SelfSAGCN: Self-Supervised Semantic Alignment for Graph Convolution Network

Xu Yang[1] , Cheng Deng[1]*, Zhiyuan Dang[1] , Kun Wei[1] , Junchi Yan[2]
[1]School of Electronic Engineering, Xidian University, Xian 710071, China
[2]Department of CSE, and MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
{xuyang.xd, zhiyuandang, chdeng.xd, weikunsk}@gmail.com,
yanjunchi@sjtu.edu.cn

Abstract

Graph convolution networks (GCNs) are a powerful deep learning approach and have been successfully applied to representation learning on graphs in a variety of real-world applications. Despite their success, two fundamental weaknesses of GCNs limit their ability to represent graph-structured data: poor performance when labeled data are severely scarce and indistinguishable features when more layers are stacked. In this paper, we propose a simple yet effective Self-Supervised Semantic Alignment Graph Convolution Network (SelfSAGCN), which consists of two crux techniques: Identity Aggregation and Semantic Alignment, to overcome these weaknesses. The behind basic idea is the node features in the same class but learned from semantic and graph structural aspects respectively, are expected to be mapped nearby. Specifically, the Identity Aggregation is applied to extract semantic features from labeled nodes, the Semantic Alignment is utilized to align node features obtained from different aspects using the class central similarity. In this way, the over-smoothing phenomenon is alleviated, while the similarities between the unlabeled features and labeled ones from the same class are enhanced. Experimental results on five popular datasets show that the proposed SelfSAGCN outperforms state-of-the-art methods on various classification tasks.

1. Introduction

Graphs, representing entities and their relationships, have been successfully applied to a wide range of applications [36, 4], such as social networks [26], knowledge graphs [34], and molecular structures [8]. In recent years, many studies focus on developing deep learning approaches for graph structure data [38], leading to rapid development in the field of graph convolution networks (GCNs) [13, 7]. GCNs, generalizing deep convolutional neural network (CNN) [30] to graph-structured data, apply the linear transformation and graph aggregation to all the neighbors of a node and adopt a nonlinear activation function to obtain low-dimensional features for graph nodes.

Normally, GCNs can be categorized into spatial and spectral convolution methods [39, 12]. The operation on node neighbor groups is utilized to define the graph convolution layer in the spatial methods. Diffusion-Convolutional Neural Network (DCNN) [1] adopts a graph diffusion module to incorporate the contextual information of the node. MoNet [24] integrates CNN and provides a unified generalization of CNN architectures on graphs. Moreover, Graph Attention Network (GAT) [29] is utilized to semi-supervised classification task by designing an attention layer, which can learn the weight of each neighbor for feature aggregation. For spectral methods, the graph convolution operation is defined by the spectral representation of graphs. The Fourier domain based on eigen-decomposition of graph Laplacian matrix [2] is proposed to define graph convolution, and the spectral filters based on Chebyshev expansion of the Laplacian are provided to avoid the high computational complexity of eigen-decomposition [6]. More recently, Kipf *et al.* [13] proposed a simple Graph Convolutional Network (GCN) for semi-supervised learning. Simplifying Graph Convolutional (SGC) [31] is proposed to capture higher-order information in the graph by applying the K-th power of the graph convolution matrix in a single neural network layer.

Despite their significant success, the performance of current GCNs drops sharply with a diminishing number of labeled nodes per class. It is easy to cause the over-fitting problem with severely scarce labeled nodes, which exhibits a large testing error even though its training error is small. The reason for this phenomenon is the GCNs rely on graph structure to realize feature propagation, whereas it cannot effectively propagate the labels to the entire graph when

*Corresponding author.

only a few labeled nodes are provided. Moreover, most of the recent GCNs are shallow and achieve their best performance with 2-layer models. Stacking more layers with nonlinearity activations tends to degrade their performances, which is called over-smoothing [17]. Such a phenomenon suggests that the receptive fields are extremely large with stacking more layers and thus the node features are inclined to converge to a certain value [20].

The propagation based on the graph structure can transfer the labeled node features to the unlabeled ones in the same class, making the node features in the same class similar. However, the graph propagation of GCNs is not enough [35], and the phenomenons of *over-fitting* and *over-smoothing* restrict their performance in many cases. Recently, some methods are taking an interest in the semantic information of the nodes. DAGNN [20] is proposed to adaptively incorporate semantic information from large receptive fields. GCNII [3] adopts identity mapping to preserve the inputs information directly. Geom-GCN [25] and non-local GNNs [21] are proposed to capture long-range dependencies from the node features according to non-local aggregators. But, few works discuss the relationship between semantic information and graph structure, and solve these problems effectively. In this paper, we present a simple yet effective Self-Supervised Semantic Alignment Graph Convolution Network (SelfSAGCN), which integrates semantic extraction and alignment into traditional GCNs, to simultaneously overcome the over-fitting and over-smoothing problems. The behind basic idea is the node features in the same class but from semantic and graph structural aspects respectively, are expected to be mapped nearby. Meanwhile, unlabeled node features should be central similar to the labeled ones.

The Self-Supervised Semantic Alignment Graph Convolution Network consists of two crux techniques: Identity Aggregation and Semantic Alignment. Specifically, we first apply the Identity Aggregation to extract semantic information layer-by-layer from the labeled nodes, which is not plagued by the over-smoothing problem. Moreover, the Semantic Alignment transfers the learned semantics to unlabeled node features obtained from the graph aggregation operations using the class central similarity optimization. In this way, the node features from semantic and graph structural aspects are mapped nearby, which has a dramatic effect on alleviating the over-smoothing. Particularly, we construct the class centroids of unlabeled nodes by assigning pseudo-labels in the similarity optimization, and update them gradually to suppress noise signals. The alignment of the labeled nodes and unlabeled ones can further enhance the performance of the model when the labeled nodes are severely scarce. Our experiments show that the SelfSAGCN model outperforms state-of-the-art methods on various classification tasks.

