NEW GCNN-BASED ARCHITECTURE FOR SEMI-SUPERVISED NODE CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

The nodes of a graph existing in a specific cluster are more likely to connect to each other than with other nodes in the graph. Then revealing some information about the nodes, the structure of the graph (the graph edges) provides this opportunity to know more information about the other nodes. From this perspective, this paper revisits the node classification task in a semi-supervised scenario by graph convolutional neural network. The goal is to benefit from the flow of information that circulates around the revealed node labels. For this aim, this paper provides a new graph convolutional neural network architecture. This architecture benefits efficiently from the revealed training nodes, the node features, and the graph structure. On the other hand, in many applications, non-graph observations (side information) exist beside a given graph realization. The non-graph observations are usually independent of the graph structure. This paper shows that the proposed architecture is also powerful in combining a graph realization and independent non-graph observations. For both cases, the experiments on the synthetic and real-world datasets demonstrate that our proposed architecture achieves a higher prediction accuracy in comparison to the existing state-of-the-art methods for the node classification task.

1 INTRODUCTION

Node classification in graphs is generally an unsupervised learning task which refers to clustering (grouping) nodes with similar features. Revealing the labels for a small proportion of nodes transforms the unsupervised node classification task to a semi-supervised learning problem. Semi-supervised node classification on a purely graphical observation (a graph realization) has been investigated in the literature on real-world networks by providing various methods. For a brief survey see Section 2.

Under the transductive semi-supervised learning setting, the goal is to predict the labels of unlabeled nodes given the adjacency matrix of a graph, the feature matrix containing a set of features for all nodes, and a few revealed node labels. There exist various methods for inferring the unlabeled nodes such as (Kipf & Welling, 2016; Veličković et al., 2017). Most of the prominent existing methods use either graph-based regularization, graph embedding, or graph convolutional neural networks in a node domain or a spectral domain.

The structure of a graph (graph edges) allows a graph convolutional neural network to use a set of fixed training nodes to predict the unlabeled nodes. Increasing the number of fixed training nodes improves the accuracy of the predictions. But in practice, a few training nodes are available in the training set. In this paper, we comprehensively investigate the way that the predicted labels can be effectively involved in the training procedure to increase the prediction accuracy.

On the other hand, in many applications, non-graph observations (side information) exist beside a given graph realization and its node feature matrix. See (Saad & Nosratinia, 2018) and references therein for a brief introduction about the effects of side information on the community detection for generative models. In practice, the feature matrix is not independent of the graph structure, while the non-graph observations may be independent. Combining the feature matrix with the non-graph observations is important especially for the case in which the quality of side information is not obvious for the estimator.

In this paper, we propose a novel graph convolutional neural network architecture that benefits from the predicted unlabeled nodes and improves the accuracy of prediction. Our proposed architecture is also able to combine the provided side information with the graph structure and its feature matrix. This combination achieves higher accuracy in comparison to the existing state-of-the-art methods. To the best of our knowledge, this is the first time that the predicted labels in a graph are revisited by a graph convolutional neural network to improve the accuracy. In addition, this is the first time that the performance of graph convolutional neural networks has been investigated in the presence of independent non-graph observations (side information).

2 Related Work

Graph-based semi-supervised methods are typically classified into explicit and implicit learning methods. In this section, we review the related work in both classes while the focus of this paper is mainly on the graph convolutional neural network which belongs to the latter.

2.1 Explicit Graph-Based Learning

In the graph-based regularization methods, it is assumed that the data samples are located in a low dimensional manifold. These methods use a regularizer to combine the low dimensional data with the graph. In the graph-based regularization methods, the objective function of optimization is a linear combination of a supervised loss function for the labeled nodes and a graph-based regularization term with a hyperparameter. The hyperparameter makes a trade-off between the supervised loss function and the regularization term. Graph Laplacian regularizer is widely used in the literature: a label propagation algorithm based on Gaussian random fields (Zhu et al., 2003), a variant of label propagation (Talukdar & Crammer, 2009), a regularization framework by relying on the local or global consistency (Zhou et al., 2004), manifold regularization (Belkin et al., 2006), a unified optimization framework for smoothing language models on graph structures (Mei et al., 2008), and deep semi-supervised embedding (Weston et al., 2012).

