Learning Robust Graph Neural Networks with Limited Supervision

Abdullah Alchihabi[*] **Yuhong Guo**[*†]
[*]School of Computer Science, Carleton University, Ottawa, Canada
[†]Canada CIFAR AI Chair, Amii, Canada
abdullahalchihabi@cmail.carleton.ca, yuhong.guo@carleton.ca

Abstract

Graph Neural Networks (GNNs) require a relatively large number of labeled nodes and a reliable/uncorrupted graph connectivity structure to obtain good performance on the semi-supervised node classification task. The performance of GNNs can degrade significantly as the number of labeled nodes decreases or the graph connectivity structure is corrupted by adversarial attacks or noise in data measurement/collection. Therefore, it is important to develop GNN models that are able to achieve good performance when there is limited supervision knowledge—a few labeled nodes and a noisy graph structure. In this paper, we propose a novel Dual GNN learning framework to address this challenging task. The proposed framework has two GNN based node prediction modules. The primary module uses the input graph structure to induce typical node embeddings and predictions with a regular GNN baseline, while the auxiliary module constructs a new graph structure through fine-grained spectral clustering and learns new node embeddings and predictions. By integrating the two modules in a dual GNN learning framework, we perform joint learning in an end-to-end fashion. This general framework can be applied on many GNN baseline models. The experimental results show that the proposed dual GNN framework can greatly outperform the GNN baseline methods and yield superior performance over many state-of-the-art methods when the labeled nodes are scarce and the graph connectivity structure is noisy.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206. Copyright 2023 by the author(s).

1 INTRODUCTION

Graph Neural Networks (GNNs) have been successfully employed to solve multiple tasks such as node classification, graph completion, and edge prediction across a variety of application domains, including computational chemistry (Shi et al., 2020), protein-protein interactions (Zitnik and Leskovec, 2017) and knowledge-base completion (Schlichtkrull et al., 2018). In particular, many GNN advancements have addressed the typical node classification task in a semi-supervised learning setting where only a subset of nodes in the graph are labeled, including the well known Graph Convolutional Networks (GCNs) (Kipf and Welling, 2017), Graph Attention Networks (GATs) (Veličković et al., 2018), Topology Adaptive Graph Convolutional Networks (TAGs) (Du et al., 2017), and Dynamic Neighborhood Aggregation networks (DNAs) (Fey, 2019). These GNN models have achieved great results on the benchmark GNN learning datasets. However, they typically require a relatively large number of labeled nodes as well as a reliable/uncorrupted graph structure to obtain good performance. Their performance can degrade significantly as the number of labeled nodes decreases (Li et al., 2018; Lin et al., 2020) or when the graph structures are noisy or corrupted (Wang et al., 2020; Chen et al., 2020).

The performance drop of GNNs with severely limited labeled data can be explained by the inability of GNNs to propagate the label information from the labeled nodes to the rest of the graph. That is, while it is known that deep GNN architectures can cause over-smoothing problems (Li et al., 2018), a relatively shallow GNN architecture can fail to propagate messages across the whole graph and cause the classifier to overfit the small neighborhoods of the labeled nodes. When there are very small number of labeled nodes, this will induce a serious overfitting problem and degrade the classification performance (Li et al., 2018; Sun et al., 2020; Lin et al., 2020). In addition, GNNs learn discriminative node embeddings for effective node classification by propagating messages across the edges of the graph. Hence noisy or corrupted graph structures can greatly impair the message passing process and degrade the ultimate node classification performance (Li et al., 2018;

Wang et al., 2020). The dependence of GNN models on a relatively large number of labeled nodes and reliable graph structures can seriously limit their applications, since providing sufficient numbers of labeled nodes and reliable clean graph structures usually requires substantial expert-level supervision knowledge and effort, which can induce unendurable high cost in many domains. Moreover, it is also very difficult to guarantee reliable/uncorrupted graph structures given the numerous sources of noise that could damage the graph connectivity structures, ranging from adversarial attacks on the graph structures to data collection or measurement noise (Wang et al., 2020; Chen et al., 2020). Therefore, it is important to develop GNN models that can learn efficiently with limited labeled nodes and robustly with corrupted/unreliable graph structures.

In this work, we propose a novel dual GNN learning framework that is resilient to very low label rates and corrupted/noisy graph structures. The proposed framework is made up of two node prediction modules. The first module can be treated as a standard primary GNN model, which takes the original graph data as input. It suffers from the aforementioned message propagation drawbacks when labeled nodes are scarce and graph structures are noisy. To address this problem, the second GNN module employs a fine-grained spectral clustering method based on the node embedding results of the first module to construct a new adjacency matrix and hence a new graph structure, aiming to alleviate graph noise, enable effective information propagation across the graph, and facilitate the subsequent node embedding and node classification learning. The two modules coordinate with each other within the integrated dual learning framework to perform end-to-end training with a joint objective function. This general dual learning framework can be applied on many standard GNN baseline models. We conduct experiments with four baseline graph neural network learning models. The experimental results show that the proposed framework can significantly improve the baseline models as well as outperform many state-of-the-art methods when the labeled nodes are very scarce and the graph structure is noisy/corrupted across multiple benchmark datasets.

2 RELATED WORKS

Graph Neural Network (GNN) learning has received a lot of attention in the research community. Many early works developed spectral approaches for GNN learning based on graph Laplacians (Defferrard et al., 2016; Niepert et al., 2016; Kipf and Welling, 2017). For example, Defferrard et al. (2016) employed the Chebyshev polynomials to parameterize the learned filters and localize them in the spectral domain. Kipf and Welling (2017) proposed a Graph Convolutional Network (GCN) model that uses a first-order approximation of the Chebyshev polynomial with a re-normalization trick to ensure numerical stability during training. Another set of works operate directly in the spatial domain by defining novel message-passing and message-aggregation functions (Veličković et al., 2018; Du et al., 2017; Fey, 2019). Veličković et al. (2018) proposed a Graph Attention Network (GAT) that uses a self-attention mechanism to assign importance weights to the edges of graphs. Du et al. (2017) proposed a Topology Adaptive Graph Convolutional Network (TAG) that aggregates messages from different powers of the adjacency matrix to enlarge the receptive field of the nodes. Fey (2019) proposed a Dynamic Neighborhood Aggregation (DNA) model that defines dynamic receptive fields for each node by allowing some nodes to aggregate global information from the graph while other nodes focus on local information. Hamilton et al. (2017) proposed an inductive approach, GraphSAGE, that employs neighborhood sampling and several different neighborhood aggregators to learn node representations. Wu et al. (2019a) proposed an efficient GNN variant (SGC) that drops the non-linear functions between the GNN layers and collapses the weight matrices of the network into a single weight matrix. Most of these previous works have focused on developing powerful GNNs with a greater representation power to solve challenging graph-related tasks, including node classification.

More recently, a number of methods have been proposed to address some notable limitations of the standard GNN models, including the vulnerability to adversarial attacks (Elinas et al., 2020; You et al., 2020; Zhang and Zitnik, 2020; Geisler et al., 2020; Wang et al., 2020), over-smoothing (Li et al., 2018; Rong et al., 2020; Min et al., 2020), and performance degradation with scarce labels (Zhou et al., 2019; Wang et al., 2020; Calder et al., 2020; Lin et al., 2020). In particular, the label scarcity problem has been addressed by using various strategies, including self-supervision (Sun et al., 2020), self-training (Zhou et al., 2019), and metric learning (Lin et al., 2020). Lin et al. (2020) proposed a new framework that employs metric learning to improve the performance of GNNs under very low label rates. Zhou et al. (2019) used dynamic self-training to increase the size of the training set and alleviate overfitting problems when the labels are scarce. Sun et al. (2020) used self-supervision and self-training to augment the training set with confidently labeled nodes, whose self-supervised labels match their predicted pseudo-labels, to increase the number of labeled samples. Wan et al. (2021) proposed a new framework, named as Graph Contrastive Graph Poisson Network (CGPN-GCN), which integrates a contrastive learning objective with a novel message-passing model—Graph Poisson Network (GPN)—to help propagate the label information across the graph.