The main contributions of this paper are in three-folds:

- We propose a simple yet effective method called Self-Supervised Semantic Alignment Graph Convolution Network (SelfSAGCN), which consists of Identity Aggregation and Semantic Alignment techniques in a synergetic fashion, to jointly mitigate the *over-fitting* and *over-smoothing* problems.
- To improve the discriminative power of node features and boost the classification performance, we explore the Identity Aggregation to extract discriminative semantic features from the labeled nodes, which has the consistent receptive fields in different layers. Then the unlabeled node features obtained from the graph aggregation operations are aligned with the semantic features by the Semantic Alignment technique for seeking extra supervised information.
- We evaluate the proposed SelfSAGCN on five popular benchmarks, including some standard citation networks and image datasets, and the experiments show that the proposed model outperforms the state-of-the-art methods on various classification tasks.

2. Related Work

A common GCNs layer performs a two-step processing similar to the depth-wise separable convolution: spatial graph aggregation and feature transformation. The first step updates each node feature using feature vectors of their neighboring nodes, which is similar to a 1×1 convolution. After that, each node feature vector is mapped into a new space through a shared linear transformation. GCN [13] and GAT [29] compute a weighted sum of node features within the 1-hop neighborhood, where the weight of each node comes from the degree of the node and the interaction between the neighboring nodes, respectively. GraphSAGE [11] presents the max pooling, while GINs [33] simply sums the node features. Different from these works, LGCN [9] directly explores the regular convolution through top-k ranking. Simplifying Graph Convolutional (SGC) [31] explores to capture higher-order information in the graph by applying the K-th power of the graph convolution matrix in a single neural network layer, which corresponds to a low-pass-type filter on the spectral domain and derives smoothing features across a graph. Moreover, some methods are taking an interest in the semantic information of the nodes. Geom-GCN [25] and non-local GNNs [21] are proposed to capture long-range dependencies for disassortative graph according to non-local aggregators. Shoestring [19] incorporates metric learning into the paradigm of graph-based semi-supervised learning.

Recently, several works attempt to tackle the problem of over-smoothing, which means that node features converge

to indistinguishable limits. In [17], it is demonstrated that the propagation process of the GCN model is a special symmetric form of Laplacian smoothing, which makes the node features in the same class similar. Meanwhile, they showed that stacking more layers may make node features from different classes indistinguishable. JKNet [34] conducts dense skip connections to combine the output of each layer in order to preserve the locality of the node features. Recently, DropEdge [27] is proposed to alleviate the impact of over-smoothing by randomly removing out a few edges from the graph. The Personalized PageRank matrix (APPNP) [14] is utilized to replace the graph convolution matrix and solve the over-smoothing problem. GDC [15] further extends APPNP by generalizing Personalized PageRank to an arbitrary graph diffusion process. Comparatively, our method introduces an identity aggregation operation to extract extra semantic information from the labeled nodes, and adopts the learned information to guide the graph propagation operations according to the semantic alignment, which can jointly mitigate the *over-fitting* and *over-smoothing* problems.

3. Preliminary

Without lose of generalization, we briefly review the fundamental idea of semi-supervised GCN [13]. Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ be input data, and $\mathbf{G} = (\mathbf{X}, \mathbf{A})$ be the graph representation of $\mathbf{X}$ with $\mathbf{A}$ encoding the pairwise relationship among data $\mathbf{X}$, where $\mathbf{A}_{ij} = 1$ if an edge exists between node i and node j. GCN normally contains several propagation (hidden) layers and one final perception layer. Given an input $\mathbf{H}^{(0)} = \mathbf{X}$ and graph $\mathbf{A}$, GCN conducts the following layer-wise propagation in hidden layers as [13]:

$$\mathbf{H}^{(k+1)} = \sigma((\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}), \quad (1)$$

where $k = 0, ..., K-1$ represents the number of graph convolution layers and $\mathbf{I}$ is an identity matrix. $\mathbf{D} = \mathrm{diag}(\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n)$ is a diagonal matrix with $\mathbf{d}_i = \sum_{j=1}^{n} \mathbf{A}_{ij}$. $\mathbf{W}^{(k)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as ReLU. Intuitively, GCN learns feature for each node by propagating neighbors' features and conducting a non-linear transformation after that. By the re-normalization trick [13], we can replace the matrix $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ by a normalized version $\tilde{\mathbf{P}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, and obtain the following Graph Convolutional Layer.

$$\mathbf{H}^{(k+1)} = \sigma(\tilde{\mathbf{P}}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}). \quad (2)$$

The last layer of GCN outputs the final node features $\mathbf{H}^{(K)}$, which can be utilized to node classification. In this task, a softmax activation function is further employed on the final node features $\mathbf{H}^{(K)}$. Let $\mathbf{H}^{out} = \mathrm{softmax}(\mathbf{H}^{(K)}) \in \mathbb{R}^{n \times c}$ be the final output, where c denotes the number of class. Then $\mathbf{H}^{out}$ is utilized to predict final labels for the graph nodes. The weight parameters of GCN network $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, ..., \mathbf{W}^{(K)}$ are trained by minimizing the cross-entropy loss over all the labeled nodes.

$$\mathcal{L}_{Semi-GCN} = -\sum_{i \in L}\sum_{j=1}^{c} \mathbf{Y}_{ij} \ln \mathbf{H}^{out}_{ij}. \quad (3)$$

where L indicates the set of labeled nodes and each row $\mathbf{Y}_{i\cdot}, i \in L$ of $\mathbf{Y}$ denotes the corresponding label indication vector for the i-th labeled node.

