Self-Consistent Graph Neural Networks for Semi-Supervised Node Classification

Yanbei Liu, Shichuan Zhao, Xiao Wang, *Member, IEEE*, Lei Geng, Zhitao Xiao, and Jerry Chun-Wei Lin, *Senior Member, IEEE*

***Abstract*—Graph Neural Networks (GNNs), the powerful graph representation technique based on deep learning, have attracted great research interest in recent years. Although many GNNs have achieved the state-of-the-art accuracy on a set of standard benchmark datasets, they are still limited to traditional semi-supervised framework and lack of sufficient supervision information, especially for the large amount of unlabeled data. To overcome this issue, we propose a novel self-consistent graph neural networks (SCGNN) framework to enrich the supervision information from two aspects: the self-consistency of unlabeled data and the label information of labeled data. First, in order to extract the self-supervision information from the numerous unlabeled nodes, we perform graph data augmentation and leverage a self-consistent constraint to maximize the mutual information of the unlabeled nodes across different augmented graph views. The self-consistency can sufficiently utilize the intrinsic structural attributes of the graph to extract the self-supervision information from unlabeled data and improve the subsequent classification result. Second, to further extract supervision information from scarce labeled nodes, we introduce a fusion mechanism to obtain comprehensive node embeddings by fusing node representations of two positive graph views, and optimize the classification loss over labeled nodes to maximize the utilization of label information. We conduct comprehensive empirical studies on six public benchmark datasets in node classification task. In terms of accuracy, SCGNN improves by an average of 2.08% over the best baseline, and specifically by 5.8% on the Disease dataset.**

***Index Terms*—Graph neural networks, self-consistent constraint, semi-supervised learning, node classification.**

Manuscript received 30 April 2022; revised 24 March 2023; accepted 3 April 2023. Date of publication 12 April 2023; date of current version 11 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 62172052 and 61901297, in part by Program for Innovative Research Team in University of Tianjin under Grant TD13-5034 and in part by the Special Foundation for Beijing Tianjin Hebei Basic Research Cooperation under Grant 21JCZXJC00170. Recommended for acceptance by K.-K. R. Choo. *(Corresponding author: Zhitao Xiao.)*

Yanbei Liu is with the School of Life Sciences, Tiangong University, Tianjin 300387, China, and also with the Jiangsu Key Laboratory of Image and Video Understanding for Social Safety, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: liuyanbei@tiangong.edu.cn).

Lei Geng and Zhitao Xiao are with the School of Life Sciences, Tianjin Key Laboratory of Optoelectronic Detection Technology and Systems, Tiangong University, Tianjin 300387, China (e-mail: genglei@tiangong.edu.cn; xiaozhitao@tiangong.edu.cn).

Shichuan Zhao is with the School of Electronics and Information Engineering, Tiangong University, Tianjin 300387, China (e-mail: zsc2569497392@163.com).

Xiao Wang is with the School of Software, Beihang University, Beijing 100191, China (e-mail: xiaowang@bupt.edu.cn).

Jerry Chun-Wei Lin is with the Department of Computer Science, Electrical Engineering and Mathematical Sciences, Western Norway University of Applied Sciences, 5063 Bergen, Norway, and also with the Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland (e-mail: jerrylin@ieee.org).

Digital Object Identifier 10.1109/TBDATA.2023.3266590

I. Introduction

Graph is a ubiquitous data structure, which has been largely applied in many fields. For examples, in e-commence, a graph-based learning system can exploit the interactions between users and products to make highly accurate recommendations [1]. In chemistry, molecules are modeled as graphs, and their bioactivity needs to be identified for drug discovery [2]. With the popularity of graphs, it is particularly important to learn effective representations of graphs and apply them to downstream tasks. With the success of deep learning, graph neural networks (GNNs) have recently emerged as the state-of-the-art learning representations on graphs [3]. GNNs adapt the effective deep representation learning approaches from euclidean to non-euclidean domains, which broadly follow a recursive message passing scheme [4] where local neighborhood information is aggregated and passed on to the neighbors. Currently, using a small number of labeled data, researchers have proposed various semi-supervised GNN architectures with different aggregation schemes. The most typical GNNs and the variants include GCN [5], GAT [6], ML-GCN [7], AP-GCN [8], ARMA [9], LGLP [10] and ASGNN [11]. Empirically, these GNNs have achieved impressive performance in many analytical tasks, such as node classification [5], [12], link prediction [10] and graph classification [11].

Despite the enormous success of these GNNs, they are still limited to the conventionally semi-supervised framework and lack of sufficient supervision information. In general, the supervision information in the semi-supervised learning comes only from labels of labeled data. However, labeled data is usually scarce, it's difficult to effectively improve the performance of GNNs. In addition, most semi-supervised GNNs update node representations by aggregating information from neighbors and train the model to predict the category labels of unlabeled nodes. However, in these models, those large amounts of unlabeled data are not effectively utilized due to the lack of additional constraints. Some studies have shown that in semi-supervised learning, maximizing the effective utilization of unlabeled data can significantly improve learning accuracy if used properly [13]. Therefore, it is crucial to extract sufficient supervision information from the whole graph data, especially for the numerous unlabeled data.

Nevertheless, it is technically challenging to effectively acquire self-supervision information from unlabeled data in the semi-supervised GNNs. (1) It is difficult to design constraints for unlabeled data and integrate into semi-supervised GNNs. Some studies on self-supervised graph representation learning, particularly these methods based on contrastive learning [14], [15], [16], [17], have achieved promising results in extracting self-supervision information from graph data. However, these methods are not directly applicable to semi-supervised learning as they fail to incorporate the scarce yet valuable label information. (2) Graph augmentation scheme over unlabeled data is not a trivial task since graphs are more complex due to the non-euclidean property. Unlike image data, researchers can use various augmentation methods, such as cropping, color distortion, and Gaussian blurs, to generate image views. Due to the discrete nature of graph structure, constructing informative views of a graph has technical difficulties.

To accommodate these issues, in this article, we propose a novel Self-consistent Graph Neural Networks (SCGNN) framework to improve node classification performance by providing additional supervision information for semi-supervised GNNs. Our aim is to enrich supervision information from two aspects, i.e., the self-consistency of unlabeled data and the label information of labeled data. First, the node representations in the original graph should be consistent with those in the augmented graph. We present a self-consistent constraint to extract the self-supervision information from massive unlabeled nodes. By maximizing the mutual information between two graph views, the self-consistency can fully exploit the intrinsic structural attributes of the graph and maximizes the node representations consistency between graph views. Second, we introduce a fusion mechanism to obtain comprehensive node embeddings by fusing the node representations of two positive graph views. It helps to fully extract the supervision information from a few labeled nodes and effectively avoid the loss of label information. Finally, we combine the unsupervised self-consistent loss of unlabeled data and the supervised loss of labeled data into the final objective function to learn node representations. Extensive experiments on six real-world datasets clearly demonstrate the superiority of SCGNN compared with existing methods on the semi-supervised node classification. The main contributions of this article can be summarized as follows:

- We propose a novel Self-consistent Graph Neural Networks (SCGNN) framework, which can combine the self-consistency of massive unlabeled data and the label information of labeled data to learn node representations. It is a simple and efficient extension for semi-supervised GNNs.
- To extract the self-supervision information from the abundant unlabeled nodes, we present a self-consistent constraint by maximizing the mutual information between two graph views, which can effectively utilize the intrinsic structure of the graph and maximizes the node representations consistency between graph views.
- Extensive experiments on six benchmark datasets demonstrate that our model significantly outperforms state-of-the-art models on node classification task by effectively incorporating supervision information from unlabeled and labeled data.

The remainder of the paper is organized as follows. Section II reviews the related work, including graph-based semi-supervised learning and contrastive learning on graph. We present mathematical notations for the graph and brief review of graph convolutional network in Section III. Details of our proposed method are presented in Section IV. In Section V, we present experimental results and analysis. Section VI concludes this article with discussions.

II. Related Work

In this section, we briefly introduce two related topics, including graph-based semi-supervised learning and contrastive learning.

A. Graph-Based Semi-Supervised Learning

Researchers have proposed a large number of approaches for semi-supervised learning using graph representations in recent years, which can be categorized into two groups, regularized graph Laplacian based methods and graph embedding based methods [5]. The first group includes label propagation with Gaussian Fields and Harmonic Functions [18], Manifold propagation [19] and deep semi-supervised embedding [20]. The second group contains some studies inspired by the Skip-Gram [21] using various of random walk and search strategies, such as DeepWalk [22] learns embeddings via the prediction of the local neighborhood of nodes, sampled from random walks on the graph. LINE [23] and node2vec [24] extend DeepWalk with more sophisticated random walk or breadth-first search schemes. Recently, research attention has been shifted to the learning of proper network embedding to facilitate the label determination ([5], [6], [25]), where Graph Convolutional Networks (GCNs) have been demonstrated to outperform traditional graph-based models due to its impressive representation ability [26], such as GCN [5], GAT [6], AGCN [27], Cluster-GCN [28], ML-GCN [7] and G^{2}CN [29]. These models extract the supervision signals from only small amount of labeled data and then predict the category labels of unlabeled data through a message passing mechanism. Different from the above methods, the proposed SCGNN imposes a self-consistent constraint term on the unlabeled data and mines the self-supervision information of the unlabeled data by maximizing the mutual information between different augmented views.