Besides the graph Laplacian regularization, there exist other methods based on the graph embedding: DeepWalk (Perozzi et al., 2014) that uses the neighborhood of nodes to learn embeddings, LINE (Tang et al., 2015) and node2vec (Grover & Leskovec, 2016) which are two extensions of DeepWalk using a biased and complex random walk algorithm, and Planetoid (Yang et al., 2016) which uses a random walk-based sampling algorithm instead of a graph Laplacian regularizer for acquiring the context information.

2.2 Implicit Graph-Based Learning

Graph convolutional neural networks have attracted increasing attention recently, as an implicit graph-based semi-supervised learning method. Several graph convolutional neural network methods have been proposed in the literature: a diffusion-based convolution method which produces tensors as the inputs for a neural network (Atwood & Towsley, 2016), a scalable and shallow graph convolutional neural network which encodes both the graph structure and the node features (Kipf & Welling, 2016), a multi-scale graph convolution (Abu-El-Haija et al., 2018), an adaptive graph convolutional networks (Li et al., 2018), graph attention networks (Veličković et al., 2017), a variant of attention-based graph neural network for semi-supervised learning (Thekumparampil et al., 2018), and dual graph convolutional networks (Zhuang & Ma, 2018).

3 Proposed Semi-Supervised Node Classification Architecture

In this section, we start by stating some quick intuitions to clarify how revealing some node labels may help the estimator to classify other nodes. We define the graph convolutional neural network semi-supervised problem and analyze our idea for revealed node labels. Then we propose our semi-supervised node classification architecture. This section is finished by providing a technique for extracting side information in the proposed architecture based on the adjacency matrix.

3.1 INTUITION

We start by a simple example to illustrate how revealed node labels may help an estimator to predict the labels of unlabeled nodes. Assume in a given graph with k classes, the labels of all nodes are revealed except for two nodes i and j. The goal is to classify node i. A Bayesian hypothesis testing problem with k hypotheses is considered. Let D_i be a vector of random variables such that l-th element denotes the number of edges from node i to other nodes with revealed labels in the cluster l. Also, let D_i' be a vector whose l-th element denotes the number of edges from node i to other unlabeled nodes (node j in this example) in the cluster l. Since the estimator does not know that node j belongs to which class, D_i' is also an unknown random variable. The random variable H takes the values in the set $\{1, \cdots, k\}$. For node i, we want to infer the value of H by observing a realization of D_i. Then we have to select the most likely hypothesis conditioned on D_i, i.e.,

$$\underset{k}{\text{maximize}}\, \mathbb{P}(H = k | D_i = d_i),$$

which is the Maximum A Posteriori (MAP) estimator. Let A denote the adjacency matrix of the graph. With no prior distribution on H, when $A_{ij} = 0$, the MAP estimator is reorganized as

$$\underset{k}{\text{maximize}}\, \mathbb{P}(D_i = d_i | H = k), \tag{1}$$

which can be solved by $k - 1$ pairwise comparisons. When $A_{ij} = 1$,

$$\mathbb{P}(H = k | D_i = d_i) = \frac{\mathbb{P}(D_i = d_i, H = k)}{\mathbb{P}(D_i = d_i)} = \frac{\sum_{d_i' \in S} \mathbb{P}(D_i = d_i, D_i' = d_i', H = k)}{\mathbb{P}(D_i = d_i)},$$

where $S \triangleq \{s = \{0,1\}^k : s^T \mathbf{1} = 1\}$. Assume there exists no prior distribution on H. Then the MAP estimator is reorganized as

$$\underset{k}{\text{maximize}} \sum_{d_i' \in S} \mathbb{P}(D_i = d_i, D_i' = d_i' | H = k). \tag{2}$$

A comparison between equation 1 and equation 2 shows that how revealing true node labels reduces the complexity of optimum estimator.

