SELF-TRAINING GCN WITH FEATURE AGGREGATION CHECKING PART

Anonymous authors
Paper under double-blind review

ABSTRACT

Recently, Graph Convolutioal Networks (GCNs) have achieved significant success in processing graph learning task. Due to its excellent ability in representation learning, Graph Convulutional networks are widely used for various tasks such as node classification, graph classification, link prediction in CV and NLP. Nevertheless, it remains difficult to use the existing GCN models to obtain good prediction, especially when the training set is quite small. In this paper, we propose a novel graph convolutional training algorithm called Self-training GCN with Feature aggregation Checking part (SGFC). The algorithm is inspired by self-supervision strategy, which introduces a checking part to add new supervised information after training some epochs and adopts multi-round strategy to make the prediction much better. In this paper, we illustrate our training algorithm and elaborate the whole architecture of our framework. Extensive experiments show that our method can achieve state-of-the-art performance on a wide range of datasets in node classification tasks compared with other methods.

1 INTRODUCTION

Graph is a kind of data structure that expresses the relationships (abstracted as edges) between entities (abstracted as nodes). Many scenarios in real life can be represented in form of graphs, such as citation networks, social networks, transportation networks, molecular structures, etc (You et al., 2020). Recently, graph representation learning has attracted great attention. Lots of studies are focusing on graph representation learning based on deep learning. Significant progress has been made in many tasks based on graph representation,such as node classification, graph classification, link prediction and node clustering (Dwivedi et al., 2020).

Graph Neural Network (GNN) is a very representative method of graph representation learning. It can learn a representation vector for each node in graph, which can be used in subsequent classification or regression. Graph Covolutional Network (GCN) is an important part of GNN. It can be seen as the migration of Convolutional Neural Network on non-Euclidean structure data. Kipf & Welling (2016) proposed GCN from spectrogram theory and validated its good performance on semi-supervised node classification task by experiments. Subsequent models such as GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2017), SGCN (Wu et al., 2019) and APPNP (Klicpera et al., 2018) designed more sophisticated neighborhood aggregation function from perspective of spatial or spectral domain. These variants further improve GNN's ability of representation and obtain much more effective results on semi-supervised node classification compared with traditional methods such as MLP, DeepWalk (Perozzi et al., 2014), etc. However, the prediction accuracy of GCN largely depends on the quantity and quality of supervised information. When the quantity of labeled nodes is quite small, the classification accuracy of GCN will significantly reduce (Li et al., 2018), which mainly because scarce supervised information is difficult to spread widely in the graph so that other nodes cannot make full use of supervised information for prediction. Therefore, many studies pay more attention to further improve the expressive ability of GCN to obtain better graph reasoning ability, especially when the labeled nodes are quite scarce. Kipf & Welling (2016) has validated that we cannot improve GCN's representation ability by simply stacking layers just like MLP. Because as for MLP, it has been proved that any continuous function can be approximated if MLP has enough depth in layers. Therefore, we don't need to worry about expressive ability when designing deep neural networks. When it comes to GNN, however, Xu et al. (2018) proposed that massage-passing based GNN has an upper bound on graph-level task. The WL

isomorphism algorithm (Weisfeiler & Lehman, 1968) is the upper bound of this problem. For node-level task, it seems that GNN can achieve the effect of a universal approximator. However, stacking too many layers tends to cause over-smoothing (Xu et al., 2018), which makes all node embeddings distinguished. Several studies are devoted to making GNN deeper by some special tricks which can suppress the bad effects of over-smoothing to a certain extent (Li et al., 2019), but how to design a perfect multi-layer GNN remains a problem. Apart from trying to improve the representation ability of the model, on the other hand, Li et al. (2018) proposed to apply co-training and self-training techniques on GNN training. In fact, it's wise to make use of self-training mechanism which don't need to enhance model's expressive ability but just add new nodes to the training set so that the nodes with high confidence in the unlabeled set are used to expand the supervised information of GNN training (Li et al., 2018). This method can improve classification accuracy, especially when there are only few nodes. However, how to design the specific method of co-training and self-training so that the improved algorithm can be effective on a wide range of datasets remains a problem. Sun et al. (2019) proposed a multi-stage self-training strategy (M3S) to increase the training set, which uses Deep Cluster (Caron et al., 2018) and an aligning mechanism to generate pseudo-labels. Since Deep Cluster only uses k-means to cluster GCN embeddings, the clustering result does not fully consider each node's structural feature, instead, it only apply deep clustering on embedding and rely on distance measures to align, without considering using more appropriate attributed graph clustering approach. Experimental results show that the effect of M3S drops as the number of labeled nodes increases. Zhou et al. (2019) proposed a dynamic self-training framework (DSGCN) to continuously increase the training set, but it directly uses the output of GCN as confidence to expand supervised information without a checking part, which is somewhat unstable. Therefore, the improvement of the prediction accuracy is not obvious. In summary, above work didn't take full advantage of self-training mechanism, and the method they adopted to expand training set didn't make full use of node features and structural information, which makes the prediction accuracy relatively limited

Based on above work, we firstly analyzes the problems of the previous self-training graph convolutional networks, and then propose our method called SGFC, a novel training algorithm for dealing with semi-supervised classification. The main difference between our framework and previos self-training work is the design of self-training mechanism. We use multi-round for training, and adopt feature aggregation and a linear SVM classifier (Cortes & Vapnik, 1995) to serve as checking part, which is different from previous studies. This framework improves the performance of node classification by continuously adding useful training information without improving the GCN's expressive ability. We also generalize the framework to different GNN models. Finally, we conduct related experiments to show the effectiveness of our framework.

