Semi-supervised Graph Learning with Few Labeled Nodes

Cong Zhang[1], Ting Bai[1,2](✉), and Bin Wu[1,2]

[1] Beijing University of Posts and Telecommunications, Beijing, China
{zhangc137,baiting,wubin}@bupt.edu.cn
[2] Beijing Key Laboratory of Intelligent Telecommunications Software and Multimedia, Beijing, China

Abstract. Graph-based semi-supervised learning, utilizing both a few labeled nodes and massive unlabeled nodes, has aroused extensive attention in the research community. However, for the graph with few labeled nodes, the performance of Graph Convolutional Networks (GCNs) will suffer from a catastrophic decline due to its intrinsic shallow architecture limitation and insufficient supervision signals. To accommodate this issue, we propose a novel Self-Training model (ST-LPGCN) which reinforces the pseudo label generation on the GCNs with Label Propagation algorithm (LPA). By making full use of the advantages of GCNs in aggregating the local node features and LPA in propagating the global label information, our ST-LPGCN improves the generalization performance of GCNs with few labeled nodes. Specifically, we design a pseudo label generator to pick out the nodes assigned with the same pseudo labels by GCN and LPA, and add them to the labeled data for the next self-training process. To reduce the error propagation of labels, we optimize the transition probability between nodes in LPA under the supervision of the pseudo labels. The extensive experimental results on four real-world datasets validate the superiority of ST-LPGCN for the node classification task with few labeled nodes.

Keywords: Graph neural networks · Label propagation · Few labels · Semi-supervised learning

1 Introduction

Recently, graph representation learning had been successfully applied into many research areas, such as social networks [8], physical process [18], knowledge graph [7], and biological networks [17]. However, due to the data privacy policy and the high cost of data annotation, the node labels on graphs are usually sparse in many real-world scenarios. To alleviate the lack of labeled data

This work is supported by the National Natural Science Foundation of China under Grant No. 62102038; the National Natural Science Foundation of China under Grant No. 61972047, the NSFC-General Technology Basic Research Joint Funds under Grant U1936220.

problem, graph-based semi-supervised learning (GSSL), which trains numerous unlabeled data along with a small amount of labeled data, has attracted increasing interests in the research community. And it had been successfully developed in various approaches, such as label propagation [36], graph regularization [35], graph autoencoder [13] and graph neural networks [8,14,25]. The success of GSSL depends on the simple assumption that the nearby nodes on a graph tend to have the same labels [21]. It spreads the labels of a few labeled nodes to the remaining massive unlabeled nodes according to the adjacent relationship within the graph to accurately classify the unlabeled nodes. The mainstream method Graph Convolutional Networks (GCNs) learning representations of each node by aggregating the information of its neighbors to facilitate the label assignments, have been demonstrated to significantly outperform the other classic GSSL methods [5,36].

Despite the noticeable progress of GCNs and the variants [9,14,31] in graph-based semi-supervised learning tasks, it still faces adverse conditions when the labeled data is extremely limited. The performance of GCNs would suffer from a catastrophic decline when the labeled data in GSSL is minimal. GCNs iteratively update each node's representation via aggregating the representations of its neighbors, and assign pseudo labels to the unlabeled nodes in the training process. GCNs with shallow architecture will restrict the efficient propagation of label signals, but when GCNs is equipped with many convolutional layers, it will suffer from the over-smoothing problem [15,30], resulting in the restriction of the performance of GCNs in the case of few labels. Some previous studies [12,15,22] expand the labeled data by assigning unlabeled nodes with pseudo labels or further use the unsupervised cluster information. Although they reduce the error information contained in pseudo labels, they can not effectively involve the global structure information, and the limited label signals can not be efficiently propagated to the entire graph.

To address this problem, we propose a novel Self-Training model (ST-LPGCN) which reinforces the pseudo label generation on the GCNs with Label Propagation algorithm (LPA), then generates pseudo labels to expand the training set (i.e., labeled nodes). In ST-LPGCN, we utilize GCNs to extract the node features and local structure information and assign the unlabeled nodes with pseudo labels. However, as described above, the shallow GCNs cannot propagate the label information through the entire graph. We further use LPA to spread the labels to the entire graph to reinforces the pseudo labels generation. By doing that, we not only solve the over smoothing problem of GCNs in propagating labels from high-order neighbors, but also incorporate both local and global semantic information, which is helpful to predict the pseudo labels of nodes. Specifically, to reduce the error propagation of labels, we learn the transition probability matrix in LPA under the supervision of the pseudo labels to optimize the transition probability between nodes. Then we carefully design a pseudo label generator to pick out the nodes assigned with the same pseudo labels by GCN and LPA, and add them to the labeled data for the next self-training process. Our contributions are summarized as follows:

