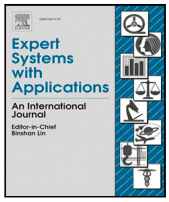
Expert Systems With Applications 226 (2023) 120178

Contents lists available at ScienceDirect

Expert Systems With Applications

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



CNIM-GCN: Consensus Neighbor Interaction-based Multi-channel Graph Convolutional Networks

Xiaofei Zhu [a,*], Chenghong Li [a], Jiafeng Guo [b], Stefan Dietze [c,d]

[a] College of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China
[b] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[c] Knowledge Technologies for the Social Sciences, Leibniz Institute for the Social Sciences, Cologne 50667, Germany
[d] Institute of Computer Science, Heinrich-Heine-University Düsseldorf, Düsseldorf 40225, Germany

ARTICLE INFO

Keywords:
Network representation learning
Deep learning
Graph convolutional networks
Node classification

ABSTRACT

Node classification plays a critical role in numerous network applications, and has attracted increasing attention in recent years. Existing state-of-the-art studies aim at maintaining common information between the topology graph and the feature graph in an implicit way, i.e., adopting a common convolution with parameter sharing strategy to preserve common information between the two graphs. Despite their effectiveness, these studies are still far from satisfactory due to the complex correlation information between the two spaces. To address this issue, we present a novel method named Consensus Neighbor Interaction-based Multi-channel Graph Convolutional Networks (CNIM-GCN). CNIM-GCN preserves the common information between the feature space and topology space in an explicit way by introducing a consensus graph for information propagation. A multi-channel graph convolutional networks is developed for effectively fusing information from different graphs. In addition, we further incorporate two types of consistency constraints, i.e., structural consistency constraint and reconstruction consistency constraint, to maintain the consistency between different channels. The former is leveraged to keep the consistency between different spaces at the structural relationship level, while the latter is used to preserve a consistency between the final node representation and the original node feature representation. We carry out extensive experiments on five real-world datasets, including ACM, BlogCatalog, CiteSeer, Flickr and UAI2010. Experimental results show that our proposed approach CNIM-GCN is superior to the state-of-the-art baselines.

1. Introduction

Node classification attempts to predict node labels by exploring topological structure or node features, and plays an important role in numerous network applications, such as social networks (Huang, Li, & Hu, 2017; Meng, Liang, Bao, & Zhang, 2019; Shen, Dai, Chung, Lu, & Choi, 2020; Yu, Wang, Liu, Böhm, & Shao, 2020), protein interaction networks (Hamilton, Ying, & Leskovec, 2017b; Yue et al., 2020), commerce networks (Hou et al., 2020; Liu, Fang, Liu and Hoi, 2021), and citation networks (Kipf & Welling, 2017; Sen et al., 2008; Tang et al., 2008; Wang, Wang, Guo, & Gong, 2021). Many approaches based on graph convolutional networks (GCNs) (Hamilton, Ying, & Leskovec, 2017a; Hang, Neville, & Ribeiro, 2021; Kipf & Welling, 2017; Veličković et al., 2018) have been developed for node classification, that learn node embeddings by propagating node feature information along the topological structure. The basic assumption of these methods is that the topological structure and node features are complementary to each other, which can be fused to achieve better performance for node classification. Kipf and Welling (2017) employ an efficient layer-wise propagation strategy for neural network models and operate directly on graphs. Hamilton et al. (2017a) learn to generate embeddings through sampling and aggregate feature information from a node's local neighborhood by training a set of aggregator functions. Veličković et al. (2018) propose a graph attention network on graph-structured data by assigning different weights to different nodes within a neighborhood. Despite these remarkable achievements, applying GCNs may not be able to adaptively extract the most correlated information between topological structure or node features. Some recent works (Klicpera, Bojchevski, & Günnemann, 2019; Li, Han, & Wu, 2018) show that the graph convolution of the GCN model would mix node features and its nearby neighbors, which can be considered as a special form of Laplacian smoothing.

Very recently, instead of propagating node feature information along the topological structure, some research efforts (Jin et al., 2021;

* Corresponding author.
E-mail addresses: zxf@cqut.edu.cn (X. Zhu), lch@2020.cqut.edu.cn (C. Li), guojiafeng@ict.ac.cn (J. Guo), stefan.dietze@gesis.org (S. Dietze).

https://doi.org/10.1016/j.eswa.2023.120178
Received 5 March 2022; Received in revised form 11 August 2022; Accepted 15 April 2023
Available online 21 April 2023

Wang et al., 2020) have been devoted to capture the underlying structure of nodes in the feature space. For example, SimP-GCN (Jin et al., 2021) integrates the node features with the structure information of the topology graph by constructing a feature graph and allowing nodes to adaptively combine information from these two graphs. The main limitation of SimP-GCN is that it simply integrates the feature graph with the topology graph in an adaptive way and may be insufficient for capturing the rich information embedded in both graphs. Different with SimP-GCN, AM-GCN (Wang et al., 2020) aims to derive a feature graph by constructing a k-nearest neighbor graph, and then propagate node features over both the topology graph and the feature graph in order to extract two corresponding embeddings in the two spaces. To capture the common information between the feature space and the topology space, it develops a common convolution module with parameter sharing to extract common embeddings shared by both topological structure and node features. Despite these significant achievements, AM-GCN models the complex correlation information between the two spaces in an implicit way by simply using a common convolution module with parameter sharing, which is distant from optimal.

In order to fully mine the rich information between topological structure and node features, we propose a novel method named Consensus Neighbor Interaction-based Multi-channel Graph Convolutional Networks (CNIM-GCN). More precisely, to address the limitation of AM-GCN, we propose to maintain the consensus information of the two graphs by explicitly introducing a novel graph referred to as consensus graph. The consensus graph can be considered as a bridge between the topology graph and the feature graph and serves for effective information propagation between them. Then a multi-channel graph convolutional networks is designed for effectively fusing information from different graphs. In order to propagate information learnt from three different channels (i.e, topology channel, feature channel and consensus channel), we leverage the consensus graph for updating the node representation of the topology graph and the feature graph. In addition, we further maintain the consistency of the three node representations from different channels with a structural consistency constraint. The assumption behind this constraint is that if two nodes are close to each other in the topology space (feature space), they should be close in the consensus space. Finally, a consistency between the final node representation and the original node feature representation is also maintained by incorporating a reconstruction consistency constraint. We carry out extensive experiments on five real-world datasets, and the results demonstrate that our proposed approach CNIM-GCN surpasses all the state-of-the-art baselines in terms of both metrics (i.e., Accuracy and F1-score). The main contributions of this paper are summarized as follows:

- We preserve the common information between the feature space and topology space in an explicit way by introducing a novel consensus graph for information propagation.
- We propose a novel consensus neighbor interaction-based multi-channel graph convolutional networks for node classification.
- We incorporate two types of consistency constraints, i.e., structural consistency constraint and reconstruction consistency constraint, to further maintain the consistency between different channels.
- Through extensive experiments on five real-world datasets, we show the effectiveness of the proposed approach CNIM-GCN over state-of-the-art baselines.

2. Related work

In the past few years, many graph convolutional network (GCN) based methods (Hamilton et al., 2017a; Kipf & Welling, 2017; Veličković et al., 2018) are proposed for the task of node classification, where each node aggregates feature information over the topology graph. For example, Kipf and Welling (2017) develop a layer-wise propagation strategy operated on graphs and propose a scalable semi-supervised classification method for graph-structured data. Hamilton et al. (2017a) propose a general framework GraphSAGE for inductive node embedding. Instead of simple convolution operations, GraphSAGE generates node embeddings via sampling and integrating features based on node's neighborhood information. Veličković et al. (2018) develop an attention-based convolution to obtain a node representation by attending to its neighbors on the topology graph.

Recent studies (Klicpera et al., 2019; Li et al., 2018) show that GCN-based methods usually perform a special form of Laplacian smoothing, which would mix node features and its nearby neighbors. To address this issue, Li et al. (2018) propose a co-training approach and a self-training approach to train GCN, where the former trains a GCN with a random walk model and the latter exploits the feature extraction capability. Abu-El-Haija et al. (2019) further present that GCN-based methods cannot learn general neighborhood mixing functions, and propose to capture neighborhood mixing relationships, e.g., averaging and delta operators, by repeatedly mixing feature representations of neighbors at different distances. Wang et al. (2020) propose to capture the underlying structure of nodes in feature space, and propagate information over both topology graph and feature graph. A common convolution module with the parameter sharing strategy is utilized to extract common embeddings shared by both topology structure and node features. Jin et al. (2021) attempt to propose a feature similarity preservation aggregation by integrating node features with the topology structure information in an adaptive way. It also employs self-supervised learning to model the complex feature similarity and dissimilarity relations between nodes.