The main contributions of this paper are summarized as follows:

- We analyze the problems in existing GCN models and self-training based GCN frameworks. Then we propose a novel self-training framework SGFC for semi-supervised node classification task.
- We extend the framework to a variety of GNNs, and show the scalability of the framework through experiments so that we can choose feasible graph convolutional strategies for classification on different datasets.
- Extensive experimental results show that the proposed framework achieves state-of-the-art results.We analyze the reason of the improvement in prediction.

2 Related work

2.1 Graph neural network

Convolutional Neural Network has made great achievement in the processing of images (Krizhevsky et al., 2012; LeCun et al., 1998; Sermanet et al., 2012), but CNN is only good at dealing with Euclidean Structure Data (Bruna et al., 2013). Therefore, in recent years more and more studies have focused on studying how to learn the representation on non-Euclidean Structure Data such as graph. GNN is an important work in the field of graph representation learning which can learn the representation of node or the whole graph. GNN mainly include Graph Convolutional Network (Kipf & Welling, 2016), Graph Recurrent Neural Network (Hajiramezanali et al., 2019), Graph Autoencoder (Pan et al., 2018). As one of the most important part of GNN, Graph Convolutional Networks can

be categorized into spectral and spatial approaches (Bruna et al., 2013). The spectral approaches define the convolution operation by the Laplacian feature decomposition of the graph, thereby filtering the graph structure in the spectral domain (Bruna et al., 2013). Defferrard et al. (2016) uses the Chebyshev polynomial of the graph Laplacian matrix to approximate the spectral domain filter. On this basis, a simpler approach GCN was proposed by Kipf & Welling (2016) which limits the filter to the first-order neighborhood around each node. By contrast, the spatial approaches implement convolution in spatial domain by defining aggregate function and transform function. GraphSAGE is a representative work to take the representation learning into a formal pattern called aggregation and combination. It also proposed some effective aggregation strtegies such as mean-aggregator and max-aggregator. GAT focuses on the diversity in connected nodes and leverages self-attention mechanism to learn the important information in neighborhoods. Althouth these models have achieved far better performance on node classification than those traditional methods which learn embeddings for each node firstly and then classify them using a classifier, however, the limitations on GCN layers make it impossible to transform the supervised information to the entire graph.

2.2 Self-training

Self-training is an ancient topic in NLP field before deep learning era. Hearst (1991); Yarowsky (1995) used self-training for word sense disambiguation. Riloff et al. (1999) used self-training for information extraction and learning subjective nonus. In more recent studies, Zhou et al. (2012); Rosenberg et al. (2005) used self-training for object recognition. Sagae (2010); McClosky et al. (2006); Huang & Harper (2009); McClosky et al. (2008) showed self-training's excellent effectiveness in parsing. Huang et al. (2009) applied self-training to part of speech tagging. Kozareva et al. (2005) introduced self-training mechanism to named entity recognition. Besides, self-training was also used in sentiment classification (Drury et al., 2011; Van Asch & Daelemans, 2016).

As for the self-training algorithm, Chen et al. (2011) pointed that it may not work well if we always select those highly confident samples with a pre-defined threshold. McClosky et al. (2006) proposed a ranked list of n-best predicted parses and chose the best one. Rosenberg et al. (2005) showed that compared with a selection metric based on the detection confidence generated by the detector, a training data selection metric that is define independently of the detector can perform better. Zhou et al. (2012) pointed that it can significantly improve the performance over standard self-training framework if we select more informative unlabeled data using a guided search algorithm. What's more, Levatić et al. (2017) proposed an algorithm to automatically select appropriate threshold.

Li et al. (2018) firstly proposed to improve GCN model by self-training mechanism and validated its improvement by experiments. Self-training is a feasible approach to combine with semi-supervision, which is natural and general. In node classification task, assuming we have a basic model or algorithm for this problem, this model can use labeled samples to predict unlabeled ones and return a related margin or confidence score. The self-training framework trains and applies base model in rounds. At the end of each round, some of the unlabeled samples which have a high predicted confidence in one class will be added to training set for the training of next round. The training set continuously expands. The new added nodes are beneficial to predict rest nodes,which can effectively improve model's performance without enhancing model's ability of representation. The M3S training algorithm proposed by Sun et al. (2019) pretrains a model over the labeled data, then assigns "pseudo-labels" to highly confident unlabeled samples, and includes them into the labeled data for the next round of training. In this pipeline, self -supervision is injected to align and refine the pseudo-labels for the unlabeled nodes. Yang et al. (2020b) proposed Self-Enhanced GNN (SEG), whose main idea is to improve the quality of the input data using the outputs of existing GNN models. Zhou et al. (2019) proposed DSGCN which generalizes and simplifies previous work. The model applies soft label confidence and provides customiziable modeules as extension for multi-stage self-training.

3 Preliminaries and problem definition

3.1 Basic notions on graph

In this problem, we are given an undirected graph $G = (V, E, X)$, where $V = \{v_1, v_2, ..., v_n\}$ consists of a set of nodes with $|V| = n$, and $E = \{e_{ij}\}$ is a set of edges between nodes. X is an

attribute feature of all the nodes, $X = \{x_1, x_2, ...x_n\} \in R^{n\times d}$, where x_i is the d-dimension attribute vector associated with node v_i. Adjacency matrix $A = \{a_{ij}\} \in R^{n\times n}$ denotes the topological structure of graph G, where $a_{ij} = 1$ if there is an edge between node i and j, otherwise $a_{ij} = 0$. We assume each node belongs to exactly one class and use y_i to denote the class label of the $i - th$ node. The aim is to design learning algorithms to predict the labels of all nodes based on the labels of a small set of training nodes provided at the beginning. We define $L \subseteq V$is the set of labeled nodes and $U = V \backslash L$ is the set of unlabeled nodes.

