# Self-supervised Consensus Representation Learning for Attributed Graph

Changshu Liu

School of Computer Science and Engineering, University of Electronic Science and Technology of China ericliu9866@gmail.com Liangjian Wen The Noah's Ark Lab, Huawei Technologies Company Limited wenliangjian1@huawei.com Zhao Kang\*

School of Computer Science and Engineering, University of Electronic Science and Technology of China zkang@uestc.edu.cn

Guangchun Luo

School of Information and Software Engineering, University of Electronic Science and Technology of China gcluo@uestc.edu.cn

# ABSTRACT

Attempting to fully exploit the rich information of topological structure and node features for attributed graph, we introduce self-supervised learning mechanism to graph representation learning and propose a novel Self-supervised Consensus Representation Learning (SCRL) framework. In contrast to most existing works that only explore one graph, our proposed SCRL method treats graph from two perspectives: topology graph and feature graph. We argue that their embeddings should share some common information, which could serve as a supervisory signal. Specifically, we construct the feature graph of node features via k-nearest neighbour algorithm. Then graph convolutional network (GCN) encoders extract features from two graphs respectively. Self-supervised loss is designed to maximize the agreement of the embeddings of the same node in the topology graph and the feature graph. Extensive experiments on real citation networks and social networks demonstrate the superiority of our proposed SCRL over the state-of-the-art methods on semi-supervised node classification task. Meanwhile, compared with its main competitors, SCRL is rather efficient. The source code is available at https://github.com/topgunlcs98/SCRL.

## **CCS CONCEPTS**

• Computing methodologies  $\rightarrow$  Semi-supervised learning settings; Neural networks.

## **KEYWORDS**

self-supervised learning, semi-supervised classification, graph convolutional network

MM '21, October 20-24, 2021, Virtual Event, China

Ling Tian

School of Computer Science and Engineering, University of Electronic Science and Technology of China lingtian@uestc.edu.cn

#### **ACM Reference Format:**

Changshu Liu, Liangjian Wen, Zhao Kang, Guangchun Luo, and Ling Tian. 2021. Self-supervised Consensus Representation Learning for Attributed Graph. In Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3475416

## **1** INTRODUCTION

With the proliferation of information collected on the Internet, besides topological structure, vertices in a graph are often associated with content information, i.e., node attributes, on the top of a plain graph. This type of system is modeled by attributed graph [21]. It poses a new challenge: how to learn a holistic representation for attributed graph? Since deep learning methods can effectively extract useful representation of data, graph neural networks have been applied on a wide range of disciplines, including social network [30], chemistry and biology [10, 32], traffic prediction [7, 31], text classification [9, 13], and knowledge graph [12, 41]. As a representative method in graph neural networks, GCN [19] has shown impressive performance for semi-supervised classification task because it propagates feature information over the graph topology, providing a new fusion strategy for topological structure and node features. By contrast, some models, like multi-layer perceptron (MLP) [27], rely exclusively on node features.

Some recent studies theoretically analyze the weakness of the fusion mechanism in GCN. Li et al. [20] indicate that the graph convolution of the GCN model is actually a special form of Laplacian smoothing on node features. However, repeatedly applying Laplacian smoothing may mix the features of vertices from different clusters and make them indistinguishable, which has a negative impact on downstream tasks [23]. [26] and [42] show that GCN only performs low-pass filtering on feature vectors of nodes and graph structure only provides a way to denoise the data. To further prove that the fusion capability of current GCN [19] is not optimal, Wang et al. [40] set up a series of experiments and demonstrate that MLP [27] and DeepWalk [29] can easily perform better than GCN even under some simple situations that the correlation between node labels and features or topology structure is very obvious. The situation becomes more complex in reality. For example, in a social network, the relationship between people is very complicated,

<sup>\*</sup>Corresponding author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

<sup>© 2021</sup> Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-8651-7/21/10...\$15.00 https://doi.org/10.1145/3474085.3475416



Figure 1: The framework of SCRL model. SCRL consists of two parts: feature extraction module and self-supervision module. First, we use node features to construct feature graph. Then, in feature extraction module, we adopt two independent GCNs to extract the latent features of nodes in two graphs, respectively. Consensus representation is learnt in the self-supervision module by solving the "exchanged prediction" problem. We use cross-entropy loss to penalize the difference between prediction and the ground-truth label.

while most of current GCN-based methods mainly consider the relationship between connected individuals and ignore other implicit information. In a citation network where nodes represent papers and there is an edge between two papers if they share the same author, if we conduct information aggregation based on the original topology structure, it will ignore the situation that an author may write papers belonging to different categories or similar papers are actually written by different authors, hence we may mistakenly aggregate different types of papers together.

Towards a better fusion strategy integrating both node features and topology structure, attention mechanism is also introduced. Graph attention network (GAT) [36] assigns attention weights to every edge, which measures the local similarities among features of neighbors. Wang *et al.* [40] propose an adaptive multi-channel GCN (AMGCN) that learns suitable weights for fusing feature and topology information. However, these attention-based approaches need to compute weights for every node, which is inefficient for large-scale graphs. These issues motivate us to design a new framework which can improve the capability of fusing topology structure and node features and, at the same time, enhance the computational efficiency.