More recently, some works propose to address the issue of limited structural information on tail nodes. Liu, Nguyen, and Fang (2021b) propose a novel graph neural network Tail-GNN. They introduce a transferable neighborhood translation to enhance representations of tail nodes by transferring information from head nodes with rich structural information to tail nodes. In addition, some research efforts (Zhang, Du, Xie, & Wang, 2021) have also been devoted to the cross-network scenarios, where a node classification model is learned by transferring knowledge from the source network to the target network. Zhang et al. (2021) propose an adversarial separation network (ASN), which is designed for cross-network domain adaptation. ASN models domain-private and domain-shared information in a separate way, where they employ two domain-private encoders to extract the domain-specific features in each network and employ a domain-invariant shared features across networks.

3. Method

In this section, we describe our proposed model, named Consensus Neighbor Interaction-based Multi-channel Graph Convolutional Networks (CNIM-GCN). The overall framework is shown in Fig. 1. The main idea of CNIM-GCN is that it introduces a novel consensus graph to maintain the information consistency between the topology space and the feature space for information propagation among different information spaces. In particular, we construct a consensus graph which captures the consensus neighbor information between the topology graph and the feature graph. Then a multi-channel graph convolutional module is utilized on topology graph, feature graph and consensus graph to propagate information. Considering that the node representations learned by the three graph convolutional modules contain different kinds of information, we allow node representations of different graphs to exchange information at each layer of GCNs. After that, we apply the attention mechanism to fuse the node representations $\mathbf{Z}_t$, $\mathbf{Z}_c$ and $\mathbf{Z}_f$ to obtain the final node representations $\mathbf{Z}$. In addition, two types of consistency constraints, i.e., structural consistency constraint and reconstruction consistency constraint, are incorporated to further maintain the consistency between different channels.

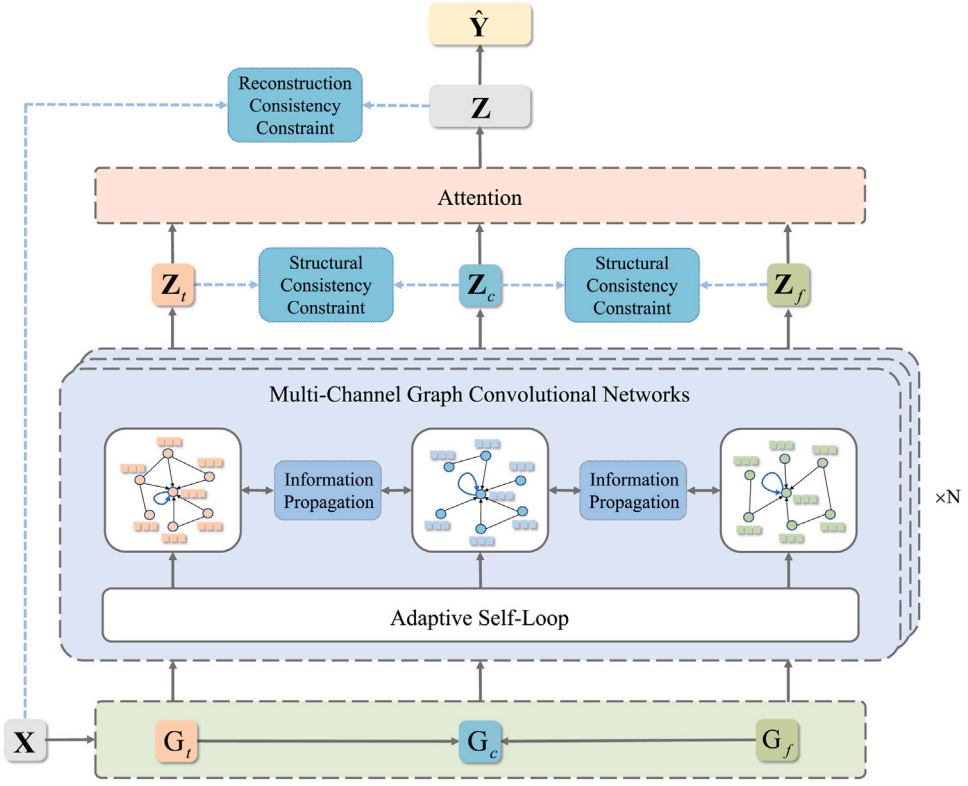


Fig. 1. The framework of CNIM-GCN model. CNIM-GCN consists of three graphs (i.e., topology graph, feature graph, and consensus graph), their corresponding graph convolutional modules, two types of consistency constraints (i.e., structural consistency constraint and reconstruction consistency constraint), and an attention module.

3.1. Problem definition

Let $G = (\mathbf{A}, \mathbf{X})$ denote a graph, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric adjacency matrix with n nodes, $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the node feature matrix, and d is the dimension of the node features. $\mathbf{Y}$ denotes the labels of nodes. Each node v_i in the graph has its feature vector $\mathbf{x}_i \in \mathbb{R}^d$ which is the ith row of $\mathbf{X}$, and its corresponding label $\mathbf{y}_i \in \{0,1\}^C$, where C represents the number of classes. Specifically, $\mathbf{A}_{ij} = 1$ indicates that there is an edge between node v_i and v_j, otherwise, $\mathbf{A}_{ij} = 0$. For semi-supervised classification, only the first m nodes ($0 < m \ll n$) have labels $\mathbf{Y}_L$ and the labels $\mathbf{Y}_U$ of the remaining data are missing. The goal is to learn a mapping function $f : \mathbf{A}, \mathbf{X}, \mathbf{Y}_L \rightarrow \mathbf{Y}_U$ to predict the missing labels $\mathbf{Y}_U$ for the unlabeled nodes.

3.2. Graph construction

The model in this paper contains three types of graphs, i.e., topology graph G_t, feature graph G_f, and consensus graph G_c. In the following, we will provide details for the processes of constructing each graph.

Topology Graph. In this work, we directly leverage node topology structure to construct the topology graph $G_t = (\mathbf{A}_t, \mathbf{X})$, where $\mathbf{A}_t = \mathbf{A} \in \mathbb{R}^{n\times n}$ is a symmetric adjacency matrix with n nodes and $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the node feature matrix, and d is the dimension of the node features.

Feature Graph. Due to the inevitable existence of noise edges in the original topology graph, applying conventional GCN on the topology graph would involve noise information propagation and lead to learn inferior node representation. A natural solution is to make full use of the rich information within node features $\mathbf{X}$ to participate in the node representation learning process (Jin et al., 2021; Nie, Jiao, Wang, Wang, & Tian, 2021; Nie et al., 2020). To this end, we construct a feature graph $G_f = (\mathbf{A}_f, \mathbf{X})$ based on the node feature matrix $\mathbf{X}$, where $\mathbf{A}_f \in \mathbb{R}^{n\times n}$ is the adjacency matrix of the kNN graph. To be specific, we first utilize the distance metric function (e.g., Cosine Similarity, Heat Kernel, Euclidean Distance) to calculate the similarity matrix $\mathbf{S} \in \mathbb{R}^{n\times n}$ among n nodes. Here we choose the Cosine Similarity to obtain $\mathbf{S}$. Specifically, denote $\mathbf{x}_i$ and $\mathbf{x}_j$ be the feature vectors of nodes v_i and v_j, we obtain:

$$\mathbf{S}_{ij} = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|\|\mathbf{x}_j\|}. \tag{1}$$

After obtaining the similarity matrix $\mathbf{S}$, we choose to the top k most similar nodes for each node to set edges, and obtain the adjacency matrix $\mathbf{A}_f$.

Consensus Graph. In order to effectively fuse the information of the topology graph G_t and the feature graph G_f, we propose to introduce a novel graph, named consensus graph, which is designed to maintain the consensus information within both graphs. The consensus graph can be considered as a bridge between the topology graph and the feature graph and serves for effective information propagation between them. For the consensus graph, it shares the same node set as the topology graph and feature graph, and there is an edge between two nodes in the consensus graph if the two nodes correlate to each other in both topology graph and feature graph. It is worth noting that if we define the correlation between two nodes as there is a direct connection between them in both topology graph and feature graph, it would suffer from the sparsity issue since two correlated nodes would not have a direct connection between them in both topology graph and feature graph. To alleviate this issue, we define the node correlation between two nodes (v_i, v_j) as follows:

Definition 1 (*Node Correlation*)**.** If the neighbors $\mathcal{N}_i$ of node v_i in one graph (e.g., topology graph) have an overlap with the neighbors $\mathcal{N}_j$ of node v_j in the other graph (e.g., feature graph), then the strength of the correlation between node v_i and node v_j is measured by the degree of the overlap of $\mathcal{N}_i$ and $\mathcal{N}_j$.

To measure the node correlation in the two different graphs, we first normalize the adjacency matrix $\mathbf{A}_t$ of the topology graph and the

adjacency matrix $\mathbf{A}_f$ of the feature graph:

$$\hat{\mathbf{A}}_t = \mathbf{D}_t^{-1}(\mathbf{A}_t + \mathbf{I}), \tag{2}$$

$$\hat{\mathbf{A}}_f = \mathbf{D}_f^{-1}(\mathbf{A}_f + \mathbf{I}), \tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{D}_t$ and $\mathbf{D}_f$ are the degree matrices of $(\mathbf{A}_t + \mathbf{I})$ and $(\mathbf{A}_f + \mathbf{I})$, respectively. Then the adjacency matrix $\hat{\mathbf{A}}_c$ of the consensus graph is obtained from $\hat{\mathbf{A}}_t$ and $\hat{\mathbf{A}}_f$ as follows:

$$\hat{\mathbf{A}}_c = \hat{\mathbf{A}}_t \cdot \hat{\mathbf{A}}_f^T, \tag{4}$$

and therefore we obtain the consensus graph $G_c = (\hat{\mathbf{A}}_c, \mathbf{X})$.