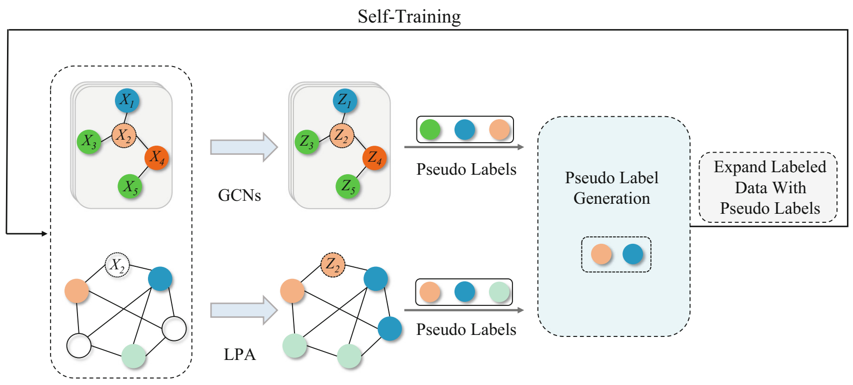

Fig. 1. An overall framework of our proposed ST-LPGCN. ST-LPGCN consists of three components: a GCN generates pseudo labels based on the embeddings learned from node features and local structure; a LPA component assigns unlabeled nodes with pseudo labels according to the global structure and label information; and the pseudo label generation component picks nodes which assigned with the same pseudo labels by GCN and LPA. The selected nodes are added to the labeled data for the next self-training process.

- We propose a novel self-training algorithm for the semi-supervised node classification tasks called ST-LPGCN, which reinforces the pseudo label generation on the GCNs with Label Propagation algorithm (LPA).
- To reduce the error propagation of labels, we optimize the transition probability between nodes in LPA under the supervision of the pseudo labels. The nodes assigned with the same pseudo labels by GCN and LPA are added to the labeled data for the next self-training process.
- We conduct extensive experiments on four benchmark graph datasets, and the experimental results demonstrate that our proposed ST-LPGCN outperforms the state-of-the-art baselines, especially in the cases with extremely few labeled data.

2 Related Works

Graph-Based Semi-supervised Learning. Graph-based semi-supervised learning (GSSL) has attracted much attention in recent years due to the wide range of applications, scalability to large-scale data, and promising performance. The early GSSL methods are mainly based on the cluster assumption that the nearby nodes on a graph tend to possess the same label. Researches along this line mainly consist of label propagation [36] and its variants such as Gaussian random fields [37], local and global consistence [34] and modified adsorption [23]. Another line is to predict the labels based on the embeddings learned from input graphs [32], such as factorization-based methods [1,3] and random-walk-based methods [5,24] which learn node embeddings based on the graph structure.

However, these methods fail to leverage the rich node features, which are significantly critical for GSSL. To jointly model the data features and graph structure, autoencoder-based methods have been proposed including SDNE [27], GAE and VGAE [13]. Later, with the introduction of GCN [14], the GNN-based methods become the dominant solution. GNNs take advantage of message passing based on the Laplacian smoothing assumption, in which messages are exchanged and updated between each pair of the nodes and capture more facts to make a more robust prediction on the unlabeled nodes. To improve the performance of GNNs in the case of few labels, Li [15] combined GCNs and self-training to expand supervision signals, M3S [22] proposed multi-stage self-training and utilized clustering method to eliminate the pseudo labels that may be incorrect. CGCN [12] further utilized an optimized graph clustering approach to strengthen the performance.

Graph Convolutional Networks. Inspired by the success of convolutional networks on grid-structure data, GCNs have been proposed to generalize the CNNs to graph-structure data and achieved state-of-the-art performance in many graph mining tasks. Generally, GCNs can be divided into spectral-based and spatial-based methods. Spectral-based approaches define the graph convolutions in the Fourier domain based on the graph signal processing. The general graph convolution framework is first proposed by Bruna [2] which needs to compute the Laplacian eigenvectors and then ChebyNet [4] using Chebyshev polynomials to optimize this method. Afterwards, Kipf [14] utilized the localized first-order approximation of spectral graph convolutions further simplified this model. Furthermore, to overcome time-consuming computation on approximated Laplace eigenvalues, spatial approaches define convolutions directly on the graph based on nodes' spatial relations. The graph convolution is defined as the weighted average function on the target node's neighbors such as GraphSAGE [8] and GAT [25]. Although GCNs and their variants have shown promising results, the performance of most existing GCNs will suffer from a catastrophic decline when the labeled data is limited. To address this problem, BGCN [33] incorporates a Bayesian method into GCN to get a more robust and generalized model for node classification. LCGNN [29] directly uses label information to reconstruct the aggregation matrix in GNNs to optimize the aggregation process. CG^3 [26] leverages contrastive learning and a hierarchical GCN model to capture nodes' information from local to global perspectives to learn better representations for the classification task.

Label Propagation Algorithm. Label propagation algorithm is a classic graph-based semi-supervised learning method that regards the labeled nodes as guides that lead the label information to propagate through the edges within the graph to assign the unlabeled nodes with predicted labels. Gaussian random fields [37] and local and global consistency [6] are the early typical work using label propagation. With the proposal of GNNs, several researches try to incorporate GNNs and LPA because they are both based on the message passing model. TPN [16] uses GNNs generating soft labels and propagates them by label propagation, GCN-LPA [28] uses the label propagation as the regularization for parameter matrix in GCN. UniMP [20] incorporates label information to the feature propagation process in GCN. Recent work Correct and Smooth [11] notes that the key to improve the performance is using labels directly, utilizes label propagation twice for correcting and smoothing and achieves performance comparable to GCNs.