B. Contrastive Learning on Graphs

Graph contrastive learning [30] is based on the InfoMax principle [31] and is a class of self-supervised approaches. It trains an encoder to contrastively discriminate between representations that depict statistical dependencies of interest and those that do not [16], that is, to learn discriminative representations by contrasting positive and negative samples. Contrastive methods [32], [33] have been shown to be effective for unsupervised learning in computer vision, learning self-supervised representations of images via minimizing the distance between two views of the same image. These methods are also applicable to graphs, and have become the current state-of-the-art in unsupervised classification of nodes and graphs. For instance, inspired by the local-global mutual information maximization viewpoints [32], Deep Graph Infomax (DGI) [14] extends deep InfoMax [32] to graphs, and learns node representations by maximizing the mutual information (MI) between node and graph encodings. InfoGraph [16], on the other hand, aims to obtain embeddings at the whole graph level for unsupervised and semi-supervised learning. Besides, MVGRL [15] creates another view for the graph by introducing graph diffusion [34], and learns node-level and graph-level representations by contrasting different views. GRACE [35] proposes a novel framework for unsupervised graph representation learning by maximizing the agreement of node representations between two graph views. GCA [36] and GraphCL [37] follow the spirit of SimCLR [38] and learn node/graph representations by directly treating other nodes/graphs as negative samples. Although contrastive learning can use the data themselves to provide the self-supervision information for feature learning, it is not directly applicable to semi-supervised learning as it fails to incorporate the label information that is scarce yet valuable in semi-supervised learning. In this article, we design a novel semi-supervised framework to extract supervision information from both labeled and unlabeled data, which helps to obtain the discriminative node representations to perform downstream tasks.

III. Preliminary

In this section, we first summarize the definition of the graph and main mathematical notations used in this article, and then briefly review the GCN method.

A. Notations

A graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices or nodes (we will use nodes throughout the paper), and $\mathcal{E}$ is the set of edges. Let $v_i \in \mathcal{V}$ denote a node and $e_{ij} = (v_i, v_j) \in \mathcal{E}$ denote an edge pointing from v_j to v_i. The neighborhood of a node v is defined as $N(v) = \{u \in \mathcal{V} | (v, u) \in \mathcal{E}\}$. The adjacency matrix $\mathbf{A}$ is a $n \times n$ matrix with $\mathbf{A}_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and $\mathbf{A}_{ij} = 0$ if $e_{ij} \notin \mathcal{E}$. The degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$, where $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ii}$ means that the degree matrix only has values on the diagonal, and all other positions are set to 0. A graph may have node attributes $\mathbf{X}$, where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is a node feature matrix with $\mathbf{x}_v \in \mathbb{R}^d$ representing the feature vector of a node v. Main notations used in this article are listed in Table I.

B. Graph Convolutional Network

Graph Convolutional Networks (GCNs) [5], [39] generalize convolution operation from euclidean domains to graph-structured data. Similar to the classic CNNs, GCNs contain several convolutional hidden layers that take a feature map matrix $\mathbf{H}^{(k)} \in \mathbb{R}^{n\times d_k}$ as the input and output a feature map $\mathbf{H}^{(k+1)} \in \mathbb{R}^{n\times d_{k+1}}$ by using a graph convolution operator [40]. In general, we set $d_{k+1} \le d_k$ and thus the convolution operation also provides a kind of low-dimensional embedding for nodes.

Though a number of different GCNs have been proposed, here we focus on a representative one proposed by Kipf and Welling

TABLE I
Main Notations Used in This Paper

Notations	Meaning
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The Graph
$\mathbf{A} \in \mathbb{R}^{n\times n}$	The graph adjacency matrix
$\mathbf{S} \in \mathbb{R}^{n\times n}$	The diffusion matrix
$\mathbf{X} \in \mathbb{R}^{n\times d}$	The node feature matrix
$\mathbf{D} \in \mathbb{R}^{n\times n}$	The degree matrix
$\mathbf{W} \in \mathbb{R}^{d\times f}$	The layer-specific trainable weight matrix
$\mathbf{x} \in \mathbb{R}^{d}$	The node feature vector
n, d	The number of nodes and the dimension of node feature vector
$\mathbf{T}$	The transition matrix

[5], which is a multi-layer graph convolutional network for semi-supervised graph learning. Given an input feature matrix $\mathbf{H}^{(0)} = \mathbf{X} \in \mathbb{R}^{n\times d}$ and graph adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, its layer-wise propagation rule is given as follows:

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

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I} \in \mathbb{R}^{n\times n}$ is the adjacency matrix, $\mathbf{I}$ is the identity matrix. $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ii}$ is the degree matrix of $\tilde{\mathbf{A}}$. $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is obtained by the "renormalization trick", representing the normalized adjacency matrix. $\mathbf{W}^{(l)} \in \mathbb{R}^{d\times f}$ is a layer-specific trainable weight matrix. $\mathbf{H}^{(l)} \in \mathbb{R}^{n\times d}$ is the matrix of activations in the l_{th} layer with row $\mathbf{H}_i^{(l)}$ containing a d-dimensional feature vector for node $v_i \in \mathcal{V}$. The first hidden layer $\mathbf{H}^{(0)} = \mathbf{X} \in \mathbb{R}^{n\times d}$, and $\sigma(\cdot)$ denotes an activation function, such as $ReLu(\cdot) = \max(0, \cdot)$.

IV. Proposed SCGNN Model

In this section, we present the proposed SCGNN model that jointly performs unsupervised self-consistent learning and supervised label learning. Fig. 1 shows our framework, consisting of four main components: (1) an augmentation mechanism, which is used to generate the other related view of the original input graph; (2) unsupervised self-consistent loss, which maximizes the local-global mutual information by applying the discriminator $\mathcal{D}$ to score the consistency between node representations and graph representation; (3) supervised cross-entropy loss, which uses a fusion module to fuse node representations of two views into final comprehensive node representations, and further optimize the classification loss over labeled nodes; (4) the objective function, consisting of above two types of loss terms and jointly optimizing to obtain final node representations.

A. Unsupervised Self-Consistent Learning

1) Augmentations: In this article, we adopt graph diffusion to transform an adjacency matrix to a diffusion matrix, providing extra global information for the graph. We treat two matrices as two congruent views of the same graph's structure [15].

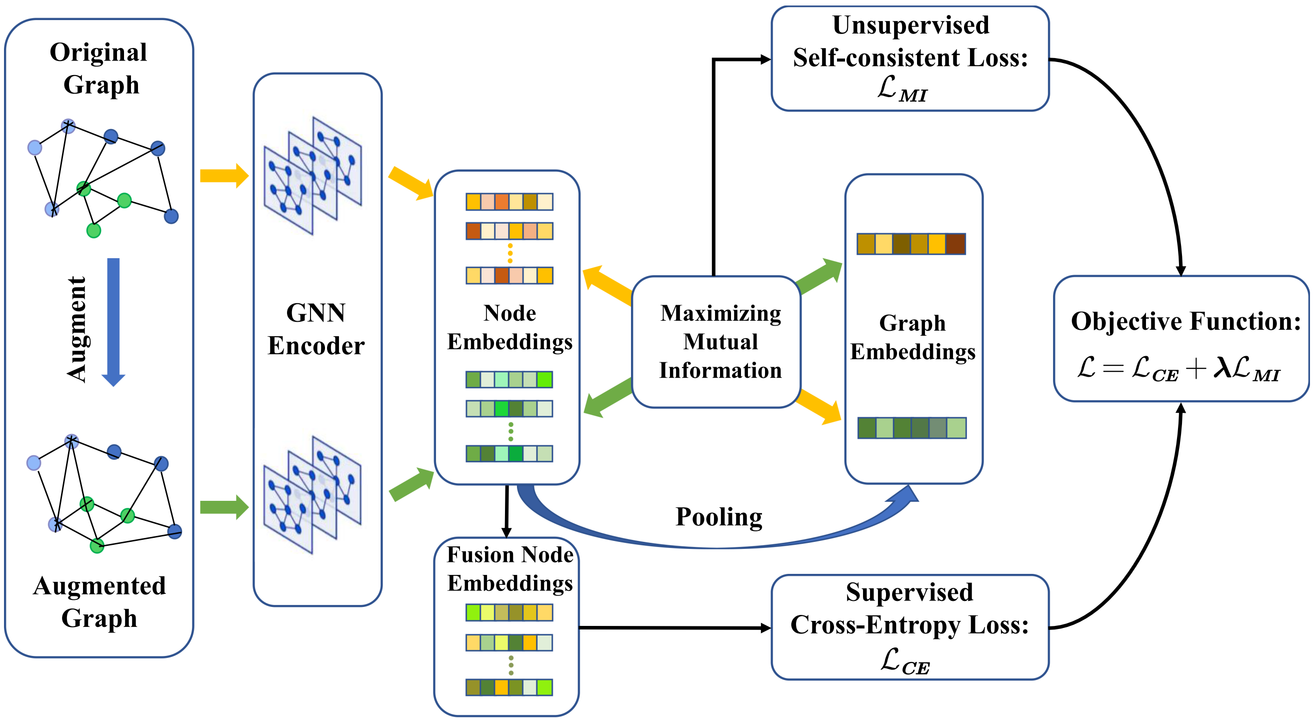

Original Graph
Augment
Augmented Graph
GNN Encoder
Node Embeddings
Maximizing Mutual Information
Graph Embeddings
Unsupervised Self-consistent Loss: $\mathcal{L}_{MI}$
Objective Function: $\mathcal{L} = \mathcal{L}_{CE} + \lambda\mathcal{L}_{MI}$
Pooling
Fusion Node Embeddings
Supervised Cross-Entropy Loss: $\mathcal{L}_{CE}$

Fig. 1. Illustration of the proposed SCGNN model. The original graph and its augmented view are fed together as inputs to two encoders with parameter sharing. To extract the self-supervision informationfrom massive unlabeled nodes, a self-consistent constraint is presented by maximizing the mutual information between outputs of two graph views. Simultaneously, we learn the label information of labeled data by fusing node representations from two graph views. Finally, we combine the two constraints to learn comprehensive node representations. λ is a hyper-parameter to balance two losses.