3.3. Multi-channel graph convolutional networks

With the three graphs, i.e., topology graph G_t, consensus graph G_c and feature graph G_f, on hand, we then introduce a multi-channel graph convolutional networks to effectively fuse information from different graphs. In the multi-channel graph convolutional networks, we first set an adaptive self-loop for each node of the three graphs, and then learn node representations with GCN on all three graph in parallel and allow node within different graphs to exchange information at each GCN layer.

Adaptive Self-loops. In the original GCN, given a node v_i, a self-loop is added to include its own features in the aggregation process. Gao and Ji (2019) believe that the features of nodes themselves may be more important in the prediction. If we want to retain more information about the original feature, we can add more self-loops to the node v_i. The importance of node features varies for different nodes (Jin et al., 2021). Therefore, an adaptive self-loop is set for each node of the three graphs:

$$\tilde{\mathbf{A}}_*^{(l)} = \hat{\mathbf{A}}_* + \gamma \mathbf{D}_*^{(l)}, \tag{5}$$

where $\tilde{\mathbf{A}}_*^{(l)}$ $(*\in \{t, c, f\})$ is the adjacency matrix of the graph G_* at the lth layer, γ is a hyperparameter that controls the contribution of the self-loop, and $\mathbf{D}_*^{(l)} = diag(k_{*1}^{(l)}, k_{*2}^{(l)}, \ldots, k_{*n}^{(l)})$ is the corresponding learnable diagonal matrix, where:

$$k_{*i}^{(l)} = \mathbf{z}_{*i}^{(l-1)} \mathbf{w}_1^{(l)} + b_1^{(l)}, \tag{6}$$

where $\mathbf{z}_{*i}^{(l-1)}$ $(*\in \{t, c, f\})$ is the hidden representation of the ith node in the graph G_* at the $(l-1)$th layer, $\mathbf{w}_1^{(l)}$ and $b_1^{(l)}$ are the learnable parameters at the lth layer.

After adding an adaptive self-loop for each node of the three graphs, we apply GCN on these graphs in parallel and exchange information among them at each GCN layer. Specifically, at the lth GCN layer, we first update the node representation of each graph G_* $(*\in \{t, c, f\})$ as follows:

$$\mathbf{Z}_*^{(l)} = ReLU(\tilde{\mathbf{A}}_*^{(l)} \tilde{\mathbf{Z}}_*^{(l-1)} \mathbf{W}_*^{(l)}), \tag{7}$$

where $\tilde{\mathbf{Z}}_*^{(l-1)}$ is the node representation of the graph G_* at the $(l-1)$th layer, $\mathbf{W}_*^{(l)}$ is the weight matrix of the lth layer, $ReLU$ is the activation function. Note that we set $\mathbf{Z}_t^{(0)} = \mathbf{Z}_c^{(0)} = \mathbf{Z}_f^{(0)} = \mathbf{X}$.

In order to propagate information learnt from three different channels (i.e, topology channel, feature channel and consensus channel), we leverage the consensus graph as a bridge which serves for updating the node representation of topology graph and feature graph. After that, we update the node representation of consensus graph based on the updated node representation of both topology graph and feature graph. Formally, we update the node representation of each graph as follows:

$$\tilde{\mathbf{Z}}_t^{(l)} = (1 - \varepsilon_1)\mathbf{Z}_t^{(l)} + \varepsilon_1 \mathbf{Z}_c^{(l)}, \tag{8}$$

$$\tilde{\mathbf{Z}}_f^{(l)} = (1 - \varepsilon_2)\mathbf{Z}_f^{(l)} + \varepsilon_2 \mathbf{Z}_c^{(l)}, \tag{9}$$

$$\tilde{\mathbf{Z}}_c^{(l)} = \frac{(\tilde{\mathbf{Z}}_t^{(l)} + \tilde{\mathbf{Z}}_f^{(l)})}{2}, \tag{10}$$

where ε_1 (ε_2) is a hyper-parameter used to balance the contribution of information propagation from the topology graph (feature graph) and the consensus graph. At last, we consider the output of the last layer of GCN in each channel, i.e., $\mathbf{Z}_t^{(N)}$, $\mathbf{Z}_c^{(N)}$, $\mathbf{Z}_f^{(N)}$, as the final node representations $\mathbf{Z}_t$, $\mathbf{Z}_c$ and $\mathbf{Z}_f$, respectively.

3.4. Attention mechanism

After we have $\mathbf{Z}_t$, $\mathbf{Z}_c$ and $\mathbf{Z}_f$, we fuse them with the attention mechanism. Specifically, for the node v_i, its embedding in $\mathbf{Z}_t$ is represented by $\mathbf{z}_t^i \in \mathbb{R}^{1\times h}$. First, we get the attention value w_t^i of the node v_i in $\mathbf{Z}_t$:

$$w_t^i = \mathbf{v}^T \cdot tanh(\mathbf{W}_2 \cdot \mathbf{z}_t^{iT} + \mathbf{b}_2), \tag{11}$$

where $\mathbf{v} \in \mathbb{R}^{m\times 1}$ is a shared attention vector, $\mathbf{W}_2 \in \mathbb{R}^{m\times h}$ is the weight matrix, and $\mathbf{b}_2 \in \mathbb{R}^{m\times 1}$ is a bias vector. Similarly, the attention values of v_i in $\mathbf{Z}_c$ and $\mathbf{Z}_f$ are w_c^i and w_f^i, respectively. Then, the $softmax$ function is used to normalize the three attention values, and the final weight of node v_i to $\mathbf{Z}_t$ is:

$$\alpha_t^i = \frac{exp(w_t^i)}{exp(w_t^i) + exp(w_c^i) + exp(w_f^i)}. \tag{12}$$

Similarly, we obtain the weights α_c^i and α_f^i. The final node embedding for node v_i is:

$$\mathbf{z}_i = \alpha_t^i \mathbf{z}_t^i + \alpha_c^i \mathbf{z}_c^i + \alpha_f^i \mathbf{z}_f^i. \tag{13}$$

At last, the final embeddings of all n nodes is $\mathbf{Z} = (\mathbf{z}_1; \cdots; \mathbf{z}_n) \in \mathbb{R}^{n\times h}$.

3.5. Loss function

After obtaining the final node embedding $\mathbf{Z}$, we feed it into a fully-connected layer and a softmax layer to produce the prediction of n nodes $\hat{\mathbf{Y}} = (\hat{\mathbf{y}_1}; \cdots; \hat{\mathbf{y}_n}) \in \mathbb{R}^{n\times C}$ as follows:

$$\hat{\mathbf{y}}_i = softmax(\mathbf{z}_i \cdot \mathbf{W}_3 + \mathbf{b}_3), \tag{14}$$

where $\mathbf{W}_3 \in \mathbb{R}^{h\times C}$ and $\mathbf{b}_3 \in \mathbb{R}^{1\times C}$ are trainable weights and bias. Then the cross entropy loss $\mathcal{L}_{class}$ for node classification over all training nodes is calculated based on the predicted labels $\hat{\mathbf{Y}}$ and the ground-truth label matrix $\mathbf{Y}$:

$$\mathcal{L}_{class} = -\sum_{i\in\mathbf{Y}_L} \sum_{c=1}^{C} \mathbf{Y}_{ic} \ln \hat{\mathbf{Y}}_{ic}, \tag{15}$$

where $\mathbf{Y}_L$ is the labeled training set, C is the number of classes, $\mathbf{Y}_{ic}$ and $\hat{\mathbf{Y}}_{ic}$ denote the real and predicted probability of the ith node to the cth class, respectively.

3.6. Structural consistency constraint

For the node representations $\mathbf{Z}_t, \mathbf{Z}_c$ and $\mathbf{Z}_f$ outputted by three different channels (i.e, topology channel, feature channel and consensus channel), despite we leverage the consensus graph as a bridge to propagate information in different channels, we further maintain the consistency of the three node representations explicitly with a structural consistency constraint. To the end, we normalize the node representation using L_2−normalization and obtain $\hat{\mathbf{Z}}_t, \hat{\mathbf{Z}}_c$ and $\hat{\mathbf{Z}}_f$. As the types of node representations are from different spaces, directly maintaining a consistency constraint over them would lead to sub-optimal performance. To deal with this issue, we propose to keep the consistency between different spaces at the structural relationship level. Specifically, we represent the structural relationships by leveraging the similarity matrices $\mathbf{S}_t, \mathbf{S}_c$ and $\mathbf{S}_f$ of n nodes, which are obtained according to $\hat{\mathbf{Z}}_t, \hat{\mathbf{Z}}_c$ and $\hat{\mathbf{Z}}_f$ respectively:

$$\mathbf{S}_* = \hat{\mathbf{Z}}_* \cdot \hat{\mathbf{Z}}_*^T. \tag{16}$$

The assumption of this constraint is that if node v_i and node v_j are close in the topology space (feature space), they should also be close in the

Table 1
The statistics of the datasets.