In fact, there are correlations between graph structure and node features and the underlying information from these two aspects could supervise each other [15]. For example, a blogger may tend to follow another one who has similar interests listed on his/her homepage. Hence, we mine the common information between node features and graph structure with an efficient self-supervised mechanism. Specifically, we argue that the shared information could boost the performance of downstream tasks. As shown in Fig.1, firstly a kNN graph is generated from the node features as the feature graph. Secondly, with the feature graph and topology graph, we use different convolution modula to extract node embeddings in feature space and topology space. Finally, in order to learn a consensus representation, we propose a self-supervised module that uses the node embedding in one graph to predict the classification result of the same node in the other graph.

Our main contributions are summarized as follows:

- We propose a novel self-supervised framework to learn a consensus representation for attributed graph. To the best of our knowledge, we are the first to explore the role of self-supervised mechanism in fusing the topology information and node feature information of graph.
- We develop an efficient graph representation learning algorithm which takes less time in training compared with other approaches.
- We conduct extensive experiments on a series of datasets. It shows that our approach outperforms many state-of-the-art methods on semi-supervised node classification task. Even with very small amounts of labeled data, our SCRL is still superior to main competitors.

## 2 RELATED WORKS

## Graph-based Semi-supervised Learning

Large numbers of approaches for semi-supervised learning on graphs have been proposed in the past two decades. Some early studies use graph Laplacian regularization and graph embedding [16, 17] to learn graph representation. Inspired by the Skip-gram model [25] for natural language processing, Perozzi *et al.* propose DeepWalk [29] that learns latent representation from sampled truncated random walks via the prediction of local neighborhood of nodes. Subsequently, some varients are proposed to improve DeepWalk, prominent examples include Line [35] and node2vec [11].

Recently, thanks to the breakthroughs in deep learning, attention has turned to various types of graph neural networks. Bruna et al. [2] propose a general graph convolution framework based on graph Laplacian. [9] then optimizes it utilizing Chebyshev polynomial approximation to improve efficiency. Besides, GCN [19] uses a localized first-order approximation to simplify the convolution operation. GAT [36] gives different attention weights to different nodes in a neighborhood to aggregate node features. Demo-Net [43] builds a degree-specific graph neural network for both node and graph classification. MixHop [1] utilizes multiple powers of adjacency matrix to learn general mixing of neighborhood information. However, these methods only use a single topology graph for node aggregation, which may cause graph structure to be emphatically considered and fails to make full use of rich feature information. AMGCN [40] utilizes attention mechanism to merge embeddings extracted from topology graph and feature graph. However, attention-based methods always require high time and space complexity.

## Self-supervised Learning

Self-supervised learning aims to learn representative features without label information, which is able to reduce human cost for annotating data. Self-supervised learning has found many successful applications ranging from language modeling [6] to computer vision [22, 28]. As a category of self-supervised learning, contrastive learning trains the network by comparing the representations learnt from augmented samples. For example, MoCo [14] and SimCLR [5] construct negative pairs and positive pairs via data augmentation techniques and then compare them. However, it becomes computationally expensive for large datasets. Another class is cluster-based approaches. For instance, Caron *et al.* [4] present a simplified training pipeline by mapping features to cluster prototypes.

Recently, there are a few works focusing on self-supervised learning in the domain of graph. M3S [34] utilizes self-supervised learning approach to improve the generalization performance of GCN. Deep Graph Infomax (DGI) [37] drives local network embeddings to capture global structural information by maximizing local mutual information. Deep graph contrastive representation learning (GRACE) [45] is a graph contrastive representation learning framework by maximizing agreement at the node level. Most graph contrastive learning approaches conduct random corruption on nodes and edges, which may bring noise into original graph data and degrade the learnt representation. Nevertheless, whether selfsupervised mechanism can improve the capability of GCN in fusing topological structure and node features still remains unexplored.

#### 3 THE PROPOSED METHODOLOGY

## 3.1 Notation

An attributed graph can be represented as  $G = \{A, X\}$ , where  $A \in \mathbb{R}^{N \times N}$  is the adjacency matrix of *N* nodes and  $X \in \mathbb{R}^{N \times d}$  is

the node feature matrix wherein every node is described by a vector with *d* dimensions. Each node belongs to one out of M classes. As for *A*,  $A_{ij} = 1$  represents that there is an edge between node *i* and node *j* while  $A_{ij} = 0$  indicates that node *i* and node *j* are not connected. In our study, we derivate the corresponding feature graph  $\hat{G} = {\hat{A}, X}$ , which shares the same *X* with *G*, but has a different adjacency matrix. Therefore, topology graph and feature graph refer to *G* and  $\hat{G}$  respectively.

#### 3.2 The Framework of SCRL

As shown in Fig.1, we use topology graph and feature graph to capture the underlying information in topology space and feature space. Our model mainly contains two components: the feature abstraction module that uses GCN to extract features from graph, the self-supervision module that measures the consistency between the representations learned from the topology graph and feature graph.

#### Feature Graph.

Merely propagating feature information over topology graph may hinder the fusion capability of GCN under some circumstances [40]. A natural idea would be to complement topology graph by fully making use of the information inside node features. Therefore we introduce feature graph into our work.

To represent the structure of nodes in the feature space, we build a kNN graph  $\hat{G}$  based on the feature matrix X. To be precise, a similarity matrix S is computed using the cosine similarity formula:

$$S_{ij} = \frac{x_i \cdot x_j}{|x_i| \cdot |x_j|} \tag{1}$$

where  $S_{ij}$  is the similarity between node feature  $x_i$  and node feature  $x_j$ . Then for each node we choose the top k nearest neighbors and establish edges. In this way, we construct feature graph.