Generalized graph diffusion [34] is defined as

$$\mathbf{S} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \tag{2}$$

where θ_k is the weight coefficient and $\mathbf{T}$ is the generalized transition matrix. (2) is guaranteed to converge, with associated constraints that $\sum_{k=0}^{\infty}\theta_k = 1, \theta_k \in [0,1]$, and that the eigenvalues of $\mathbf{T}$ are bounded by $\lambda_i \in [0,1]$. Two common examples of transition matrix $\mathbf{T}$ include the random walk transition matrix $\mathbf{T}_{rw} = \mathbf{AD}^{-1}$ and the symmetric transition matrix $\mathbf{T}_{sym} = \mathbf{D}^{-\frac{1}{2}}\mathbf{AD}^{-\frac{1}{2}}$ [34].

For (2), two popular examples of graph diffusion are Personalized PageRank (PPR) [41] and the heat kernel [42]. PPR and heat kernel are defined by setting $\mathbf{T} = \mathbf{T}_{rw}$, $\theta_k = \alpha(1-\alpha)^k$ and $\theta_k = e^{-t}\frac{t^k}{k!}$, respectively. Formally, given the adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, the identity matrix $\mathbf{I} \in \mathbb{R}^{n\times n}$ and the diagonal degree matrix $\mathbf{D} \in \mathbb{R}^{n\times n}$, (2) can be reformulated as

$$\mathbf{S}^{PPR} = \alpha\left(\mathbf{I}_n - (1-\alpha)\,\mathbf{D}^{-\frac{1}{2}}\mathbf{AD}^{-\frac{1}{2}}\right)^{-1}, \tag{3}$$

$$\mathbf{S}^{heat} = \exp\left(t\mathbf{AD}^{-1} - t\right), \tag{4}$$

where $\alpha \in (0,1)$ is the teleport probability and t denotes diffusion time [15]. $\mathbf{S}^{heat}$ and $\mathbf{S}^{PPR}$ as two congruent augmented views of the same graph's structure are used in our encoders. To distinguish two views, we assume that the original view is $\mathbf{V}_\alpha$ and the augmented view is $\mathbf{V}_\beta$.

2) Encoders: Our framework allows various choices of the network architecture without any constraints. Here we focus on Graph Convolutional Network (GCN) [5]: a flexible class of node embedding architecture, which generates node representations by repeated aggregation over local node neighborhoods. As shown in Fig. 1, we consider adjacency and diffusion matrices as two congruent structural views $(\mathbf{V}_\alpha, \mathbf{V}_\beta)$, defining the general formulation of GCN layers similar to (1)

$$\mathbf{H}_\alpha^{(l+1)} = \sigma\left(\hat{\mathbf{A}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}\right), \mathbf{H}_\beta^{(l+1)} = \sigma\left(\hat{\mathbf{S}}\mathbf{X}^{(l)}\mathbf{W}^{(l)}\right), \tag{5}$$

where $\hat{\mathbf{A}}$ is the $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\hat{\mathbf{S}} \in \mathbb{R}^{n\times n}$ is a diffusion matrix. As mentioned previously, $\hat{\mathbf{S}}$ belongs to the other view obtained through the augmentation mechanism. Our aim is to learn a graph encoder $\mathcal{E}_\theta(\cdot)$: $\mathcal{E}_\theta(\hat{\mathbf{A}}\mathbf{XW}, \hat{\mathbf{S}}\mathbf{XW}) \rightarrow \mathcal{E}_\theta(\mathbb{R}^{n\times n} \times \mathbb{R}^{n\times d} \times \mathbb{R}^{d\times f}, \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times d} \times \mathbb{R}^{d\times f}) \rightarrow (\mathbf{H}_\alpha, \mathbf{H}_\beta) \rightarrow (\mathbf{H}_\delta)$, where $\mathbf{H}_\alpha = (\vec{h}_{\alpha 1}, \vec{h}_{\alpha 2}, \ldots.., \vec{h}_{\alpha n})^T \in \mathbb{R}^{n\times f}$ and $\mathbf{H}_\beta = (\vec{h}_{\beta 1}, \vec{h}_{\beta 2}, \ldots.., \vec{h}_{\beta n})^T \in \mathbb{R}^{n\times f}$ are the respective node representations of two views. Then, we use the graph pooling (readout) function $\mathcal{P}(\cdot)$: $\mathbb{R}^{n\times f} \rightarrow \mathbb{R}^f$ to aggregate node representations $\mathbf{H}_\alpha$ and $\mathbf{H}_\beta$ of two views into their respective graph-level summary vector representations $\vec{s}$. Specifically, we perform a direct average function to aggregate the graph-level information, which is represented by

$$\vec{s} = \mathcal{R}\left(\mathbf{H}_\alpha, \mathbf{H}_\beta\right) = \varepsilon\left(\frac{1}{n}\sum_{i=1}^{n}\vec{h}_i\right) \in \mathbb{R}^f, \tag{6}$$

where $\varepsilon(\cdot)$ is the logistic sigmoid function. Applying the readout function to node representations results in two graph representations, i.e., $\vec{s}_\alpha, \vec{s}_\beta$, each associated with one of views. $(\mathbf{H}_\alpha, \mathbf{H}_\beta) \rightarrow (\mathbf{H}_\delta)$ means that we fuse the node representations of two positive graph views, i.e., $\mathbf{H}_\delta = \mathbf{H}_\alpha + \mathbf{H}_\beta$ to obtain comprehensive node embeddings $\mathbf{H}_\delta \in \mathbb{R}^{n\times f}$. The positive graph view refers to a view that doesn't disturb the initial node feature matrix $\mathbf{X}$. To sum up, our model obtains the node representations $\mathbf{H}_\alpha$ and $\mathbf{H}_\beta$ of two views and their respective graph representations $\vec{s}_\alpha$ and $\vec{s}_\beta$, as well as the comprehensive node representations $\mathbf{H}_\delta$.

3) Mutual Information Maximization: The mutual information maximization module is regarded as $\mathcal{D}_\varphi(\cdot)\colon \mathbb{R}^f \times \mathbb{R}^f \rightarrow \mathbb{R}$. $\mathcal{D}_\varphi(\vec{h}_i, \vec{s})$ represents the probability scores of high-level representation $\vec{h}_i$ of each node and graph representation $\vec{s}$ of this graph view. We follow the settings of DGI [14] to calculate the discriminator scores by applying a simple bilinear scoring function

$$\mathcal{D}_\varphi\left(\vec{h}_i, \vec{s}\right) = \sigma\left(\vec{h}_i^T \mathbf{W}' \vec{s}\right), \tag{7}$$

where $\mathbf{W}'$ is a learnable scoring matrix and σ is the logistic sigmoid nonlinearity to convert scores into probabilities of $(\vec{h}_i, \vec{s})$ being a positive example.

Specifically, we discard subscripts of our notations for a moment. The node representations of positive graph views, i.e., $\mathbf{H} = (\vec{h}_1, \vec{h}_2, \ldots.., \vec{h}_n)^T$ represents the node embeddings learned by the graph encoder through initial node feature matrix $\mathbf{X}$. To generate negative graph views, we use a random shuffle function $\mathcal{C}$ [14] to randomly shuffle the feature matrix $\mathbf{X}$ of the original graph $\mathcal{G}$. We define it as $\widetilde{\mathcal{G}} = \mathcal{C}(\mathbf{X})$ and the node representations are denoted as $\widetilde{\mathbf{H}} = (\vec{\tilde{h}}_1, \vec{\tilde{h}}_2, \ldots.., \vec{\tilde{h}}_n)^T$. The objective function is defined as a binary cross-entropy loss as follows:

$$\mathcal{L}_{MI} = \frac{1}{2n}\left(\sum_{i=1}^{n} \mathbb{E}_G\left[\log \mathcal{D}_\varphi\left(\vec{h}_{\beta(\alpha)i}, \vec{s}_{\alpha(\beta)}\right)\right] + \sum_{j=1}^{n} \mathbb{E}_{\widetilde{G}}\left[\log\left(1 - \mathcal{D}_\varphi\left(\vec{\tilde{h}}_{\beta(\alpha)j}, \vec{s}_{\alpha(\beta)}\right)\right)\right]\right), \tag{8}$$

where i and j denote the index values of the node representation matrix for positive and negative graph views (i.e., G and $\tilde{G}$), respectively. By minimizing the above objective function, the proposed method can effectively maximize the mutual information between the node representations and graph representation in two views $\mathbf{V}_\alpha$ and $\mathbf{V}_\beta$.