	Nodes	Edges	Classes	Features	Training	Validation	Test
ACM	3025	13 128	3	1870	60/120/180	500	1000
BlogCatalog	5196	171 743	6	8189	120/240/360	500	1000
CiteSeer	3327	4732	6	3703	120/240/360	500	1000
Flickr	7575	239 738	9	12 047	180/360/540	500	1000
UAI2010	3067	28 311	19	4973	380/760/1140	500	1000

consensus space. Formally, we define the structural consistency loss as follows:

$$\mathcal{L}_{stru} = \frac{(\|\mathbf{S}_t - \mathbf{S}_c\|_2^2 + \|\mathbf{S}_f - \mathbf{S}_c\|_2^2)}{2}. \tag{17}$$

3.7. Reconstruction consistency constraint

The reconstruction consistency constraint is proposed to keep a consistency between the final node representation $\mathbf{Z}$ and the original node feature representation $\mathbf{X}$. Similar to the structural consistency constraint, we first map both final node representation $\mathbf{Z}$ and node feature representation $\mathbf{X}$ into their corresponding similarity matrices $\mathbf{S}_x$ and $\mathbf{S}_z$, which are defined as follows:

$$\mathbf{S}_x = \hat{\mathbf{X}} \cdot \hat{\mathbf{X}}^T, \tag{18}$$

$$\mathbf{S}_z = \hat{\mathbf{Z}} \cdot \hat{\mathbf{Z}}^T, \tag{19}$$

where $\hat{\mathbf{X}}$ and $\hat{\mathbf{Z}}$ are the corresponding normalized node representation of $\mathbf{X}$ and $\mathbf{Z}$ with L_2-normalization, respectively. Then the reconstruction consistency loss is defined as follows:

$$\mathcal{L}_{rec} = \|\mathbf{S}_z - \mathbf{S}_x\|_2^2. \tag{20}$$

3.8. Learning objectives

Finally, we combine the cross entropy loss and the two consistency losses to obtain the overall loss function:

$$\mathcal{L} = \mathcal{L}_{class} + \theta_1 \mathcal{L}_{stru} + \theta_2 \mathcal{L}_{rec}, \tag{21}$$

where θ_1 and θ_2 are hyper-parameters that control the contribution of consistency constraints. Under the guidance of labeled data from the training set, the model is trained through back propagation to learn node representations for classification.

4. Experiments

In this section, we first give the experimental setup, then compare the proposed approach CNIM-GCN with some state-of-the-art baseline methods. After that, we conduct ablation study and analyze the importance of different channels. We also visualize the learnt node representations and analyze several important parameters of our proposed model. At last, we investigate the convergence speed of the proposed method.

4.1. Experimental setup

Datasets. Our proposed CNIM-GCN is evaluated on five real world datasets which are summarized in Table 1.

- **ACM** (Wang et al., 2019): The nodes in this dataset are papers published in some conferences, and there is an edge between two nodes if they have the same author. The features are the bag-of-words representations of paper keywords.
- **BlogCatalog** (Meng et al., 2019): This dataset is extracted from the BlogCatalog website, which consists of social relationships of bloggers. It contains 5196 user nodes, and 171743 edges representing user interactions. Each node's attributes are constructed by keywords generated by users as a short description of their blogs. The node labels represent the topic categories in which users register their blogs.
- **CiteSeer** (Kipf & Welling, 2017): CiteSeer is a network of citations where nodes are papers and edges are citation links. Node attributes are constructed based on the bag-of-words representations of the papers, and all nodes are divided into six categories (i.e., Agents, AI, DB, IR, ML, and HCI).
- **Flickr** (Meng et al., 2019): Flickr is a social network where users can share images and videos. Nodes represent users and edges represent their relationships. All the nodes are divided into 9 classes according to interest groups of users.
- **UAI2010** (Wang, Liu, Jiao, Chen, & Jin, 2018): This dataset is extracted from Wikipedia and includes 3067 nodes and 28311 edges. Nodes are documents and edges are links. Node attributes are represented based on the bag-of-words representations of the documents, and all nodes are divided into 19 distinct categories.

Baselines. We compare our proposed CNIM-GCN with eleven state-of-the-art baseline methods:

- Deepwalk (Perozzi, Al-Rfou, & Skiena, 2014): Deepwalk is a network embedding method, which utilizes structural regularities within short random walks and learns local information from truncated random walks to obtain network representations.
- LINE (Tang et al., 2015): LINE is a network embedding model which can handle large-scale networks with millions of vertices and billions of edges. It attempts to preserve both the first-order and second-order proximities, and propose an effective and efficient edge-sampling method for model inference.
- Chebyshev (Defferrard, Bresson, & Vandergheynst, 2016): It proposes to build a spectral graph theoretical formulation of CNNs on graphs on established tools in graph signal processing (GSP) (Shuman, Narang, Frossard, Ortega, & Vandergheynst, 2013). The computational complexity of the proposed model is linear with the dimensionality of the data.
- GCN (Kipf & Welling, 2017): This method introduces a simple layer-wise propagation rule for neural network models based on a first-order approximation of spectral convolutions on graphs. Similar to Chebyshev (Defferrard et al., 2016), it scales linearly in the number of graph nodes.
- kNN-GCN (Wang et al., 2020): kNN-GCN uses the sparse k-nearest neighbor graph (i.e., feature graph) derived from the node features instead of the traditional topology graph as the input graph of GCN.
- GAT (Veličković et al., 2018): GAT aggregates information from each node's neighbors, and obtain its hidden representation with a self-attention strategy. It assigns different weights to different neighboring nodes in the process of graph convolution.
- DEMO-Net (Wu, He, & Xu, 2019): It is a generic degree-specific graph neural network, which applies the same graph convolution for nodes with the same degree value. A degree-specific multi-task graph convolution function is proposed to learn the node representation.
- MixHop (Abu-El-Haija et al., 2019): MixHop proposes to repeatedly mix neighbors' feature representations at various distances in order to capture neighborhood relationships. It introduces a graph convolutional layer to learn neighborhood information without additional memory or computational complexity.

Table 2
Comparison results with baselines on all datasets in terms of Acc and F1 (%). "L/C" denotes the number of labeled nodes for training per class, and "OOM" denotes "out of memory". The best results on each dataset are in bold, and the second-best ones are underlined.

Datasets	Metrics	L/C	DeepWalk	LINE	Chebyshev	GCN	kNN-GCN	GAT	DEMO-Net	MixHop	Tail-GCN	AM-GCN	SimP-GCN	CNIM-GCN
ACM	Acc	20	62.69	41.28	75.24	87.80	78.52	87.36	84.48	81.08	88.27	90.40	88.00	**91.00**
		40	63.00	45.83	81.64	89.06	81.66	88.60	85.70	82.34	89.37	90.76	89.54	**91.56**
		60	67.03	50.41	85.43	90.54	82.00	90.40	86.55	83.09	91.00	91.42	90.88	**92.20**
	F1	20	62.11	40.12	74.86	87.82	78.14	87.44	84.16	81.40	88.23	90.43	87.99	**90.93**
		40	61.88	45.79	81.26	89.00	81.53	88.55	84.83	81.13	89.27	90.66	89.51	**91.56**
		60	66.99	49.92	85.26	90.49	81.95	90.39	84.05	82.24	90.94	91.36	90.84	**92.11**
BlogCatalog	Acc	20	38.67	58.75	38.08	69.84	75.49	64.08	54.19	65.46	75.83	81.98	85.40	**89.52**
		40	50.80	61.12	56.28	71.28	80.84	67.40	63.47	71.66	76.63	84.94	86.82	**92.32**
		60	55.02	64.53	70.06	72.66	82.46	69.95	76.81	77.44	78.43	87.30	90.44	**92.74**
	F1	20	34.96	57.75	33.39	68.73	72.53	63.38	52.79	64.89	75.00	81.36	84.83	**89.21**
		40	48.61	60.72	53.86	70.71	80.16	66.39	63.09	70.84	75.93	84.32	86.24	**92.12**
		60	53.56	63.81	68.37	71.80	81.90	69.08	76.73	76.38	78.00	86.94	90.05	**92.50**
CiteSeer	Acc	20	43.47	32.71	69.80	70.30	61.35	72.50	69.50	71.40	71.03	**73.10**	71.74	71.72
		40	45.15	33.32	71.64	73.10	61.54	73.04	70.44	71.48	74.27	**74.70**	72.88	74.48
		60	48.86	35.39	73.26	74.48	62.38	74.76	71.86	72.16	74.93	**75.56**	72.96	75.50
	F1	20	38.09	31.75	65.92	67.50	58.86	68.14	67.84	66.96	66.80	68.42	**68.57**	67.26
		40	43.18	32.42	68.31	69.70	59.33	69.58	66.97	67.40	69.32	**69.81**	69.03	69.45
		60	48.01	34.37	70.31	71.24	60.07	**71.60**	68.22	69.31	70.79	70.92	70.01	70.41
Flickr	Acc	20	24.33	33.25	23.26	41.42	69.28	38.52	34.89	39.56	OOM	75.26	77.80	**81.98**
		40	28.79	37.67	35.10	45.48	75.08	38.44	46.57	55.19	OOM	80.06	82.88	**84.78**
		60	30.10	38.54	41.70	47.96	77.94	38.96	57.30	64.96	OOM	82.10	84.46	**85.60**
	F1	20	21.33	31.19	21.27	39.95	70.33	37.00	33.53	40.13	OOM	74.63	78.21	**81.52**
		40	26.90	37.12	33.53	43.27	75.40	36.94	45.23	56.25	OOM	79.36	83.17	**84.62**
		60	27.28	37.77	40.17	46.58	77.97	37.35	56.49	65.73	OOM	81.81	84.48	**85.28**
UAI2010	Acc	20	42.02	43.47	50.02	49.88	66.06	56.92	23.45	61.56	62.70	70.10	57.44	**73.06**
		40	51.26	45.37	58.18	51.80	68.74	63.74	30.29	65.05	65.47	73.14	65.88	**75.10**
		60	54.37	51.05	59.82	54.40	71.64	68.44	34.11	67.66	67.43	74.40	70.04	**77.60**
	F1	20	32.93	37.01	33.65	32.86	52.43	39.61	16.82	49.19	48.60	55.61	46.13	**60.80**
		40	46.01	39.62	38.80	33.80	54.45	45.08	26.36	53.86	50.15	64.88	54.99	**65.68**
		60	44.43	43.76	40.60	34.12	54.78	48.97	29.05	56.31	51.43	65.99	58.99	**69.74**