In addition, several works have proposed methods to induce robustness to noisy or corrupted graph structures. Zhu et al. (2019) proposed a RGCN model, where Gaussian distributions are used to represent the embeddings of nodes in order to absorb the negative effect of the adversarial attacks to the covariance of the distributions. Wu et al. (2019b) proposed a preprocessing method, GCN-Jaccard, to defend against adversarial attacks by deleting edges that connect nodes with low Jaccard similarities. Entezari et al. (2020) proposed a pre-processing method, GCN-SVD, to obtain low-rank estimates for the attacked graph structures using truncated SVD. Jin et al. (2020) presented a Pro-GNN model that learns a clean graph structure jointly with the GNN model by optimizing the low-rank, sparsity and homophily properties of the estimated graph structure. A few other works have proposed to defend against adversarial attacks on graph structures by using iterative deep graph learning frameworks (Chen et al., 2020), novel message aggregation functions (Geisler et al., 2020), and variational inferences (Elinas et al., 2020).

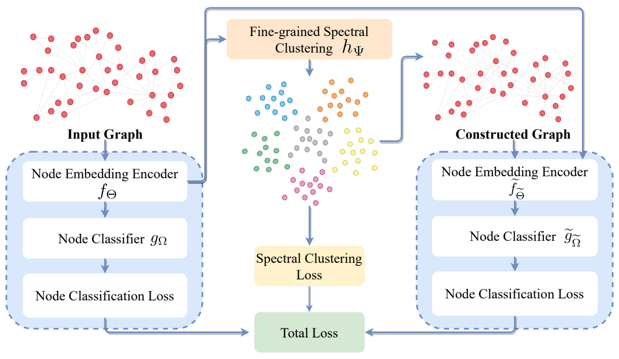

Fine-grained Spectral Clustering h_Ψ
Input Graph
Constructed Graph
Node Embedding Encoder f_Θ
Node Classifier g_Ω
Node Classification Loss
Spectral Clustering Loss
Total Loss
Node Embedding Encoder $\tilde{f}_{\tilde{\Theta}}$
Node Classifier $\tilde{g}_{\tilde{\Omega}}$
Node Classification Loss

Figure 1: An illustration of the proposed Dual GNN learning framework. The framework contains two modules: The primary module is on the left side, and the auxiliary module is on the right side.

3 METHOD

3.1 Problem Setup

We consider the following transductive semi-supervised node classification setting. The input is a graph $G = (V, E)$, where V is the set of nodes with size $|V| = N$ and E is the set of edges. E is typically represented by an adjacency matrix A of size $N \times N$, which can be either symmetric (for undirected graphs) or asymmetric (for directed graphs), with either binary indicator values or real-valued weights. This adjacency matrix A encodes the input graph structure, which can be corrupted. Each node in the graph G is associated with a corresponding feature vector of size D. The feature vectors of all the nodes in the graph are represented by an input feature matrix $X \in \mathbb{R}^{N\times D}$. In the context of transductive semi-supervised node classification, the nodes in V are split into two subsets: a subset of labeled nodes V_ℓ and a subset of unlabeled nodes V_u. The labels for V_ℓ are represented using a label indicator matrix $Y^\ell \in \{0,1\}^{N_\ell \times C}$, where C is the number of classes and N_ℓ is the number of labeled nodes.

3.2 Dual GNN Learning Framework

In this section, we present the proposed Dual GNN learning framework for semi-supervised node classification, which empowers a given standard GNN base model to handle the difficult learning scenarios with scarce labeled nodes and noisy graph structures. The framework, as shown in Figure 1, is made up of two GNN based node prediction modules, each of which consists of its own node embedding encoder f and node classifier g. The first module can be treated as a standard *primary GNN model*, which takes the initial node feature matrix X and adjacency matrix A as input. It produces the primary node classifier, which aims to learn discriminative node embeddings by propagating messages from the labeled nodes to the rest of the graph using the input adjacency matrix. However, as aforementioned, due to the scarcity of labeled nodes and the noisy input graph structure, the first GNN module may only be able to propagate the messages to the local neighbourhoods of the labeled nodes while failing to propagate messages to a larger portion of the graph. The second GNN module is designed to address this issue by constructing a new graph adjacency matrix with a fine-grained spectral clustering method based on the node embedding results of the first module. The new adjacency matrix aims to enable effective message propagation across the whole graph to induce new node embeddings and classifier. It can be treated as an *auxiliary module*. The two GNN modules are trained in a joint learning framework, where the learned node embedding of the primary GNN module is updated based on both the local and global information obtained from the primary node classifier and auxiliary node classifier, respectively. Below we present the details of the two modules and the fine-grained clustering component.

3.2.1 Primary GNN Module for Node Prediction

This module maintains the learning capacity of a standard GNN baseline with the original graph information. It takes the initial node feature matrix X and adjacency matrix A as input to learn node embeddings and class prediction probabilities as follows:

$$H = f_\Theta(X, A), \qquad P = g_\Omega(H) \tag{1}$$

where f_Θ denotes the GNN encoder that takes the adjacency matrix A and the initial node features X as input and uses message passing and message aggregation to learn a new embedding matrix $H \in \mathbb{R}^{N\times d}$ for the nodes; g_Ω denotes the node classifier, which takes the learned node embedding H as input and outputs class prediction probability matrix P of the nodes. Θ and Ω denote the model parameters for f and g, respectively. In general, this module can be any GNN model developed in the literature that performs standard graph neural network learning. It can be trained by minimizing the following supervised node classification loss (i.e., cross-entropy loss):

$$\mathcal{L}_{CE} = \sum\nolimits_{i \in V_\ell} \ell_{CE}(P_i, Y_i^\ell) \tag{2}$$

where ℓ_{CE} denotes the cross-entropy function, P_i and Y_i^ℓ denote the predicted class probability vector and the true label indicator vector for node i, respectively.

3.2.2 Fine-grained Spectral Clustering

To build the auxiliary module, we deploy a fine-grained spectral clustering component to construct a new adjacency matrix on the input graph. The spectral clustering function h is a multi-layer perceptron that takes the node embedding matrix H learned from the primary GNN module as input and returns a fine-grained soft-clustering assignment on the graph nodes:

$$S = h_\Psi(H) \tag{3}$$

where $S \in \mathbb{R}^{N \times K}$ is the node soft clustering assignment matrix, K is the number of clusters, and Ψ denotes the parameters of the spectral clustering function h. As each class could contain multiple clusters, we consider fine-grained clustering with a K value that is much larger than the number of classes C.

The fine-grained clustering function h_Ψ is learned by minimizing the following relaxed min-cut spectral clustering loss function $\mathcal{L}_{sc}$:

$$\mathcal{L}_{sc} = -\frac{Tr(S^T \tilde{A} S)}{Tr(S^T \tilde{D} S)} + \left\| \frac{S^T S}{\|S^T S\|_F} - \frac{I_K}{\sqrt{K}} \right\|_F \tag{4}$$

where $\|.\|_F$ denotes the Frobenius norm, I_K is the identity matrix of size $K \times K$, $\tilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ is the normalized input adjacency matrix, and D and $\tilde{D}$ are the degree matrices of A and $\tilde{A}$ respectively, which are calculated as follows:

$$D_{ii} = \sum\nolimits_j A_{ij}, \qquad \tilde{D}_{ii} = \sum\nolimits_j \tilde{A}_{ij} \tag{5}$$

We adopt a commonly used relaxed min-cut loss function $\mathcal{L}_{sc}$ to learn fine-grained clustering through h_Ψ. This formulation offers several advantages over the traditional spectral clustering approach. First, it does not require the expensive eigen-decomposition of the graph Laplacian. Second, it takes both the node embedding matrix H and the original graph connectivity structure A into account. Minimizing the first term of $\mathcal{L}_{sc}$ pushes strongly connected nodes to be clustered together while enforcing the cluster assignment to be learnable from H with function h_Ψ. The second term of $\mathcal{L}_{sc}$ is a regularization term that penalizes degenerated clustering assignments and pushes h_Ψ to generate non-overlapping orthogonal clusters with relatively similar sizes. This fine-grained spectral clustering module is not directly useful but serves as a base for building our auxiliary GNN module below.