B. Supervised Label Learning

After fusing node representations of two views, we obtain the final comprehensive representation for each node, i.e., $\vec{h} = (h_1, h_2, \ldots, h_C) \in \mathbb{R}^C$, where C represents the number of categories. Then we apply softmax, i.e., $S_i = \exp(h_i)/\sum_{j=1}^{C}\exp(h_j)$ and obtain a score vector $(S_1, S_2 \ldots, S_C)$. The indexed category with the maximum score is assumed to be the estimated label. Herein we define two popular loss function, one of which is Mean Square Error (MSE) loss defined as follows:

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i}\sum_{c=1}^{C}(S_c - y'_c)^2, \tag{9}$$

where y' represents the category label one-hot vector and S_c is the score vector.

In addition, the other one is Cross-Entropy (CE) loss function given as follows:

$$\mathcal{L}_{CE} = -\frac{1}{n}\sum_{i}\sum_{c=1}^{C} y_{i,c}\log(S_c), \tag{10}$$

where $y_{i,c}$ is a binary indicator of class label for each node. S_c represents the score vector mentioned above.

Algorithm 1: Model Training for SCGNN.

```
Input: Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, node feature matrix $\mathbf{X} \in \mathbb{R}^{n\times d}$,
  initial adjacency matrix $\mathbf{A}_\alpha \in \mathbb{R}^{n\times n}$, augmentation
  function $\mathcal{T}$, pooling function $\mathcal{P}(\cdot)$, shuffle function $\mathcal{C}$,
  graph encoder $\mathcal{E}_\theta(\cdot)$ and discriminator $\mathcal{D}_\varphi(\cdot)$
Output: Node embedding $\vec{h}_v$ for the node $v \in \mathcal{V}$
1: Sample one stochastic augmentation function $\mathcal{T}$: $\mathcal{T}^{PPR}$
   or $\mathcal{T}^{heat}$;
2: Generate specific augmentation graph view $\mathbf{A}_\beta$ by
   performing $\mathcal{T}$;
3: for $epoch \leftarrow 1, 2, \ldots$ do
4:   // Unsupervised Self-consistent Learning
5:   Generate negative graph views by $\widetilde{\mathbf{X}} = \mathcal{C}(\mathbf{X})$;
6:   Obtain positive node representations by
     $\mathbf{H}_\alpha = \mathcal{E}_\theta(\mathbf{A}_\alpha, \mathbf{X}, \mathbf{W})$ and $\mathbf{H}_\beta = \mathcal{E}_\theta(\mathbf{A}_\beta, \mathbf{X}, \mathbf{W})$;
7:   Obtain negative node representations by
     $\widetilde{\mathbf{H}}_\alpha = \mathcal{E}_\theta(\mathbf{A}_\alpha, \widetilde{\mathbf{X}}, \mathbf{W})$ and $\widetilde{\mathbf{H}}_\beta = \mathcal{E}_\theta(\mathbf{A}_\beta, \widetilde{\mathbf{X}}, \mathbf{W})$;
8:   Obtain graph representation $\vec{s}_\alpha$, $\vec{s}_\beta$ with (6);
9:   // Supervised Label Learning
10:  Obtain final node representations by $\mathbf{H}_\delta = \mathbf{H}_\alpha + \mathbf{H}_\beta$;
11:  Minimize the objective function in (11) and update the
     parameters $\theta$ and $\varphi$ by applying Adam optimizer [43];
12: end for
13: return $\mathbf{H}_\delta$.
```

C. Model Optimization

In this section, we will describe how SCGNN jointly optimizes the encoder $\mathcal{E}_\theta(\cdot)$ and the discriminator $\mathcal{D}_\varphi(\cdot)$ for node classification, where $\theta = \mathbf{W}$, $\varphi = \mathbf{W}'$. Specifically, we jointly train the above supervised cross-entropy loss (i.e., $\mathcal{L}_{CE}$) and unsupervised self-consistent loss (i.e., $\mathcal{L}_{MI}$). The final objective function can be summarized as follows:

$$\mathcal{L} = \mathcal{L}_{CE}(y_{i,c}, S_c) + \lambda\mathcal{L}_{MI}\left(\vec{h}_i, \vec{s}\right), \tag{11}$$

where $y_{i,c}$ is the binary indicator and S_c is the score vector of each node predicted by the encoder. $\vec{h}_i$ represents high-level representations for each node. $\vec{s}$ is the graph-level representation obtained by the graph readout function $\mathcal{P}(\cdot)$. $\lambda \in [0, 1]$ is a hyper-parameter for balance $\mathcal{L}_{CE}$ and $\mathcal{L}_{MI}$.

TABLE II
STATISTICS OF THE DATASETS

Dataset	Type	Nodes	Edges	Features	Classes	Training	Test
Cora	Citation network	2,708	5,429	1,433	7	140	1,000
Citeseer	Citation network	3,327	4,732	3,703	6	120	1,000
Pubmed	Citation network	19,717	44,338	500	3	60	1,000
Amazon Computer	Co-purchase graph	13,752	491,772	767	10	200	1,000
Disease	Tree structure graph	1,044	1,043	1,000	2	40	1,000
BlogCatalog	Social network	5,196	171,743	8,189	6	120	1,000

To summarize, our proposed method has the following merits: (1) it combines unsupervised self-consistent learning with supervised label learning, i.e., jointly taking into account the supervision information of labeled and unlabeled nodes, providing rich supervision information for model learning node representations; (2) it adopts unsupervised self-consistent constraint to extend contrastive learning into the semi-supervised learning on graphs by effectively utilizing the structural information and feature information of unlabeled data; (3) it can learn informative and comprehensive node representations by effectively fusing node representations of two positive graph views, and further extract more supervision information from labeled nodes via optimizing the classification loss on labeled nodes. The proposed SCGNN algorithm is summarized in the *Algorithm*.

V. Experiments

In this section, we conduct extensive experiments to verify the effectiveness of the proposed SCGNN. Specifically, we first introduce datasets and various state-of-the-art baselines, and then give the experimental setup of main analysis tasks. Finally, we discuss experimental results, including node classification, visualization of node representations, ablation study and hyper-parameter sensitivity.

A. Experimental Setup

1) Datasets: We conduct experiments on the following six public datasets, the statistics of which are provided in Table II.

- *Cora, Citeseer, Pubmed* [44]: The Cora, Citeseer and Pubmed datasets are citation networks, where nodes are publications and edges are citation links. Node attributes are bag-of-words representations of the papers.
- *Amazon Computer* [45]: Amazon Computers is segment of the Amazon co-purchase graph, where nodes represent goods, edges indicate that two goods are frequently bought together. Node features are bag-of-words encoded product reviews, and class labels are given by the product category.
- *Disease* [46]: The Disease dataset is a graph with tree structure, where node features indicate the susceptibility to the disease.
- *BlogCatalog* [47]: It is a social network with bloggers and their social relationships from the BlogCatalog website. Node attributes are constructed by the keywords of user profiles. The labels represent the topic categories provided by the authors. The nodes are divided into 6 classes.

2) Baselines: Semi-supervised node classification is a classical task in the graph learning and statistical learning communities. We compare our method with the following state-of-the-art methods:

- *DeepWalk* [22] is a network embedding method which uses random walk to obtain contextual information and uses skip-gram algorithm to learn network representations.
- *GCN* [5] is a semi-supervised graph convolutional network model which learns node representations by aggregating information from neighbors.
- *Chebyshev* [39] is a GCN-based method utilizing Chebyshev filters.
- *TAGCN* [48] is a novel graph convolutional network defined in the vertex domain.
- *GraphSAGE* [49] is a general inductive framework that leverages node feature information to efficiently generate node embeddings for previously unseen data.
- *GAT* [6] is a graph neural network model using attention mechanism to aggregate node features.
- *MixHop* [50] is a GCN-based method which mixes the feature representations of higher-order neighbors in one graph convolution layer.
- *ARMA* [9] proposes a novel graph convolutional layer inspired by the Auto-Regressive Moving Average (ARMA) filter.
- *LGCN* [51] proposes a new hyperbolic graph convolution network, which builds the graph operations of hyperbolic GCNs with Lorentzian version.
- *Ortho-GConv* [52] proposes a novel orthogonal feature transformation, which can stabilize the GNN model training and improve the model's generalization performance.
- *STABLE* [53] proposes a contrastive method with robustness-oriented augmentations, which can effectively capture structural information of nodes.

3) Parameters Setting: For all datasets, we select the label rates for training set (20 labeled nodes per class), and choose 500 and 1,000 nodes as the validation set and the test set, respectively. Specifically, for citation network datasets, we choose the same dataset splits as in [44]. For Amazon Computer and BlogCatalog datasets, the settings are the same as [45] and [47], respectively. Since the Disease dataset has only 1,044 nodes, we just split it into training and test sets according to the above split strategy. In addition, we further evaluate the performance by using different label rates (5%, 10%, 15%, 20%) on citation network datasets. In this case, although we expand the training set, the testing set remains unchanged.