#### Feature Extraction Module.

To extract meaningful features from graphs, we adopt GCN that is comprised of multiple graph convolutional layers. With the input graph G, the (l + 1)-th layer's output  $H^{(l+1)}$  can be represented as:

$$H^{(l+1)} = ReLU(D^{-\frac{1}{2}}AD^{-\frac{1}{2}}H^{(l)}W^{(l)})$$
(2)

where ReLU is the Relu activation function  $(ReLU(\cdot) = max(0, \cdot))$ , D is the degree matrix of A,  $W^{(l)}$  is a layer-specific trainable weight matrix,  $H^{(l)}$  is the activation matrix in the *l*-th layer and  $H^{(0)} = X$ . In our study we use two GCNs to exploit the information in topology and feature space. The output is donated by  $x_t = \{x_{t1}, x_{t2}, \cdots, x_{tN}\}$  and  $x_f = \{x_{f1}, x_{f2}, \cdots, x_{fN}\}$ , respectively.

#### Self-Supervision Module.

An important component of our framework is self-supervision module that is used to fuse the representations learnt by feature extraction modules. To convert the learnt representations into vectors of class scores, the representations of the i-th node  $x_{ti}$  and  $x_{fi}$  are followed by a *prototype head*  $C: \mathbb{R}^U \to \mathbb{R}^B$ . U refers to the dimension of  $x_{ti}$  and  $x_{fi}$ . B refers to the number of prototypes. In *prototype head* C, we store a set of prototype vectors:  $\{c_1, \dots, c_B\}$ . Every prototype vector projects the node feature to a prototype/cluster. By computing the dot product between  $x_{ti}$  and  $c_b$ , we can get the class score of node i corresponding to prototype b in topology graph. Specifically,

$$z_i^{(t)} = C(x_{ti}) = x_{ti}^{\top} C$$
(3)

$$z_i^{(f)} = C(x_{fi}) = x_{fi}^\top C \tag{4}$$

 ${\cal C}$  is a linear layer. Prototypes are initialized randomly and are learnt along with the training process.

Then we compute the probability of assigning prototype j to node representation  $x_i$  by taking the softmax of its projection:

$$p_{ij}^{(t)} = \frac{exp(\frac{1}{\tau}z_{ij}^{(t)})}{\sum_{i'}^{B} exp(\frac{1}{\tau}z_{ii'}^{(t)})}$$
(5)

$$p_{ij}^{(f)} = \frac{exp(\frac{1}{\tau}z_{ij}^{(f)})}{\sum_{i'}^{B} exp(\frac{1}{\tau}z_{ij'}^{(f)})}$$
(6)

where  $\tau$  is a temperature parameter. We use  $p_i^{(t)} = \left\{ p_{i1}^{(t)}, \cdots, p_{iB}^{(t)} \right\}$ and  $p_i^{(f)} = \left\{ p_{i1}^{(f)}, \cdots, p_{iB}^{(f)} \right\}$  to denote the probabilities of the *i*-th node belonging to different prototypes in topology graph and feature graph.

A key question in our self-supervision module is to find a 'target' for the projection of prototype vectors. Inspired by previous works [?], we utilize pseudo label in this module. We cast pseudo label assignment problem as an optimal transport problem and compute the soft labels using the iterative Sinkhorn algorithm [8]. The outputs are denoted by  $q_i^{(t)}$  and  $q_i^{(f)}$  for  $z_i^{(t)}$  and  $z_i^{(f)}$  respectively.

outputs are denoted by  $q_i^{(t)}$  and  $q_i^{(f)}$  for  $z_i^{(t)}$  and  $z_i^{(f)}$  respectively. With  $p_i^{(t)}$ ,  $p_i^{(f)}$  and  $q_i^{(t)}$ ,  $q_i^{(f)}$ , we set up the "exchanged prediction" problem. We assume that both topology graph and feature graph should produce the same label. Therefore the pseudo labels obtained from one graph are predicted using the other graph. To be precise, for every node, it is our goal to minimize the cross entropy of two pairs of probabilities:  $q_i^{(f)}$ ,  $p_i^{(t)}$  and  $q_i^{(t)}$ ,  $p_i^{(f)}$ . The loss function of exchanged prediction can be defined as:

$$l_{ss}^{(i)} = l(p_i^{(t)}, q_i^{(f)}) + l(p_i^{(f)}, q_i^{(t)})$$
(7)

$$l(p_i^{(t)}, q_i^{(f)}) = -\sum_{b=1}^{B} q_{ib}^{(f)} \log p_{ib}^{(t)}$$
(8)

$$l(p_i^{(f)}, q_i^{(t)}) = -\sum_{b=1}^B q_{ib}^{(t)} \log p_{ib}^{(f)}$$
(9)

Summing this loss over all nodes leads to the following loss function for the "exchanged prediction" problem.

$$L_{ss} = \frac{1}{N} \sum_{i=1}^{N} l(p_i^{(t)}, q_i^{(f)}) + l(p_i^{(f)}, q_i^{(t)})$$
  
$$= -\frac{1}{N} \sum_{i=1}^{N} [\frac{1}{\tau} x_{ti}^{\mathsf{T}} C q_i^{(f)} + \frac{1}{\tau} x_{fi}^{\mathsf{T}} C q_i^{(t)}$$
  
$$- \log \sum_{b=1}^{B} \exp(\frac{x_{ti}^{\mathsf{T}} c_b}{\tau}) - \log \sum_{b=1}^{B} \exp(\frac{x_{fi}^{\mathsf{T}} c_b}{\tau})]$$
 (10)

This loss function is jointly minimized with respect to prototypes C and the parameters of the GCN encoders that produce representations  $x_t$  and  $x_f$ .