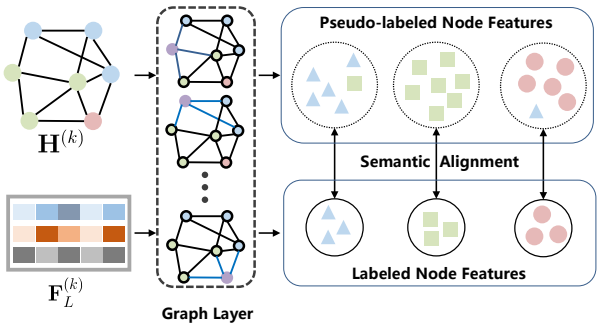


Figure 1. Illustration of each layer from SelfSAGCN. There are two inputs of each graph convolutional layer, and the outputs in the same class are aligned.

4. Methodology

In this section, we propose a Self-Supervised Semantic Alignment Graph Convolution Network (SelfSAGCN), which consists of two crux techniques: Identity Aggregation and Semantic Alignment, to simultaneously address the over-fitting and over-smoothing in the deep graph neural network. The flow chart of the proposed method is shown in Figure 1.

4.1. Identity Aggregation

Different with tradition model GCN [13], the proposed SelfSAGCN has two different inputs. Given an input $\mathbf{H}^{(0)} = \mathbf{X}$ and graph $\mathbf{A}$, SelfSAGCN conducts the following layer-wise propagation in hidden layers as,

$$\mathbf{H}^{(k+1)} = \sigma(\tilde{\mathbf{P}}\mathbf{H}^{(k)}\mathbf{W}^{(k+1)}). \quad (4)$$

Other than that, we adopt the Identity Aggregation to extract the semantic information of the labeled nodes, where the input is $\mathbf{F}^{(0)} = \mathbf{X}$ with an identity graph $\mathbf{I}$, and the output of the hidden layers is:

$$\mathbf{F}^{(k+1)} = \sigma(\mathbf{I}\mathbf{F}^{(k)}\mathbf{W}^{(k+1)}). \quad (5)$$

The last layer of SelfSAGCN outputs the final node features $\mathbf{H}^{(K)}$ and $\mathbf{F}^{(K)}$, and the softmax activation function is utilized to generate the final outputs as

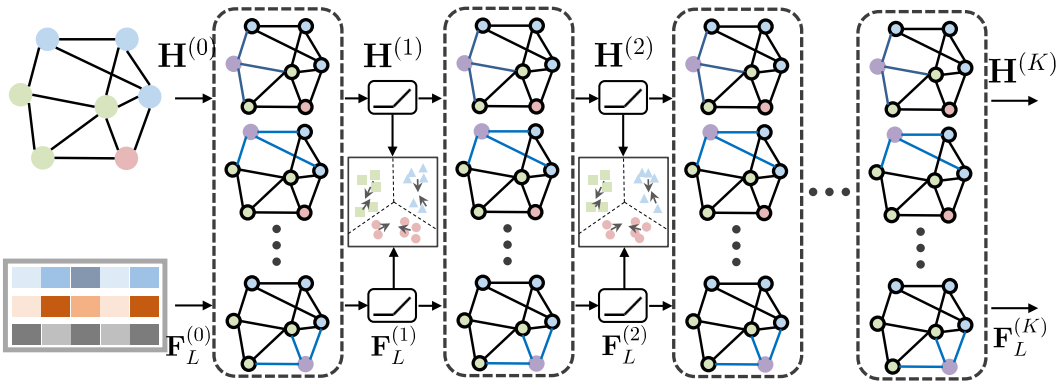


Figure 2. Illustration of the overall architecture. We first extract the node features from semantic and graph structural aspects with different input graphs, and the semantic alignment layer-by-layer is utilized to make the node features from different aspects being central similar.

$\mathbf{H}^{out} = \text{softmax}(\mathbf{H}^{(K)}), \mathbf{F}^{out} = \text{softmax}(\mathbf{F}^{(K)}) \in \mathbb{R}^{n\times c}$. The weight parameters of SelfSAGCN network $\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, ..., \mathbf{W}^{(K)}$ are trained by minimizing the cross-entropy loss over all the labeled nodes L.

$$\mathcal{L}_{Semi} = -\sum_{i\in L}\sum_{j=1}^{c}(\mathbf{Y}_{ij}\ln\mathbf{H}_{ij}^{out} + \mathbf{Y}_{ij}\ln\mathbf{F}_{ij}^{out}). \quad (6)$$

where L indicates the set of labeled nodes and each row $\mathbf{Y}_{i\cdot}, i \in L$ of $\mathbf{Y}$ denotes the corresponding label indication vector for the i-th labeled node.

The node features after graph aggregation may become consistent when the number of layers continues to increase, whereas the second term in $\mathcal{L}_{Semi}$ can effectively suppress this phenomenon due to its receptive fields are consistent. In other words, $\mathbf{F}^{(k)}$ is utilized to provide corresponding discriminative semantic features from the input nodes. Then, our goal is to transfer the learned semantic information to all node features obtained from graph aggregation operations, while making the semantic and graph structural features of each node more similar.

4.2. Semantic Alignment

Since $\mathbf{F}^{(k)}$ and $\mathbf{H}^{(k)}$ are obtained from the same network parameters, and the distribution of nodes from the same class should be similar ideally. $\mathbf{F}^{(k)}$ is not plagued by the over-smoothing phenomenon with stacking more layers. Thus, we adopt the labeled parts of $\mathbf{F}^{(k)}$ to be semantic guidance information for the features $\mathbf{H}^{(k)}$. Let $\mathbf{F}_L^{(k)}$ be the semantic features learned from the set of labeled nodes L. We employ the Semantic Alignment to ensure features of the same class in $\mathbf{F}_L^{(k)}$ and $\mathbf{H}^{(k)}$ are mapped nearby.