TABLE III
ACCURACY (%) FOR NODE CLASSIFICATION TASK. (BOLD: BEST)

Dataset	Cora	Citeseer	Pubmed	Amazon Computer	Disease	BlogCatalog
DeepWalk	67.2	43.2	65.3	70.3 ± 0.7	61.8 ± 0.5	49.2 ± 2.0
Chebyshev	81.2	69.8	74.4	70.9 ± 0.5	74.0 ± 0.6	69.0 ± 0.4
GCN	81.5	70.3	79.0	73.0 ± 0.6	76.0 ± 0.5	69.9 ± 0.2
TAGCN	83.3 ± 0.7	71.4 ± 0.5	$\mathbf{81.1 \pm 0.4}$	77.5 ± 0.5	74.7 ± 0.7	67.9 ± 0.5
GraphSAGE	76.4 ± 0.3	63.4 ± 0.3	78.6 ± 0.3	71.4 ± 0.6	75.3 ± 1.0	62.0 ± 0.8
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	78.0 ± 0.4	73.0 ± 0.8	64.1 ± 0.4
MixHop	81.9 ± 0.4	71.4 ± 0.8	80.8 ± 0.6	76.3 ± 1.2	55.4 ± 2.0	63.4 ± 1.2
ARMA	83.4 ± 0.6	72.5 ± 0.4	78.9 ± 0.3	78.2 ± 0.6	73.6 ± 0.8	74.7 ± 0.3
LGCN	83.5 ± 0.5	72.5 ± 0.6	79.6 ± 0.6	75.8 ± 0.6	79.5 ± 0.4	79.2 ± 0.6
Ortho-GConv	83.3 ± 0.4	71.5 ± 0.2	79.5 ± 0.5	78.0 ± 1.5	75.9 ± 1.2	70.3 ± 0.9
STABLE	84.1 ± 0.5	73.1 ± 0.4	80.3 ± 0.3	78.2 ± 1.0	75.2 ± 1.2	74.5 ± 0.7
SCGNN	$\mathbf{84.5 \pm 0.3}$	$\mathbf{73.5 \pm 0.5}$	80.8 ± 0.5	$\mathbf{79.8 \pm 0.4}$	$\mathbf{85.3 \pm 0.4}$	$\mathbf{81.4 \pm 0.3}$

For our model, we utilize two-layer GCN as backbone of the model, and train it with the Adam optimizer [43]. The weight decay is set to 5e-4. The hidden layer dimension is set to 512 and the hyper-parameter λ is chosen from $\{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. We set the initial learning rate to 0.001 for three citation network datasets and 0.01 for the other datasets. We set ReLU as the activation function and apply a dropout rate of 0.5 to further prevent overfitting. For three citation network datasets, the number of epochs is set to 200. As for other three datasets, we use an early stop mechanism and set the hyper-parameter patience of early stop p to 10. Furthermore, in our experiments, we choose the cross-entropy loss function to calculate the classification loss of supervised label learning.

For all baselines, we adopt the implementations of Chebyshev, GCN, TAGCN, GraphSAGE, GAT and ARMA from the PyTorch Geometric library [54] in all experiments. For the remaining baselines DeepWalk,[1] MixHop[2] and LGCN,[3] Ortho-GConv,[4] STABLE,[5] we provide all the code websites for reproducibility. It should be noted that in Ortho-GConv [52], the author uses three kinds of backbone networks. To ensure fair comparison, we chose to use GCN as the backbone in our experiments. All baselines are initialized with same parameters suggested by their papers and we also further carefully turn parameters to get optimal performance.

B. Node Classification

We evaluate the semi-supervised node classification performance of SCGNN against state-of-the-art baselines. We mainly consider two cases: (1) training 20 labels per class on all six datasets; (2) using label rates of 5%, 10%, 15% and 20% on three citation network datasets. For all methods, we report the mean and standard deviation results over 10 independent trials with different random seeds.

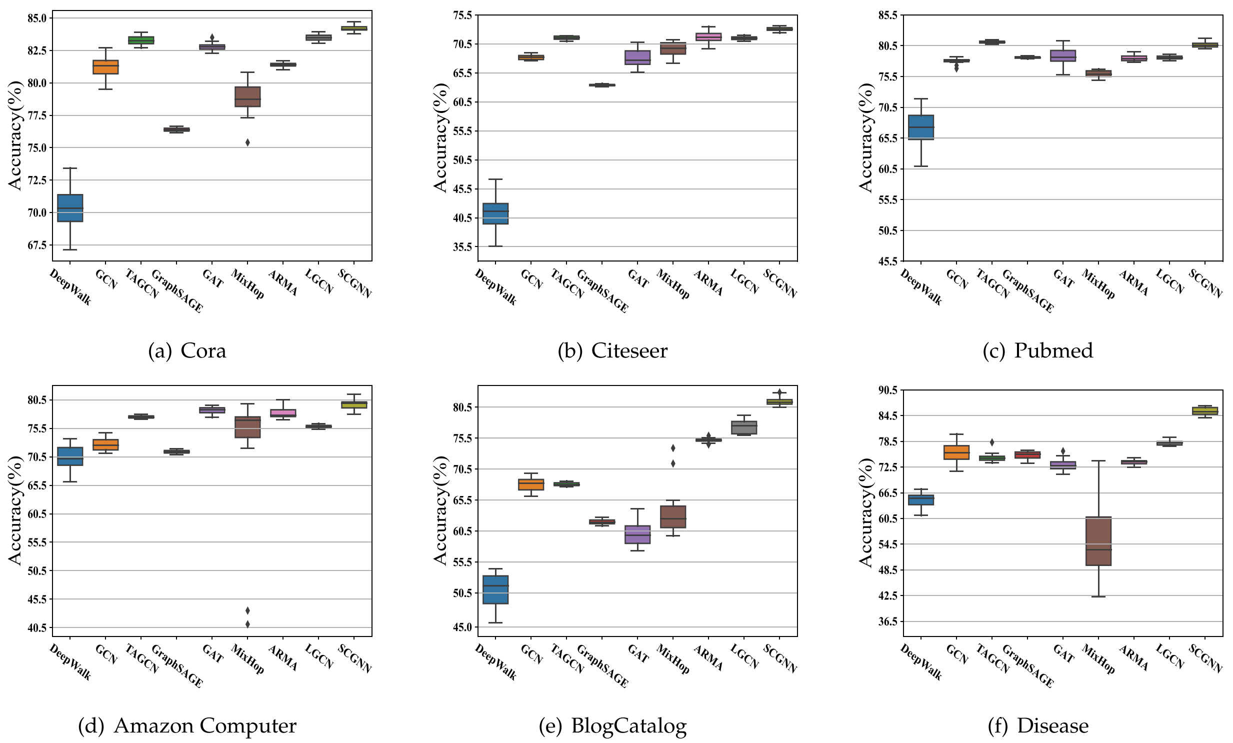
1) 20 Labels per Class: The classification results are shown in Table III and Fig. 2. We use Accuracy (ACC) as the metric for evaluation. By analyzing experimental results, we have the following observations.

- From the Table III, it can be seen that SCGNN consistently performs better than all baselines on five datasets. On Pubmed dataset, SCGNN is slightly lower than TAGCN, yet it still achieves competitive performance. The possible reason is that TAGCN is designed with learnable filters that can adequately adapt to the topology of the graph, which has advantages on large datasets. Compared with the best performance of baselines, especially on Disease and BlogCatalog datasets, SCGNN achieves a large gain, i.e., 85.3% versus 79.5% and 81.4% versus 79.2%, respectively. In addition, on Amazon Computer dataset, the performance is also improved by 1.6% compared with the best baseline, which indicates the effectiveness of SCGNN.
- The Boxplots in Fig. 2 show the accuracy distribution and sensitivity (robustness) of all nine methods. We can observe that, in general, SCGNN has higher accuracy and lower variance on all datasets, which proves that our proposed method is superior to other methods and is more stable. It is worth noting that MixHop has a large variance on the Disease dataset. We assume that there are only two node categories in the dataset and node attributes are scattered. MixHop mixes the feature representations of higher-order neighbors in a graph convolution layer, resulting in relatively discrete node representations.

2) Label Rates of 5%, 10%, 15% and 20%: The classification results are shown in Table IV and Fig. 3, in which we adopt ACC, Macro-F1 score (F1) and Recall, Precision as evaluation metrics. Based on experimental results, we can draw the following conclusions.

- As shown in Table IV, SCGNN performs best on all datasets and label rates compared with all baselines. Especially, in terms of ACC, SCGNN gains improvements of 1.1% on Cora and 1.5% on Citeseer compared with the best baselines. In terms of F1, SCGNN achieves an improvement of 1.9% on Pubmed dataset. Furthermore, SCGNN consistently outperforms GCN on all datasets, the possible reason is that SCGNN can extract more supervision information from graph data than GCN, indicating the effectiveness of the feature fusion mechanism and self-consistent constraint.