3.2 PROBLEM DEFINITION & ANALYSIS

The focus of this paper is on the graph-based semi-supervised node classification. For a given graph with n nodes, let A denote an $n \times n$ adjacency matrix and X denote an $n \times m$ feature matrix, where m is the number of features. Under a transductive learning setting, the goal is to infer unknown labels Y_u, given the adjacency matrix A, the feature matrix X, and L revealed labels denoted by Y_l (fixed training nodes). Without loss of generality, assume the first L nodes of the graph are the revealed labels. Then $Y \triangleq [Y_l, Y_u]$ denotes the vector of all node labels (labeled and unlabeled nodes). On the other hand, assume there exists a genie that gives us a vector of side information Y_s with length n such that conditioned on the true labels, Y_s is independent of the graph edges. Without loss of generality, it is assumed that the entries of Y_s are a noisy version of the true labels. In this paper, we suppose that the feature matrix X depends on the graph, conditioned on the true labels. To infer the unlabeled nodes, the Maximum A Posteriori (MAP) estimator for this configuration is

$$\mathbb{P}(Y | A, X, Y_s, Y_l) = \frac{\mathbb{P}(A, X, Y_s, Y_l | Y)\mathbb{P}(Y)}{\mathbb{P}(A, X, Y_s, Y_l)} \propto \mathbb{P}(A, X, Y_l | Y)\mathbb{P}(Y_s | Y),$$

where Y is drawn uniformly from the set of labels, i.e., there is no prior distribution on node labels. Then we are interested in the optimal solution of the following maximization:

$$f \triangleq \underset{Y}{\text{maximize}} \log \mathbb{P}(A, X, Y_l | Y) + \log \mathbb{P}(Y_s | Y).$$

Assume $\hat{Y}$ and $\tilde{Y}$ are the primal optimal solutions of maximizing $\log \mathbb{P}(A, X, Y_l | Y)$ and $\log \mathbb{P}(Y_s | Y)$, respectively. Then,

$$\log \mathbb{P}(A, X, Y_l | \tilde{Y}) + \log \mathbb{P}(Y_s | \tilde{Y}) \leq f \leq \log \mathbb{P}(A, X, Y_l | \hat{Y}) + \log \mathbb{P}(Y_s | \tilde{Y}),$$

or equivalently

$$\log \mathbb{P}(A, X, Y_l|\tilde{Y}) \leq f - \log \mathbb{P}(Y_s|\tilde{Y}) \leq \log \mathbb{P}(A, X, Y_l|\hat{Y}).$$

For squeezing $f - \log \mathbb{P}(Y_s|\tilde{Y})$ from above and below, it suffices to provide an algorithm to make $\hat{Y}$ and $\tilde{Y}$ as close as possible by changing the entries of training labels Y_l. Recall the assumption that there exists a genie that provides an independent graph side information. This assumption can be relaxed and the side information can be extracted from either the feature matrix or the adjacency matrix of a graph. Note that extracting side information from both the feature matrix and the adjacency matrix makes the side information completely dependent on both inputs of a graph convolutional neural network.

3.3 Proposed Model

In this paper, the side information either is given directly or generated from the feature or the adjacency matrix. Figure 1 shows our proposed architecture with three blocks.

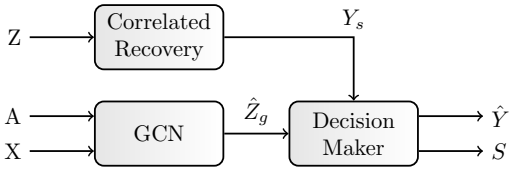


Figure 1: The block diagram of the proposed architecture.

The GCN block is a variant of classical graph convolutional neural network (Kipf & Welling, 2016) that takes X and A as inputs and returns $\hat{Z}_g$ which is an $n \times k$ matrix, where k is the number of classes. Note that $\hat{Z}_g(i, j)$ determines the probability that the node i belongs to the class j in the graph. The correlated recovery block is applied when the side information is not given directly. In community detection, the correlated recovery refers to the recovering of node labels better than random guessing. The input of the correlated recovery is either the feature matrix X or a function of the adjacency matrix A. The output of the correlated recovery block is Y_s which is a vector with length n. The decision maker decides how to combine the provided side information Y_s and the output of the GCN block $\hat{Z}_g$. The decision maker returns the predicted labels $\hat{Y}$ which is a vector with length n and a set of node indices S that are used for defining the loss function. Then the loss function for this architecture is defined as

$$\mathcal{L}(Y_s, \hat{Y}, S) \triangleq \frac{1}{|S|} \sum_{i \in S} \mathcal{L}_0(Y_s(i), \hat{Y}(i)),$$

where $\mathcal{L}_0(\cdot, \cdot)$ is the cross-entropy loss function, and the index i in $\hat{Y}(i)$ and $Y_s(i)$ refers to i-th entry.

Let E index the epochs during the training procedure and E_u denote the epoch in which the decision maker starts to make a change in the number of training nodes.