#### Node Classification.

Ideally,  $X_t$  and  $X_f$  should be close to each other. To preserve the information from feature graph and topology graph,  $X_t$  and  $X_f$  are concatenated as the consensus representation R. Then we use R for semi-supervised classification with a linear transformation and a softmax function. Y' is the prediction result and  $Y'_{ij}$  is the probability of node i belonging to class j. W and a are weights and bias of the linear layer, respectively.

$$Y' = \operatorname{softmax}(W \cdot R + a) \tag{11}$$

Suppose there are *T* nodes with labels in the training set. We adopt cross-entropy to measure the difference between prediction label  $Y'_{ii}$  and ground truth label  $Y_{ij}$ :

$$L_{ce} = -\sum_{i=1}^{T} \sum_{j=1}^{M} Y_{ij} \ln Y'_{ij}$$
(12)

Finally, by combining  $L_{ss}$  and  $L_{ce}$ , overall loss function of our SCRL model can be represented as:

$$L = L_{ce} + L_{ss} \tag{13}$$

The parameters of the whole framework are updated via backpropagation. The consensus representation of the attributed graph can be learnt at the same time. The detailed description of our algorithm is provided in Algorithm 1.

Algorithm 1: The proposed algorithm SCRL							
<b>Input:</b> Node feature matrix <i>X</i> ; adjacency matrix <i>A</i> ; node label							
matrix <i>Y</i> ; maximum number of iterations $\eta$ ; temperature							
parameter $ au$							
1 Compute the feature graph $G$ according to $X$ by running kNN							
algorithm.							
$2 \text{ for } it = 0 \text{ to } \eta \text{ do}$							
3 /* Consensus representation learning */							
4 $x_t = GCN(G) // \text{ embeddings of two graphs}$							
5 $x_f = \text{GCN}(\hat{G})$							
$ c = z^{(t)} = \operatorname{prototype}(x_t) $							
7 $z^{(f)} = \operatorname{prototype}(x_f)$							
8 $p^{(t)} = \operatorname{softmax}(z^{(t)} / \tau)$							
9 $p^{(f)} = \operatorname{softmax}(z^{(f)} / \tau)$							
$q^{(t)} = \operatorname{sinkhorn}(z^{(t)})$							
$q^{(f)} = \operatorname{sinkhorn}(z^{(f)})$							
Calculate the overall loss with Equation(13)							
Update all parameters of framework according to the overall							
loss							
4 end for							
5 Predict the labels of unlabeled nodes based on the trained							
framework.							

**Output:** Classification results Y'

## 4 EXPERIMENT

In this section, we conduct extensive experiments to evaluate the effectiveness of the self-supervised consensus representation learning framework for attributed graph.



Figure 2: The t-SNE demonstration of node representations of ACM and Flickr during training.

Table 1: The statistics of datasets.

Datasets	Nodes	Edges	Dimensions	Classes
Citeseer	3327	4732	3703	6
PubMed	19717	44338	500	3
ACM	3025	13128	1870	3
BlogCatalog	5196	171743	8189	6
UAI2010	3067	28311	4973	19
Flickr	7575	239738	12047	9

## 4.1 Datasets

For the graph dataset, we select four commonly used citation networks (Citeseer [19], PubMed [33], UAI2010 [38], ACM [39]) and two social networks (BlogCatalog [24], Flickr [24]).

Specifically, Citeseer consists of 3327 scientific publications extracted from the CiteSeer digital library classified into one of six classes. PubMed consists of 19717 scientific publications from PubMed database pertaining to diabetes classified into one of three classess. ACM network is extracted from ACM database where publications are represented by nodes and those with the same author are connected by edges. UAI2010 contains 3067 nodes in 19 classes and it has been tested in GCN for community detection. BlogCatalog is a social blog directory containing links between 5196 blogs. Flickr is widely used by photo researchers and bloggers to host images that they embed in blogs and social media. Flickr is composed of 7575 users and they are classified into nine groups. The statistical information of datasets is summarized in Table 1.

#### 4.2 Baselines

We thoroughly verify the performance of our proposed SCRL with representative baselines.

DeepWalk [29] is a graph embedding approach that merely takes into account the structure of the graph. LINE [35] is a graph embedding method for very large graph that utilizes both first-order and second-order proximity of the network. ChebNet [9] is a spectralbased GCN that uses Chebyshev polynomials to reduce computational complexity. GCN [19] further solves the efficiency problem by introducing first-order approximation of ChebNet. For comparison, we use the sparse k-nearest neighbor graph calculated from feature matrix as the input graph of GCN and name it kNN-GCN. GAT [36] adopts attention mechanism to learn the relative weights between two connected nodes. Demo-Net [43] is a degree-specific graph neural network for node classification. MixHop [1] is a GCNbased method that concatenates embeddings aggregated using the transition matrices of k-hop random walks before each layer. DGI [37] leverages local mutual information maximization across the



Figure 3: Visualization of learnt representations of different methods on BlogCatalog dataset.

graph's patch representations. M3S [34] is a self-supervised framework that utilizes DeepCluster [3] to choose nodes with precise pseudo labels. GRACE [45] is a recently proposed graph contrastive learning framework. It generates two graph views by corruption and learns node representations by maximizing the agreement of node representations in these two views. AMGCN [40] extracts embeddings from node features, topological structure, and uses the attention mechanism to learn the adaptive importance weights of embeddings.