More importantly, in the semi-supervised learning task, we cannot obtain the labels of all nodes. Thus, we use pseudo-labels to achieve class-level semantic alignment. Specifically, for $\mathbf{F}_L^{(k)}$ and the labeled nodes of $\mathbf{H}^{(k)}$, we directly apply their corresponding labels as the class information. For the unlabeled nodes of $\mathbf{H}^{(k)}$, we adopt the SelfSAGCN to assign pseudo-labels to the nodes with current network parameters. Particularly, the central similarity is utilized to mitigate the negative impact of false pseudo-labels on the results.

$$\mathcal{L}_{Sema} = \sum_{j=1}^{c} d(\mathbf{C}_j(\mathbf{F}_L^{(k)}), \mathbf{C}_j(\mathbf{H}^{(k)})), \quad (7)$$

where $\mathbf{C}_j(\mathbf{F}_L^{(k)})$ represents the centroid of the features belonging to the j-th class in $\mathbf{F}_L^{(k)}$, and $\mathbf{C}_j(\mathbf{H}^{(k)})$ is the centroid of the features belonging to the j-th class in $\mathbf{H}^{(k)}$. d is the squared Euclidean distance function. Particularly, the pseudo-labels of $\mathbf{H}^{(k)}$ are adopted from $\mathbf{H}^{out}$ and false signals in pseudo-labeled nodes are restrained through centroid alignment as possible [32].

The framework of the proposed SelfSAGCN is shown in Figure 2. As the number of network layers increases, the semantic alignment in each layer is utilized to fully provide the discriminative information to the node features obtained from graph propagation operations, which can alleviate over-smoothing. Meanwhile, the central similarity of labeled nodes and unlabeled ones can provide extra supervised information and further enhance the classification accuracy of unlabeled nodes.

$$\mathcal{L}_{Sema} = \sum_{k=1}^{K}\sum_{j=1}^{c} d(\mathbf{C}_j(\mathbf{F}_L^{(k)}), \mathbf{C}_j(\mathbf{H}^{(k)})). \quad (8)$$

More formally, our totally objective can be written as follows:

$$\mathcal{L} = \mathcal{L}_{Semi} + \lambda\mathcal{L}_{Sema}. \quad (9)$$

The centroids constructed from the pseudo-labels may lack stability in the semi-supervised classification task. Hence, in each iteration, we first calculate the centroids

Algorithm 1 Self-Supervised Semantic Alignment Graph Convolution Network

Input: Input nodes $\mathbf{X}$, graph $\mathbf{A}$, $\mathbf{Y}_L$ of the labeled nodes, number of layers K and number of classes c, hyper-parameters λ and α;

while not converge **do**

1: Calculate $\mathbf{H}^{(k)}$ by Eq. (4);
2: Calculate $\mathbf{F}_L^{(k)}$ by Eq. (5);
3: Calculate the centroids of each class according to Eq. (10).
4: Jointly update all the parameters by minimizing Eq. (9).

end while

Output: Classification results $\mathbf{H}^{out}$.

of $\mathbf{C}_j^t(\mathbf{F}_L^{(k)})$ and $\mathbf{C}_j^t(\mathbf{H}^{(k)})$ using their corresponding features $\mathbf{F}_L^{(k)}$ and $\mathbf{H}^{(k)}$. Then, the centroids of last iteration are added to suppress the instability.

$$\begin{aligned}\mathbf{C}_j^t(\mathbf{F}_L^{(k)}) &\leftarrow (1-\alpha)\mathbf{C}_j^t(\mathbf{F}_L^{(k)}) + \alpha\mathbf{C}_j^{(t-1)}(\mathbf{F}_L^{(k)}) \\ \mathbf{C}_j^t(\mathbf{H}^{(k)}) &\leftarrow (1-\alpha)\mathbf{C}_j^t(\mathbf{H}^{(k)}) + \alpha\mathbf{C}_j^{(t-1)}(\mathbf{H}^{(k)}),\end{aligned} \tag{10}$$

where $\alpha \in [0, 1)$ is the balance weight. Finally, our proposed SelfSAGCN algorithm is sketched in Algorithm 1.

4.3. Further Analysis

First, the class of a node can be predicted by its features in an ideal setting. Under the assumption that the neighbor nodes usually belong to the same class, the propagation based on the graph structure can transfer semantic information from the labeled nodes to the unlabeled nodes in the same class. It makes the node features in the same class similar, which is very beneficial for the semi-supervised classification task. However, the class of a node should be determined by its feature, not by the relationship between the node and other neighbor nodes [20]. Hence, we can extract the node features two different aspects, semantic and graph structure. The learned semantic information can be utilized to guide the feature learning of the graph propagation operations.

In addition, the semantic alignment is utilized to guarantee the obtained features from different aspects being central similar, which mainly includes the similarity of semantic features and graph structural features, as well as the central similarity of labeled nodes and unlabeled nodes belonging to the same class. The ultimate goal is to enforce the distribution of the features from different aspects tend to be consistent, which can provide more discriminative information to the node features and boost the performance.