- Tail-GCN (Liu, Nguyen, & Fang, 2021a): Tail-GCN is developed for learning robust tail node embeddings. It introduces a novel concept of transferable neighborhood translation to capture the relational tie between a node and its neighboring nodes, and bridges the gap between head and tail nodes.
- AM-GCN (Wang et al., 2020): AM-GCN aims to extract two specific embeddings in both feature space and topology space by propagating node features over both feature graph and topology graph. To maintain the common embedding shared by the two spaces, it develops a common convolution module with parameter sharing.
- SimP-GCN (Jin et al., 2021): SimP-GCN aims to preserve the original node similarity. It proposes an adaptive strategy that coherently integrates the graph structure and node features, and explicitly encodes the pairwise feature relation to preserve feature and structural similarity.

Parameter Settings. We choose three label rates (i.e., 20, 40, 60 labeled nodes per class) as the training set, and select 500 nodes and 1000 nodes as the development set and test set respectively. Three two-layer networks with the same hidden layer dimension ($nhid1$) and the same output dimension ($nhid2$) are trained simultaneously, where $nhid1 \in \{512, 768\}$, $nhid2 \in \{128, 256\}$. The dropout rate is set to 0.5, and weight decay $\in \{5e^{-5}, 5e^{-4}, 5e^{-3}\}$. We utilize Adam as the optimizer with a learning rate of $1e^{-4} \sim 1e^{-3}$. For the k-nearest neighbor graph (i.e., feature graph), we take $k \in \{2, \ldots, 20\}$. The adaptive self-loop coefficient (i.e., γ) and the information propagation balance coefficients (i.e., ε_1 and ε_2) are searched in $\{0, 0.1, 0.2, \ldots, 1\}$. The consistency constraint coefficients (i.e., θ_1 and θ_2) are searched in $\{0, 0.001, 0.01, 0.1, 1\}$. To evaluate the performance of the proposed method, we employ Accuracy (Acc) and macro F1-score (F1) in this work.

4.2. Node classification results

We compare the performance of our proposed CNIM-GCN with eleven baselines in Table 2. The results demonstrate that CNIM-GCN has the state-of-the-art performance as compared to other competitive baselines in terms of both metrics (i.e., Acc and F1), which ascertains the superiority of our proposed method. The main observations are as follows:

- Compared with baselines which rely on either the topology graph or the feature graph, the methods (i.e., AM-GCN, SimP-GCN and CNIM-GCN) fusing both graphs achieve a superior performance. Among them, our proposed method CNIM-GCN mostly outperforms the two best performing baselines, i.e., AM-GCN and SimP-GCN, by a large margin. This is mainly attributed to that CNIM-GCN maintains the common information of the topology graph and the feature graph by explicitly modeling the consensus graph.
- We can also observe that with the increase of the number of labeled nodes for training, the performance of CNIM-GCN raises correspondingly and it is consistently superior to the two best performing baselines AM-GCN and SimP-GCN on all datasets except the dataset CiteSeer. For example, on the BlogCatalog dataset, CNIM-GCN achieves F1 scores of 89.21%, 92.12%, 92.5% with label rate 20, 40 and 60, respectively, which are 9.65% (5.16%), 9.25% (6.82%) and 6.40% (2.72%) better than the corresponding performance of AM-GCN (SimP-GCN). In addition, when less labeled nodes are available for training, the performance improvement of CNIM-GCN over both AM-GCN and SimP-GCN becomes larger. This reveals that our proposed method can still effectively model the information of both topology space and feature space when the task become more challenge.

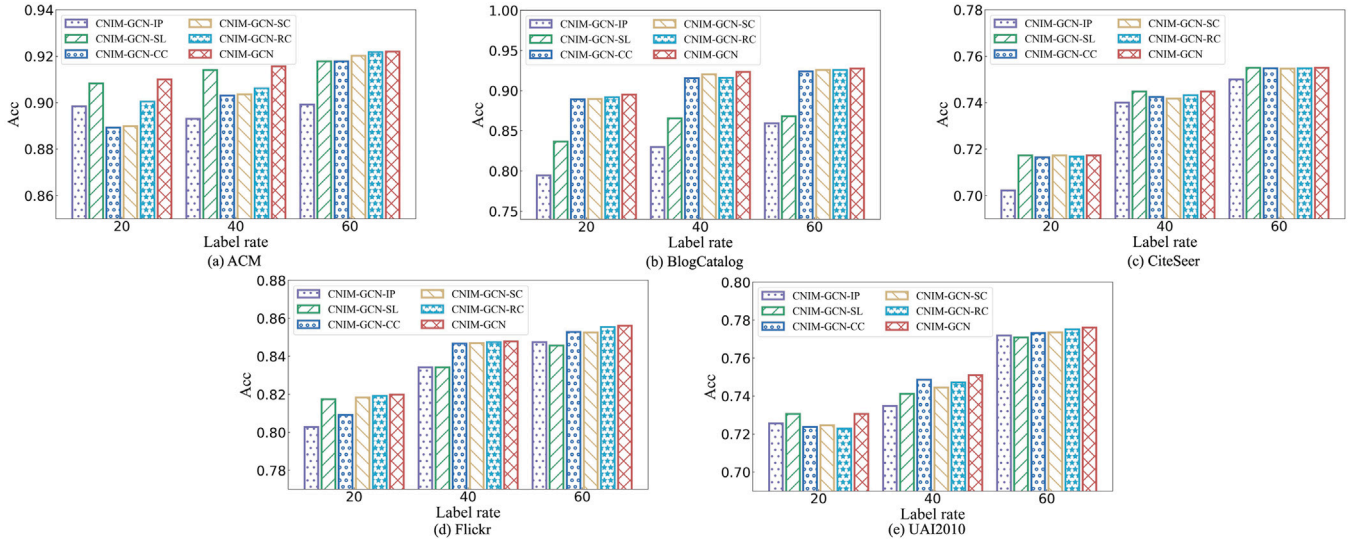


Fig. 2. The ablation study of CNIM-GCN on all datasets.

- The performance improvements of CNIM-GCN over both AM-GCN and SimP-GCN on the three datasets, i.e., BlogCatalog, Flickr, UAI2010, are considerably larger than that on the remaining two datasets, i.e., ACM and CiteSeer. For example, the improvements of F1 score of CNIM-GCN over AM-GCN and SimP-GCN on BlogCatalog, Flickr, UAI2010 are 9.65% (5.16%), 9.23% (4.23%), 9.33% (31.80%) respectively when the label rate is 20. However, the improvements of F1 score of CNIM-GCN over AM-GCN on ACM and CiteSeer are relatively small, e.g., 0.55% (3.34%) and −1.70% (−1.91%), when the label rate is 20. The main reason is that the corresponding feature graphs of the BlogCatalog, Flickr, UAI2010 are more informative than the corresponding topology graph, e.g., the performance of the baseline kNN-GCN is better than that of the baseline GCN. In contrast, on the datasets ACM and CiteSeer, the feature graphs are less informative, which leads to inferior performance improvement of our method.

4.3. Ablation study

In this section, we study the contributions of five main components, including the information propagation, the adaptive self-loop, the consistency constraint, the structural consistency constraint, and the reconstruction consistency constraint to our proposed method.