3 The Proposed Model

In this section, we introduce our Self-Training framework (ST-LPGCN) which reinforces the pseudo label generation on the GCNs with Label Propagation algorithm. The overall architecture is shown in Fig. 1. Before introducing the ST-LPGCN, we provide the preliminary notations in this paper.

3.1 Preliminary Notations

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ is the set of nodes, $\mathcal{E}$ represents the set of edges. The feature vector of node v_i is denoted as $\mathbf{x}_i \in \mathbb{R}^{1\times d}$ and $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; ...; \mathbf{x}_n] \in \mathbb{R}^{n\times d}$ denotes the feature matrix of all nodes. $\mathbf{A} \in \{0, 1\}^{n\times n}$ denotes the adjacency matrix of the $\mathcal{G}$, in which each entry A_{ij} represents the state of connection between node v_i and v_j. $A_{ij} = 1$ indicates that there exists an edge between v_i and v_j; otherwise, $A_{ij} = 0$. The label of a node v_i is represented as a one-hot vector $\mathbf{y}_i \in \mathbb{R}^{1\times C}$, where C is the number of classes. Given a node v_i, our task is to predict the label of v_i. If $y_{ij} = 1$, the label of node v_i is $j \in C$.

3.2 Generating Pseudo Labels with GCN

GCN is a widely used message passing model for semi-supervised node classification. A GCN model usually contains multiple layers, and each layer aggregates the first-order neighbors' information and generate a low dimensional vector for each node. The simple message passing process can be formulated as follows [33]:

$$\begin{aligned} \mathbf{H}^{(1)} &= \sigma(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}), \\ \mathbf{H}^{(l+1)} &= \sigma(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \end{aligned} \tag{1}$$

where $\hat{\mathbf{A}}$ is the normalized adjacency matrix, defined as $\widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$. $\widetilde{\mathbf{A}}$ is the adjacent matrix with self-loops and $\widetilde{\mathbf{D}}$ is the degree matrix of $\widetilde{\mathbf{A}}$. $\mathbf{W}^{(l)}$ is the parameter matrix at layer l and $\mathbf{H}^{(l)}$ is the output features matrix from layer $l-1$. σ is a non-linear activation function.

We adopted the vanilla GCN [14] with two convolutional layers in our paper, formulated as:

$$\mathbf{Z} = softmax(\hat{\mathbf{A}}(ReLU(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)}))\mathbf{W}^{(1)}), \tag{2}$$

where $\mathbf{W}^{(0)} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times c}$ are the parameter matrices, and h is the dimension of the hidden layer. The non-linear activation function in first layer is rectified linear unit (ReLU). $\mathbf{Z} \in \mathbb{R}^{n \times c}$ is a probability distribution, where each row represents the probabilities that the node belongs to the corresponding labels.

The GCN model is optimized by minimizing the cross-entropy loss function:

$$\mathcal{L}_{gcn} = -\sum_{i \in L} \sum_{c=1}^{C} Y_{ic} \ln Z_{ic}, \tag{3}$$

where L is the set of the labeled nodes and C is the number of the labels.

The node is finally classified into the class with the maximal probability, and the pseudo labels assigned by GCN is $\hat{\mathbf{Z}}$. Following [15], we select top-t confident nodes in each class and assign the pseudo labels to them. Instead of adding them to the labeled data directly, we feed the pseudo labels generated by GCN into pseudo label generation component for further selection.

3.3 Generating Pseudo Labels with LPA

Traditional LPA. Label propagation algorithm (LPA) is a classic GSSL approach, which propagates the node labels according to the similarity between nodes. Given an initial label matrix $\mathbf{Y}^{(0)}$, which consists of one-hot label indicator vectors $y_i^{(0)}$ for the few labeled nodes and zero vectors for the unlabeled ones. The process of traditional LPA can be formulated as following:

$$\hat{\mathbf{Y}}^{(k+1)} = \mathbf{D}^{-1}\mathbf{A}\hat{\mathbf{Y}}^{(k)}, \tag{4}$$

where $\mathbf{A}$ and $\mathbf{D}$ are the adjacent matrix and its degree matrix respectively. The labeled nodes propagate labels to their neighbors according to the normalized adjacent matrix $\mathbf{D}^{-1}\mathbf{A}$, namely transition probability matrix. The greater the similarity between each node and its neighbors, the greater the probability propagated from the label of its neighbors.

However, in the above equation, the fixed transition probability of labels in LPA will lead to the avalanche effect that the initial errors will be magnified with the iterative process, which would have a catastrophic impact on the predictions. In addition, the connection of nodes from different classes in the graph may be a noise for LPA, which would further degrade the performance of LPA. To reduce the error propagation of labels, we optimize the traditional LPA by making the transition probability matrix learnable under the supervision of the pseudo labels.