Method	20 labels per class		
	Cora	Citeseer	Pubmed
MLP	61.6(0.2)	61.0(0.3)	74.2(0.2)
GCN [13]	81.2(0.4)	71.1(0.7)	78.5(1.0)
GAT [29]	83.1(0.7)	70.8(0.9)	71.1(1.2)
SGC [31]	81.7(0.6)	71.3(1.1)	78.9(1.3)
Shoestring [19]	81.9(2.1)	69.5(2.4)	79.7(4.5)
APPNP [14]	83.3(0.4)	71.8(0.9)	80.1(1.3)
DAGNN [20]	**84.4**(0.4)	73.3(0.7)	80.5(0.9)
SelfSAGCN	83.8(0.5)	**73.5**(1.2)	**80.7**(1.5)

Table 1. Summary of classification accuracy (%) on citation networks with 20 labeled nodes.

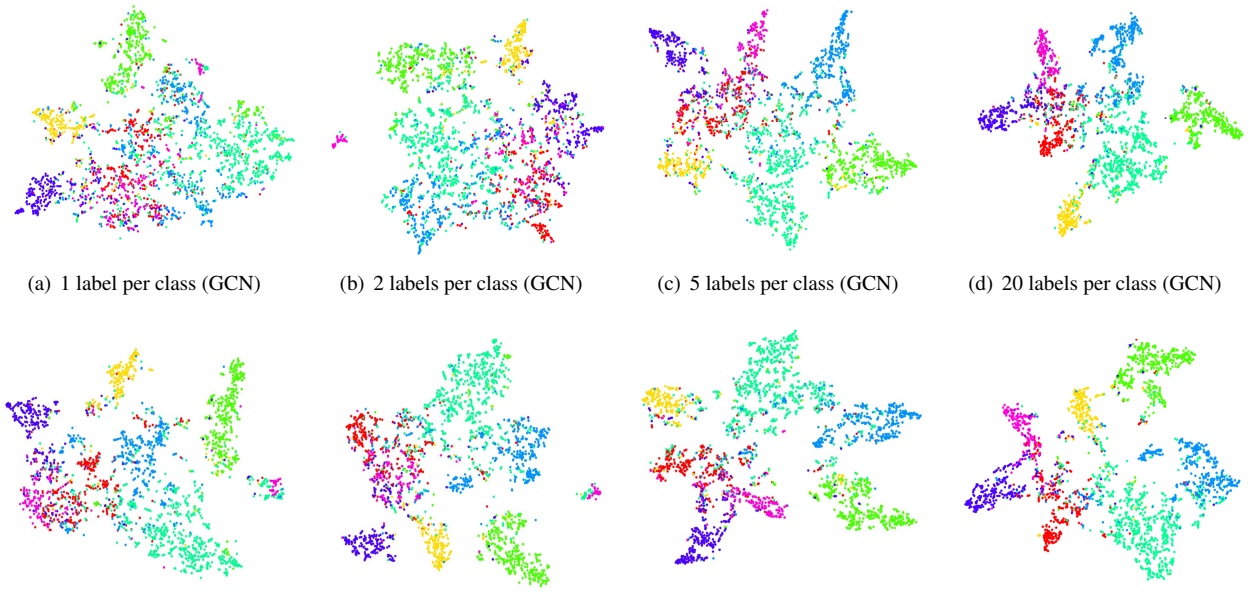
5. Experiments

In this section, we evaluate the effectiveness of the proposed SelfSAGCN with five benchmarks on several semi-supervised classification tasks.

5.1. Datasets

We test the proposed method on five datasets including three standard citation benchmarks datasets (Cora [23], Citeseer [10], and Pubmed [28]) and two widely used image datasets (STL-10 and CIFAR-10 [5]). The details of these datasets are as follows:

- **Cora** contains 2708 nodes and 5429 edges. Each node has a 1433 dimension feature vector and all nodes are falling into seven classes.
- **Citeseer** is a citation network that contains 3327 nodes and 4732 edges. The nodes of this network are falling into six classes and each node has a 3703 dimension feature vector.
- **Pubmed** is a larger network data that contains 19717 nodes and 44338 edges in all. Each node has a 500 dimension feature vector and all the nodes are falling into three classes.
- **STL-10** consists of color images of 96×96 pixel size, in which there are ten classes with 1,300 examples each. Following [37], we extract the deep features of STL-10, and construct a k-nearest neighbor graph ($k = 10$) with nodes denoting images and edges representing the neighborhood relationship between images, which are then used to test the performance of all baselines.
- **CIFAR-10** is a natural image dataset with 60,000 samples from 10 classes. Then, we extract the deep features and construct a k-nearest neighbor graph ($k = 20$) with nodes denoting samples and edges representing the neighborhood relationship between samples.



(a) 1 label per class (GCN) (b) 2 labels per class (GCN) (c) 5 labels per class (GCN) (d) 20 labels per class (GCN)

(e) 1 label per class (SelfSAGCN) (f) 2 labels per class (SelfSAGCN) (g) 5 labels per class (SelfSAGCN) (h) 20 labels per class (SelfSAGCN)

Figure 3. Visualization of the discriminative capability of the node features on Cora. Colors represent node classes. Top: results from GCN, Bottom: results from SelfSAGCNs.