## 4.3 Experimental Setup

The experiments are run on the PyTorch platform using an Intel(R) Core(TM) i7-8700 CPU, 64G RAM and GeForce GTX 1080 Ti 11G GPU. Technically two layer GCN are built and we train our model by utilizing the Adam [18] optimizer with learning rate ranging from 0.0001 to 0.0005. In order to prevent over-fitting, we set the dropout rate to 0.5. In addition, we set weight decay  $\in \{1e - 4, \dots, 5e - 3\}$ and  $k \in \{2, \dots, 9\}$  for the kNN graphs. Two popular metrics are applied to quantitatively evaluate the semi-supervised node classification performance: Accuracy (ACC) and F1-Score (F1). For fairness, we follow Wang et al. [40] and Yang et al. [44] and select 20, 40, 60 nodes per class for training and 1000 nodes for testing. For example, there are 19 types of nodes in UAI2010, therefore we train our model on training set with 380/760/1140 nodes, corresponding to label rate of 12.39%, 24.78%, 37.17%, respectively. The selection of labeled nodes on each dataset is identical for all compared baselines. We repeatedly train and test our model for 5 times with the same partition of dataset and then report the average of ACC and F1.

## 4.4 Node Classification Results

The results of experiments are summarized in Table 2, where the best performance is highlighted in bold. Some results are directly taken from [40]. We have following findings:

• It can be seen that our proposed method boosts the performance of the listed baselines across most evaluation metrics on six datasets, which proves its effectiveness. Particularly,

Dataset			AC	ACM BlogCatalog								
L/C	2	20	4	.0 60		20 40			60			
Label Rate	1.9	8%	3.9	97%	5.9	95%	2.3	81%	4.6	52%	6.9	3%
Metrics	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk [29]	62.69	62.11	63.00	61.88	67.03	66.99	38.67	34.96	50.80	48.61	55.02	53.36
LÎNE [35]	41.28	40.12	45.83	45.79	50.41	49.92	58.75	57.75	61.12	60.72	64.53	63.81
ChebNet [9]	75.24	74.86	81.64	81.26	85.43	85.26	38.08	33.39	56.28	53.86	70.06	68.37
GCN [19]	87.80	87.82	89.06	89.00	90.54	90.49	69.84	68.73	71.28	70.71	72.66	71.80
kNN-GCN [40]	78.52	78.14	81.66	81.53	82.00	81.95	75.49	72.53	80.84	80.16	82.46	81.90
GAT [36]	87.36	87.44	88.60	88.55	90.40	90.39	64.08	63.38	67.40	66.39	69.95	69.08
Demo-Net [43]	84.48	84.16	85.70	84.83	86.55	84.05	54.19	52.79	63.47	63.09	76.81	76.73
MixHop [1]	81.08	81.40	82.34	81.13	83.09	82.24	65.46	64.89	71.66	70.84	77.44	76.38
GRACE [45]	89.04	89.00	89.46	89.36	91.08	91.03	76.56	75.56	76.66	75.88	77.66	77.08
AMGCN [40]	90.40	90.43	90.76	90.66	91.42	91.36	81.89	81.36	84.94	84.32	87.30	86.94
SCRL	91.82	91.79	92.06	92.04	92.82	92.80	90.22	89.89	90.26	89.90	91.58	90.76
Dataset			Fli	ckr					UAI	2010		
L/C	2	20	4	10	6	0	2	20	4	0	6	0
Label Rate	2.3	8%	4.7	75%	7.1	.3%	12.	39%	24.	78%	37.	17%
Metrics	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk [29]	24.33	21.33	28.79	26.90	30.10	27.28	42.02	32.92	51.26	46.01	54.37	44.43
LINE [35]	33.25	31.19	37.67	37.12	38.54	37.77	43.47	37.01	45.37	39.62	51.05	43.76
ChebNet [9]	23.26	21.27	35.10	33.53	41.70	40.17	50.02	33.65	58.18	38.80	59.82	40.60
GCN [19]	41.42	39.95	45.48	43.27	47.96	46.58	49.88	32.86	51.80	33.80	54.40	32.14
kNN-GCN [40]	69.28	70.33	75.08	75.40	77.94	77.97	66.06	52.43	68.74	54.45	71.64	54.78
GAT [36]	38.52	37.00	38.44	36.94	38.96	37.35	56.92	39.61	63.74	45.08	68.44	48.97
Demo-Net [43]	34.89	33.53	46.57	45.23	57.30	56.49	23.45	16.82	30.29	26.36	34.11	29.03
MixHop [1]	39.56	40.13	55.19	56.25	64.96	65.73	61.56	49.19	65.05	53.86	67.66	56.31
GRACE [45]	49.42	48.18	53.64	52.61	55.67	54.61	65.54	48.38	66.67	49.50	68.68	51.51
AMGCN [40]	75.26	74.63	80.06	79.36	82.10	81.81	70.10	55.61	73.14	64.88	74.40	65.99
SCRL	79.52	78.89	84.23	84.03	84.54	84.51	72.90	57.80	74.58	67.40	74.90	67.54
Dataset			Cite	eseer					Pub	Med		
L/C	2	20	4	10	6	0	2	20	4	0	6	0
Label Rate	3.6	1%	7.2	21%	10.	82%	0.3	30%	0.6	51%	0.9	1%
Metrics	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1	ACC	F1
DeepWalk [29]	43.47	38.09	45.15	43.18	48.86	48.01	-	-	-	-	-	-
LINE [35]	32.71	31.75	33.32	32.42	35.39	34.37	-	-	-	-	-	-
ChebNet [9]	69.80	65.92	71.64	68.31	73.26	70.31	74.20	73.51	76.00	74.92	76.51	75.83
GCN [19]	70.30	67.50	73.10	69.70	74.48	71.24	79.00	78.45	79.98	79.17	80.06	79.65
kNN-GCN [40]	61.35	58.86	61.54	59.33	62.38	60.07	71.62	71.92	74.02	74.09	74.66	75.18
GAT [36]	72.50	68.14	73.04	69.58	74.76	71.60	-	-	-	-	-	-
Demo-Net [43]	69.50	67.84	70.44	66.97	71.86	68.22	-	-	-	-	-	-
MixHop [1]	71.40	66.96	71.48	67.40	72.16	69.31	-	-	-	-	-	-
GRACE [45]	71.70	68.14	72.38	68.74	74.20	70.73	79.50	79.33	80.32	79.64	80.24	80.33
AMGCN [40]	73.10	68.42	74.70	69.81	75.56	70.92	76.18	76.86	77.14	77.04	77.74	77.09
SCRL	73.62	69.78	75.08	70.68	75.96	72.84	79.62	78.88	80.74	80.24	81.03	80.55