3.2.3 Auxiliary GNN Module for Node Prediction

To deploy an auxiliary dual module, we first use the learned clustering assignment matrix S to construct a new adjacency matrix A_{sc} as follows:

$$A_{sc}(i,j) = \begin{cases} r(S_{i,:}, S_{j,:}), & \text{if } r(S_{i,:}, S_{j,:}) \geq \alpha \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $r(.,.)$ is the Pearson correlation function that measures the similarity between the clustering assignment vectors on a pair of nodes, and α is a hyper-parameter that controls the sparsity of the constructed adjacency matrix. We expect the fine-grained spectral clustering function can capture the global geometric information of the original graph structure A and the learned node embedding H. By constructing the adjacency matrix A_{sc} from the clustering assignment matrix S, we aim to address the problems of scarce labeled nodes and noisy edges in the original graph structure. Specifically, as a pair of nodes that are either connected in A or have similar embeddings in H tend to have similar clustering assignments through the spectral clustering function, it enables A_{sc} not only to maintain the connectivity of the original A, but also to add more edges based on the embedding similarities of the nodes. This helps propagate the local messages and node label information to a larger portion of the graph, addressing the underlying local overfitting problem caused by scarce labeled nodes and corrupted graph structures with many deleted edges.

Given the new adjacency matrix A_{sc}, the auxiliary GNN node prediction module has a standard architecture with a GNN encoder $\tilde{f}$ and a classifier $\tilde{g}$. The encoder $\tilde{f}$ takes the node embedding H from the primary module and the new constructed adjacency matrix A_{sc} as input and outputs a node embedding matrix $\tilde{H}$, while the node classifier $\tilde{g}$ further predicts the classification probability matrix $\tilde{P}$ on all the nodes:

$$\tilde{H} = \tilde{f}_{\tilde{\Theta}}(H, A_{sc}), \qquad \tilde{P} = \tilde{g}_{\tilde{\Omega}}(\tilde{H}) \tag{7}$$

where $\tilde{\Theta}$ and $\tilde{\Omega}$ denote the model parameters for the encoder and classifier respectively. Specifically we adopt the encoder of the standard Graph Convolutional Networks (GCN) (Kipf and Welling, 2017) as our encoder $\tilde{f}$. We choose to use the GCN encoder due to two reasons. First, GCNs are simple and efficient with relatively fewer learnable parameters. Second, unlike other GNN models such as GATs, GCNs allow us to utilize the weights of the constructed adjacency matrix A_{sc}. This auxiliary GNN module can be trained in a standard way by minimizing the node classification loss (i.e., cross-entropy loss):

$$\tilde{\mathcal{L}}_{CE} = \sum\nolimits_{i \in V_\ell} \ell_{CE}(\tilde{P}_i, Y_i^\ell) \tag{8}$$

3.2.4 Joint Dual GNN Learning

Finally we can integrate the classification loss functions, $\mathcal{L}_{CE}$ and $\tilde{\mathcal{L}}_{CE}$, from the two modules and the clustering loss, $\mathcal{L}_{sc}$, together to form a joint dual GNN learning problem as follows:

$$\min_{\Theta,\Omega,\tilde{\Theta},\tilde{\Omega},\Psi} \mathcal{L} = \mathcal{L}_{CE} + \tilde{\mathcal{L}}_{CE} + \mathcal{L}_{sc} \tag{9}$$

Note as the embedding matrix H from the primary module is used as inputs for the spectral clustering and the auxiliary module, the auxiliary loss $\tilde{\mathcal{L}}_{CE}$ and the clustering loss $\mathcal{L}_{sc}$ are also functions of the primary encoding parameters Θ. With such a joint learning framework, the two modules can interactively impact each other through the shared encoder f_Θ. The entire network is trained in an end-to-end manner by minimizing the joint loss function $\mathcal{L}$.

4 EXPERIMENTS

We tested the proposed Dual GNN learning framework by applying it on multiple GNN baseline models and conducted experiments with three different experimental setups: learning with few labeled nodes, learning with noisy graph structures, and learning with both few labeled nodes and noisy graph structures.

4.1 Experiment Settings

4.1.1 Datasets & Baselines

We used the citation network datasets (Cora, CiteSeer) (Sen et al., 2008) and the Wikipedia network dataset (Chameleon) (Pei et al., 2019). The three datasets, Cora, CiteSeer and Chameleon, have seven, six and five classes respectively, and their average node degrees are 4.89, 3.73 and 28.55 respectively. For the citation datasets, we used the same train/validation/test/unlabeled node splits as in (Yang et al., 2016), where the training set consists of 20 nodes per class, the validation set consists of 500 nodes, the test set consists of 1000 nodes, while the remaining nodes in the graph are used as unlabeled nodes. For the Wikipedia dataset, we randomly generated the validation and test sets with equal corresponding sizes to those of the citation datasets. We applied our proposed Dual GNN framework on four GNN baselines by using each of them as our primary module: Graph Convolution Networks (GCN) (Kipf and Welling, 2017), Graph Attention Networks (GAT) (Veličković et al., 2018), Topology Adaptive Graph Convolutional Networks (TAG) (Du et al., 2017), and Dynamic Neighborhood Aggregation in Graph Neural Networks (DNA) (Fey and Lenssen, 2019).

4.1.2 Implementation Details

For the Dual GNN framework and the baselines, the node embedding encoders are made up of two message-passing layers, while the node classifiers contain a single fully connected layer. The spectral clustering function h is a one-layer perceptron. The non-linear activation function applied on each message-passing layer is the exponential linear units (ELUs). We apply L_2 regularization to all model parameters, with a weight decay hyperparameter value of $1e^{-3}$ on the citation datasets and $1e^{-4}$ on Chameleon. GAT has a single attention head with an attention dropout rate of 0.5. DNA has an additional single fully-connected layer prior to the message-passing layers to project the initial node embeddings to a lower dimensional space with size 32. All networks are trained for 500 training epochs using the Adam optimizer with a learning rate of $1e^{-2}$ and a scheduler (step size = 50 training iterations) with a learning rate decay factor of $\gamma = 0.5$. For the Dual GNN framework, the number of clusters is set to $K = 10 \times C$, where C is the number of classes, and the sparsity threshold for constructing the new adjacency matrix is $\alpha = 0.7$. In the case of learning with noisy additional edges, we first sparsify the input graph structure by dropping edges between nodes with pairwise similarities below a similarity threshold β before applying the proposed Dual GNN. The Pearson correlation function is used to measure the similarity between each pair of nodes based on their input features. The similarity threshold β takes value 0.1 for the two citation datasets (Cora and CiteSeer) and takes value 0.01 for the Chameleon dataset.

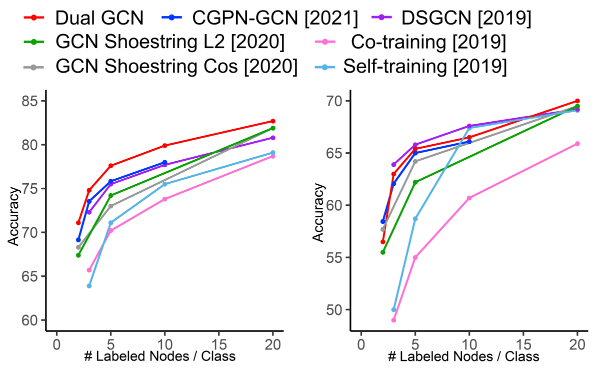
4.2 Experiments with Few Labeled Nodes

In this set of experiments, we aim to investigate the performance of the Dual GNN framework with significantly fewer labeled nodes. The public split of the citation datasets contains 20 labeled training nodes per class. We tested several much smaller label rates by using $\{2, 3, 5, 10\}$ labeled nodes per class, respectively. On the citation datasets, for each label rate, we generated 10 random subsets of labeled nodes from the training set in the public split. For each random subset, we repeated 5 runs for each comparison method. We report the mean test accuracy results and the corresponding standard deviations over the total 50 runs. We also conducted experiments by using all the 20 labeled nodes per class in the public training split of the citation datasets (Yang et al., 2016) and report the average results over 5 runs. For the Chameleon dataset, the 10 labeled subsets for each label rate are randomly selected from the set of all graph nodes, excluding the validation set and the test set.