3.2 SEMI-SUPERVISED LEARNING ON GRAPH

The aim of semi-supervised node classification is to learn a classifier from a set D with N samples, $|D| = N$. All the samples are split into an unlabeled set $U = X_U = \{x_{Ui}\}_{i=1}^{|U|}$ and a labeled set $L = (X_L, Y_L) = \{(x_{Li}, y_{Li})\}_{i=1}^{|L|}$, $|D| = |U| + |L|$. y_{Li} is the label for C classes corresponding to x_{Li} in D. y_{Li} can be an one-hot coding or other styles. We only use the labeled nodes as supervised information to train a model to predict the rest unlabeled nodes' classes on the same graph. We difine the model as $fun(.)$, $\tilde{Y}_U = fun(X_L, Y_L, X_U)$. $\tilde{Y}_U$ is the predicted labels of unlabeled nodes. We want the predicted value $\tilde{Y}_U$ to be as close as possible to the ground-truth Y_U, which can be defined as an optimization problem as follows, θ is the parameter of model $fun(.)$.

$$\arg\min_{\theta} \|\tilde{Y}_U - Y_U\|^2 = \arg\min_{\theta} \|fun(X_L, Y_L, X_U) - Y_U\|^2$$

3.3 GCN MODEL

The GCN applied in our framework is proposed by (Kipf & Welling, 2016). GCN can be explained in both spectral and spatial domain. It learns the representations of each node by iteratively aggregating the embeddings of its neighbors, which is a good model for semi-supervised classification. GCN consists of $L > 0$ layers with the same propagation rule and a non-linear transformation. In the $l - th$ layer, the hidden representations $H^{(l-1)}$ are averaged among one-hop neighbors as :

$$H^{(l)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l-1)}W^{(l)})$$

Where $\tilde{A} = A + I$ is the adjacency matrix of G after adding self-loops, $\tilde{D} = diag(\tilde{A})$, which is a diagonal matrix of $\tilde{A}$. $W^{(l)}$ is a trainable weight matrix of the $l - th$ layer, and σ is a nonlinear activation function e.g., $Relu(.) = max(0, .)$ and $softmax(.)$ for the last layer. $H^{(l)} \in R^{n\times d}$ denotes hidden feature matrix of the $l - th$ layer and $H^{(0)} = X$. For more details, please refer to Kipf & Welling (2016). The spatial graph convolutional network recursively aggregate the features from neighbors (Veličković et al., 2017):

$$h_v^{(k)} = AGG(\{h_i^{(k-1)}\}_{i\in\tilde{N}(v)}\}) = \sum_{i\in\tilde{N}(v)} a_{vi}h_i^{(k-1)}W^{(k)}$$

Where $h_v^{(k)}$ denotes the representation of target node v at the $k - th$ layer, $W^{(k)}$ is the weight matrix (model parameter) to do feature transformation at the $k - th$ layer, and the initial feature representation $h_v^{(0)}$ can be obtained from the original feature matrix X.

4 PROPOSED METHOD

In this section, we will elaborate our proposed framework for semi-supervised node classification called Self-training GCN with Feature aggregation Checking part (SGFC). Firstly, we explain the importance of checking part in self-training framework. Secondly, we illustrate the design of checking part in our framework combined with analysis of the characteristics of common graph networks, and show the superiority of our checking part in graph networks. Then we elaborate every part of the framework and display the SGFC training algorithm. Finally we expand our framework to other GNN models.

4.1 ANALYSIS OF CHECKING PART IN MULTI-ROUND SELF-TRAINING

Li et al. (2018) firstly proposed to apply co-training and self-training techniques on node classification. After this work, some later studies combined the idea of co-training and self-training. These work use multi-stage(multi-round) strategies to train a GNN classifier for several rounds. They designed various co-training algorithms and added them to self-training framework. The co-training algorithm serves as a checking part to judge whether a node could be added into the training set for the next round. For example, M3S (Sun et al., 2019) uses Deep Cluster as checking part to assign each node a peusdo-label to make the checking. Checking part plays an important role in self-training framework on node classification task. A poorly-designed checking part cannot correctly select useful nodes from test set. If a node i in test set has ground-truth class label c, it might be assigned with peusdo-label c' ($c' \neq c$) by a poorly-designed checking part and added into training set in the next groud. The wrong supervised information (we call them bad training nodes) would constrain the performance of the model and even baddly degrade the prediction of next round. As is shown in Table 1, we study the performance of GCN on different ratio of bad training nodes on Cora with standard split as Kipf & Welling (2016). It's obvious that the accuracy continously decreases as the ratio of bad training nodes increases, which indicates that a poorly designed checking part adding more bad nodes into training set would degrade GNN performance. Therefore, it's vital to design a fast and efficient checking part to ensure the high quality of new added training nodes.