Table 2: Node classification results(%). L/C refers to the number of labeled nodes per class.

compared with AMGCN, SCRL achieves a maximum improvement of 8.33% for ACC and 8.53% for F1 on BlogCatalog. Additionally, on Flickr our method can exceed AMGCN by 4.26% and 4.67% for ACC and F1, respectively. This is mainly attributed to the self-supervision component.

- Our SCRL achieves better performances than GRACE on most of the metrics, especially when the label rate is relatively low. This could be explained by the fact that GRACE performs corruption by randomly adding/deleting edges to generate views for graphs that may damage the original graph topology, hence degrading performance of classification.
- On some occasions, feature graph produces better result than topology graph. For example, on BlogCatalog, Flickr, and UAI2010, kNN-CGN easily outperforms GCN. This confirms the necessity of incorporating the feature graph into our framework. On all datasets, SCRL achieves impressive improvement compared with both GCN and kNN-GCN, which indicates that the consensus representation of two graphs provides more holistic information than a single graph.

For a more intuitive understanding, we use t-SNE to visualize the evolution of the representation learnt by our SCRL in the training process. As shown in Fig.2, at the beginning the representation of ACM is chaotic and scattered. At apoch 10, compact clusters begin



Figure 4: Averaged time cost per epoch of AMGCN, GRACE and SCRL for six datasets. (\*) indicates out-of-memory error and vertical axis is in log-scale.

to form. By epoch 20, a well learnt representation is established that makes it easier to separate data points into different groups. The representation of Flickr has a similar evolution process in the training. To further show the advantage of our proposed method, we also visualize the embedding results of BlogCatalog generated by GCN, GRACE, AMGCN and SCRL, which are shown in Fig.3. It can be observed that the embedding generated by our proposed SCRL exhibits clearer cluster structure compared to other three methods.

To verify the efficiency of SCRL, we report the averaged training time per epoch when training SCRL, GRACE and AMGCN in Fig.4. Experiments are conducted with a GeForce GTX 1080 Ti 11G GPU. It can be seen that SCRL always costs much less time to train than others. For instance, for Flickr our proposed SCRL costs 65.9ms per epoch but AMGCN and GRACE need 237.7ms and 122.8ms, respectively. For BlogCatalog, SCRL needs 78.2ms per epoch while AMGCN needs 784.5ms and SCRL needs 243.8ms. What's more, for larger datasets like PubMed, AMGCN and GRACE are subject to out-of-memory error. As mentioned above, AMGCN introduces attention mechanism and computes attention weights for every node. GRACE divides node in two views into positive pairs, inter-view negative pairs and intra-view negative pairs, and then makes pair-wise comparisons in the contrastive loss. Therefore, both AMGCN and GRACE become computationally inefficient and resource-consuming during training.

# 4.5 Few Labeled Classification

To further investigate the capability of our proposed SCRL in dealing with scarce supervision data, we conduct experiments when the number of labeled examples is extremely small. Taking Citeseer and PubMed for example, we strictly follow Li *et al.* [20] and select a small set of labeled examples for model training. Specifically, for Citeseer, we select 3, 6, 12, 18 nodes per class, corresponding to four label rates: 0.5%, 1%, 2% and 3%. For PubMed, we select 2, 3, 7 nodes per class, corresponding to three label rates: 0.03%,

Table 3: Classification accuracy on Citeseer and PubMed with low label rates.