Trainable LPA. To propagate the label information more correctly, we propose a trainable LPA method to learn a optimized transition probability matrix where the nodes in the same class connect more strongly, while the nodes from different classes connect weakly. The trainable LPA can be formulated as:

$$\hat{\mathbf{Y}}^{(k+1)} = \mathbf{D}^{-1}\mathbf{A}\mathbf{W}^{(k)}\hat{\mathbf{Y}}^{(k)}, \tag{5}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^{n\times n}$ is the parameter matrix at the k-th iteration. The trainable LPA are optimized by cross-entropy loss function:

$$\mathcal{L}_{lpa} = -\sum_{i\in L}\sum_{c=1}^{C} Y_{ic} \ln \hat{Y}_{ic}, \tag{6}$$

where the $\hat{Y}_{ic}$ is the predicted label by LPA.

In this way, the transition probability matrix can be corrected by the pseudo labels, which alleviates the errors magnification problem in the iterative process. We select top-t confident nodes in each class and assign the pseudo labels to them. The same as the pseudo labels in GCN, we feed the pseudo labels generated by trainable LPA into the pseudo label generation component for further selection.

```
Algorithm 1: ST-LPGCN Algorithm
   Input: Adjacent matrix $\mathbf{A}$, feature matrix $\mathbf{X}$, label matrix $\mathbf{Y}$, initial labeled
          and unlabeled set $Y_0$, $U_0$.
   Output: Predicted labels for each unlabeled node
 1 Random initialize the parameter matrices of GCN and trainable LPA;
 2 for $k = 1, 2, ..., K$ do
 3     Initial the pseudo-label sets $\hat{Z}_{gcn} = \emptyset$ and $\hat{Y}_{lpa} = \emptyset$;
 4     Train GCN and LPA on the labeled set $Y_{k-1}$ to obtain the predictions $\hat{Z}$
         and $\hat{Y}$;
 5     Sort nodes according to the confidence scores in the unlabeled set $U_{k-1}$;
 6     for each class $c$ do
 7         Select the top $t$ nodes from $\hat{Z}$ and $\hat{Y}$, and add them to $\hat{Y}_{gcn}$ and $\hat{Y}_{lpa}$
             respectively;
 8         for each node do
 9             if the label of node $v$ are same in both $\hat{Y}_{gcn}$ and $\hat{Y}_{lpa}$ then
10                 Add it to the labeled data $Y_{k-1}$ with pseudo label $c$;
11                 Delete it from the unlabeled set $U_{k-1}$;
12     Train the GCN and LPA on the new labeled data $Y_k$;
13 Conduct label prediction based on the final trained GCN;
```

3.4 Pseudo Label Generation

The core of self-training lies in the accuracy of pseudo labels assigned on the nodes. It is important to generate reliable labels to avoid error propagation. In this component, we use the pseudo labels from GCN and LPA components for mutual checking. The GCN pseudo labels are generated based on the nodes' feature similarities in the feature space. While the LPA pseudo labels are generated by using the label information and the global structure information, which makes up for the limitations of the shallow GCN.

Specifically, given a node, we check their pseudo-labels generated in the GCN and trainable LPA. If the labels are the same, we assign the label of the node to the labeled data for the next self-training process; otherwise, we remove them to diminish potential error information. Given a node v which had been generated pseudo labels in GCN and LPA, the reliable pseudo label can be generated by:

$$\hat{\mathbf{Y}}_{final}(v) = \left\{ v | \hat{\mathbf{Z}}_{gcn}(v) = \hat{\mathbf{Y}}_{lpa}(v) \right\}, \tag{7}$$

where $\hat{\mathbf{Z}}_{gcn}(v)$, $\hat{\mathbf{Y}}_{lpa}(v)$ denote the pseudo labels of node v generated by GCN and LPA respectively. $\hat{\mathbf{Y}}_{final}(v)$ is the final pseudo label of v that will be added to the labeled data for next self-training process.

3.5 Self-training Process

Following M3S [22], we repeat K times of the self-training process to provides more supervision information from the learnable pseudo labels. The final loss function to optimized is:

$$\mathcal{L} = \mathcal{L}_{gcn} + \lambda \mathcal{L}_{lpa}, \tag{8}$$

where λ is the trade-off hyper-parameter. The GCN and LPA component are trained simultaneously (see in Algorithm 1).

4 Experiments

4.1 Experiment Setup

Datasets. The four widely used benchmark datasets are Cora, CiteSeer, Pubmed [19] and ogbn-arxiv [10]. In these datasets, each node represents a document and has a feature vector. The label of each node represents the topic that it belongs to. The statistics of the datasets are shown in Table 1.

Baseline Methods. In the experiments, we compare our method with the state-of-the-art semi-supervised node classification methods, including:

- **LPA** [36]: Label propagation is a classical semi-supervised learning algorithm. It iteratively assigns labels to the unlabeled nodes by propagating the labels through the graph.

Table 1. Statistics of the datasets.