Table 1: Performance of GCN on different ratio of bad training nodes

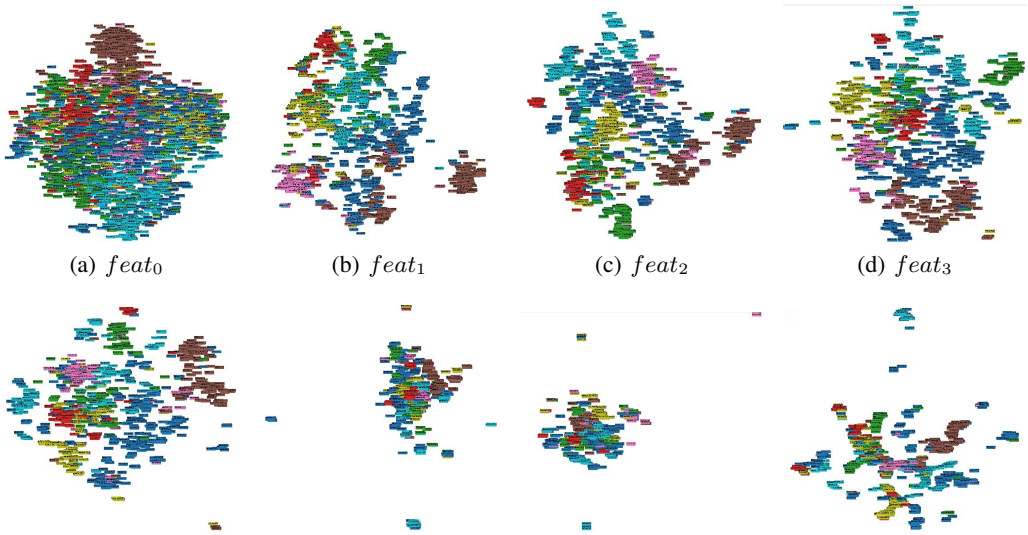
Ratio	**0%**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**
Acc	81.9%	77.9%	71.5%	69.1%	65.4%	57.9%	56.3%

4.2 CHECKING PART WITH FEATURE AGGREGATION

Most of graph networks in real world are small word networks (Watts & Strogatz, 1998) with small diameters, such as socal networks and citation networks. In these graphs, we always reach a destination node from start node only in few hops. Nodes that have the same label tend to connect with each other and even generate a cluster. Under this circumstance, we don't need to capture remote information of a node, which means a shallow GCN could deal with node classification perfectly. Futhermore, we find that these graphs require little graph representation ability of GNN. The reason of GNN's great performance on these graphs might mainly lies on the neighborhood aggregation. We use citation network Cora as an example. We aggregate the features of the nodes in Cora for different times, we choose mean-aggregator, and apply t-SNE (Maaten & Hinton, 2008) to visualize the aggregation result in 2-D space. As is shown in Figure 1, $feat_i$ means the original features of nodes are aggregated for i times, $feat_0$ is the t-SNE result of original features without aggregation. In Figure 1 (a), all the labels are mixed toghter and difficult to distinguish. This can explain the poor classication performance of MLP and other methods only based on node features. However, as the fusion of node features continues to deepen, nodes with same label tend to aggregate into cluster in 2-D space. In the case of $feat_1$, $feat_2$ and $feat_3$, we can already see significant clustering phenomena. When it comes to $feat_{15}$ and $feat_{20}$, sporadic isolated nodes appear, which dues to the existence of connected components that contain few nodes. When the aggregation time goes up to a certain level (such as $feat_{15}$ and $feat_{20}$), the cluster boundaries which is clear in $feat_2$, $feat_3$ begin to blur again. In $feat_{100}$, all the nodes have been confused again and difficult to distinguish. From Figure 1 we can see that when the aggregation time is suitable (2-5 times), nodes can be clearly distinguished in 2-D space. We further find after aggregation, the nodes with original high-dimension features are linearly separable. As Table 2 shows, we use feature aggregation and linear SVM classifier (Cortes & Vapnik, 1995) to make classification. $feat_i$ means we conduct neighboor aggregation i times and then apply a linear SVM to make classification. $Concat$ means we concatenate from $feat_0$ to $feat_{20}$ and make classification. Experiments show our aggregation method has almost the same performance as GNN in citation networks. Note that linear svm can get pretty good performance even on $feat_{20}$, which is much better than a GCN model with same layers (Kipf & Welling, 2016). Xu et al. (2018) attributes the poor performance of deep GCN to over-smoothing, but our experiments show that the adverse effect of nerghboorhood aggregation on GCN seems to be overestimated. More recently, Yang et al. (2020a) argued that over-smoothing is

not the main factor resulting in the failure of deep GCN. They advocate that overfitting is the main factor causing this problem. The linear transformation and activation in every GCN layers may help alleviate the over-smoothing. Our extensive experiments in Figure 2 validated its rationality.

As feature aggregation has satiesfying performance and it's a fast method which doesn't spend lots of time on training epochs. It is a suitable method that serves as checking part in our self-training method. Figure 3 shows the number of bad training ndoes added into training set by different checking mechanisms. We compare our method with deep cluster and plain self-training method (Li et al., 2018) which selects nodes only by the confidence of GCN output without checking part. a-c represent Cora with 0.5%, 1%, 2% label rate. d-f represent Citeseer with 0.5%, 1%, 2% label rate. Our method introduces least bad training nodes compared with other methods, which accounts for the better performance on node classification shown in Section 5.



(a) $feat_0$ (b) $feat_1$ (c) $feat_2$ (d) $feat_3$

(e) $feat_4$ (f) $feat_{15}$ (g) $feat_{20}$ (h) $feat_{100}$

Figure 1: T-SNE visualization of neighborhood aggregation.

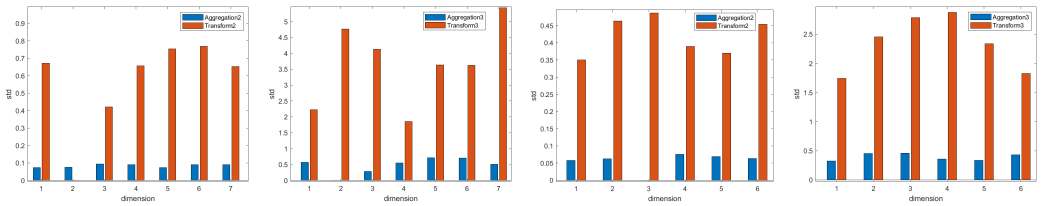
Table 2: Performance of aggregation and classic methods