Datasets		Cite	seer		PubMed		
L/C	3	6	12	18	2	3	7
Label Rate	0.5%	1%	2%	3%	0.03%	0.05%	0.10%
ChebNet [9]	19.7	59.3	62.1	66.8	55.9	62.5	69.5
GCN [19]	33.4	46.5	62.6	66.9	61.8	68.8	71.9
GAT [36]	45.7	64.7	69.0	69.3	65.7	69.9	72.4
DGI [37]	60.7	66.9	68.1	69.8	60.2	68.4	70.7
M3S [34]	56.1	62.1	66.4	70.3	59.2	64.4	70.5
GRACE [45]	55.4	59.3	63.4	67.8	64.4	67.5	72.3
AMGCN [40]	60.2	65.7	68.5	70.2	60.5	62.4	70.8
SCRL	62.4	67.3	69.8	73.3	67.9	71.9	73.4



Figure 5: The influence of iteration number in Sinkhorn algorithm.

0.05% and 0.10%. To make a fair comparison, we report mean classification accuracy of 10 runs.

We report the result in Table 3. We can observe that SCRL outperforms all state-of-the-art approaches. It can be seen that the accuracy of GCN, ChebNet, and GAT decline severely when the label rate is very low, especially on 0.5% Citeseer, due to insufficient propagation of label information. By contrast, self-supervised/contrastive approaches, i.e., DGI, M3S, GRACE, are obviously much better because they additionally exploit supervisory signals with data themselves. Different from them, SCRL explores supervisory signals in feature graph, which is ignored by other self-supervised methods. Specifically, SCRL improves DGI, M3S, GRACE by 3.03%, 5.29%, and 5.12% on average, respectively.

## 4.6 Parameter Analysis

In this section, we analyze the sensitivity of parameters of our method on ACM, BlogCatalog, Citeseer, and UAI2020.

*Number of prototypes B*. In Table 4, we evaluate the influence of prototype number on Flickr via varying the value of *B*. We can observe that when *B* is bigger than 27, increasing prototype number



Figure 6: The influence of parameter k in feature graph.

Table 4: The influence of prototype number on Flickr.

Number of prototypes	9(M)	27(3M)	45(5M)	90(10M)
20 L/C	78.54	79.52	78.82	78.75
40 L/C	83.82	84.15	84.23	83.90
60 L/C	84.03	84.54	84.48	84.42

may not improve the performance substantially. This suggests that B has little influence as long as there are "enough" prototypes. In fact, using too many prototypes increases computation time. Throughout this paper, we set B in the range of M to 3M when we train SCRL.

**Number of iterations p**. We investigate the impact of normalization steps performed during Sinkhorn algorithm [8]. We test it by setting p to 5 values: 3, 5, 10, 15, 30. The results are shown in Fig.5. As the number of iterations increases, the accuracy normally raises first and then holds steady. We observe that 5 iterations are usually good enough for the model to reach an ideal accuracy.

**Parameter k**. Finally, we study the impact of parameter k in the kNN feature graph with various k ranging from 2 to 9. We conduct experiments on ACM and BlogCatalog and fix the iteration number p to 5. For every k, we repeat experiments 10 times and record ACC to further calculate its average and standard deviation, which are displayed in Fig.6. For ACM, with the increasing of k, the performance usually becomes better as well. It is because a larger k can provide more information about the relationship between nodes in feature space. Generally, BlogCatalog has a similar trend. However, with 60 labeled nodes per class, SCRL achieves the best performance when k is 7. That is perhaps because too many neighbor nodes in feature graph may introduce some noisy edges.

## **5 ABLATION STUDY**

As mentioned above, our proposed SCRL model employs self-supervised loss to learn a consensus representation from topological structure and node features. To elucidate the contribution of self-supervision module, we report the classification results of SCRL when this component is removed on three datasets: Citeseer, ACM, and Flickr. For fairness, the split of data is identical with the experimental setting in Section 4. We adopt "SCRL(w/o SSL)" to represent the simplified model when self-supervised loss (SSL) is removed. The comparison is shown in Table 5. Apparently, ACC and F1 decrease when the aforementioned component is dropped from the framework. It reveals that our proposed self-supervision module is able to improve

Table 5: The influence of self-supervised module.

		1		SCDI
Dataset	Metrics	L/C	SCRL	SCRL
				(w/o SSL)
		20	79.52	76.28
	ACC	40	84.23	79.10
Flieler		60	84.54	83.58
FIICKI		20	78.89	74.70
	F1	40	84.03	78.37
		60	84.51	83.12
		20	91.82	90.24
	ACC	40	92.06	90.32
ACM		60	92.82	91.50
ACM		20	91.79	90.20
	F1	40	92.04	90.28
		60	92.80	91.45
		20	73.50	71.82
	ACC	40	75.08	74.48
Citeseer		60	75.96	74.30
		20	69.91	68.26
	F1	40	70.41	69.35
		60	72.81	70.74

the performance on semi-supervised learning task substantially. For example, the accuracy can be boosted by over 5% in some cases due to the introduction of self-supervised loss. In addition, the improvement of SCRL(sw/o SSL) over GCN verifies the importance of feature graph.

## 6 CONCLUSION

In this paper, we propose a self-supervised consensus representation learning framework for semi-supervised classification on attributed graph. We make the first attempt to introduce the idea of selfsupervised learning to integrate the correlated information from the topology structure and node features. Specifically, we require that the embeddings of feature graph and topology graph should be consistent and generate the same labels. It is realized by the "exchanged prediction" in the self-supervised module. Extensive experiments well demonstrate its superior performance over the state-of-the-art models on real world datasets. For example, our SCRL improves AMGCN by 2.74% and 2.94% for ACC and F1 on average across all datasets; SCRL improves GRACE by 9.28% and 10.58% for ACC and F1.