- **Phase (1):** when $E < E_u$, the decision maker embeds the fixed training labels Y_l inside the side information Y_s, resulting in $Y_s(i) = Y_l(i)$ for all $i \in \{\text{fixed training node indices}\}$. The decision maker returns $\hat{Y}$ and the set of training nodes S.
- **Phase (2):** when $E \geq E_u$, the decision maker first embeds the fixed training labels Y_l inside the side information Y_s. Then the decision maker uses $\hat{Z}_g$ and determines a set of nodes S_1 such that each element of S_1 belongs to a specific class with a probability at least P_{th}. Note that P_{th} is a threshold that evaluates the quality of the selected nodes. On the other hand, the decision maker obtains a set of nodes S_2 such that for each element of S_2 both the corresponding side information and the prediction of the graph convolutional neural network refer to the same class. Then,

$$S \triangleq (S_1 \cap S_2) \cup \{\text{fixed training node indices}\}.$$

Phase (2) continues until the prediction accuracy for the fixed training nodes be grater than F_{th}; Otherwise, the training continues based on the last obtained set S. In this procedure, E_u, P_{th}, and F_{th} are three hyperparameters that should be tuned.

Assume at epoch E_u, the optimal solution for maximizing $\mathbb{P}(A, X, Y_l|Y)$ is $\hat{Y}_g^{E_u}$ which is extracted from $\hat{Z}_g$. The decision maker uses $\hat{Z}_g$ and Y_s to obtain a set of nodes S that is used for the next training iteration. Also, since the neural networks are robust to the noisy labels (Rolnick et al., 2017; Hendrycks et al., 2018; Ghosh et al., 2017), the selected nodes will have enough quality to be involved in the training process by choosing an appropriate value for P_{th}. Note that the hyperparameter P_{th} determines the quality of the selected nodes. Then at epoch $E_u + 1$, the training is based on a new training set $Y_l^{E_u} \triangleq \{Y_s(i) : i \in S\}$ which includes the fixed training labels in Y_l. Let $\hat{Y}_g^{E_u+1}$ be the optimal solution for maximizing $\mathbb{P}(A, X, Y_l^{E_u}|Y)$. Note that the side information Y_s is more similar to $Y_l^{E_u}$ than Y_l. Then $\tilde{Y}$ is more similar to $\hat{Y}_g^{E_u+1}$ than $\hat{Y}_g^{E_u}$ and the idea follows.

3.4 Extracting Side Information

For extracting side information that is as much as possible independent from the output of the GCN block, the side information is extracted either from the given feature matrix or the adjacency matrix of the graph. Define the r-neighborhood matrix A_r as

$$[A_r]_{ij} \triangleq \frac{|N_i(r) \cap N_j(r)|}{|N_i(r) \cup N_j(r)|},$$

where $N_i(r)$ is the set of nodes that are in a distance with radius r of node i. For extracting side information from the adjacency matrix, a classifier is trained by the r-neighborhood matrix and the training nodes, while r is a hyperparameter that must be tuned. A similar idea is represented in (Abbe & Sandon, 2015) in which the authors use a variant of r-neighborhood matrices and solve a set of linear equations to theoretically determine whether a pair of nodes are in the same cluster or not. On the other hand, for extracting side information from the feature matrix, a classifier is trained directly based on the feature matrix X and the training nodes.

4 Experiments

The proposed architecture in Section 3 is tested under a number of experiments on synthetic and real-world datasets: semi-supervised document classification on three real citation networks, semi-supervised node classification under the stochastic block models with a different number of classes, and semi-supervised node classification in the presence of noisy labels side information which is independent of graph edges for both the synthetic and real datasets.

4.1 Datasets & Side Information

Citation Networks: Cora, Citeseer, and Pubmed are three common citation networks that have been investigated in previous studies. In these networks, articles are considered as nodes. The article citations determine the edges connected to the corresponding node. Also, a sparse bag-of-words vector, extracted from the title and the abstract of each article, is used as a vector of features for that node. Table 1 shows the properties of these real datasets in detail.

Table 1: The properties of the real datasets for the semi-supervised node classification.