Dataset	**MLP**	**ChebyNet**	**GCN**	**GAT**	**feat5**	**concat**	**feat20**	**feat50**
Cora	55.1%	81.2%	81.5%	83.0%	83.2%	83.2%	78.8%	34.2%
CiteSeer	46.5%	69.8%	70.3%	72.5%	72.3%	72.3%	70.8%	65.5%
PubMed	71.4%	74.4%	79.0%	79.0%	78.8%	78.2%	73.5%	62.6%

4.3 Our approach

Inspired by Multi-stage Training Framework as generalization for self-training method proposed by previous work. We propose a more generalized end-to-end self-training framework named SGFC. The flow chart of our approach is displayed in Figure 4. Instead of using deep cluster and aligning mechanism, we apply feature aggregation and linear SVM classifier as a checking part to determine whether an unlabeled node could be added to training set for the next round. The selected new nodes are determined by both GCN confidence and the checking part. The design of checking part is directly related to whether the suitable new supervision information can be selected to training set, so it determines whether the framework can effectively improve the classification accuracy.

The aggregation time should be determined by the structure of graph networks, for a small aggregation time will not sufficiently disseminate neighborhood information and a large one will casuse confusion. We can concatenate from $feat_0$ to $feat_{10}$ and put this embbeding into SVM. Although



(a) cora aggre/trans-2 (b) cora aggre/trans-3 (c) citeseer aggre/trans-2 (d) citeseer aggre/trans-3

Figure 2: Standard deviation after each layer's aggregation and transformation. We respectively train a 2 layer GCN (exclude input layer) with hidden units 7 and 6 respectively for Cora and CiteSeer. We show the standard deviation of the middle embeddings' each dimension. From (a)-(d) we can see the transformation of each GCN layer make the standard deviation larger, which increases the diversity in each dimension. This phenomenon may support Yang et al. (2020a)

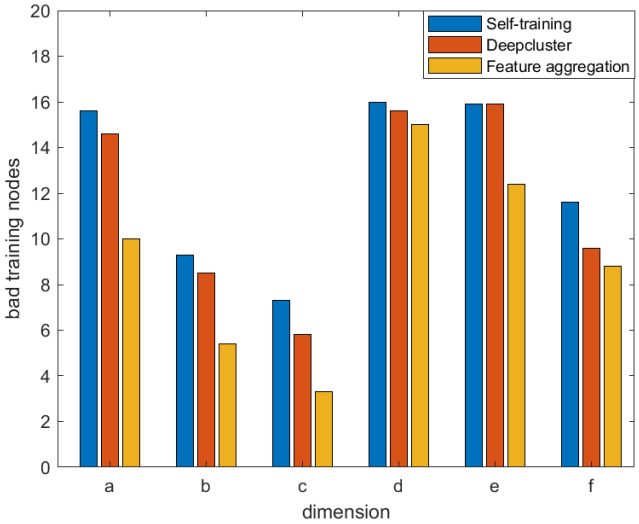


Figure 3: Number of bad training nodes added into training set by different checking mechanism

the concatenation of ten features might be a high-dimension vector, it still fast to train a linear SVM classifier on it. If the pseudo-label generated by checking part is consistent with the prediction output by the GCN model in this round, the node is added to the training set because we can conclude this node belongs to the class with very high certainty. In this way, each round can add high-confident nodes into training set to enhance the next training. We can add all the eligible nodes into training set to fully expand supervised information. However, this method could still add some bad training nodes, making the framework ineffective. So we choose to select a fixed number of eligible nodes with highest confidence in each class into training set just like many previous work have done (Li et al., 2018; Sun et al., 2019). In fact, when the initial training set is quite small, our method can also quickly expand the size of training set so that the accuracy of prediction can reach a relatively high level within a few rounds. Repeat the above steps K times, then we can get the final prediction result. We provide a detailed description of SGFC approach in Algorithm 1.

As for the training round K, it should be adapted to the label rate, a dataset with small label rate should be assigned with a large K. Specifically, we use the function $K = \frac{w}{r+b}$ to fit the graph dataset. w and b are weight and bias of the function respectively. For example, in Cora we have w=9.3, b=-0.3. In CiteSeer and PubMed we have w=46.7, b=1.2 and w=0.6, b=0.1 respectively. These two parameters are related to the characteristics of the graph. For example we find the weight w is correlated with the graph smoothness. We reference and modify the label smoothness λ_l and feature smoothness λ_f proposed by Hou et al. (2019) to define graph smoothness. λ_l (Hou et al.,

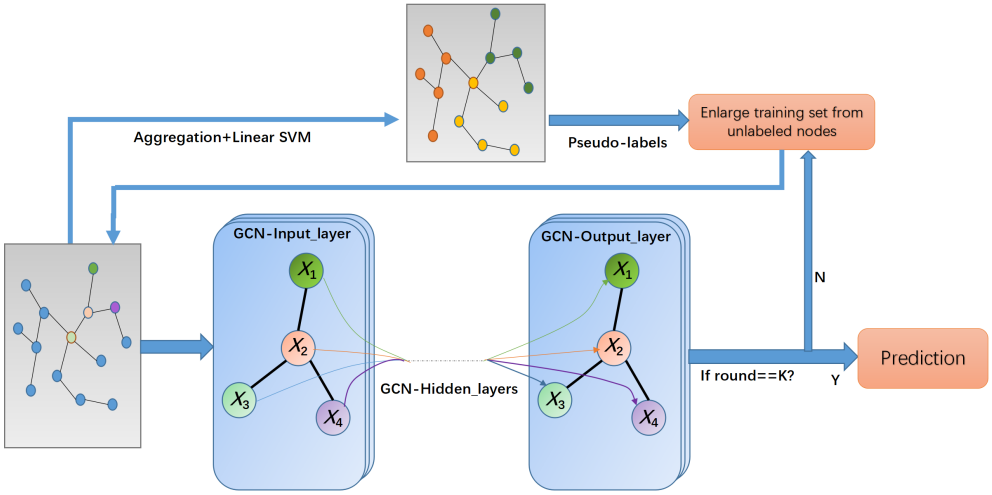


Figure 4: The overall framework of SGFC on node classifiation task.