## ACKNOWLEDGMENTS

This paper was in part supported by Grants from the Natural Science Foundation of China (Nos. 61806045,U19A2059).

## REFERENCES

- Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.
- [2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In 2nd International Conference on Learning Representations, ICLR 2014.
- [3] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep clustering for unsupervised learning of visual features. In Proceedings of the European Conference on Computer Vision (ECCV). 132–149.

- [4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. 2020. Unsupervised Learning of Visual Features by Contrasting Cluster Assignments. (2020).
- [5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In Proceedings of the 37th International Conference on Machine Learning, ICML, Vol. 119. 1597–1607.
- [6] Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. Advances in Neural Information Processing Systems 32, 7059–7069.
- [7] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems* 21, 11 (2019), 4883–4894.
- [8] Marco Cuturi. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. Advances in neural information processing systems 26 (2013), 2292–2300.
- [9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems 29 (2016), 3844–3852.
- [10] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. 2015. In Advances in Neural Information Processing Systems, Vol. 28.
- [11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. 855–864.
- [12] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. 1802–1808.
- [13] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems. 1025–1035.
- [14] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 9729–9738.
- [15] Zhao Kang, Xiao Lu, Jian Liang, Kun Bai, and Zenglin Xu. 2020. Relation-guided representation learning. *Neural Networks* 131 (2020), 93–102.
- [16] Zhao Kang, Haiqi Pan, Steven C.H. Hoi, and Zenglin Xu. 2020. Robust Graph Learning From Noisy Data. IEEE Transactions on Cybernetics 50, 5 (2020), 1833– 1843.
- [17] Zhao Kang, Chong Peng, Qiang Cheng, Xinwang Liu, Xi Peng, Zenglin Xu, and Ling Tian. 2021. Structured Graph Learning for Clustering and Semi-supervised Classification. *Pattern Recognition* 110 (2021), 107627.
- [18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015.
- [19] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In International Conference on Learning Representations (ICLR).
- [20] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI* conference on artificial intelligence.
- [21] Zhiping Lin and Zhao Kang. 2021. Graph Filter-based Multi-view Attributed Graph Clustering. In Proceedings of the 30th International Joint Conference on Artificial Intelligence, IJCAI 2021, August 19-26, 2021.
- [22] Juncheng Lv, Zhao Kang, Xiao Lu, and Zenglin Xu. 2021. Pseudo-supervised Deep Subspace Clustering. *IEEE Transactions on Image Processing* 30 (2021), 5252–5263. https://doi.org/10.1109/TIP.2021.3079800
- [23] Zhengrui Ma, Zhao Kang, Guangchun Luo, Ling Tian, and Wenyu Chen. 2020. Towards Clustering-friendly Representations: Subspace Clustering via Graph Filtering. In Proceedings of the 28th ACM International Conference on Multimedia. 3081–3089.
- [24] Zaiqiao Meng, Shangsong Liang, Hongyan Bao, and Xiangliang Zhang. 2019. Coembedding attributed networks. In Proceedings of the twelfth ACM international conference on web search and data mining. 393–401.
- [25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems. 3111–3119.
- [26] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. arXiv preprint arXiv:1905.09550 (2019).
- [27] Sankar K Pal and Sushmita Mitra. 1992. Multilayer perceptron, fuzzy sets, classifiaction. (1992).
- [28] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. 2016. Context encoders: Feature learning by inpainting. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2536–2544.
- [29] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. 701–710.

- [30] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2110–2119.
- [31] Afshin Rahimi, Trevor Cohn, and Timothy Baldwin. 2018. Semi-supervised User Geolocation via Graph Convolutional Networks. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2009–2019.
- [32] Sungmin Rhee, Seokjun Seo, and Sun Kim. 2018. Hybrid Approach of Relation Network and Localized Graph Convolutional Filtering for Breast Cancer Subtype Classification. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18. 3527–3534.
- [33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. AI magazine 29, 3 (2008), 93–93.
- [34] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 5892–5899.
- [35] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-Scale Information Network Embedding. In Proceedings of the 24th International Conference on World Wide Web. 1067–1077.
- [36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. International Conference on Learning Representations.
- [37] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax. In International Conference on Learning Representations.
- [38] Wenjun Wang, Xiao Liu, Pengfei Jiao, Xue Chen, and Di Jin. 2018. A Unified Weakly Supervised Framework for Community Detection and Semantic Matching. In Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, 218–230.
- [39] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous Graph Attention Network. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019.* ACM, 2022– 2032.
- [40] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020. ACM, 1243–1253.
- [41] Zhichun Wang, Qingsong Lv, Xiaohan Lan, and Yu Zhang. 2018. Cross-lingual Knowledge Graph Alignment via Graph Convolutional Networks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 349–357.
- [42] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. 6861–6871.
- [43] Jun Wu, Jingrui He, and Jiejun Xu. 2019. Net: Degree-specific graph neural networks for node and graph classification. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 406– 415.
- [44] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semisupervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.
- [45] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. In ICML Workshop on Graph Representation Learning and Beyond.