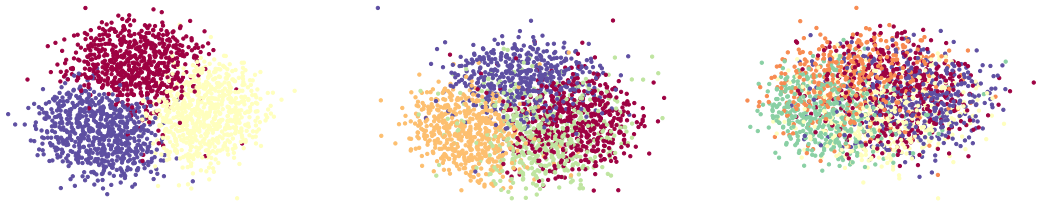
Real Dataset	Nodes	Edges	Classes	Features	Training Nodes
Cora	2708	5429	7	1433	140
Citeseer	3327	4732	6	3703	120
Pubmed	19717	44338	3	500	60

Stochastic Block Model (SBM): The stochastic block model is a generative model for random graphs which produces graphs containing clusters. Here, we consider a stochastic block model with $n = 2000$ nodes and k classes. Without loss of generality, assume the true label for each node is

drawn uniformly from the set $\{0, \cdots, k-1\}$. Under this model, if two nodes belong to the same class then an edge is drawn between them with probability p; Otherwise, these nodes are connected to each other with probability q. Table 2 summarizes the properties of the stochastic block models in our experiments. Also, Figure 2 shows three realizations of the described generative model with the parameters in Table 2. In this paper, a realization of the stochastic block model, based on the parameters in Table 2 with k classes, is briefly called k-SBM dataset.

Table 2: The properties of the synthetic dataset for the semi-supervised node classification.

Synthetic Dataset	Nodes	Classes	p	q	Training Nodes
k-SBM	$n = 2000$	$k \in \{3, 4, 5\}$	$5 \times \frac{\log n}{n}$	$1 \times \frac{\log n}{n}$	$20k$



(a) $k = 3$ (b) $k = 4$ (c) $k = 5$

Figure 2: Three realizations of k-SBM with 2000 nodes and a various number of classes.

Noisy Labels Side Information: We consider a noisy version of the true label for each node as synthetic side information. This information is given to the decision maker to investigate the effect of a non-graph observation which is completely independent of the graph edges. Under the noisy labels side information, the decision maker observes the true label of each node with probability α; Otherwise, the decision maker observes a value that is drawn uniformly from the incorrect labels.

4.2 Experimental Settings

For the GCN block in Figure 1, a two-layer graph convolutional neural network is trained with ReLu and softmax activation functions at the hidden and output layers, respectively. For real datasets, we exactly follow the same data splits in (Kipf & Welling, 2016) including 20 nodes per class for training, 500 nodes for the validation, and 1000 nodes for the test. For k-SBM datasets, we follow a data splitting similar to the one used for the real datasets. Then it is randomly considered 20 nodes per class for the training, 500 nodes for the validation, and 1000 nodes for the test. The weights of the neural networks are initialized by the Glorot initialization in (Glorot & Bengio, 2010). Adam (Kingma & Ba, 2014) optimizer with specific learning rates for phase (1) and phase (2) is applied. Also, the cross-entropy loss is used for all datasets. Table 3 summarizes the values of hyperparameters that are picked for each dataset in the experiments.

Table 3: Hyperparameters for the proposed architecture experiments.

Hyperparameters	Cora	Citeseer	Pubmed	k-SBM
P_{th}	0.55	0.80	0.70	0.50
F_{th}	0.99	0.80	1.00	0.50
E_u	50	80	80	150
Neurons	128	128	64	16
Maximum Epochs	250	200	200	300
L2 Regularization Factor	8×10^{-5}	8×10^{-5}	4×10^{-4}	5×10^{-5}
Learning Rate for Phase 1	0.01	0.01	0.01	0.01
Learning Rate for Phase 2	0.005	0.05	0.002	0.01
Correlated Recovery Input(s)	A_4	X	A_1	A, A_1
Correlated Recovery Classifier	GBC	GBC	GBC	GCNN

Throughout this paper, a gradient boosting classifier and a graph convolution neural network classifier are used for real and synthetic datasets, respectively, as a classifier in the correlated recovery block.