We applied the Dual GNN framework over four baseline models. The comparison results over each baseline model and the corresponding Dual method are reported in Table 1. We can see that the performance of all four baselines (GCN, GAT, TAG, DNA) degrades as the number of labeled nodes per class decreases. The performance of the DNA baseline degrades more substantially compared to the

Table 1: Mean classification accuracy (standard deviation is within brackets) on the Cora (left part), CiteSeer (middle part) and Chameleon (right part) datasets with a few labeled nodes per class (2, 3, 5, 10, 20).

	Cora					**CiteSeer**					**Chameleon**				
	2	3	5	10	20	2	3	5	10	20	2	3	5	10	20
GCN	63.5$_{(4.4)}$	69.2$_{(3.5)}$	75.1$_{(3.1)}$	78.8$_{(0.9)}$	81.5$_{(0.6)}$	48.1$_{(7.9)}$	56.4$_{(6.2)}$	62.2$_{(3.5)}$	66.3$_{(1.8)}$	68.5$_{(0.7)}$	25.1$_{(5.0)}$	24.5$_{(4.3)}$	30.8$_{(3.9)}$	35.5$_{(4.4)}$	40.4$_{(3.4)}$
Dual GCN	**71.1**$_{(4.0)}$	**74.8**$_{(2.4)}$	**77.6**$_{(2.6)}$	**79.9**$_{(1.0)}$	**82.7**$_{(0.5)}$	**56.5**$_{(9.1)}$	**63.0**$_{(5.7)}$	**65.4**$_{(4.1)}$	**66.5**$_{(2.4)}$	**70.0**$_{(1.5)}$	**25.7**$_{(4.9)}$	**26.6**$_{(4.0)}$	**31.1**$_{(5.0)}$	**35.9**$_{(4.7)}$	**40.9**$_{(3.1)}$
GAT	63.5$_{(3.9)}$	69.6$_{(3.4)}$	74.0$_{(3.5)}$	79.0$_{(1.2)}$	81.3$_{(0.5)}$	49.2$_{(8.1)}$	56.0$_{(6.6)}$	61.8$_{(3.3)}$	65.2$_{(1.9)}$	69.0$_{(0.6)}$	25.5$_{(4.7)}$	25.4$_{(4.0)}$	30.1$_{(4.5)}$	36.1$_{(4.0)}$	42.2$_{(3.2)}$
Dual GAT	**65.7**$_{(5.2)}$	**71.6**$_{(2.6)}$	**75.4**$_{(2.7)}$	**79.1**$_{(1.4)}$	**81.6**$_{(0.3)}$	**50.8**$_{(7.3)}$	**58.5**$_{(5.3)}$	**63.2**$_{(3.5)}$	**65.9**$_{(2.0)}$	**69.0**$_{(0.5)}$	**26.3**$_{(5.0)}$	**26.5**$_{(4.4)}$	**30.5**$_{(4.5)}$	**36.6**$_{(4.7)}$	**43.3**$_{(2.2)}$
TAG	**67.7**$_{(4.3)}$	73.0$_{(2.7)}$	76.3$_{(2.8)}$	79.9$_{(1.2)}$	**82.6**$_{(0.4)}$	**50.7**$_{(9.8)}$	**58.5**$_{(6.6)}$	**63.6**$_{(3.1)}$	**67.7**$_{(1.3)}$	**70.0**$_{(0.9)}$	23.5$_{(4.5)}$	24.1$_{(3.7)}$	28.1$_{(4.3)}$	32.2$_{(4.5)}$	38.2$_{(4.3)}$
Dual TAG	67.5$_{(6.8)}$	**75.2**$_{(3.5)}$	**77.5**$_{(2.6)}$	**80.0**$_{(1.5)}$	82.0$_{(0.5)}$	49.0$_{(1.1)}$	57.2$_{(8.5)}$	61.9$_{(4.3)}$	64.7$_{(2.6)}$	67.9$_{(1.2)}$	**23.6**$_{(5.2)}$	**24.2**$_{(4.3)}$	**29.8**$_{(4.8)}$	**35.8**$_{(5.2)}$	**44.3**$_{(2.5)}$
DNA	56.7$_{(5.3)}$	65.1$_{(3.2)}$	70.3$_{(3.3)}$	77.0$_{(1.2)}$	**81.3**$_{(0.5)}$	43.9$_{(7.1)}$	53.0$_{(5.8)}$	58.6$_{(3.5)}$	64.4$_{(2.1)}$	69.5$_{(0.7)}$	**25.3**$_{(3.7)}$	26.9$_{(3.9)}$	29.0$_{(4.5)}$	**33.9**$_{(3.2)}$	**35.0**$_{(2.9)}$
Dual DNA	**62.8**$_{(5.0)}$	**69.2**$_{(3.5)}$	**74.0**$_{(3.0)}$	**77.6**$_{(1.2)}$	81.0$_{(0.3)}$	**51.5**$_{(8.0)}$	**60.8**$_{(5.0)}$	**64.3**$_{(3.5)}$	**67.1**$_{(2.1)}$	**70.1**$_{(0.6)}$	24.3$_{(4.1)}$	**27.2**$_{(3.9)}$	**30.2**$_{(4.5)}$	33.3$_{(3.1)}$	34.6$_{(3.0)}$



(a) Cora (b) CiteSeer

Figure 2: Comparison results in terms of mean classification accuracy for the proposed Dual GCN and several state-of-the-art methods on the Cora and CiteSeer datasets with a few labeled nodes.

other three baselines on Cora and CiteSeer, which can be explained by its relatively large number of learnable parameters. The TAG baseline performs better than the other baselines at low label rates on Cora and CiteSeer, which can be attributed to its inherently large receptive field that allows messages to be propagated to a larger portion of the graph (Luan et al., 2019). This can also explain the relatively similar performance between Dual TAG and TAG on the citation datasets. The proposed framework nevertheless consistently improves the performance of all the other three base models, GCN, GAT and DNA, across all label rates on the citation datasets. The performance gains achieved by the Dual GNN framework are particularly remarkable with smaller label rates. For example, Dual GCN outperforms GCN by 7.6% and 8.4% on Cora and CiteSeer, respectively, with two labeled nodes per class. On Chameleon, where the graph is significantly more densely connected than the citation graphs, our Dual GNN framework still improves the GCN, GAT and TAG baselines across all label rates. These results suggest that the proposed Dual GNN framework is very beneficial for GNN learning with smaller numbers of labeled nodes.

We also compared the proposed Dual GNN framework with a number of methods developed in the literature that address the performance degradation of GNNs with few labeled nodes, including Contrastive Graph Poisson Networks (CGPN-GCN) (Wan et al., 2021), two Shoestring methods (GCN Shoestring L2, GCN Shoestring Cos) (Lin et al., 2020), and three Dynamic self-training methods (Co-training, Self-training, and DSGCN) (Zhou et al., 2019). We used the reported results from (Wan et al., 2021), (Lin et al., 2020) and (Zhou et al., 2019). We used GCN as the primary module and compared our proposed Dual GCN with these six comparison methods with different label rates. The comparison results on the two citation datasets, Cora and CiteSeer, are reported in Figure 2. We can see that Dual GCN consistently outperforms all the other methods across all label rates with notable performance gains on the Cora dataset – the average performance gain is about +2%. On CiteSeer, although Dual GCN is outperformed by DSGCN, the difference between them is very small, while Dual GCN outperforms the other comparison methods.