2019) is defined as:

$$\lambda_l = \sum_{e_{vi,vj} \in \varepsilon} (1 - I(v_i \simeq v_j))/|\varepsilon|$$

ε is the set of all the edges in graph. $I(v_i \simeq v_j) = 1$ when v_i and v_j have same label and 0 when they have different labels. A smaller λ_l implies that nodes with same labels tend to be connected together. Similarly, λ_f is defined as:

$$\lambda_f = \frac{\| \sum_{v \in V} (\sum_{v' \in N_v} (x_v - x_{v'}))^2 \|_2}{|\varepsilon| d}$$

where $\| \ . \ \|_2$ is the $l - 2$ norm and d is the dimension of node feature. A larger λ_f means the connected nodes in graph are more likely to be dissimilar. The smoothness values are shown in Table 3 (Hou et al., 2019). The three graph have similar λ_f and distinct λ_l. Cora and PubMed has comparable λ_l while CiteSeer has a larger one, which means the node in Cora and PubMed can gain much positive information from its surrounding to get its peak accuracy quickly and thus have a smaller w. Compared PubMed with Cora, PubMed has a larger λ_f, which means the graph's feature sets have high frequency and the context of a node can obtain more information gain from its neighboors and may quickly get its peak accuracy and thus has a smaller w than Cora. After the selection of w, we choose a suitable bias b to fine-tune the function.

Table 3: λ_l on different dataset

Dataset	**Cora**	**CiteSeer**	**PubMed**
λ_l	0.19	0.26	0.20
λ_f	4.67e-4	1.22e-4	9.49e-4

Our framework can be easily ported to many other GNN models. In addition to GCN, the GNN part can be replaced with GAT, GraphSage and other models according to specific tasks.

5 EXPERIMENT

In this section, we conduct extensive experiments to validate the effectiveness of our proposed framework on semi-supervised node classification task.

Algorithm 1 SGFC Training Algorithm

Input:
Features matrix of all nodes, X;
Adjacent matrix of graph G(V,E), A;
Labeled and unlabeled set, L_0, U_0;
GCN model, $GCNconv(.)$;
Classifier, $Classify(.)$;
Self-training rounds, K

Output:
Labels of all the unlabeled nodes, Y_u;

Let $L = L_0$, $U = U_0$, $K = |L_0|$;
for $k = 0$ to K **do**
  Execute feature aggregation for nine times to generate $feat_1, \ldots, feat_9$;
  Make concatenation: $feat = (X||feat_1||\ldots||feat_9)$;
  Execute classifier to get pseudo-labels: $\check{Y}_u = Classify(feat, L, U)$;
  Train GCN model and get GCN predictions and confidence matix: $\check{Y}_u, M_c = GCNconv(A, X, L, U)$;
  for each class j **do**
    Find the top t nodes in U according to confidence matrix M_c.
    for each node i of selected t nodes **do**
      if $\check{Y}_{ui}$ equals $\check{Y}_{ui}$ **then**
        Add node i to labeled set L;
        Delete node i from unlabeled set U;
      end if
    end for
  end for
  Update L and U;
end for
Get the labels of nodes in U_0: $Y_u = GCNconv(A, X, L, U)$
return Y_U

Table 4: Datasets statistics

	Class	Features	Nodes(tain/val/test)	Edges
Cora	7	1,433	2,485(140/200/1000)	5,069
CiteSeer	6	3,703	2,110(120/200/1000)	3,668
PubMed	3	500	19,717(60/200/1000)	44,324

5.1 DATASET

We conduct experiments on 3 open graph datasets based on citation network(including Cora, CiteSeer, PubMed) (Sen et al., 2008) for semi-supervised node classification task.

Citetion networks Cora, CiteSeer and PubMed are representative citation network datasets where nodes and edges denote documents and their relationships, respectively. Node features are formed by bag-of-words representations for documents. Each node has a label indicating what field the corresponding document belongs to.

The detailed statistics of the 3 datasets are summarized in Table 4.

5.2 EXPERIMENTAL SETTINGS

As is illustrated above, our framework can be ported to many GNN models. In this experiment, we select plain GCN (Kipf & Welling, 2016) and use our proposed checking part to provide pseudo-labels. We set the number of GCN layers n_layers=2, learning rate lr=1e-2, training epochs=600, weight_decay=5e-4. We will find the top 10 nodes (t=10 in Algorithm 1) for each class. As for training round K, We use the function $K = \frac{w}{r+b}$ proposed in Section 4.3 to assign a most suitable K for

Table 5: Results on Cora dataset

Label Rate	**0.5%**	**1%**	**2%**	**3%**	**4%**
Node2Vec	32.4%	44.4%	50.2%	54.5%	57.4%
LP	57.6%	61.0%	63.5%	64.3%	65.7%
Cheby	38.0%	52.0%	62.4%	70.8%	74.1%
GCN	50.6%	58.4%	70.0%	75.7%	76.5%
GAT	49.0%	60.7%	72.8%	77.7%	79.7%
Co-training	53.9%	57.0%	69.7%	74.8%	75.6%
Self-training	56.8%	60.4%	71.7%	76.8%	77.7%
MultiStage	61.1%	63.7%	74.4%	76.1%	77.2%
M3S	61.5%	67.2%	75.6%	77.8%	78.0%
SGFC	**62.8**%	**68.5**%	**76.1**%	**78.0**%	**80.3**%

each experiment. For Cora dataset we choose K as 47,13,5,3,3. And we choose K as 27,21,15,11,9 for CiteSeer; 5,4,3 for PubMed. The label rate indicates the amount of labeled nodes. We use the labeled nodes as training set and all the remaining nodes are used as test set. For each label rate we randomly split the training set and test set and we take extra measures to ensure that the nodes of each class in training set is balanced. For each experiment we repeat 10 times and get the mean accuracy.