[1]https://github.com/phanein/deepwalk
[2]https://github.com/benedekrozemberczki/MixHop-and-N-GCN
[3]https://github.com/ydzhang-stormstout/LGCN
[4]https://github.com/KaiGuo20/Ortho-GConv
[5]https://github.com/likuanppd/STABLE



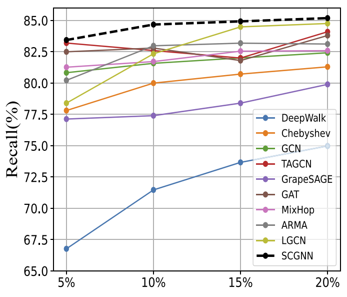
(a) Cora (b) Citeseer (c) Pubmed

(d) Amazon Computer (e) BlogCatalog (f) Disease

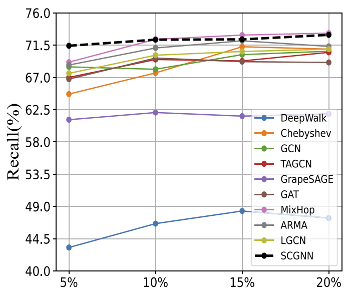
Fig. 2. Boxplots of the classification results of different methods on the six datasets. The black horizontal line represents the mean value and whiskers extend to the minimum and maximum of data points.

TABLE IV
NODE CLASSIFICATION RESULTS (%) WITH DIFFERENT LABEL RATES. (BOLD: BEST)

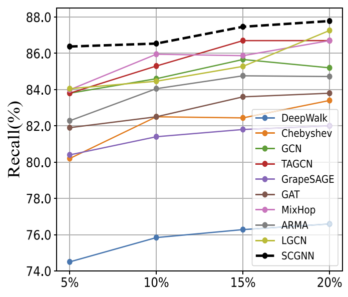
Dataset	Metrics	Ratio	DeepWalk	Chebyshev	GCN	TAGCN	GraphSAGE	GAT	MixHop	ARMA	LGCN	SCGNN
Cora	ACC	5%	68.1 ± 0.1	79.4 ± 0.5	81.5 ± 0.2	83.3 ± 0.7	76.4 ± 0.3	83.0 ± 0.7	81.9 ± 0.4	83.4 ± 0.6	83.5 ± 0.5	**84.5 ± 0.3**
		10%	74.4 ± 0.5	82.0 ± 0.2	84.6 ± 0.3	85.3 ± 0.3	78.4 ± 0.3	84.5 ± 0.5	83.0 ± 0.4	85.1 ± 0.3	84.2 ± 0.7	**86.2 ± 0.6**
		15%	75.5 ± 0.4	82.8 ± 0.5	84.5 ± 0.6	85.5 ± 0.2	81.5 ± 0.2	85.3 ± 0.3	84.4 ± 0.7	84.5 ± 0.5	85.1 ± 0.6	**86.6 ± 0.3**
		20%	76.0 ± 0.2	83.6 ± 0.1	85.3 ± 0.5	86.3 ± 0.3	81.7 ± 0.2	85.9 ± 0.5	84.2 ± 0.5	85.1 ± 0.3	85.0 ± 0.6	**87.4 ± 0.6**
	F1	5%	66.1 ± 0.5	78.0 ± 0.4	79.8 ± 0.3	82.2 ± 0.6	75.5 ± 0.2	82.0 ± 0.3	81.0 ± 0.3	81.9 ± 0.3	80.8 ± 0.5	**83.0 ± 0.4**
		10%	73.5 ± 0.3	79.9 ± 0.5	83.0 ± 0.4	83.3 ± 0.5	76.9 ± 0.5	82.3 ± 0.5	81.9 ± 0.2	83.1 ± 0.3	82.8 ± 0.6	**84.5 ± 0.4**
		15%	74.5 ± 0.8	79.7 ± 0.6	82.5 ± 0.6	83.5 ± 0.3	79.8 ± 0.5	83.4 ± 0.5	82.9 ± 0.6	83.3 ± 0.3	83.5 ± 0.6	**84.8 ± 0.3**
		20%	74.6 ± 0.4	82.3 ± 0.1	83.5 ± 0.4	84.5 ± 0.5	80.2 ± 0.2	84.3 ± 0.5	83.0 ± 0.6	84.2 ± 0.4	83.4 ± 0.4	**85.5 ± 0.7**
Citeseer	ACC	5%	46.5 ± 0.3	70.0 ± 0.5	73.2 ± 0.5	71.8 ± 0.2	65.2 ± 0.2	72.4 ± 0.3	72.0 ± 0.6	72.9 ± 0.5	72.5 ± 0.6	**74.7 ± 0.4**
		10%	48.2 ± 0.1	74.5 ± 0.5	76.0 ± 0.3	75.3 ± 0.5	67.0 ± 0.1	75.6 ± 0.3	75.0 ± 0.3	76.2 ± 0.3	73.0 ± 0.7	**76.4 ± 0.4**
		15%	51.2 ± 0.4	75.4 ± 0.3	76.4 ± 0.5	75.4 ± 0.3	67.4 ± 0.3	76.1 ± 0.2	76.1 ± 0.3	76.5 ± 0.3	74.5 ± 0.5	**76.8 ± 0.2**
		20%	52.4 ± 0.5	75.5 ± 0.2	76.7 ± 0.4	75.7 ± 0.5	68.8 ± 0.3	76.4 ± 0.6	76.4 ± 0.4	76.5 ± 0.3	74.8 ± 0.4	**77.4 ± 0.4**
	F1	5%	42.3 ± 0.1	64.5 ± 0.4	69.7 ± 0.3	68.0 ± 0.1	61.6 ± 0.2	68.7 ± 0.4	69.2 ± 0.5	69.8 ± 0.5	69.2 ± 0.7	**71.2 ± 0.2**
		10%	44.1 ± 0.1	65.0 ± 0.3	69.9 ± 0.5	70.9 ± 0.2	60.0 ± 0.3	70.4 ± 0.4	71.6 ± 0.7	72.1 ± 0.3	69.8 ± 0.7	**72.3 ± 0.5**
		15%	46.8 ± 0.5	68.1 ± 0.3	70.1 ± 0.5	70.5 ± 0.3	61.6 ± 0.3	70.7 ± 0.5	72.0 ± 0.4	72.1 ± 0.3	70.0 ± 0.5	**72.4 ± 0.2**
		20%	48.1 ± 0.2	68.3 ± 0.3	68.3 ± 0.5	71.0 ± 0.5	62.2 ± 0.3	70.7 ± 0.4	72.3 ± 0.8	72.2 ± 0.1	70.5 ± 0.4	**73.0 ± 0.3**
Pubmed	ACC	5%	77.9 ± 0.6	83.6 ± 0.4	85.0 ± 0.5	84.3 ± 1.0	81.7 ± 0.3	83.0 ± 0.4	83.6 ± 0.6	83.1 ± 0.3	84.8 ± 0.6	**86.3 ± 0.4**
		10%	78.9 ± 0.4	83.8 ± 0.2	85.7 ± 0.7	86.2 ± 0.3	82.2 ± 0.1	84.7 ± 0.2	86.2 ± 0.5	85.0 ± 0.2	84.7 ± 0.4	**86.7 ± 0.5**
		15%	79.1 ± 0.3	85.1 ± 0.2	86.7 ± 0.2	87.7 ± 0.3	83.1 ± 0.3	85.7 ± 0.4	86.6 ± 0.5	85.3 ± 0.4	86.2 ± 0.3	**88.1 ± 0.2**
		20%	79.3 ± 0.2	85.5 ± 0.3	86.4 ± 0.3	87.9 ± 0.1	83.3 ± 0.3	86.9 ± 0.1	87.0 ± 0.6	85.9 ± 0.3	86.6 ± 0.6	**88.5 ± 0.3**
	F1	5%	76.2 ± 0.6	83.1 ± 0.5	83.9 ± 0.5	84.0 ± 0.2	80.6 ± 0.4	82.0 ± 0.3	82.5 ± 0.7	82.0 ± 0.3	83.9 ± 0.4	**85.9 ± 0.3**
		10%	77.3 ± 0.4	83.1 ± 0.2	83.8 ± 0.5	85.0 ± 0.3	80.8 ± 0.2	83.2 ± 0.2	85.0 ± 0.5	83.5 ± 0.4	84.0 ± 0.2	**86.3 ± 0.4**
		15%	77.6 ± 0.4	84.4 ± 0.3	85.7 ± 0.1	86.9 ± 0.2	81.9 ± 0.3	84.6 ± 0.5	85.5 ± 0.4	84.2 ± 0.5	85.1 ± 0.6	**87.1 ± 0.3**
		20%	77.8 ± 0.1	84.6 ± 0.2	85.2 ± 0.3	87.5 ± 0.2	82.1 ± 0.2	85.8 ± 0.1	85.8 ± 0.4	84.7 ± 0.4	85.5 ± 0.7	**87.8 ± 0.2**



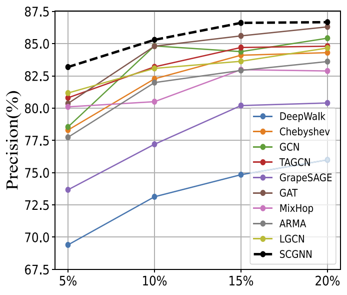
(a) Cora



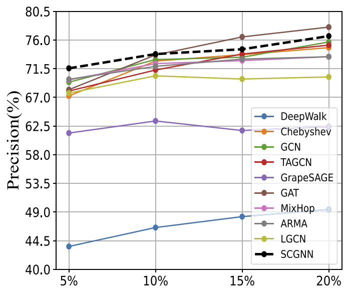
(b) Citeseer



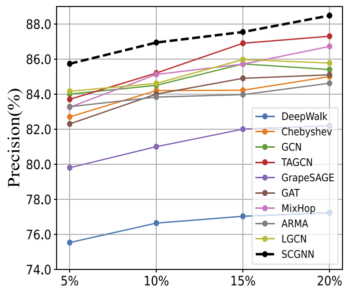
(c) Pubmed



(d) Cora

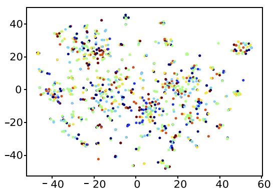


(e) Citeseer

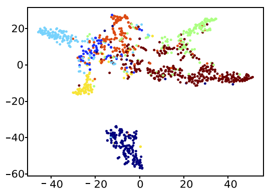


(f) Pubmed

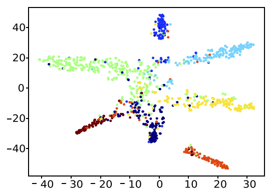
Fig. 3. Recall and Precision of different methods under different label rates on Cora, Citeseer and Pubmed.