4.3 BASELINES

For the synthetic dataset either with or without the synthetic side information, the proposed architecture is compared with the architecture in (Kipf & Welling, 2016). When the synthetic side information is not available, our architecture benefits from correlated recovery to extract the side information. For the real datasets, the architecture is compared with several state-of-the-art methods. These methods have been listed in Table 7 including graph Laplacian regularized methods (Brandes et al., 2007; Zhu et al., 2003; Zhou et al., 2004; Yang et al., 2016) and deep graph embedding methods (Veličković et al., 2017; Zhuang & Ma, 2018; Du et al., 2017; Abu-El-Haija et al., 2018). The comparisons are based on the reported prediction accuracy in each paper for each dataset.

5 RESULTS

In this section, we report the average prediction accuracy on the test set for the proposed architecture by running 100 repeated runs with random initializations for each dataset. The presented results have three parts: investigating the effect of various classifiers (in the correlated recovery block) on the accuracy performance of the proposed architecture, showing the merit of the presented method in dealing with a non-graph observation which is independent of the graph edges, and expressing the superiority of the proposed architecture in comparison to the existing methods. Unless otherwise noted, the experiments in this section follow the hyperparameters represented in Table 3.

Table 4 compares the prediction accuracy of various classifiers in the correlated recovery block in Figure 1. In Table 4, for each dataset, either the r-neighborhood matrix A_r or the feature matrix X is considered as the classifier input. For each classifier and each dataset, A_r and other classifier hyperparameters have been chosen appropriately to maximize the accuracy on the validation set. Note that for k-SBM datasets the feature matrix does not exist, i.e., $X = \mathbf{I}$. Then the extracted side information only based on the feature matrix is not reliable.

Table 4: Prediction accuracy (in percent) of the proposed architecture using various classifiers in the correlated recovery.

Classifier	Input(s)	Cora	Citeseer	Pubmed	k-SBM		
					$k=3$	$k=4$	$k=5$
Neural Network	X	80.3	56.9	78.5	66.9	39.7	25.8
Neural Network	A_r	83.4	74.0	79.8	99.1	94.9	83.4
Gradient Boosting	X	83.5	**74.8**	79.5	33.3	24.8	19.8
Gradient Boosting	A_r	**84.7**	73.0	**81.0**	99.1	95.1	84.8
Graph Convolution Network	X, A	83.1	74.2	79.5	93.8	82.4	64.3
Graph Convolution Network	A_r, A	83.4	73.5	80.4	**99.3**	**96.8**	**91.2**

Table 5 summarizes the results which compare the proposed method with the GCN (Kipf & Welling, 2016) for both real and synthetic datasets. The results show that without independent side information, the accuracy of the proposed method outperforms the traditional GCN method because it benefits from the extracted side information. Also, Table 5 makes a comparison between the quality of the extracted side information and synthetic noisy labels side information with various noise parameters α.

Note that in Table 5, the synthetic side information is not combined with the feature matrix because it is assumed that the quality of the side information is unknown. If the synthetic side information has enough and acceptable quality, it can be embedded in the feature matrix. This embedding improves the accuracy of both the classical GCN and the proposed architecture. But if the side information does not have enough quality, embedding reduces the accuracy of both methods dramatically. Considering this fact, Table 6 shows the results when the synthetic side information is combined with the feature matrix for both classical GCN and the proposed architecture. Then we need to create a

Table 5: Prediction accuracy (in percent) of the proposed architecture and GCN (Kipf & Welling, 2016) in the presence of extracted or synthetic side information.

Method	Synthetic Side Information	Cora	Citeseer	Pubmed	k-SBM		
					$k=3$	$k=4$	$k=5$
GCN (Kipf & Welling, 2016)	without	81.5	70.3	79.0	96.5	86.9	75.1
Active GCN (ours)	without	84.7	74.8	81.0	99.3	96.7	90.6
Active GCN (ours)	$\alpha = 0.7$	85.8	75.3	80.9	99.4	97.2	93.0
Active GCN (ours)	$\alpha = 0.5$	85.1	74.9	80.3	99.2	96.7	90.8
Active GCN (ours)	$\alpha = 0.3$	84.1	73.9	79.4	99.0	95.5	86.3

new feature matrix by combining the side information with the feature matrix X. Therefore, for real datasets, the new feature matrix is created by stacking the one-hot representation of synthetic side information to the given feature matrix. Also, for synthetic datasets, the one-hot representation of side information is used as a newly created feature matrix instead of the identity matrix.

Table 6: Prediction accuracy (in percent) of the proposed architecture and GCN (Kipf & Welling, 2016) in the presence of synthetic side information embedded in the feature matrix.