5.3 BASELINE&RESULTS

The baseline of our experiments include traditional deep learning method such as MLP, Node2Vec (Grover & Leskovec, 2016) ,LP (Wu et al., 2012) and classic GNN approach such as GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017) and ChebNet (Defferrard et al., 2016). We also include Co-training and Self-training proposed by Li et al. (2018) and other self-learning based approaches MultiStage (Sun et al., 2019), M3S (Sun et al., 2019) as baseline. The relevant experimental settings and results are all taken from original papers.

The results on all the 3 dataset are summarized in Table 5, Table 6 and Table 7. It can be seen from the three tables that the GNN based approaches is obviously better than non-GNN-based ones. And the self-training method can achieve better result than those without self-training, especially when the label rate is quite small. So the self-training mechanism do help GNN perform better. In summary, our SGFC achieves superior performance on all these 3 datasets, which significantly demonstrates the effectiveness and generalizability of our proposed model. Our experimental results prove the superiority of using self-training mechanism and a good checking part to generate pseudo-labels, which provides the GNN model with more valuable supervised information. Compared with plain self-training algorithm (Li et al., 2018), our framework applies an extra checking part to expand supervised information, which is more reliable than only use GNN output confidence. Compared with M3S, our framework adopts a better checking part, the Deep Cluster adopted by M3S doesn't fully utilize node's structural information, and the checking part must work after the GNN has generated embeddings. Our checking part makes full use of both node features and structural information by feature aggregation and is fast to execute.

5.4 ABLATION EXPERIMENT

Our framework can have better performance compared with a single GNN model. But the hyper-parameter training round K is determined by graph dataset. So we study the influence of training rounds K on the classification accuracy. Due to the random split of initial training set and test set, the experimental results vary within a certain range. We take CiteSeer as an example and select five representative experiments to study this question.

As we can see in Figure 5(a), accuracy grows during the first few rounds in all the five experiments. Note that when round is 0, the framework degrades to plain GCN. So Figure 5(a) shows our method based on self-training does improve the performance without designing more complicated GNN architecture. Especially when the label rate is small, there is a significant margin between our framework and plain GCN. In addition, when the label rate is small, accuracy tends to converge

Table 6: Results on CiteSeer dataset

Label Rate	**0.5%**	**1%**	**2%**	**3%**	**4%**
Node2Vec	24.9%	29.1%	34.4%	35.7%	38.8%
LP	37.7%	41.6%	41.9%	44.4%	44.8%
Cheby	31.7%	42.8%	59.9%	66.2%	68.3%
GCN	44.8%	54.7%	61.2%	67.0%	69.0%
GAT	44.6%	53.7%	64.6%	66.4%	69.3%
Co-training	42.0%	50.0%	58.3%	64.7%	65.3%
Self-training	51.4%	57.1%	64.1%	67.8%	68.8%
MultiStage	53.0%	57.8%	63.8%	68.0%	69.0%
M3S	56.1%	62.1%	66.4%	70.3%	70.5%
SGFC	**58.3%**	**66.4%**	**70.2%**	**70.3%**	**70.8%**

Table 7: Results on PubMed dataset

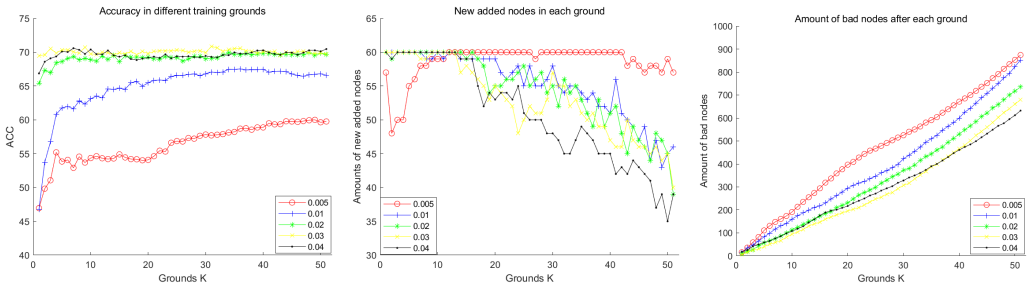
Label Rate	**0.03%**	**0.05%**	**0.1%**
LP	37.2%	38.2%	42.9%
LP	58.3%	61.3%	63.8%
Cheby	40.4%	47.3%	51.2%
GCN	51.1%	58.0%	67.5%
GAT	50.6%	59.1%	65.0%
Co-training	55.5%	61.6%	67.8%
Self-training	56.3%	63.6%	70.0%
MultiStage	57.4%	64.3%	70.2%
M3S	59.2%	64.4%	70.6%
SGFC	**60.1%**	**65.2%**	**70.7%**

when the round is pretty large. However when the label rate is large, accuracy reaches maximum value rapidly and even decreases when in large round. So the round K should be adapted to the label rate, a dataset with small label rate should be assigned with a large K. The result in Figure 5(a) is consistent with the phenomenon in Figure 5(b). When the label rate is small, our framework can still add a lot of eligible new nodes to improve accuracy. But when the label rate is large, the framework can dig out enough nodes to training set but fails when the round becomes large, which results in the stopping of growth in Figure 5(a). In Figure 5(c), we can see that our framework still generate many bad training nodes in each round which constrain the performance of the framework. In fact, the aim of checking part is to alleviate this problem. We would devote to further solving this problem in future work.