4.3 Experiments with Noisy Graph Structures

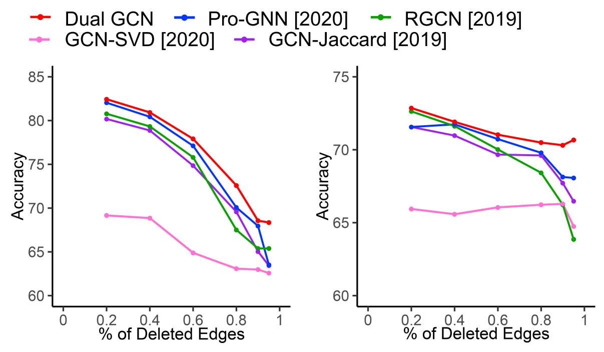
In the second set of experiments, we investigate the robustness of the proposed Dual GNN framework to noisy/corrupted graph structures with missing or additional edges. We simulate untargeted adversarial attacks by randomly deleting edges or adding noisy edges in the input graphs, which is similar to the adversarial setups in some previous works (Elinas et al., 2020; Geisler et al., 2020; Chen et al., 2020). To corrupt the input graph structure using missing edges, we randomly drop a portion of the existing edges in the adjacency matrix of the graph. We considered a set of different edge drop ratios, i.e., corruption rates, in $\{0.25, 0.50, 0.75, 0.90, 0.95\}$. To corrupt the input graph structure using noisy additional edges, we randomly add a number of noisy edges to the adjacency matrix of the graph. We considered the following set of different numbers of noisy additional edges: $\{2000, 3000, 5000, 7000, 10,000\}$. For each corruption rate or each number of noisy additional edges, we generate 10 corrupted input adjacency matrices and experiment using each resulting graph for 5 times with 20 labeled nodes per class. We report the mean test classification accuracy and the corresponding standard deviation across all runs for each case.

Table 2: Mean classification accuracy (standard deviation is within brackets) on the Cora (left part), CiteSeer (middle part) and Chameleon (right part) datasets with different edge corruption (deletion) rates (0.25, 0.5, 0.75, 0.9, 0.95).

	Cora					**CiteSeer**					**Chameleon**				
	0.25	0.5	0.75	0.90	0.95	0.25	0.5	0.75	0.90	0.95	0.25	0.5	0.75	0.90	0.95
GCN	79.1$_{(1.0)}$	74.7$_{(1.5)}$	67.4$_{(1.8)}$	60.1$_{(1.7)}$	57.9$_{(1.6)}$	67.2$_{(1.1)}$	64.5$_{(1.1)}$	58.7$_{(1.7)}$	55.7$_{(1.8)}$	54.9$_{(1.7)}$	40.0$_{(3.5)}$	39.6$_{(3.6)}$	38.3$_{(3.3)}$	36.7$_{(3.0)}$	37.2$_{(1.9)}$
Dual GCN	**79.9**$_{(0.8)}$	**75.9**$_{(1.5)}$	**69.0**$_{(1.7)}$	**64.1**$_{(2.0)}$	**60.8**$_{(1.8)}$	**67.7**$_{(1.4)}$	**66.2**$_{(2.6)}$	**64.2**$_{(3.1)}$	**60.6**$_{(3.1)}$	**61.4**$_{(2.1)}$	**40.9**$_{(3.1)}$	**40.2**$_{(3.3)}$	**39.5**$_{(2.5)}$	**37.8**$_{(2.3)}$	**37.9**$_{(1.8)}$
GAT	78.7$_{(1.1)}$	75.2$_{(1.3)}$	68.9$_{(1.8)}$	**61.4**$_{(1.8)}$	58.2$_{(1.6)}$	67.2$_{(1.1)}$	65.1$_{(1.2)}$	59.4$_{(1.9)}$	56.4$_{(2.3)}$	54.3$_{(2.3)}$	**42.6**$_{(2.8)}$	**42.4**$_{(2.5)}$	**40.5**$_{(2.0)}$	**38.7**$_{(2.1)}$	**38.0**$_{(1.3)}$
Dual GAT	**78.9**$_{(0.8)}$	**75.3**$_{(1.5)}$	**68.9**$_{(1.6)}$	60.6$_{(2.3)}$	**59.3**$_{(2.3)}$	**67.6**$_{(1.3)}$	**65.2**$_{(1.6)}$	**59.6**$_{(1.9)}$	**57.3**$_{(2.7)}$	**59.2**$_{(2.2)}$	41.6$_{(2.5)}$	**42.4**$_{(2.1)}$	40.2$_{(1.9)}$	38.3$_{(2.3)}$	**38.0**$_{(1.2)}$
TAG	77.6$_{(1.0)}$	70.6$_{(1.6)}$	60.0$_{(1.9)}$	55.3$_{(1.6)}$	54.6$_{(1.6)}$	**66.3**$_{(1.1)}$	62.2$_{(1.1)}$	55.3$_{(0.9)}$	53.9$_{(1.2)}$	53.9$_{(0.8)}$	37.7$_{(5.0)}$	37.6$_{(4.3)}$	36.6$_{(4.0)}$	37.1$_{(4.5)}$	38.7$_{(3.8)}$
Dual TAG	**78.8**$_{(1.5)}$	**72.4**$_{(1.7)}$	**64.2**$_{(2.0)}$	**57.4**$_{(1.6)}$	**57.8**$_{(2.1)}$	65.0$_{(1.8)}$	**62.6**$_{(2.8)}$	**57.7**$_{(3.3)}$	**57.8**$_{(4.3)}$	**60.6**$_{(2.5)}$	**43.7**$_{(2.8)}$	**43.2**$_{(2.8)}$	**41.3**$_{(3.7)}$	**40.5**$_{(2.9)}$	**40.1**$_{(2.8)}$
DNA	78.7$_{(0.7)}$	74.9$_{(1.1)}$	68.9$_{(1.5)}$	62.6$_{(1.8)}$	60.3$_{(1.0)}$	67.6$_{(1.2)}$	65.5$_{(1.1)}$	60.7$_{(1.1)}$	57.9$_{(1.3)}$	57.1$_{(1.3)}$	34.9$_{(2.7)}$	34.9$_{(2.4)}$	36.4$_{(3.5)}$	37.0$_{(2.3)}$	37.6$_{(2.5)}$
Dual DNA	**79.0**$_{(1.3)}$	**76.0**$_{(1.3)}$	**70.2**$_{(1.3)}$	**63.9**$_{(1.9)}$	**61.6**$_{(1.5)}$	**68.9**$_{(1.2)}$	**67.6**$_{(1.1)}$	**62.9**$_{(1.8)}$	**60.0**$_{(2.2)}$	**60.8**$_{(1.9)}$	**35.5**$_{(3.9)}$	**35.5**$_{(4.5)}$	**37.3**$_{(3.5)}$	**37.4**$_{(2.2)}$	**37.8**$_{(2.2)}$

Table 3: Mean classification accuracy (standard deviation is within brackets) on the Cora (left part), CiteSeer (middle part) and Chameleon (right part) datasets with different numbers of noisy additional edges (2000, 3000, 5000, 7000, 10,000).

