jiangxi Normal University, Jiangxi, China, in 2021. She is currently working the M.S. degree with the College of Computing and Data Science, Fuzhou University, Fuzhou, China. Her research interests include machine learning, deep learning, and differentiable programming.

Shiping Wang received his Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China in 2014. He worked as a research fellow in Nanyang Technological University from August 2015 to August 2016. He is currently a professor with the College of Computer and Data Science, Fuzhou University, Fuzhou, China. His research interests include machine learning, computer vision and granular computing.