Table 8: Performance of our framework based on different GNN models

GNN Model	**GCN**	**GAT**	**APPNP**	**GS-M**	**GS-P**
Acc of normal GNN	44.8%	46.8%	45.7%	40.9%	34.9%
Acc in our framework	58.3%	59.8%	57.6%	59.1%	51.0%

SGFC can be easily ported to many other GNN model. Different GNNs have slightly different inference ability on graph dataset. The selection of GNN model depends on the specific task. We extend our framework to various GNN models. In Table 8, we test our framework with different GNN model on CiteSeer with label rate 0.5%. GS-M, GS-P respectively represent GraphSage with mean, maxpool aggregator. It can be seen that on a certain dataset, different GNN models perform differently under this framework. On CiteSeer with 0.5% label rate, these models have nearly similar performance and GAT has the best performance among them.



(a) Accuracy (b) New added nodes (c) Bad training nodes

Figure 5: Accuracy, new added nodes and bad training nodes in each round

6 Conclusion

In this paper, we firstly analyze the limitations of graph convolutional networks in dealing with sem-supervised node classification task. In order to overcome these limitations we introduce a widely used method self-training to the task and provide a systematical analysis on this issue and argue that self-training and checking part are of great importance in enhancing GNN's accuracy of classification. Then we propose a novel node classification framework namely Self-training GCN with Future aggregation Checking part (SGFC), which adds confident data selected by the output of both GCN and checking part to the labeled set to enlarge the training set. In addition, we apply feat aggregation and linear SVM classifier on the graph nodes as checking part to improve our method. Our finally proposed approach, SGFC training algorithm, outperforms other state-of-the-art methods on all the selected three dataset, especially when the ratio of labeled nodes is quite small. We carry out ablation study and extend our framework to many other GNN models.

References

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

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 132–149, 2018.

Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *Advances in neural information processing systems*, pp. 2456–2464, 2011.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

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

Brett Drury, Luis Torgo, and Jose Joao Almeida. Guided self training for sentiment classification. In *Proceedings of Workshop on Robust Unsupervised and Semisupervised Methods in Natural Language Processing*, pp. 9–16, 2011.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In *Advances in neural information processing systems*, pp. 10701–10711, 2019.

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

Marti Hearst. Noun homograph disambiguation using local context in large text corpora. *Using Corpora*, pp. 185–188, 1991.

Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard TB Ma, Hongzhi Chen, and Ming-Chang Yang. Measuring and improving the use of graph information in graph neural networks. In *International Conference on Learning Representations*, 2019.

Zhongqiang Huang and Mary Harper. Self-training pcfg grammars with latent annotations across languages. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 832–841, 2009.

Zhongqiang Huang, Vladimir Eidelman, and Mary Harper. Improving a simple bigram hmm part-of-speech tagger by latent annotation and self-training. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pp. 213–216, 2009.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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

Zornitsa Kozareva, Boyan Bonev, and Andres Montoyo. Self-training and co-training applied to spanish named entity recognition. In *Mexican International conference on Artificial Intelligence*, pp. 770–779. Springer, 2005.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jurica Levatić, Michelangelo Ceci, Dragi Kocev, and Sašo Džeroski. Self-training for multi-target regression with tree ensembles. *Knowledge-based systems*, 123:41–60, 2017.

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*, pp. 9267–9276, 2019.

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

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

David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pp. 152–159, 2006.

David McClosky, Eugene Charniak, and Mark Johnson. When is self-training effective for parsing? In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pp. 561–568, 2008.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. Adversarially regularized graph autoencoder for graph embedding. *arXiv preprint arXiv:1802.04407*, 2018.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

Ellen Riloff, Rosie Jones, et al. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pp. 474–479, 1999.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. 2005.

Kenji Sagae. Self-training without reranking for parser domain adaptation and its impact on semantic role labeling. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pp. 37–44, 2010.

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.

Pierre Sermanet, Soumith Chintala, and Yann LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 3288–3291. IEEE, 2012.

Ke Sun, Zhanxing Zhu, and Zhouchen Lin. Multi-stage self-supervised learning for graph convolutional networks. *arXiv preprint arXiv:1902.11038*, 2019.

Vincent Van Asch and Walter Daelemans. Predicting the effectiveness of self-training: Application to sentiment classification. *arXiv preprint arXiv:1601.03288*, 2016.

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

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Boris Weisfeiler and Andrei A Lehman. A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16, 1968.

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.

Xiao-Ming Wu, Zhenguo Li, Anthony M So, John Wright, and Shih-Fu Chang. Learning with partially absorbing random walks. In *Advances in neural information processing systems*, pp. 3077–3085, 2012.

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

Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. Revisiting" over-smoothing" in deep gcns. *arXiv preprint arXiv:2003.13663*, 2020a.

Han Yang, Xiao Yan, Xinyan Dai, and James Cheng. Self-enhanced gnn: Improving graph neural networks using model outputs. *arXiv preprint arXiv:2002.07518*, 2020b.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pp. 189–196, 1995.

Yuning You, Tianlong Chen, Zhangyang Wang, and Yang Shen. When does self-supervision help graph convolutional networks? *arXiv preprint arXiv:2006.09136*, 2020.

Yan Zhou, Murat Kantarcioglu, and Bhavani Thuraisingham. Self-training with selection-by-rejection. In *2012 IEEE 12th international conference on data mining*, pp. 795–803. IEEE, 2012.

Ziang Zhou, Shenzhong Zhang, and Zengfeng Huang. Dynamic self-training framework for graph convolutional networks. *arXiv preprint arXiv:1910.02684*, 2019.