Dataset	Nodes	Edges	Classes	Features
CiteSeer	3,327	4,732	6	3,703
Cora	2,708	5,429	7	1,433
PubMed	19,717	44,338	3	500
ogbn-arxiv	169,343	1,166,243	40	128

- **GCN** [14]: GCN is a widely used graph neural network, and it learns node representations based on the first-order approximation of spectral graph convolutions.
- **GAT** [25]: GAT leverages attention mechanism to improve the model performance for node classification task.
- **Union and Intersection** [15]: Two methods **Co-training** and **Self-training** are proposed in [15] for semi-supervised node classification. Co-training uses ParWalk to select the most confident nodes and add their pseudo labels to the labeled data to train GCN. While **Self-Training** selects the pseudo node labels from GCN to boost the model performance. **Union and Intersection** takes the union or intersection of the pseudo labels generated by Co-training and Self-training as the additional supervision information.
- **M3S** [22]: It leverages the deep cluster to check the pseudo labels assigned by GCN and add to the labeled data for self-training on GCN.
- **GCN-LPA** [28]: It learns the optimal edge weights by LPA, which are leveraged as the regularization in GCN to separate different node classes.
- **LCGNN** [29]: The LCGCN and LCGAT use label information to reconstruct the aggregation matrix based on the label-consistency for GCN and GAT, which alleviate noise from connected nodes with different labels.
- **BGCN** [33]: It combines Bayesian approach with the GCN model, and views the observed graph as a realization from a parametric family of random graphs.
- **CGCN** [12]: It generates pseudo labels by combining variational graph auto-encoder with Gaussian mixture models, which can be used to boost the performance of semi-supervised learning.
- **CG^3** [26]: It leverages the contrastive learning on different views of graph and captures the similarity and graph topology information to facilitate the node classification task.

The above methods cover different kinds of approaches in graph-based semi-supervised node classification task. LPA is a classic GSSL method that only propagates the label signals to the entire graph directly. GCN and GAT are the widely used GNN-based model which aggregates node features and local structure. GCN-LPA, LCGNN and CG^3 further leverage the label information or global structure information to improve the performance of GCNs. However, they cannot be fully trained with few labeled nodes, which leads to the under-performance. Self-Training model, M3S and CGCN all utilize the self-training process based on the GCNs to expand the labeled data with pseudo labels.

The most similar work to our method is M3S. It uses the unsupervised method cluster to diminish the potential errors in pseudo label generation. The differences between our model and M3S lie in: (1) M3S does not leverage the global structure and label information of graph; (2) M3S needs more layers to incorporate the useful features on GCN which is time-consuming and would lead to the over-smoothing problem of GCN. Our ST-LPGCN leverages a LPA to make full use of the global structure information and propagate the label information to enrich the pseudo label signals. The trainable LPA enables ST-LPGCN to alleviate the potential errors in label propagation simultaneously, which would help to improve the accuracy of pseudo labels assigned on the nodes in the self-training process.

Parameters Settings. We follow the experimental settings in M3S [22] for fair comparation. We conducted experiments with different label rates, i.e., 0.5%, 1%, 2%, 3%, 4% on Cora and CiteSeer, 0.03%, 0.05%, 0.1% on PubMed and 1%, 2%, 5%, 10% on ogbn-arxiv dataset respectively. The layers of GCN is 3, 3, 2, 2 and the number of propagation times in LPA is 3 in Cora, CiteSeer, PubMed and ogbn-arxiv datasets respectively in each self-training stage. For the top-t most confident pseudo labels, t is set to 60 in GCN and LPA in all datasets. The times K repeated in self-training process are 3, 4, 5 in CiteSeer, PubMed and ogbn-arxiv datasets. While for Cora dataset, it varies depending on the label rates, which is set to 5, 4, 4, 2, 2. For each baseline method, grid search is applied to find the optimal settings, and all the results are the mean accuracy of 10 runs on all datasets. The hyperparameters of GCN are set as follows: the learning rate is 0.002, dropout rate is 0.6, L2 regularization weight is 5×10^{-4}, and the dimension of hidden layers in GCNs is 16. For other baseline methods, we adopt their public code and tune hyperparameters for the best performance. The trade-off hyperparameter λ in Eq. 8 is set to 1.

4.2 Result Analysis

We compute the accuracy of node classification task on different methods. The results on four datasets with the different label rates are shown in the Table 2 and 3. For the baseline methods which are restricted by the memory size to deal with the large-scale dataset ogbn-arxiv, comparisons are only carried out on Cora, CiteSeer and Pubmed datasets. We have the following observations:

(1) LPA performs worst on CiteSeer and ogbn-arxiv datasets. Although it can propagate the global label information through the entire graph to obtain relatively strong supervision information, without the node feature information, it can not achieve good performance due to the lack of using node semantic information.
(2) GCN and GAT perform worst on Cora and PubMed datasets. They aggregate the local node features and structure information, but they can not provide sufficient supervision information for model training with few labels, especially when the labeled data is extremely scarce.

Table 2. Node classification accuracy on Cora.