Method	1 label per class			2 labels per class			5 labels per class		
	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
MLP	42.7(0.2)	28.7(0.7)	50.1(0.4)	49.6(0.5)	34.1(0.9)	56.8(0.7)	60.4(0.3)	43.5(0.8)	64.5(0.3)
GCN [13]	43.1(0.7)	29.2(0.9)	51.5(1.8)	58.0(0.6)	40.5(0.7)	60.4(2.3)	69.1(0.5)	53.7(0.9)	68.4(3.2)
GAT [29]	45.2(0.6)	32.5(1.0)	53.5(2.3)	59.7(0.4)	43.2(0.7)	62.2(2.2)	70.2(0.3)	55.2(0.7)	69.5(1.1)
SGC [31]	44.5(0.5)	31.4(0.9)	55.2(2.3)	58.6(0.5)	45.8(0.6)	63.5(2.9)	69.5(0.4)	54.6(0.8)	71.4(2.9)
APPNP [14]	45.2(0.3)	33.5(1.2)	52.3(3.2)	58.3(0.7)	50.1(0.6)	62.7(1.3)	71.2(0.5)	55.9(0.9)	70.7(1.3)
ICGN [18]	43.3(0.5)	35.0(1.1)	54.2(1.5)	63.5(2.7)	43.9(1.9)	62.6(3.2)	72.2(2.5)	55.9(0.9)	70.7(1.3)
DAGNN [20]	59.2(0.4)	45.2(0.7)	58.5(2.4)	65.6(0.8)	54.3(0.8)	65.3(2.7)	72.0(0.4)	58.1(0.9)	71.4(2.2)
Shoestring [19]	60.2(0.9)	52.2(1.3)	60.3(6.1)	68.3(0.9)	60.1(1.3)	63.5(5.7)	73.0(1.2)	64.2(1.5)	68.4(6.3)
SelfSAGCN	**63.5**(0.8)	**57.5**(3.7)	**65.0**(3.3)	**71.7**(0.6)	**68.3**(1.2)	**70.9**(1.9)	**75.1**(0.7)	**71.9**(1.3)	**73.0**(2.1)

Table 2. Summary of classification accuracy (%) on citation networks with severely limited labeled nodes.

5.2. Baselines

We consider the following baselines: Multilayer Perceptron (MLP), Graph Convolutional Network (GCN) [13], Graph Attention Network(GAT) [29], SGC [31], APPNP [14], ICGN [18], DAGNN [20], Shorstring [19]. Moreover, we adopt several works as the baselines, which try to tackle the problem of over-smoothing, including DropEdge [27], ResGCN [16], JKNet [34], IncepGCN [27], GCNII [3]. We aim to provide a rigorous and fair comparison between different models on each dataset by using the same dataset splits and training procedure.

5.3. Implementation Details

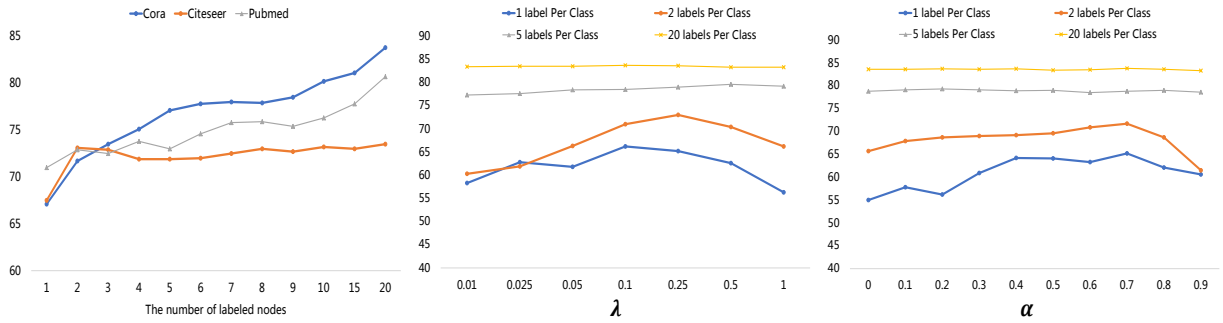
We use the Adam SGD optimizer with a learning rate of 0.01, 0.5 dropout rate, 5×10^{-4} weight decay. We set $\alpha = 0.7$ in the experiments, $\lambda \leftarrow \lambda(\frac{2}{1+e^{-10\times p}} - 1)$ is utilized to suppress noisy labels at the early stages of training. p represents the training epoch and $\frac{2}{1+e^{-10\times p}} - 1 \in (0, 1)$ gradually increases with training. We adopt ReLU as the non-linear activation and the last layer is softmax. We run our method by 20 random trials and report the average performance and the error range.

5.4. Performance with Several Limited Labels

We verify the performance of the proposed SelfSAGCN on different deep graph models. As shown in Table 1, our SelfSAGCN model performs better than other baselines with 20 labeled nodes per class. Moreover, the results for 1, 2, and 5 labeled nodes are shown in Table 2. From the table, we can observe that the proposed method outperforms the competing methods on these benchmark datasets when

Method	1 label per class		2 labels per class		5 labels per class		20 labels per class	
	STL-10	CIFAR-10	STL-10	CIFAR-10	STL-10	CIFAR-10	STL-10	CIFAR-10
MLP	61.4(0.3)	40.8(0.9)	72.2(0.4)	48.6(0.8)	84.8(0.3)	58.3(0.5)	88.2(0.5)	64.5(0.8)
GCN [13]	82.3(0.5)	60.2(0.7)	86.4(0.4)	67.5(0.7)	88.2(0.5)	71.2(0.6)	93.1(0.7)	77.7(0.3)
GAT [29]	84.2(0.4)	58.7(0.8)	86.9(0.7)	65.5(0.9)	89.1(0.8)	71.5(0.3)	94.5(0.6)	78.5(0.7)
APPNP [14]	85.7(0.5)	59.1(0.5)	85.3(0.6)	64.7(0.7)	88.9(0.9)	68.2(0.7)	93.7(0.8)	77.3(0.5)
SGC [31]	84.7(0.7)	59.7(0.9)	85.0(0.4)	65.9(0.6)	88.1(0.7)	70.1(0.8)	93.1(0.5)	78.6(0.6)
DAGNN [20]	87.5(0.5)	61.2(0.8)	89.0(0.9)	67.7(0.5)	92.2(0.4)	72.2(0.9)	94.2(0.5)	79.4(0.4)
SelfSAGCN	**92.1**(0.2)	**65.8**(0.7)	**94.3**(0.3)	**71.2**(0.9)	**95.1**(0.3)	**74.7**(0.8)	**94.9**(0.2)	**80.2**(0.6)

Table 3. Summary of classification accuracy (%) results on various datasets with severely limited labeled samples.