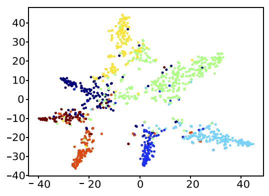
(a) DeepWalk



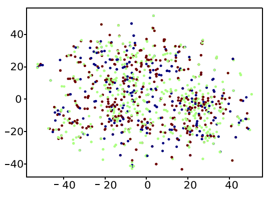
(b) GCN



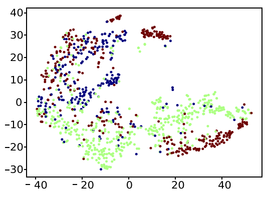
(c) GAT



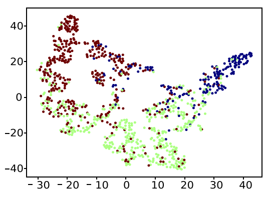
(d) SCGNN



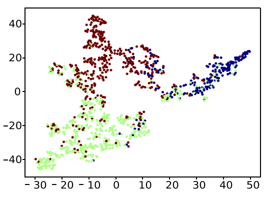
(e) DeepWalk



(f) GCN



(g) GAT



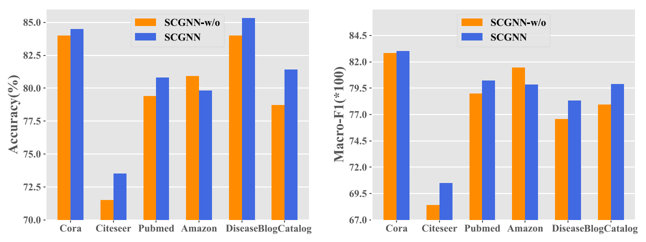
(h) SCGNN

Fig. 4. Visualization representations on Cora (first row) and Pubmed (second row).

- By comparing the results in Table IV, we can see that the performance of SCGNN also improves significantly as the label rate increases, indicating that our proposed method is promising in semi-supervised node classification tasks, especially when labeled data is not easy to obtain.
- The line charts in Fig. 3 show the Recall and Precision of varied number of labeled nodes on three citation network datasets. It is obvious that our model shows excellent performance on Cora and Pubmed datasets. In particular, on Pubmed dataset, SCGNN is significantly better than the best baselines in terms of both Recall and Precision. In addition, on Citeseer dataset, it slightly underperforms MixHop and GAT, but outperforms most other methods.

C. *Visualization*

For a more intuitive comparison, we perform the task of node visualization. Specifically, we use t-SNE [55] to project the learned node embeddings into a 2-dimensional space and visualize them. Taking Cora and Pubmed datasets as examples, we visualize the learned node embeddings in Fig. 4, where each point represents a node and its color indicates the real label.



(a) Accuracy (b) Macro-F1

Fig. 5. Ablation experimental results on six datasets.

From Fig. 4, we can find that the results of DeepWalk, GCN, and GAT are not satisfactory, because the nodes with different labels are mixed together. Compared with GCN, GAT slightly performs better. However, the boundary is still blurry. Apparently, the visualization of SCGNN performs best, where the learned embeddings have a more compact structure, the highest intra-class similarity and the clearest distinct boundaries among different classes, which indicates that our method is beneficial to extract more comprehensive node representations for downstream tasks.

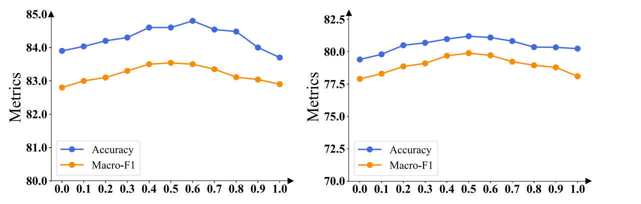
D. Ablation Study

In this section, we compare SCGNN with its variant on all datasets to verify the effectiveness of two components. SCGNN-w/o represents SCGNN without constraint $\mathcal{L}_{MI}$. The results in terms of Accuracy and Macro-F1 are shown in Fig. 5(a) and (b), respectively.

From the results, there are three main conclusions as follows. (1) The performance of SCGNN on the Amazon Computer dataset is slightly lower than that of SCGNN-w/o. The possible reason is that Amazon dataset has too many edge relations, including a lot of noise interference. This causes our model to extract some harmful information and degrades the classification performance. (2) Overall, SCGNN performs significantly better than SCGNN-w/o on the other five datasets, indicating that the constraint $\mathcal{L}_{MI}$ is effective. (3) As a variant of our model, SCGNN-w/o still performs well with the fusion mechanism only. This phenomenon is reasonable and explainable. Since the mechanism fully utilizes the label information in two graph views to obtain comprehensive node representations, our model achieves the accurate prediction during testing, which further demonstrates the superiority of our proposed SCGNN.

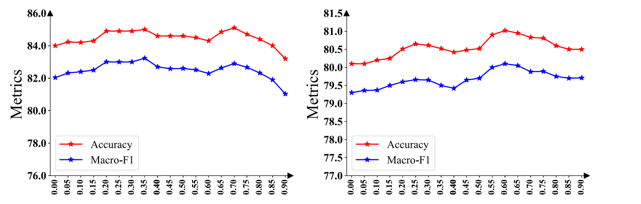
E. Hyper-Parameter λ

In this section, we investigate the sensitivity of hyper-parameter on Cora and Pubmed datasets. First, for the parameter λ,from results of Fig. 6 we can see that SCGNN gradually improves with the increase of λ, and then decreases with the larger λ. It can achieve the best performance with the range between $[0.4, 0.7]$, indicating that the self-consistent constraint can effectively provide self-supervision information. Within this range, the Accuracy and Macro-F1 basically tend to be stable.



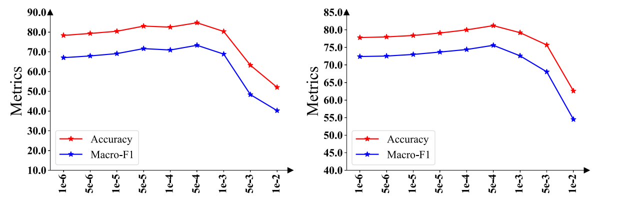
(a) Cora (b) Pubmed

Fig. 6. Analysis of hyper-parameter λ on Cora and Pubmed datasets.



(a) Cora (b) Pubmed

Fig. 7. Analysis of hyper-parameter dropout rate on Core and Pubmed datasets.



(a) Cora (b) Pubmed

Fig. 8. Analysis of hyper-parameter weight decay on Cora and Pubmed datasets.

overall, our model is robust to the perturbation of the hyper-parameter λ. In addition, we also analyze the effect of parameters dropout and weight decay on the performance prediction. As suggested in [56], [57], we use the following hyper-parameter space: dropout rate in the range [0, 0.9] and weight decay in the range [1e-6, 1e-2]. From the results in Figs. 7 and 8, we can observe that for parameter dropout rate, there are two peak ranges in the classification performance on two datasets, ranging between [0.2, 0.35] and [0.55, 0.75], respectively. For parameter weight decay, the performance basically tends to be stable in the range of [1e-6, 5e-4], but drops with larger than 5e-4. Therefore, we set the weight decay value as 5e-4 in our experiments.

F. Effect of Different Views

Finally, we investigate three types of structural views including adjacency matrix, PPR and heat diffusion matrices. Adjacency matrix is processed into a symmetrically normalized adjacency matrix (see Section III-B), PPR and heat diffusion matrices are computed using (3) and (4), respectively. We report the results of different views in terms of ACC and F1 on Cora, Citeseer and Pubmed datasets in Table V. We can find that test results under the adjacency and PPR views perform better,

TABLE V
NODE CLASSIFICATION RESULTS (%) OF DIFFERENT VIEWS. (BOLD: BEST)

Dataset	Metrics	PPR-HEAT	ADJ-HEAT	ADJ-PPR
Cora	ACC	84.0 ± 0.4	83.6 ± 0.2	**84.5 ± 0.3**
	F1	82.7 ± 0.3	82.4 ± 0.1	**83.0 ± 0.4**
Citeseer	ACC	72.3 ± 0.3	73.0 ± 0.3	**73.5 ± 0.5**
	F1	68.5 ± 0.5	69.4 ± 0.3	**70.3 ± 0.4**
Pubmed	ACC	79.0 ± 0.4	78.3 ± 0.5	**80.8 ± 0.4**
	F1	78.3 ± 0.3	77.8 ± 0.4	**80.2 ± 0.3**

and the performance of the adjacency and heat views is not satisfactory. Especially on Pubmed dataset, ACC and F1 score are 80.8% versus 78.3% and 80.2% versus 77.8%, respectively.