	Cora					**CiteSeer**					**Chameleon**				
	2000	3000	5000	7000	10,000	2000	3000	5000	7000	10,000	2000	3000	5000	7000	10,000
GCN	**76.0**$_{(0.9)}$	74.3$_{(1.2)}$	70.6$_{(1.1)}$	67.2$_{(1.3)}$	63.3$_{(1.6)}$	62.5$_{(1.0)}$	59.4$_{(1.5)}$	55.8$_{(1.8)}$	51.5$_{(1.1)}$	48.9$_{(1.9)}$	**40.5**$_{(2.2)}$	**40.5**$_{(2.5)}$	**39.2**$_{(2.6)}$	**38.8**$_{(2.0)}$	**38.6**$_{(2.2)}$
Dual GCN	75.1$_{(1.1)}$	**74.6**$_{(1.3)}$	**73.3**$_{(1.2)}$	**72.4**$_{(1.2)}$	**70.5**$_{(1.1)}$	**66.5**$_{(2.0)}$	**66.3**$_{(1.8)}$	**65.7**$_{(1.9)}$	**65.6**$_{(2.1)}$	**63.6**$_{(1.5)}$	39.2$_{(2.3)}$	38.2$_{(2.2)}$	37.9$_{(2.6)}$	38.4$_{(2.0)}$	38.1$_{(1.7)}$
GAT	75.5$_{(0.9)}$	73.3$_{(1.4)}$	69.7$_{(1.4)}$	66.4$_{(1.3)}$	62.8$_{(1.6)}$	60.9$_{(1.8)}$	57.8$_{(1.3)}$	53.2$_{(1.6)}$	49.8$_{(1.7)}$	46.1$_{(2.4)}$	39.2$_{(3.6)}$	**39.9**$_{(2.9)}$	38.5$_{(2.9)}$	38.4$_{(3.0)}$	**38.8**$_{(2.5)}$
Dual GAT	**75.5**$_{(0.9)}$	**74.5**$_{(1.0)}$	**73.1**$_{(1.2)}$	**72.0**$_{(1.2)}$	**70.1**$_{(1.5)}$	**66.6**$_{(1.1)}$	**66.5**$_{(1.0)}$	**66.0**$_{(1.1)}$	**64.7**$_{(1.5)}$	**63.3**$_{(1.6)}$	**39.2**$_{(2.9)}$	39.5$_{(3.4)}$	**38.9**$_{(2.5)}$	**39.3**$_{(2.8)}$	38.5$_{(2.5)}$
TAG	**76.6**$_{(1.0)}$	**76.2**$_{(0.9)}$	73.4$_{(1.2)}$	71.6$_{(1.3)}$	69.7$_{(1.2)}$	64.3$_{(1.2)}$	**63.1**$_{(1.4)}$	62.7$_{(1.0)}$	61.9$_{(0.8)}$	61.2$_{(1.5)}$	40.5$_{(3.1)}$	40.2$_{(2.8)}$	39.5$_{(2.5)}$	39.4$_{(2.5)}$	39.9$_{(2.2)}$
Dual TAG	76.2$_{(1.2)}$	74.6$_{(1.2)}$	**73.7**$_{(1.5)}$	**72.6**$_{(1.7)}$	**71.6**$_{(1.8)}$	**65.0**$_{(2.5)}$	62.9$_{(2.7)}$	**63.1**$_{(3.3)}$	**63.1**$_{(3.2)}$	**64.0**$_{(3.1)}$	**41.8**$_{(3.1)}$	**42.2**$_{(3.0)}$	**42.1**$_{(2.8)}$	**41.5**$_{(2.7)}$	**41.9**$_{(3.0)}$
DNA	**75.9**$_{(0.8)}$	**74.2**$_{(1.1)}$	70.3$_{(1.2)}$	67.2$_{(1.6)}$	63.3$_{(1.4)}$	64.5$_{(1.0)}$	61.7$_{(1.3)}$	58.3$_{(1.6)}$	55.5$_{(1.4)}$	51.6$_{(2.0)}$	32.6$_{(2.8)}$	33.1$_{(3.1)}$	32.3$_{(3.0)}$	31.9$_{(2.9)}$	30.8$_{(2.4)}$
Dual DNA	74.6$_{(0.8)}$	73.9$_{(1.1)}$	**73.3**$_{(0.9)}$	**72.3**$_{(1.3)}$	**71.3**$_{(1.1)}$	**67.9**$_{(1.2)}$	**67.9**$_{(1.2)}$	**68.1**$_{(1.3)}$	**67.5**$_{(1.3)}$	**66.4**$_{(1.5)}$	**35.8**$_{(2.6)}$	**35.4**$_{(2.6)}$	**35.3**$_{(2.2)}$	**35.9**$_{(2.1)}$	**35.0**$_{(2.6)}$



(a) Cora (b) CiteSeer

Figure 3: Comparison results for the proposed Dual GCN and several state-of-the-art methods for learning with noisy graph structures on Cora and CiteSeer.

Again we applied the proposed framework on four baseline models. The comparison results with corrupted missing edges on the three datasets are reported in Table 2. The table shows the vulnerability of GNNs to edge deletion attacks on the underlying graph structures; the performance of all GNN baselines on the citation datasets declines substantially as the edge corruption rate increases. This is due to the dependence of GNNs on the graph structures for propagating messages/labels across the graph. The TAG baseline is particularly sensitive to noise in graph structures due to its large but static receptive field, which is clearly demonstrated by its notably larger performance drop relative to the other baselines. Nevertheless, the proposed Dual learning framework consistently improves the performance of all four GNN baselines across all corruption rates on the citation datasets. The performance gain of our framework grows as the corruption rate increases. On the Chameleon dataset, where the underlying graph is densely connected, the GNN baselines are more robust to attacks on the graph structure, while our Dual framework still improves the performance of GCN, TAG and DNA.

We also compared the proposed Dual GNN framework with several GNN defense methods in the literature that address the performance degradation problem of GNNs in the case of noisy/corrupted graph structures, including RGCN (Zhu et al., 2019), GCN-SVD (Entezari et al., 2020), GCN-Jaccard (Wu et al., 2019b) and Pro-GNN (Jin et al., 2020). We used GCN as the primary module and compared our proposed Dual GCN with these four comparison methods under random adversarial attacks that delete random edges from the graph structure with different edge deletion rates: $\{20\%, 40\%, 60\%, 80\%, 90\%, 95\%\}$. The comparison results on the Cora and CiteSeer datasets are reported in Figure 3. The figure clearly shows that Dual GCN consistently outperforms all the other comparison methods across all the considered edge deletion rates on both Cora and CiteSeer. The performance gains are particularly remarkable with higher edge deletion rates, exceeding 4% and 2% on Cora and CiteSeer respectively.

For the experiment setting with noisy additional edges, we compared the proposed Dual GNN framework with the four baseline models and report the comparison results on the three datasets in Table 3. The table clearly shows the vulnerability of GNNs to attacks of injecting noisy additional

Table 4: Mean classification accuracy (standard deviation is within brackets) on Cora (left part), CiteSeer (middle part) and Chameleon (right part) with few labeled nodes per class (3, 5, 10) and noisy graph structures with different edge corruption (deletion) rates (0.25, 0.5, 0.75, 0.9, 0.95).