Label rate	Cora					CiteSeer					PubMed		
	0.5%	1%	2%	3%	4%	0.5%	1%	2%	3%	4%	0.03%	0.05%	0.1%
LPA	56.8	62.0	64.2	66.3	69.6	39.6	42.2	44.0	44.6	45.1	57.6	62.0	64.5
GCN	50.1	60.3	69.8	75.5	76.7	44.7	54.2	62.3	68.0	69.5	50.9	58.2	68.1
GAT	50.3	61.2	70.6	76.4	77.1	45.3	56.4	61.9	69.6	71.2	52.1	59.8	70.4
Co-training	55.1	61.3	70.1	74.9	76.8	45.6	54.0	59.6	63.5	64.8	58.2	65.7	69.8
Self-training	56.6	62.4	71.7	76.8	77.1	43.6	57.5	64.2	67.2	68.5	56.7	65.5	70.1
Union	57.0	67.6	74.5	77.1	78.7	47.0	59.3	63.1	65.9	66.2	57.1	65.2	68.8
Intersection	51.3	63.2	71.1	76.2	77.3	43.3	60.4	65.8	69.8	70.2	56.0	59.9	68.0
GCN-LPA	52.5	66.9	72.0	72.9	74.3	42.1	50.3	59.4	67.8	69.0	63.6	64.4	69.5
BGCN	56.7	69.8	74.8	76.9	78.1	52.0	58.6	68.6	70.8	72.3	63.4	66.2	70.2
M3S	61.5	67.2	75.6	77.8	78.0	56.1	62.1	66.4	70.3	70.5	59.2	64.4	70.6
LCGCN	69.7	73.4	75.6	79.1	80.0	61.1	69.1	70.3	71.1	72.9	68.8	69.4	75.9
LCGAT	70.9	75.8	77.1	79.6	81.3	62.3	68.9	70.4	71.3	72.2	68.6	69.0	75.7
CGCN	64.3	72.4	76.8	79.8	81.3	59.3	63.1	69.5	72.4	72.7	64.7	69.2	77.8
CG^3	69.3	74.1	76.6	79.9	81.4	62.7	70.6	70.9	71.3	72.5	68.3	70.1	73.2
ST-LPGCN	**75.9**	**78.0**	**80.3**	**81.0**	**81.6**	**65.2**	**70.7**	**71.3**	**72.6**	**73.4**	**75.8**	**77.6**	**80.1**

(3) The performance of Self-training, Co-Training, Union and Intersection is not consistent through different datasets and label rates. The variant models with self-training process achieve better performance, showing the usefulness of leveraging the pseudo labels in model training.

(4) M3S and CGCN with self-training outperform the algorithm without self-training such as GCN-LPA and BGCN on Cora, CiteSeer and PubMed datasets, which also indicates that the self-training can provide more supervision information to improve the model performance. Besides, M3S and CGCN utilize the clustering method to reduce the incorrect label information, which leads to better performance. It indicates that reducing the propagation error of pseudo labels can improve the model performance in self-training process.

Table 3. Node classification accuracy on ogbn-arxiv.

Label rate	1%	2%	5%	10%
LPA	51.0	56.2	62.0	65.6
GCN	60.7	63.3	65.0	65.8
Self-training	62.2	63.9	66.1	66.5
M3S	63.1	64.3	66.9	68.6
ST-LPGCN	**64.2**	**65.8**	**67.9**	**69.2**

(5) LCGCN, LCGAT and CG^3 outperforms M3S and CGCN on Cora, CiteSeer and PubMed datasets. LCGCN (LCGAT) directly uses the label information

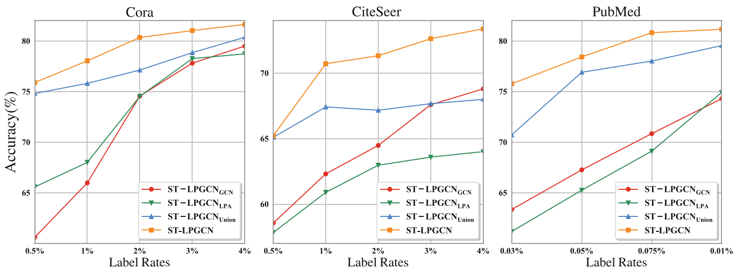

Fig. 2. Accuracy of ST-LPGCN on different datasets using different pseudo label sets.

to reconstruct the aggregation matrix and then train the GCN (GAT) by aggregating the node feature information. These approaches further use label information to supervise the aggregation process of GCN (GAT), which achieves better performance compared with the model learning from pseudo labels. CG^3 designs a hierarchical GCN model to capture nodes' information from local to global perspectives. It incorporates the global structure information to learn a powerful node representation for the classification task.

(6) ST-LPGCN outperforms the state-of-the-art baselines on all datasets with different label rates, verifying the effectiveness of our proposed method. The trainable LPA in ST-LPGCN makes it possible to capture the global structure and label information, and meanwhile alleviate the error propagation of the pseudo labels in self-training process.

(7) As the number of labels decreasing, the performance of all methods becomes worse on all datasets. However, the improvements of accuracy of our ST-LPGCN becomes increasing compared with the state-of-the-art baseline, showing the effectiveness of our model to deal with the data with extremely few labels.

4.3 Ablation Study

In this section, we first make ablation studies to demonstrate the effectiveness of our pseudo label generation strategy and the trainable LPA in our paper. Then we analyze the impact of the amount of pseudo labels added in the self-training.

Pseudo Label Generation Strategy. In the pseudo label generation, we select the nodes with the same pseudo labels generated by the GCN and LPA, and add them to the training set for the next training process. To verify the effectiveness of the pseudo label generation strategy, we adopt the following strategies to make comparing.