VI. CONCLUSION

In this article, we propose a novel self-consistent graph neural networks (SCGNN) method that effectively provides additional discriminative supervision information for semi-supervised GNNs to improve node classification performance. SCGNN could enrich the supervision information from both unlabeled and labeled data. We first extend contrastive learning into semi-supervised learning on graphs by using a self-consistent constraint. Then we introduce a fusion mechanism to fully utilize the data label information in two views to obtain comprehensive node representations. By combining unsupervised self-consistent learning and supervised label learning, the model can fully mine supervision signals in graph data. Experiments conducted on six benchmark datasets demonstrate that our proposed method achieves the state-of-the-art classification performance in semi-supervised learning. Our method can help existing semi-supervised GNNs effectively improve classification performance even with insufficient labeled data.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, “A comprehensive survey on graph neural networks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[2] Z. Zhang, P. Cui, and W. Zhu, “Deep learning on graphs: A survey,” *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[3] Y. Zhang, X. Wang, C. Shi, X. Jiang, and Y. Ye, “Hyperbolic graph attention network,” *IEEE Trans. Big Data*, vol. 8, no. 6, pp. 1690–1701, Dec. 2022.

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Neural message passing for quantum chemistry,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[5] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.

[7] M. Shi, Y. Tang, X. Zhu, and J. Liu, “Multi-label graph convolutional network representation learning,” *IEEE Trans. Big Data*, vol. 8, no. 5, pp. 1169–1181, Oct. 2022.

[8] I. Spinelli, S. Scardapane, and A. Uncini, “Adaptive propagation graph convolutional network,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4755–4760, Oct. 2021.

[9] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, “Graph neural networks with convolutional ARMA filters,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3496–3507, Jul. 2022.

[10] L. Cai, J. Li, J. Wang, and S. Ji, “Line graph neural networks for link prediction,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, Sep. 2022.

[11] Y. Xie, S. Lv, Y. Qian, C. Wen, and J. Liang, “Active and semi-supervised graph neural networks for graph classification,” *IEEE Trans. Big Data*, vol. 8, no. 4, pp. 920–932, Aug. 2022.

[12] Y. Liu, X. Wang, S. Wu, and Z. Xiao, “Independence promoted graph disentangled networks,” in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 4916–4923.

[13] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 433.

[14] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[15] K. Hassani and A. H. Khasahmadi, “Contrastive multi-view representation learning on graphs,” in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.

[16] F.-Y. Sun, J. Hoffmann, V. Verma, and J. Tang, “InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization,” in *Proc. Int. Conf. Learn. Representation*, 2020, pp. 1–16.

[17] J. Yuan, H. Yu, M. Cao, M. Xu, J. Xie, and C. Wang, “Semi-supervised and self-supervised classification with multi-view graph neural networks,” in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 2466–2476.

[18] X. Zhu, Z. Ghahramani, and J. D. Lafferty, “Semi-supervised learning using Gaussian fields and harmonic functions,” in *Proc. Int. Conf. Mach. Learn.*, 2003, pp. 912–919.

[19] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” *J. Mach. Learn. Res.*, vol. 7, no. 11, pp. 2399–2434, 2006.

[20] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, “Deep learning via semi-supervised embedding,” in *Neural Networks: Tricks of the Trade*, Berlin, Germany: Springer, 2012, pp. 639–655.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, “DeepWalk: Online learning of social representations,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.

[23] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, “LINE: Large-scale information network embedding,” in *Proc. Int. Conf. World Wide Web*, 2015, pp. 1067–1077.

[24] A. Grover and J. Leskovec, “node2vec: Scalable feature learning for networks,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.

[25] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, “Hierarchical graph convolutional networks for semi-supervised node classification,” in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 4532–4539.

[26] S. Zhu, S. Pan, C. Zhou, J. Wu, Y. Cao, and B. Wang, “Graph geometry interaction learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 7548–7558.

[27] R. Li, S. Wang, F. Zhu, and J. Huang, “Adaptive graph convolutional neural networks,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, Art. no. 434.

[28] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, “Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 257–266.

[29] M. Li, X. Guo, Y. Wang, Y. Wang, and Z. Lin, “: Graph Gaussian convolution networks with concentrated graph filters,” in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 12782–12796.

[30] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji, “Self-supervised learning of graph neural networks: A unified review,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 2, pp. 2412–2429, Feb. 2023.

[31] R. Linsker, “Self-organization in a perceptual network,” *Computer*, vol. 21, no. 3, pp. 105–117, Mar. 1988.

[32] R. D. Hjelm et al., “Learning deep representations by mutual information estimation and maximization,” in *Proc. Int. Conf. Learn. Representation*, 2019, pp. 1–24.

[33] Y. Tian, D. Krishnan, and P. Isola, “Contrastive multiview coding,” in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 776–794.

[34] J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13354–13366.

[35] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, “Deep graph contrastive representation learning,” in *Proc. Int. Conf. Mach. Learn. Workshop*, 2020, pp. 1–17.

[36] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, 2021, pp. 2069–2080.

[37] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 5812–5823.

[38] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.

[39] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.

[40] B. Jiang, D. Lin, J. Tang, and B. Luo, "Data representation and learning with graph diffusion-embedding networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10414–10423.

[41] S. Brin, "The PageRank citation ranking: Bringing order to the web," *Proc. ASIS*, vol. 98, no. 1998, pp. 161–172, 1998.

[42] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 315–322.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–15.

[44] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 40–48.

[45] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," in *Proc. Int. Conf. Neural Inf. Process. Syst. Workshop*, 2018, pp. 1–11.

[46] I. Chami, Z. Ying, C. Ré, and J. Leskovec, "Hyperbolic graph convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 4868–4879.

[47] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *Proc. ACM Int. Conf. Web Search Data Mining*, 2019, pp. 393–401.

[48] J. Du, S. Zhang, G. Wu, J. M. Moura, and S. Kar, "Topology adaptive graph convolutional networks," 2017, *arXiv:1710.10370*.

[49] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1025–1035.

[50] S. Abu-El-Haija et al., "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 21–29.

[51] Y. Zhang, X. Wang, C. Shi, N. Liu, and G. Song, "Lorentzian graph convolutional networks," in *Proc. Web Conf.*, 2021, pp. 1249–1261.

[52] K. Guo, K. Zhou, X. Hu, Y. Li, Y. Chang, and X. Wang, "Orthogonal graph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3996–4004.

[53] K. Li et al., "Reliable representations make a stronger defender: Unsupervised structure refinement for robust GNN," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 925–935.

[54] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *Proc. Int. Conf. Learn. Representation Workshop*, 2019, pp. 1–9.

[55] L. Van Der Maaten, "Accelerating t-SNE using tree-based algorithms," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3221–3245, 2014.

[56] R. Zhu, Z. Tao, Y. Li, and S. Li, "Automated graph learning via population based self-tuning GCN," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 2096–2100.

[57] X. Yang, J. Wang, X. Zhao, S. Li, and Z. Tao, "Calibrate automated graph neural network via hyperparameter uncertainty," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 4640–4644.

Yanbei Liu received the PhD degree from the School of Electronic and Information Engineering, Tianjin University, Tianjin, China, in 2017. He is currently an associate professor with Tiangong University. His current research interests include graph representation learning, data mining, and machine learning. Until now, he has published more than 30 papers in these fields.

Shichuan Zhao received the BS degree from the College of Information Science and Technology, Hebei Agricultural University, Baoding, China, in 2020, where he is currently working toward the MS degree with the School of Electronic and Information Engineering, Tiangong University, Tianjin, China. His research interests include graph neural network, data mining and machine learning.

Xiao Wang (Member, IEEE) received the PhD degree from the School of Computer Science and Technology, Tianjin University, Tianjin, China, in 2016. He is an associate professor with the School of Software, Beihang University. He was a postdoctoral researcher with the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His current research interests include data mining and machine learning. Until now, he has published more than 50 papers in conferences such as AAAI, IJCAI, KDD, etc. and journals such as *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Trans. on Cybernetics*, etc.

Lei Geng received the BE and ME degrees from the Shaondong University of Technology, in 2003 and 2006, respectively, and the PhD degree from Tianjin University, Tianjin, China, in 2012. He is currently a professor with Tiangong University. His current research interests include pattern recognition, image processing, and machine vision.

Zhitao Xiao received the BE and ME degrees from the Hebei University of Technology, in 1994 and 1997, respectively, and the PhD degree from Tianjin University, Tianjin, China, in 2003. He is currently a professor with Tiangong University. His current research interests include pattern recognition, medical image processing, and intelligent signal processing.

Jerry Chun-Wei Lin (Senior Member, IEEE) received the PhD degree from the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, in 2010. He is currently a full professor with the Department of Computer Science, Electrical Engineering and Mathematical Sciences Western Norway University of Applied Sciences, Bergen, Norway and with the Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, Poland. He has published more than 600 research articles in refereed journals (*IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Fuzzy Systems*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Cybernetics*) and international conferences (IEEE ICDE, IEEE ICDM, PKDD, PAKDD). He is the fellow of IET (FIET), ACM distinguished member (scientist).