(a) (b) (c)

Figure 4. Results with different hyper-parameters in the terms of classification accuracy (%). a) Results with different training set sizes; b) Results with different loss weight λ on Cora; c) Results with different balance weight α on Cora.

several limited labeled nodes are provided. The performance of our method improves greatly when there is only one label node per class. Particularly, the performance with 5 labels per class on Citeseer is close to the performance with 20 labeled nodes per class. Our method has achieved better results on various classification tasks compared with Shoestring, which adopts a metric learning model to address the problem in the presence of limited labeled nodes. For a fair comparison, we choose its extension on GCN as the comparison. More importantly, the proposed model will not incur extra network parameters.

Figure 3 is the distributions of the node features using t-SNE visualization [22] on Cora data with different labeled nodes. The results demonstrate that the node features from our method have a clearer structure, which can also be justified by the increased classification accuracy. Results with different training label set sizes on various datasets in terms of classification accuracy are shown in Figure 4(a). The classification accuracy will gradually increase when the labeled nodes continues to increase.

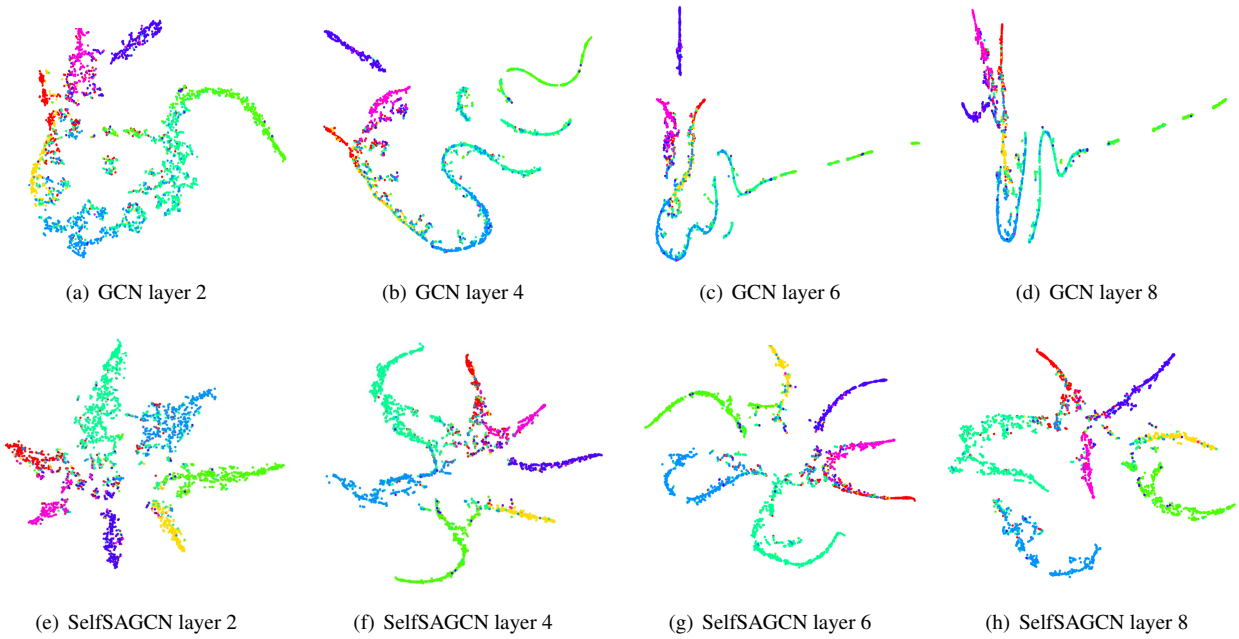
Our method is also applicable to other semi-supervised classification tasks of image datasets. For dataset STL-10 and CIFAR-10, we randomly select 1000 samples for validation purpose and use 5000 samples as test samples, respectively. The results shown in Table 3 demonstrate that the proposed method can be readily applied to a variety of datasets and achieve competitive performances.

5.5. Parameters Analysis

We also investigate the parameter sensitivity in the proposed method. Figure 4(b) represents the change of accuracies with different loss weight λ, which indicates that our method is insensitive to the parameter λ in the range of [0.1,0.5]. In addition, the proposed method becomes more stable when more labeled nodes are utilized. Moreover, Figure 4(c) illustrates that the proposed central similarity optimization method can improve the performance effectively, which is mainly because the gradually update mechanism can suppress the noise signals from the pseudo-labels.

5.6. Performance with Deeper Models

Table 4 summaries the results for the deep models with various numbers of layers using 20 labeled nodes per class, which suggests that the proposed model can alleviate the over-smoothing problem and extend the GCN into a deep model. From the t-SNE visualization on Cora in Figure 5, the discriminative power of the node features derived by different numbers of GCN layers becomes similar gradually. The node features generated by multiple GCN layers, such as 6-th and 8-th layers, are very difficult to be separated, however, the node features obtained from different SelfSAGCN layers are still discriminative. The node features



(a) GCN layer 2 (b) GCN layer 4 (c) GCN layer 6 (d) GCN layer 8

(e) SelfSAGCN layer 2 (f) SelfSAGCN layer 4 (g) SelfSAGCN layer 6 (h) SelfSAGCN layer 8

Figure 5. t-SNE visualization of node features derived by different models with different numbers of layers on Cora. Colors represent node classes. Top: results from GCN, Bottom: results from SelfSAGCN.