- CNIM-GCN-IP: We remove the information propagation component to investigate the influence of leveraging the consensus graph, which is used as a bridge serving for updating the node representation of topology graph and the feature graph.
- CNIM-GCN-SL: We discard the adaptive self-loop component in order to study how important that the features of nodes themselves to the performance of the classification.
- CNIM-GCN-CC: We remove the consistency constraints, including the structural consistency constraint as well as the reconstruction consistency constraint.
- CNIM-GCN-SC: We remove the structural consistency constraint which maintains the consistency of the node representations $\mathbf{Z}_t, \mathbf{Z}_c$ and $\mathbf{Z}_f$ generated by the three different channels.
- CNIM-GCN-RC: We remove the reconstruction consistency constraint which is utilized to keep a consistency between the final node representation $\mathbf{Z}$ and the original node feature representation $\mathbf{X}$.

As shown in Fig. 2, we can observe that each component plays a critical role in improving the performance of CNIM-GCN. On ACM dataset, removing the information propagation component (i.e., CNIM-GCN-IP) causes a significant drop of Acc scores, which are 1.29%, 2.53%, and 2.54% worse than that of CNIM-GCN with respect to the label rate 20, 40 and 60, respectively. Similar results are observed on other datasets. This reveals the effectiveness of incorporating the consensus graph as a bridge for exchanging information between the topology graph and the feature graph.

Besides, removing the adaptive self-loop component (i.e., CNIM-GCN-SL) will also lead to a decrease of the performance of CNIM-GCN. Specifically, on the datasets with rich feature information (i.e., BlogCatalog, Flickr, and UAI2010), CNIM-GCN-SL will lead a considerable performance decrease of CNIM-GCN. However, on the datasets where the feature information is less informative (i.e., ACM and CiteSeer), the performance of CNIM-GCN does not decrease significantly.

At last, the influence of removing the consistency constraints (i.e., CNIM-GCN-CC) to CNIM-GCN varies according to the datasets. For example, it results in a considerable decrease of the model performance on the dataset ACM with all three label rates. While on the dataset CiteSeer, the performance decrease is not significant. This may be because the consistency of node representations from different channels has already been well maintained by leveraging the consensus graph as a bridge to propagate information in different channels. Moreover, removing the structural consistency constraint (i.e., CNIM-GCN-SC) affects more the performance of our proposed method CNIM-GCN and mostly leads to a large performance drop as compared with removing the reconstruction consistency constraint (i.e., CNIM-GCN-RC).

4.4. Importance analysis of different channels

In this section, we investigate the contribution of node representations learned from different channels, i.e., topology channel, consensus channel, and feature channel, to the model performance. Fig. 3 shows the attention weights assigned to different channels on all five datasets with a label rate of 60.

As we can observe from Fig. 3, on the datasets ACM and CiteSeer, the proposed model assigns the highest attention to the topology channel, which reveals the important role of the topology channel for node classification on these datasets. In contrast, on the datasets BlogCatalog, Flickr, and UAI2010, the feature channel obtains higher attention weights as compared to the other two channels. This is because the feature representation of three datasets (i.e., BlogCatalog, Flickr, and UAI2010) are more informative as compared to the other two datasets (i.e., ACM and CiteSeer), which is also consistent to the results reported in Section 4.2.

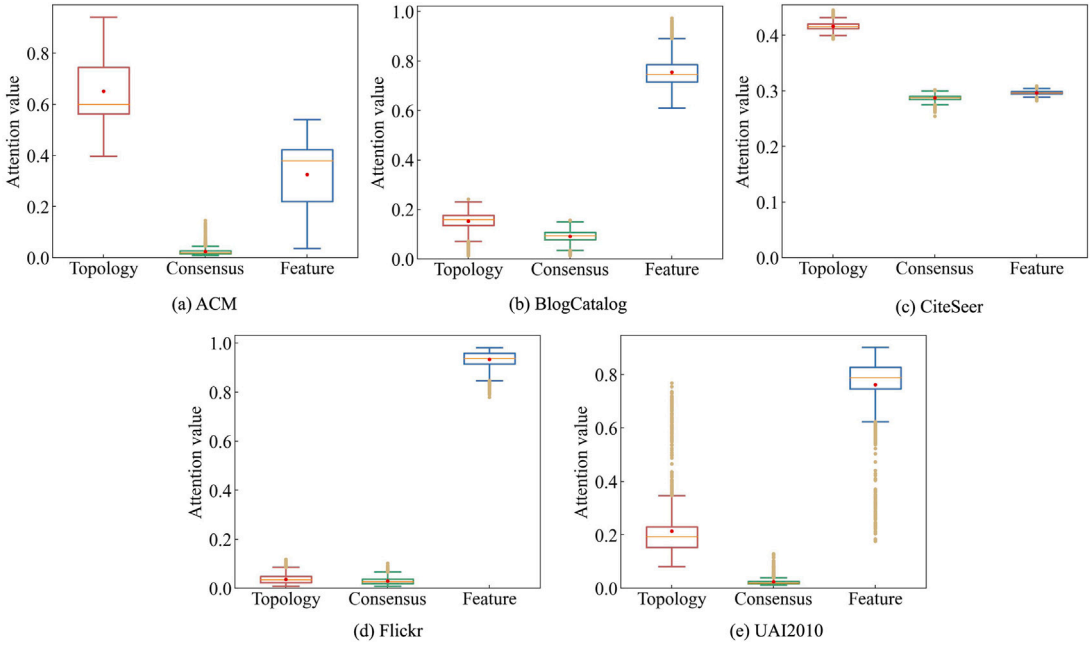


Fig. 3. Analysis of attention weights assigned to different channels on all datasets with a label rate of 60.

In addition, the attention weight of the consensus channel is smaller than both topology graph and feature graph in all cases. This is attributed to that the consensus information can be transmitted to topology graph and feature graph via the information propagation component in order to enhance their corresponding node representations. The experimental results verify that our proposed approach can adaptively assign reasonable attention weights to different channels according to their capability.

4.5. Visualization

To show the effectiveness of our proposed method in a more effective way, we further conduct a visualization on the BlogCatalog dataset with label rates of 20, 40, and 60, respectively. We use the node representations of the last layer of CNIM-GCN (GCN, AM-GCN and SimP-GCN) before the softmax and plot the representations using t-SNE (Van der Maaten & Hinton, 2008). The results are shown in Fig. 4, in which nodes are colored based on the ground-truth labels.

We can find from Fig. 4 that, compared with GCN, node embedding learned by both AM-GCN and SimP-GCN can clearly distinguish different labels. However, the results of AM-GCN and SimP-GCN are not satisfactory as many nodes with different labels are mixed together. The learned node representations of CNIM-GCN is superior to other methods because they are more compact and have better intra-class tightness as well as clear boundary of each class.

In addition, we can observe a better performance of CNIM-GCN with the increase of the label rate. For example, the model performance of CNIM-GCN with a label rate of 60 is considerably better than that of its counterparts with a label rate of 20 and 40. This indicates that CNIM-GCN can learn more compact node embedding with high intra-class similarity and clear inter-class boundaries when more labeled nodes are available, which also corresponds to the results of node classification in Section 4.2.

4.6. Parameter study

In this section, we will analyze the impact of hyperparameters of our CNIM-GCN, i.e., k, γ, ε_1, ε_2, θ_1 and θ_2. We only report results in terms of accuracy as similar findings are observed in terms of F1 score.

Analysis of Parameter k. The parameter k indicates the number of neighbors employed for building the kNN graph (i.e., feature graph). We vary k from 2 to 20, and the performance of CNIM-GCN is shown in Fig. 5. On the ACM dataset when the label rate is 60, we can see that the accuracy keeps to raise and reaches the peak when $k = 9$. If we continue to increase k, the accuracy will decrease gradually. Similar results can be observed for other two label rates. The reason is that when we increase the number of neighbors for constructing the feature graph, more useful semantic structure information can be explored via the feature channel. However, if k becomes too large, more noise neighboring nodes would be introduced, and lead to a degradation of classification performance. On the Flickr dataset, it shows a similar trend as on the ACM dataset except that the best performance is reached at a larger k, e.g., $k = 14$ when the label rate is 60.

Analysis of Parameter γ. The parameter γ reflects the impact of introducing the adaptive self-loops which encourages the model to retain more information about nodes' original features. As shown in Fig. 6, with the increase value of γ, the accuracy on the ACM dataset with the label rate of 60 first increases and obtains the best performance when γ is around 0.4, which follows by a performance degradation when we keep increasing γ. When the label rate is small, the model performance will reach a peak with a lower γ value, e.g., the best γ for label rate of 40 is 0.1. Besides, the model performance also drops faster with a small label rate. Similar performance changing trends are observed on the Flickr dataset.