		Cora					**CiteSeer**					**Chameleon**				
		0.25	0.5	0.75	0.9	0.95	0.25	0.5	0.75	0.9	0.95	0.25	0.5	0.75	0.9	0.95
3 Labels	GCN	$65.6_{(3.5)}$	$58.2_{(4.3)}$	$48.4_{(3.6)}$	$40.2_{(3.8)}$	$39.4_{(3.9)}$	$54.4_{(5.4)}$	$50.5_{(5.7)}$	$42.1_{(4.3)}$	$40.9_{(2.9)}$	$39.9_{(3.0)}$	$25.5_{(3.9)}$	$25.1_{(3.5)}$	$25.6_{(3.9)}$	$24.2_{(3.2)}$	$26.0_{(3.4)}$
	Dual GCN	$\mathbf{72.4}_{(2.3)}$	$\mathbf{64.7}_{(4.5)}$	$\mathbf{54.3}_{(4.8)}$	$\mathbf{41.9}_{(6.8)}$	$\mathbf{40.0}_{(7.3)}$	$\mathbf{62.2}_{(4.9)}$	$\mathbf{55.0}_{(8.8)}$	$\mathbf{46.7}_{(11.0)}$	$\mathbf{41.8}_{(8.9)}$	$\mathbf{42.7}_{(6.9)}$	$\mathbf{25.7}_{(3.3)}$	$\mathbf{25.6}_{(3.9)}$	$\mathbf{26.1}_{(3.7)}$	$\mathbf{25.3}_{(2.5)}$	$\mathbf{26.7}_{(3.0)}$
	DNA	$61.5_{(3.8)}$	$54.9_{(4.6)}$	$47.6_{(4.2)}$	$41.7_{(4.0)}$	$39.8_{(4.2)}$	$52.6_{(4.6)}$	$49.6_{(4.7)}$	$43.4_{(4.5)}$	$41.8_{(2.8)}$	$40.8_{(3.8)}$	$26.8_{(3.6)}$	$27.3_{(3.3)}$	$27.4_{(3.7)}$	$27.6_{(4.2)}$	$\mathbf{28.8}_{(3.5)}$
	Dual DNA	$\mathbf{66.3}_{(3.7)}$	$\mathbf{60.3}_{(4.4)}$	$\mathbf{52.0}_{(5.1)}$	$\mathbf{43.2}_{(4.6)}$	$\mathbf{41.3}_{(4.1)}$	$\mathbf{59.1}_{(4.8)}$	$\mathbf{52.2}_{(6.1)}$	$\mathbf{45.6}_{(5.5)}$	$\mathbf{42.7}_{(4.3)}$	$\mathbf{42.2}_{(4.0)}$	$\mathbf{27.4}_{(3.5)}$	$\mathbf{27.6}_{(4.1)}$	$\mathbf{27.7}_{(3.8)}$	$\mathbf{27.9}_{(4.2)}$	$28.2_{(3.8)}$
5 Labels	GCN	$71.4_{(3.2)}$	$64.5_{(5.5)}$	$56.1_{(3.3)}$	$45.0_{(3.3)}$	$42.0_{(3.0)}$	$59.0_{(3.6)}$	$53.6_{(4.5)}$	$47.6_{(3.8)}$	$44.3_{(3.8)}$	$44.7_{(2.6)}$	$\mathbf{30.4}_{(3.5)}$	$29.7_{(4.1)}$	$30.1_{(4.1)}$	$29.9_{(3.3)}$	$28.3_{(3.2)}$
	Dual GCN	$\mathbf{74.9}_{(3.0)}$	$\mathbf{68.8}_{(4.8)}$	$\mathbf{60.9}_{(3.2)}$	$\mathbf{47.8}_{(7.0)}$	$\mathbf{44.1}_{(6.9)}$	$\mathbf{62.4}_{(5.3)}$	$\mathbf{58.8}_{(6.6)}$	$\mathbf{53.1}_{(1.0)}$	$\mathbf{45.7}_{(9.7)}$	$\mathbf{46.4}_{(9.7)}$	$29.6_{(4.0)}$	$\mathbf{30.9}_{(5.0)}$	$\mathbf{30.8}_{(4.2)}$	$\mathbf{30.7}_{(4.4)}$	$\mathbf{29.2}_{(3.1)}$
	DNA	$67.2_{(3.3)}$	$61.4_{(5.1)}$	$54.1_{(2.7)}$	$46.2_{(2.6)}$	$44.2_{(2.3)}$	$57.1_{(2.9)}$	$53.2_{(3.8)}$	$48.2_{(3.4)}$	$46.2_{(3.1)}$	$45.2_{(2.6)}$	$\mathbf{29.5}_{(5.1)}$	$29.4_{(5.5)}$	$29.9_{(3.3)}$	$30.2_{(3.7)}$	$30.3_{(3.6)}$
	Dual DNA	$\mathbf{70.2}_{(3.6)}$	$\mathbf{65.8}_{(4.9)}$	$\mathbf{58.3}_{(3.2)}$	$\mathbf{48.4}_{(4.6)}$	$\mathbf{44.9}_{(3.7)}$	$\mathbf{62.0}_{(4.0)}$	$\mathbf{57.5}_{(4.9)}$	$\mathbf{50.9}_{(4.8)}$	$\mathbf{47.6}_{(3.8)}$	$\mathbf{47.7}_{(3.8)}$	$\mathbf{29.5}_{(5.0)}$	$\mathbf{30.1}_{(4.8)}$	$\mathbf{30.1}_{(3.3)}$	$\mathbf{31.2}_{(3.5)}$	$\mathbf{30.6}_{(3.3)}$
10 Labels	GCN	$76.2_{(0.9)}$	$69.9_{(1.7)}$	$62.1_{(2.5)}$	$54.1_{(2.7)}$	$52.7_{(2.6)}$	$63.3_{(2.4)}$	$59.4_{(3.6)}$	$52.9_{(3.3)}$	$50.1_{(3.5)}$	$50.4_{(2.5)}$	$34.2_{(4.8)}$	$34.7_{(4.5)}$	$33.2_{(3.4)}$	$33.1_{(2.5)}$	$32.1_{(2.6)}$
	Dual GCN	$\mathbf{77.7}_{(1.1)}$	$\mathbf{72.8}_{(1.7)}$	$\mathbf{65.4}_{(2.7)}$	$\mathbf{57.9}_{(4.0)}$	$\mathbf{56.6}_{(4.0)}$	$\mathbf{64.8}_{(2.9)}$	$\mathbf{62.9}_{(3.7)}$	$\mathbf{59.5}_{(5.5)}$	$\mathbf{54.0}_{(6.5)}$	$\mathbf{55.8}_{(5.2)}$	$\mathbf{35.1}_{(3.9)}$	$\mathbf{35.6}_{(4.1)}$	$\mathbf{34.3}_{(3.7)}$	$\mathbf{34.2}_{(2.6)}$	$\mathbf{32.4}_{(2.4)}$
	DNA	$74.3_{(1.5)}$	$69.2_{(1.2)}$	$63.3_{(1.8)}$	$56.5_{(1.9)}$	$55.2_{(2.3)}$	$62.6_{(2.5)}$	$59.7_{(2.9)}$	$54.1_{(3.2)}$	$52.3_{(3.0)}$	$52.1_{(2.3)}$	$33.5_{(3.9)}$	$33.2_{(3.7)}$	$33.2_{(3.1)}$	$\mathbf{34.3}_{(2.2)}$	$\mathbf{34.5}_{(2.5)}$
	Dual DNA	$\mathbf{75.9}_{(1.3)}$	$\mathbf{71.0}_{(1.3)}$	$\mathbf{65.7}_{(1.6)}$	$\mathbf{58.2}_{(2.2)}$	$\mathbf{55.8}_{(2.4)}$	$\mathbf{66.0}_{(2.3)}$	$\mathbf{63.7}_{(2.5)}$	$\mathbf{56.8}_{(3.0)}$	$\mathbf{53.6}_{(3.3)}$	$\mathbf{55.1}_{(3.0)}$	$\mathbf{34.0}_{(3.1)}$	$\mathbf{34.1}_{(3.1)}$	$\mathbf{33.7}_{(1.8)}$	$34.1_{(2.6)}$	$34.3_{(2.5)}$

edges into the input graph structure; the performance of all the four GNN baselines on the citation datasets degrades considerably as the number of noisy additional edges increases. This is due to the fact that the noisy additional edges corrupt the message passing process with a large number of noisy messages being propagated across the additional edges of the graph, causing the learned node embeddings to be noisy and less discriminative. The proposed Dual GNN learning framework obtains consistent improvement over the performance of the four GNN baselines across all numbers of noisy additional edges on the citation datasets. The improvement in the performance of the proposed framework grows as the number of noisy additional edges increases. In the case of the Chameleon dataset, the underlying graph structure is significantly more densely connected than those of the citation datasets. This diminishes the corrupting effect of the noisy additional edges, which explains the relatively small performance drops for the four baselines across all numbers of noisy additional edges. Nevertheless, our proposed Dual framework improves the performance of the DNA and TAG baselines and obtains similar performance to the GCN and GAT baselines. These results demonstrate the wide applicability of the proposed framework in alleviating the negative impact of graph structure corruptions.

4.4 Experiments with Both Few Labeled Nodes and Noisy Graph Structures

We conducted further experiments to investigate the performance of the proposed Dual learning framework with both few labeled nodes and noisy graph structures. We used three label rates with $3, 5$, and 10 labeled nodes per class, respectively. For each label rate, we create 10 different training sets as before and test the performance of the proposed Dual GNN framework using two baselines, GCN and DNA, with different edge deletion rates from $\{0.25, 0.50, 0.75, 0.90, 0.95\}$. We record the mean test classification accuracy and standard deviation results for each label rate across all the five considered edge corruption (deletion) rates.