- ST-LPGCN$_{GCN}$: the pseudo labels generated by the GCN.
- ST-LPGCN$_{LPA}$: the pseudo labels generated by the LPA.
- ST-LPGCN$_{Union}$: the pseudo labels are generated by adding all pseudo labels in GCN or LPA

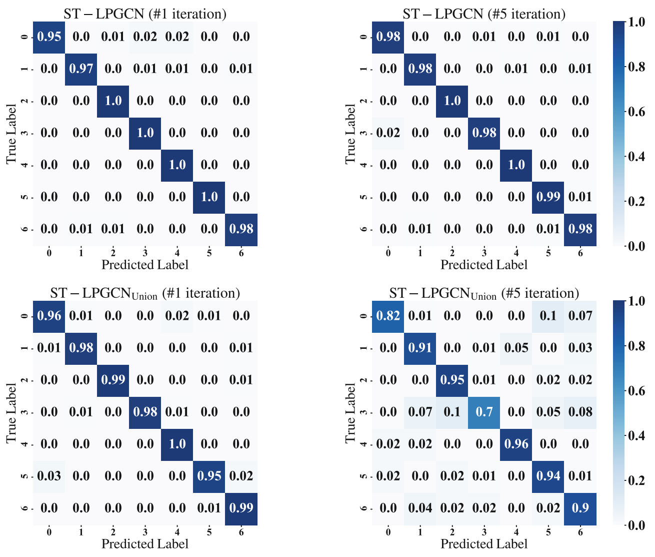

Fig. 3. Confusion matrix (the reliability of pseudo label prediction compared with true label) for ST-LPGCN and ST-LPGCN$_{Union}$ in different iterations. The larger diagonal value is, the more reliable of the predicted pseudo label. The left column represents the first iteration and the right column represents the fifth iteration of ST-LPGCN and ST-LPGCN$_{Union}$ respectively.

As shown in Fig. 2, ST-LPGCN$_{Union}$ obtains better performance than ST-LPGCN$_{GCN}$ and ST-LPGCN$_{LPA}$ due to that it uses additional label (node features) information. Our ST-LPGCN achieves the best performance, showing the effectiveness of our generation strategy to select more reliable pseudo labels of nodes, so as to alleviate the error propagation in self-training process.

As shown in Fig. 3, we further analyze the error propagation of pseudo labels in the training process, and visualize the confidence scores of pseudo labels compared with the true labels at the first iteration and the fifth iteration. We can see that the decreasing of confidence scores is more in the first iteration in ST-LPGCN$_{Union}$. And the error of pseudo labels is magnified as the iteration, leading to the large gap in the fifth iteration compared with the pseudo label generation in ST-LPGCN.

Effectiveness of Trainable LPA. The trainable LPA designed in ST-LPGCN can assign higher transition probability to the node labels belonging to the same class and reduce the probability of node labels in different classes, which may reduce the propagation of error information from the possible noise in the adjacency matrix itself. To verify this, we conducted experiments to analyze the effectiveness of the trainable transition probability matrix in LPA. We keep the same settings and change the pseudo labels generated by traditional LPA for a fair comparison. The results are reported in Table 4. We find that the trainable

LPA consistently outperforms the traditional LPA on two representative datasets with all label rates, showing the effectiveness of trainable LPA in ST-LPGCN.

Table 4. Effect of the trainable LPA in ST-LPGCN.

Dataset	Method	0.5%	1%	2%	3%	4%
Cora	Traditional LPA	72.6	74.4	77.6	79.3	80.1
	Trainable LPA	75.9	78.0	80.3	81.0	81.6
CiteSeer	Traditional LPA	61.4	67.8	69.5	70.1	71.2
	Trainable LPA	65.2	70.7	71.3	72.6	73.4

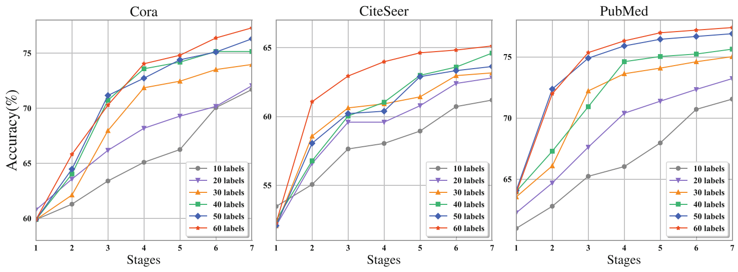

Fig. 4. Sensitivity analysis of the model to the number of pseudo labels in the training process.

Sensitivity of the Number of Pseudo Labels. In this subsection, we explore the sensitivity of the number of pseudo labels added to the labeled data. We conducted experiments taking different numbers of pseudo labels in each self-training process with a label rate of 0.5% on all datasets. As shown in Fig. 4, at the beginning of the training process, the more pseudo-labels generated during each training process, the better the performance of our model. As the training iteration increasing, the performance will approach stabilization with different numbers of pseudo labels. The performance of our model gains slight improvements when a certain number of pseudo labels are added to training data. Based on this observation, we select 60 pseudo labels to expand our training data in experiments.