Analysis of Parameter ε_1 and ε_2. The parameter ε_1 (ε_2) are introduced to balance the contribution of information propagation from the topology (feature) channel and the consensus graph. A higher ε_1 (ε_2) indicates the consensus graph serves a more important role for updating the node representation of the topology (feature) graph. We vary both ε_1 and ε_2 from 0 to 1 with a step size of 0.1, and the results are demonstrated in Fig. 7. On the ACM dataset, we can observe that a relative larger ε_1 and ε_2 result in better performance. The best performance will be reached when $\varepsilon_1 = 0.9$ and $\varepsilon_2 = 0.8$, which demonstrates the effect of incorporating the consensus graph as a bridge channel for updating both the topology and feature channel. While on the Flickr dataset, the model performance is less sensitive to ε_1, and mainly affected by ε_2. The performance keeps to increase when we increase ε_2 and reaches the peak when $\varepsilon_2 = 0.6$. If we continue raise ε_2, the performance will drop quickly. The main reason is that the

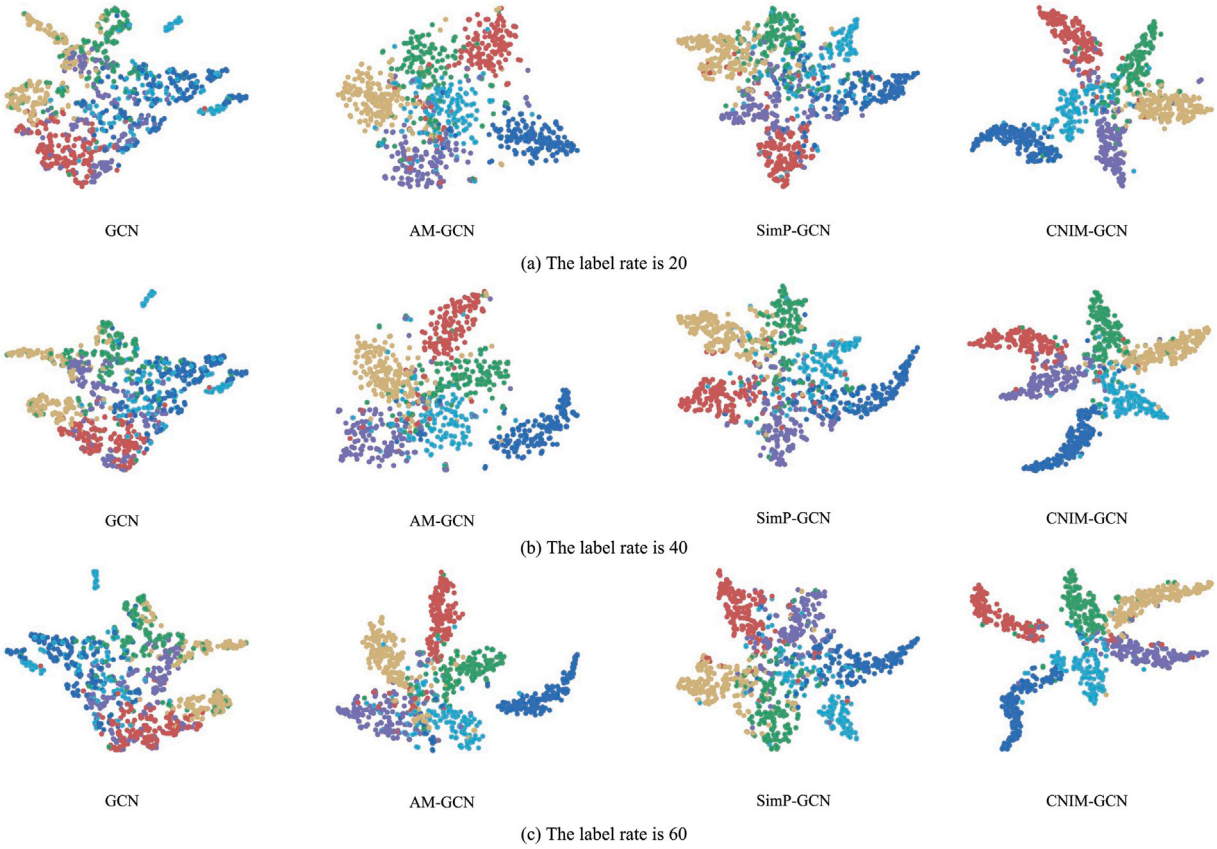


Fig. 4. Visualization of the learned node embeddings on BlogCatalog dataset.

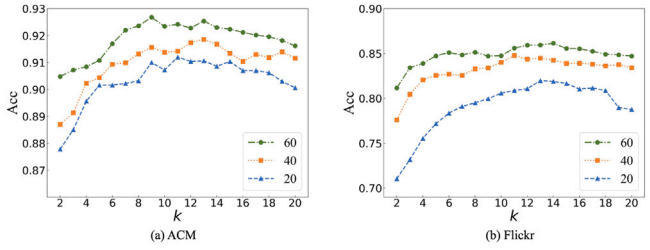


Fig. 5. Analysis of parameter k.

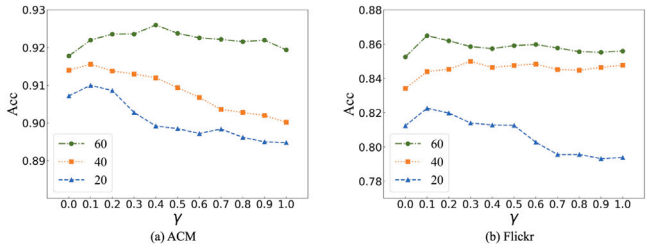


Fig. 6. Analysis of parameter γ.

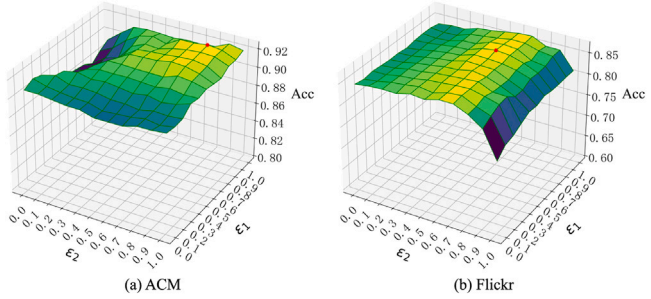


Fig. 7. Analysis of parameter ϵ_1 and ϵ_2.

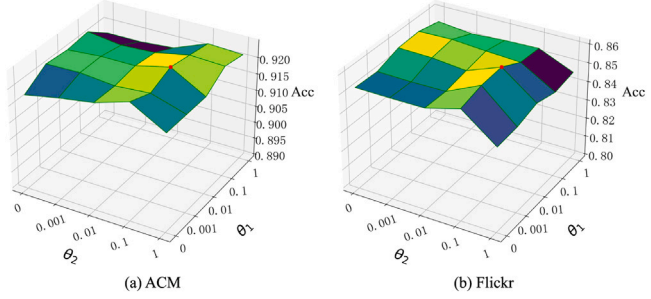


Fig. 8. Analysis of consistency coefficient θ_1 and θ_2.

feature channel on the Flickr dataset is more informative as compared to the topology channel (see Section 4.4).

Analysis of Parameter θ_1 and θ_2. The parameters θ_1 and θ_2 reflect the effects of the structural consistency constraint and the reconstruction consistency constraint to our proposed model, respectively. We