Method	Synthetic Side Information	Cora	Citeseer	Pubmed	k-SBM		
					$k=3$	$k=4$	$k=5$
GCN (Kipf & Welling, 2016)	$\alpha = 0.7$	86.4	76.4	82.7	98.4	96.0	92.4
Active GCN (ours)	$\alpha = 0.7$	88.7	80.6	83.3	98.6	96.2	92.9
GCN (Kipf & Welling, 2016)	$\alpha = 0.5$	83.6	72.6	74.7	87.1	84.2	78.7
Active GCN (ours)	$\alpha = 0.5$	86.7	77.7	74.5	83.5	83.2	78.6
GCN (Kipf & Welling, 2016)	$\alpha = 0.3$	81.0	68.8	66.8	32.1	42.0	44.4
Active GCN (ours)	$\alpha = 0.3$	84.0	74.0	60.9	29.1	37.3	43.1

Finally, the accuracy of the proposed architecture is compared with the reported accuracy of several state-of-the-art methods. The results are summarized in Table 7. The proposed architecture achieves higher accuracy in comparison to all existing methods for Cora, Citeseer, and Pubmed datasets. The results verify the proposed idea in Section 3 that improves the prediction accuracy by revealing more node labels and allowing the nodes of a graph to access more information about the other nodes.

Table 7: Prediction accuracy (in percent) of various semi-supervised node classification methods.

Method	Cora	Citeseer	Pubmed
Modularity Clustering (Brandes et al., 2007)	59.5	60.1	70.7
SemiEmb (Weston et al., 2012)	59.0	59.6	71.1
DeepWalk (Zhou et al., 2004)	67.2	43.2	65.3
Gaussian Fields (Zhu et al., 2003)	68.0	45.3	63.0
Graph Embedding (Planetoid) (Yang et al., 2016)	75.7	64.7	77.2
DCNN (Atwood & Towsley, 2016)	76.8	-	73.0
GCN (Kipf & Welling, 2016)	81.5	70.3	79.0
MoNet (Monti et al., 2017)	81.7	-	78.8
N-GCN (Abu-El-Haija et al., 2018)	83.0	72.2	79.5
GAT (Veličković et al., 2017)	83.0	72.5	79.0
AGNN (Thekumparampil et al., 2018)	83.1	71.7	79.9
TAGCN (Du et al., 2017)	83.3	72.5	79.0
DGCN (Zhuang & Ma, 2018)	83.5	72.6	80.0
LSM-GAT (Ma et al., 2019)	82.9	73.1	77.6
SBM-GCN (Ma et al., 2019)	82.2	74.5	78.4
Active GCN (ours)	**84.7**	**74.8**	**81.0**

REFERENCES

Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 670–688. IEEE, 2015.

Sami Abu-El-Haija, Amol Kapoor, Bryan Perozzi, and Joonseok Lee. N-gcn: Multi-scale graph convolution for semi-supervised node classification. *arXiv preprint arXiv:1802.08888*, 2018.

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

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (Nov):2399–2434, 2006.

Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE transactions on knowledge and data engineering*, 20(2):172–188, 2007.

Jian Du, Shanghang Zhang, Guanhang Wu, José MF Moura, and Soummya Kar. Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*, 2017.

Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

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.

Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in neural information processing systems*, pp. 10456–10465, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive graph convolutional neural networks. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. A flexible generative framework for graph-based semi-supervised learning. In *Advances in Neural Information Processing Systems*, pp. 3276–3285, 2019.

Qiaozhu Mei, Duo Zhang, and ChengXiang Zhai. A general optimization framework for smoothing language models on graph structures. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 611–618, 2008.

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

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.

David Rolnick, Andreas Veit, Serge Belongie, and Nir Shavit. Deep learning is robust to massive label noise. *arXiv preprint arXiv:1705.10694*, 2017.

Hussein Saad and Aria Nosratinia. Community detection with side information: Exact recovery under the stochastic block model. *IEEE Journal of Selected Topics in Signal Processing*, 12(5): 944–958, 2018.

Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 442–457. Springer, 2009.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

Kiran K Thekumparampil, Chong Wang, Sewoong Oh, and Li-Jia Li. Attention-based graph neural network for semi-supervised learning. *arXiv preprint arXiv:1803.03735*, 2018.

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

Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade*, pp. 639–655. Springer, 2012.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, pp. 499–508, 2018.