Method	4 Layers			8 Layers			16 Layers		
	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed	Cora	Citeseer	Pubmed
GCN [13]	80.4(0.4)	67.6(0.9)	79.0(1.2)	69.5(0.1)	30.2(0.3)	61.2(1.5)	64.9(1.4)	18.3(1.1)	40.9(0.7)
GCNdrop [27]	82.0(0.7)	70.2(0.8)	79.1(1.4)	75.8(0.4)	61.4(0.5)	78.1(0.7)	75.7(1.1)	57.2(0.9)	78.5(1.5)
JKNet[34]	80.2(0.5)	68.7(0.4)	78.0(1.0)	80.7(0.6)	67.7(0.3)	78.1(1.4)	80.2(1.0)	69.8(0.7)	72.6(1.4)
IncepGCN [27]	77.6(0.5)	69.3(0.6)	77.0(0.7)	76.5(0.8)	68.4(0.5)	77.9(1.2)	81.7(0.7)	70.2(0.9)	74.9(1.7)
ResGCN [16]	78.8(0.6)	70.5(0.6)	78.6(1.5)	75.6(0.5)	65.0(0.4)	78.1(1.3)	72.0(0.8)	66.5(0.5)	75.5(1.0)
DAGNN [20]	82.1(0.5)	68.5(0.9)	77.5(0.9)	83.2(0.6)	70.2(0.4)	78.4(0.9)	81.7(0.8)	68.4(0.9)	79.8(1.0)
GCNII [3]	82.0(0.4)	68.7(0.4)	78.6(0.4)	**84.1**(0.3)	70.6(0.5)	**79.4**(0.6)	**84.6**(0.8)	**72.9**(0.9)	**80.2**(1.0)
SelfSAGCN	**82.3**(0.3)	**72.3**(0.8)	**79.5**(1.2)	81.9(0.5)	**71.0**(1.1)	78.7(1.3)	80.5(0.3)	68.0(1.2)	78.1(1.4)

Table 4. Summary of classification accuracy (%) results on citation networks with various numbers of layers.

from semantic and graph structural aspects are enforced to map nearby, which can provide discriminative semantic information to the node features and effectively alleviate the over-smoothing problem.

6. Conclusion

In this paper, we attempt to integrate semantic extraction and alignment into traditional deep graph networks to simultaneously address the over-fitting and over-smoothing problems. To achieve this goal, we first apply the Identity Aggregation to extract semantic information layer-by-layer from the labeled nodes, which is not plagued by the over-smoothing phenomenon. We then align the node features obtained form semantic and graph structural aspects respectively using the central similarity optimization, which has a dramatic effect on alleviating the over-smoothing. Particularly, we construct the class centroids of unlabeled nodes by assigning pseudo-labels, and update the centroids gradually to suppress noise signals. In this way, the distributions of the node features from different aspects tend to be consistent, thus the results in unlabeled nodes will be boosted. We evaluate the proposed SelfSAGCN on five popular benchmarks, and the experiments demonstrate the proposed model outperforms the state-of-the-art methods on various classification tasks. More importantly, the proposed model will not incur extra network parameters.

7. Acknowledgement

Our work was supported in part by the National Natural Science Foundation of China under Grant 62071361, the National Key R&D Program of China under Grant 2017YFE0104100, and CERNET Innovation Project under Grant NGII20190904.

References

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in neural information processing systems*, pages 1993–2001, 2016.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[3] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. *arXiv preprint arXiv:2007.02133*, 2020.

[4] Zhiyuan Dang, Cheng Deng, Xu Yang, and Heng Huang. Multi-scale fusion subspace clustering using similarity constraint. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Zhiyuan Dang, Cheng Deng, Xu Yang, and Heng Huang. Doubly contrastive deep clustering. *arXiv preprint arXiv:2103.05484*, 2021.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[7] Simon S Du, Kangcheng Hou, Barnabás Póczos, Ruslan Salakhutdinov, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *arXiv preprint arXiv:1905.13192*, 2019.

[8] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Advances in neural information processing systems*, pages 6530–6539, 2017.

[9] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1416–1424, 2018.

[10] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pages 89–98, 1998.

[11] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[12] Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang, and Bin Luo. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11313–11320, 2019.

[13] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[14] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

[15] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, pages 13354–13366, 2019.

[16] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9267–9276, 2019.

[17] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *arXiv preprint arXiv:1801.07606*, 2018.

[18] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9582–9591, 2019.

[19] Wanyu Lin, Zhaolin Gao, and Baochun Li. Shoestring: Graph-based semi-supervised classification with severely limited labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4174–4182, 2020.

[20] Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 338–348, 2020.

[21] Meng Liu, Zhengyang Wang, and Shuiwang Ji. Non-local graph neural networks. *arXiv preprint arXiv:2005.14612*, 2020.

[22] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[23] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[24] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[25] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

[26] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. Deepinf: Social influence prediction with deep learning. In *KDD*, pages 2110–2119, 2018.

[27] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2019.

[28] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[30] Kun Wei, Cheng Deng, and Xu Yang. Lifelong zero-shot learning.

[31] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[32] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 5423–5432, 2018.

[33] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[34] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *ICML*, 2018.

[35] Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

[36] Xu Yang, Cheng Deng, Tongliang Liu, and Dacheng Tao. Heterogeneous graph attention network for unsupervised multiple-target domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[37] Xu Yang, Cheng Deng, Kun Wei, Junchi Yan, and Wei Liu. Adversarial learning for robust deep clustering. *Advances in Neural Information Processing Systems*, 33, 2020.

[38] Xu Yang, Cheng Deng, Feng Zheng, Junchi Yan, and Wei Liu. Deep spectral clustering using dual autoencoder network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4066–4075, 2019.

[39] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*, pages 79–91. Springer, 2018.