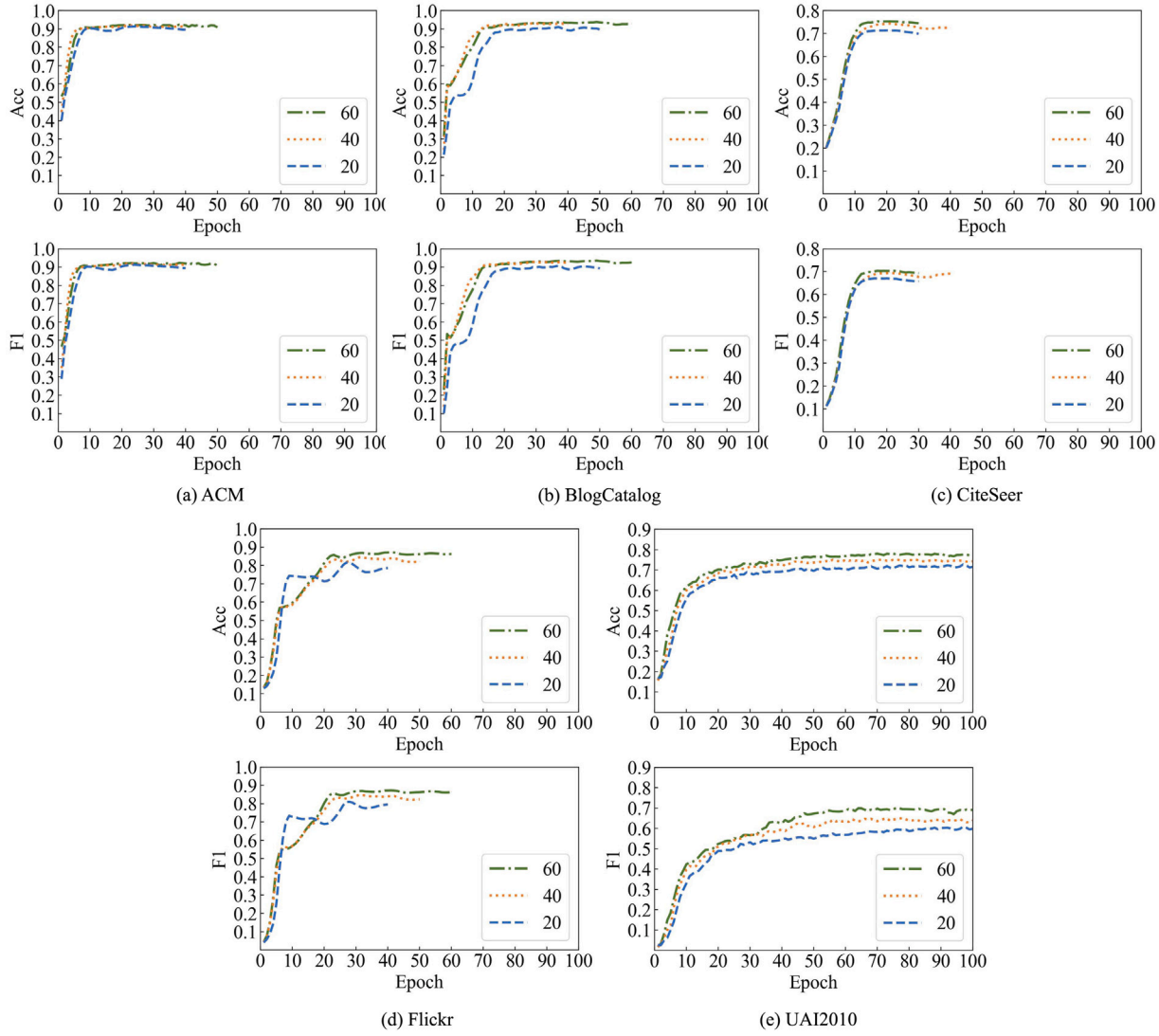


Fig. 9. Learning curves of the proposed model CNIM-GCN.

vary both θ_1 and θ_2 as {0, 0.001, 0.01, 0.1, 1}. The results on ACM are shown in Fig. 8(a), and we observe an increase of model performance when paying more attention to the structural consistency (i.e., a larger value of θ_1). The performance also increases when we raise the value of θ_2, and drops considerably when θ_2 becomes too large. The best model performance is obtained when $\theta_1 = 0.01$ and $\theta_2 = 0.1$. In Fig. 8(b), we can see similar trends of model performance on the Flickr dataset.

4.7. Learning curve

To analyze the converging speed of our proposed model, we further run the model on all datasets with three different label rates (i.e., 20, 40 and 60) and report the accuracy and F1 score for each epoch. The results are shown in Fig. 9, and we can see that our proposed model converges fast, especially on the datasets ACM and CiteSeer, where CNIM-GCN achieves the best performance with less than 10 epochs. While on the other three datasets (i.e., BlogCatalog, Flickr, and UAI2010), CNIM-GCN requires relatively more epochs to obtain the best performance, e.g., on the UAI2010 dataset, it needs around 50 to 90 epochs with respect to the label rate 20, 40, 60, respectively to achieve the best performance. This indicates that our proposed model may converge faster on datasets with less informative node features.

5. Conclusion

In this paper, we propose a new approach named Consensus Neighbor Interaction-based Multi-channel Graph Convolution Networks (CNIM-GCN) for node classification. Specifically, we introduce a consensus graph to preserve the common information between the topology graph and the feature graph in an explicit way, and develop a multi-channel graph convolutional networks for effectively fusing information from different graphs. Moreover, two types of consistency constraints, i.e., structural consistency constraint and reconstruction consistency constraint, are incorporated to maintain the consistency between different channels. Extensive experiments on five real-world datasets show that our proposed method obtains substantially better performance than state-of-the-art baselines. In addition, the model analysis suggests that each introduced component is helpful in learning better node representations.

CRediT authorship contribution statement

Xiaofei Zhu: Conceptualization, Methodology, Writing – original draft, Writing – review & editing, Supervision. **Chenghong Li:** Methodology, Software, Writing – original draft. **Jiafeng Guo:** Writing – review

& editing, Supervision. **Stefan Dietze:** Writing – review & editing, Supervision.

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 supported by the National Natural Science Foundation of China [grant number 62141201]; the Major Project of Science and Technology Research Program of Chongqing Education Commission of China [grant number KJZD-M202201102]; the Natural Science Foundation of Chongqing, China [grant number CSTB2022NSCQ-MSX1672]; the Federal Ministry of Education and Research, Germany [grant number 01IS21086].

References

Abu-El-Haija, S., Perozzi, B., Kapoor, A., Alipourfard, N., Lerman, K., Harutyunyan, H., et al. (2019). Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th international conference on machine learning* (pp. 21–29).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the 30th international conference on neural information processing systems* (pp. 3844–3852).

Gao, H., & Ji, S. (2019). Graph u-nets. In *Proceedings of the 36th international conference on machine learning* (pp. 2083–2092).

Hamilton, W. L., Ying, R., & Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of the 31st international conference on neural information processing systems* (pp. 1025–1035).

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017b). Inductive representation learning on large graphs. In *Proceedings of the 30th advances in neural information processing systems* (pp. 1024–1034).

Hang, M., Neville, J., & Ribeiro, B. (2021). A collective learning framework to boost GNN expressiveness for node classification. In *Proceedings of machine learning research*: *vol. 139*, *Proceedings of the 38th international conference on machine learning* (pp. 4040–4050).

Hou, Y., Zhang, J., Cheng, J., Ma, K., Ma, R. T. B., Chen, H., et al. (2020). Measuring and improving the use of graph information in graph neural networks. In *Proceedings of the 8th international conference on learning representations*.

Huang, X., Li, J., & Hu, X. (2017). Label informed attributed network embedding. In *Proceedings of the 10th ACM international conference on web search and data mining* (pp. 731–739).

Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., & Tang, J. (2021). Node similarity preserving graph convolutional networks. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 148–156).

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th international conference on learning representations*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2019). Predict then propagate: Graph neural networks meet personalized PageRank. In *Proceedings of the 7th international conference on learning representations*.

Li, Q., Han, Z., & Wu, X. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI conference on artificial intelligence* (pp. 3538–3545).

Liu, Z., Fang, Y., Liu, C., & Hoi, S. C. H. (2021). Relative and absolute location embedding for few-shot node classification on graph. In *Proceedings of the 35th AAAI conference on artificial intelligence* (pp. 4267–4275).

Liu, Z., Nguyen, T., & Fang, Y. (2021a). Tail-GNN: Tail-node graph neural networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1109–1119).

Liu, Z., Nguyen, T.-K., & Fang, Y. (2021b). Tail-GNN: Tail-node graph neural networks. In *Proceedings of the 27th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1109–1119).

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

Meng, Z., Liang, S., Bao, H., & Zhang, X. (2019). Co-embedding attributed networks. In *Proceedings of the 12th ACM international conference on web search and data mining* (pp. 393–401).

Nie, L., Jiao, F., Wang, W., Wang, Y., & Tian, Q. (2021). Conversational image search. *IEEE Transactions on Image Processing*, *30*, 7732–7743.

Nie, L., Li, Y., Feng, F., Song, X., Wang, M., & Wang, Y. (2020). Large-scale question tagging via joint question-topic embedding learning. *ACM Transactions on Information Systems*, *38*(2), 20:1–20:23.

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 701–710).

Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93–106.

Shen, X., Dai, Q., Chung, F., Lu, W., & Choi, K. (2020). Adversarial deep network embedding for cross-network node classification. In *Proceedings of the 34th AAAI conference on artificial intelligence* (pp. 2991–2999).

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., & Vandergheynst, P. (2013). The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, *30*(3), 83–98.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., & Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international world wide web conference* (pp. 1067–1077).

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). ArnetMiner: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 990–998).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *Proceedings of the 6th international conference on learning representations*.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., et al. (2019). Heterogeneous graph attention network. In *Proceedings of the 28th international world wide web conference* (pp. 2022–2032).

Wang, W., Liu, X., Jiao, P., Chen, X., & Jin, D. (2018). A unified weakly supervised framework for community detection and semantic matching. In *Proceedings of the Pacific-Asia conference on knowledge discovery and data mining* (pp. 218–230).

Wang, Z., Wang, J., Guo, Y., & Gong, Z. (2021). Zero-shot node classification with decomposed graph prototype network. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 1769–1779).

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., & Pei, J. (2020). Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1243–1253).

Wu, J., He, J., & Xu, J. (2019). DEMO-Net: Degree-specific graph neural networks for node and graph classification. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 406–415).

Yu, Z., Wang, H., Liu, Y., Böhm, C., & Shao, J. (2020). Community attention network for semi-supervised node classification. In *Proceedings of the 20th IEEE international conference on data mining* (pp. 1382–1387).

Yue, X., Wang, Z., Huang, J., Parthasarathy, S., Moosavinasab, S., Huang, Y., et al. (2020). Graph embedding on biomedical networks: methods, applications and evaluations. *Bioinformatics*, *36*(4), 1241–1251.

Zhang, X., Du, Y., Xie, R., & Wang, C. (2021). Adversarial separation network for cross-network node classification. In *Proceedings of the 30th ACM international conference on information and knowledge management* (pp. 2618–2626).