5 Conclusion

Graph-based semi-supervised learning is a hot topic, but there is relatively little work focusing on semi-supervised learning tasks when the labeled data is quite few, leading to a significant decline in the performance of many existing approaches. In this paper, we propose a novel Self-Training model (ST-LPGCN) which reinforces the pseudo label generation on the GCNs with Label Propagation algorithm. Our ST-LPGCN improves the effect of GCNs in propagating labels from high-order neighbors with shallow architecture, and incorporates both local and global semantic information, which is helpful to predict the pseudo labels of nodes. In the future work, we will further verify the effectiveness of ST-LPGCN in other graph learning tasks, for example, link prediction and so on.

References

1. Ahmed, A., Shervashidze, N., Narayanamurthy, S.M., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: WWW, pp. 37–48 (2013)
2. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs. In: ICLR (2014)
3. Cao, S., Lu, W., Xu, Q.: GraRep: learning graph representations with global structural information. In: CIKM, pp. 891–900 (2015)
4. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: NIPS, pp. 3837–3845 (2016)
5. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
6. Gui, J., Hu, R., Zhao, Z., Jia, W.: Semi-supervised learning with local and global consistency. Int. J. Comput. Math. **91**(11), 2389–2402 (2014)
7. Hamaguchi, T., Oiwa, H., Shimbo, M., Matsumoto, Y.: Knowledge transfer for out-of-knowledge-base entities: a graph neural network approach. In: IJCAI, pp. 1802–1808 (2017)
8. Hamilton, W.L., Ying, R., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1025–1035 (2017)
9. Hong, H., Guo, H., Lin, Y., Yang, X., Li, Z., Ye, J.: An attention-based graph neural network for heterogeneous structural learning. In: AAAI 2020, pp. 4132–4139 (2020)
10. Hu, W., et al.: Open graph benchmark: datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687 (2020)
11. Huang, Q., He, H., Singh, A., Lim, S.N., Benson, A.R.: Combining label propagation and simple models out-performs graph neural networks. arXiv preprint arXiv:2010.13993 (2020)
12. Hui, B., Zhu, P., Hu, Q.: Collaborative graph convolutional networks: unsupervised learning meets semi-supervised learning. In: AAAI 2020, pp. 4215–4222 (2020)
13. Kipf, T.N., Welling, M.: Variational graph auto-encoders. arXiv preprint arXiv:1611.07308 (2016)
14. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
15. Li, Q., Han, Z., Wu, X.: Deeper insights into graph convolutional networks for semi-supervised learning. In: AAAI, pp. 3538–3545 (2018)
16. Liu, Y., et al.: Learning to propagate labels: transductive propagation network for few-shot learning. In: ICLR (2019)
17. Pavlopoulos, G.A.: Using graph theory to analyze biological networks. BioData Min. **4**, 10 (2011)
18. Sanchez-Gonzalez, A., et al.: Graph networks as learnable physics engines for inference and control. In: ICML, pp. 4467–4476 (2018)

19. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Mag. **29**(3), 93–106 (2008)
20. Shi, Y., Huang, Z., Wang, W., Zhong, H., Feng, S., Sun, Y.: Masked label prediction: unified message passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509 (2020)
21. Song, Z., Yang, X., Xu, Z., King, I.: Graph-based semi-supervised learning: a comprehensive review. arXiv preprint arXiv:2102.13303 (2021)
22. Sun, K., Lin, Z., Zhu, Z.: Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In: AAAI, pp. 5892–5899 (2020)
23. Talukdar, P.P., Crammer, K.: New regularized algorithms for transductive learning. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009. LNCS (LNAI), vol. 5782, pp. 442–457. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04174-7_29
24. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
25. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: ICLR (2018)
26. Wan, S., Pan, S., Yang, J., Gong, C.: Contrastive and generative graph convolutional networks for graph-based semi-supervised learning. In: AAAI, vol. 35, pp. 10049–10057 (2021)
27. Wang, D., Cui, P., Zhu, W.: Structural deep network embedding. In: KDD, pp. 1225–1234 (2016)
28. Wang, H., Leskovec, J.: Unifying graph convolutional neural networks and label propagation. arXiv preprint arXiv:2002.06755 (2020)
29. Xu, B., Huang, J., Hou, L., Shen, H., Gao, J., Cheng, X.: Label-consistency based graph neural networks for semi-supervised node classification. In: SIGIR, pp. 1897–1900 (2020)
30. Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., Jegelka, S.: Representation learning on graphs with jumping knowledge networks. In: ICML, pp. 5449–5458 (2018)
31. Xu, N., Wang, P., Chen, L., Tao, J., Zhao, J.: MR-GNN: multi-resolution and dual graph neural network for predicting structured entity interactions. In: IJCAI, pp. 3968–3974 (2019)
32. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML, pp. 40–48. PMLR (2016)
33. Zhang, Y., Pal, S., Coates, M., Üstebay, D.: Bayesian graph convolutional neural networks for semi-supervised classification. In: AAAI 2019, pp. 5829–5836 (2019)
34. Zhou, D., Bousquet, O., Lal, T.N., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS, pp. 321–328 (2004)
35. Zhou, D., Huang, J., Schölkopf, B.: Learning from labeled and unlabeled data on a directed graph. In: ICML, vol. 119, pp. 1036–1043 (2005)
36. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation (2003)
37. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using Gaussian fields and harmonic functions. In: ICML, pp. 912–919 (2003)