Table 4 presents the comparison results on the three datasets. It has three sections. The top section of the table reports the comparison results with different edge deletion ratios for 3 labeled nodes per class, the middle section reports the comparison results for 5 labeled nodes per class, and the bottom section reports the results for 10 labeled nodes per class. We can see that the combination of small label rates and large edge deletion ratios causes substantial performance degradation on the baseline models. Our proposed Dual GNN framework consistently and significantly improves the performance of both GCN and DNA on the citation datasets across all the three label rates and different graph corruption ratios. On the Chameleon dataset, the Dual GNN framework also improves the performance of each baseline in most cases. This again validates the general efficacy of the proposed Dual GNN framework.

4.5 Ablation Study

We also conducted an ablation study to investigate the contribution of the two modules in the proposed Dual GNN framework. Specifically, we consider two variants of the Dual GNN learning framework: (1) *Primary+Cluster*. For this variant, we drop the auxiliary module but keep the primary module and the fine-grained spectral clustering loss. That is, we perform training with the joint loss objective of $\mathcal{L}_{CE} + \mathcal{L}_{sc}$. (2) *Auxiliary+Cluster*. For this variant, we drop the primary module but keep its encoder f_Θ, while performing training with the joint loss objective of $\tilde{\mathcal{L}}_{CE} + \mathcal{L}_{sc}$. We conducted experiments by using GCN as the baseline model, and compared the full Dual GCN model with the two variants and the base GCN under different label rates and various edge deletion based structure corruption rates.

The comparison results with different label rates are reported in Table 5. From the table, we can see that all variants have performance drops from the full Dual GCN model except the *Primary+Cluster* on Chameleon with

Table 5: Ablation study results in terms of mean classification accuracy (standard deviation is within brackets) on Cora, CiteSeer and Chameleon with a few labeled nodes per class (2, 3, 5, 10, 20). "Prim.+C" and "Aux.+C" are abbreviations for the two variants "Primary+Cluster" and "Auxiliary+Cluster", respectively.

	Cora					**CiteSeer**					**Chameleon**				
	2	3	5	10	20	2	3	5	10	20	2	3	5	10	20
GCN	$63.5_{(4.4)}$	$69.2_{(3.5)}$	$75.1_{(3.1)}$	$78.8_{(0.9)}$	$81.5_{(0.6)}$	$48.1_{(7.9)}$	$56.4_{(6.2)}$	$62.2_{(3.5)}$	$66.3_{(1.8)}$	$68.5_{(0.7)}$	$25.1_{(5.0)}$	$24.5_{(4.3)}$	$30.8_{(3.9)}$	$35.5_{(4.4)}$	$40.4_{(3.4)}$
Dual GCN	$\mathbf{71.1}_{(4.0)}$	$\mathbf{74.8}_{(2.4)}$	$\mathbf{77.6}_{(2.6)}$	$\mathbf{79.9}_{(1.0)}$	$\mathbf{82.7}_{(0.5)}$	$\mathbf{56.5}_{(9.1)}$	$\mathbf{63.0}_{(5.7)}$	$\mathbf{65.4}_{(4.1)}$	$\mathbf{66.5}_{(2.4)}$	$\mathbf{70.0}_{(1.5)}$	$\mathbf{25.7}_{(4.9)}$	$\mathbf{26.6}_{(4.0)}$	$\mathbf{31.1}_{(5.0)}$	$35.9_{(4.7)}$	$40.9_{(3.1)}$
Prim.+C	$63.1_{(5.0)}$	$69.6_{(3.1)}$	$75.1_{(3.3)}$	$78.6_{(1.2)}$	$81.6_{(0.4)}$	$49.0_{(7.2)}$	$57.2_{(5.7)}$	$62.7_{(3.4)}$	$66.0_{(1.9)}$	$69.2_{(0.4)}$	$25.4_{(4.3)}$	$25.9_{(4.2)}$	$30.5_{(4.8)}$	$\mathbf{36.4}_{(4.2)}$	$\mathbf{41.3}_{(3.1)}$
Aux.+C	$25.6_{(9.1)}$	$24.4_{(9.7)}$	$25.3_{(11.0)}$	$22.4_{(9.9)}$	$24.6_{(8.9)}$	$26.0_{(8.4)}$	$27.1_{(9.1)}$	$27.6_{(8.4)}$	$28.7_{(10.1)}$	$25.6_{(7.4)}$	$20.3_{(1.6)}$	$20.5_{(2.0)}$	$20.9_{(3.3)}$	$21.2_{(3.3)}$	$21.2_{(3.1)}$

Table 6: Ablation study results in terms of mean classification accuracy (standard deviation is within brackets) on Cora, CiteSeer and Chameleon with edge deletion corrupted graph structures (corruption rates: 0.25, 0.5, 0.75, 0.9, 0.95). "Prim.+C" and "Aux.+C" are abbreviations for the two variants "Primary+Cluster" and "Auxiliary+Cluster", respectively.

	Cora					**CiteSeer**					**Chameleon**				
	0.25	0.5	0.75	0.9	0.95	0.25	0.5	0.75	0.9	0.95	0.25	0.5	0.75	0.9	0.95
GCN	$79.1_{(1.0)}$	$74.7_{(1.5)}$	$67.4_{(1.8)}$	$60.1_{(1.7)}$	$57.9_{(1.6)}$	$67.2_{(1.1)}$	$64.5_{(1.1)}$	$58.7_{(1.7)}$	$55.7_{(1.8)}$	$54.9_{(1.7)}$	$40.0_{(3.5)}$	$39.6_{(3.6)}$	$38.3_{(3.3)}$	$36.7_{(3.0)}$	$37.2_{(1.9)}$
Dual GCN	$\mathbf{79.9}_{(0.8)}$	$\mathbf{75.9}_{(1.5)}$	$\mathbf{69.0}_{(1.7)}$	$\mathbf{64.1}_{(2.0)}$	$\mathbf{60.8}_{(1.8)}$	$\mathbf{67.7}_{(1.4)}$	$\mathbf{66.2}_{(2.6)}$	$\mathbf{64.2}_{(3.1)}$	$\mathbf{60.6}_{(3.1)}$	$\mathbf{61.4}_{(2.1)}$	$40.9_{(3.1)}$	$\mathbf{40.2}_{(3.3)}$	$\mathbf{39.5}_{(2.5)}$	$\mathbf{37.8}_{(2.3)}$	$\mathbf{37.9}_{(1.8)}$
Prim.+C	$78.9_{(0.8)}$	$74.7_{(1.5)}$	$67.2_{(1.7)}$	$62.1_{(1.9)}$	$59.9_{(1.1)}$	$67.0_{(1.1)}$	$64.7_{(1.3)}$	$62.5_{(2.1)}$	$60.1_{(1.5)}$	$59.6_{(1.9)}$	$\mathbf{41.1}_{(2.8)}$	$39.7_{(2.7)}$	$38.0_{(2.4)}$	$35.2_{(2.1)}$	$31.8_{(3.1)}$
Aux.+C	$26.2_{(1.1)}$	$36.7_{(1.0)}$	$39.4_{(7.7)}$	$39.8_{(8.8)}$	$36.1_{(7.6)}$	$32.5_{(1.0)}$	$42.4_{(8.9)}$	$43.4_{(8.2)}$	$44.3_{(8.8)}$	$44.6_{(9.9)}$	$21.5_{(2.9)}$	$23.5_{(7.6)}$	$23.6_{(5.7)}$	$35.7_{(2.2)}$	$34.5_{(2.2)}$

larger label rates of {10, 20} and the performance degradation is substantial with smaller label rates. The variant of Primary+Cluster has a similar performance as the base model GCN. This suggests the fined-grained clustering loss alone cannot help the primary module, while the auxiliary module, which constructs new graph structures from the fine-grained clustering, plays an essential role in promoting message propagation across the graph and overcoming the local overfitting problem caused by small label rates. By dropping the primary module, the performance of Auxiliary+Cluster is significantly poorer than the full model. This suggests that the auxiliary module can be easily misled by its constructed dense adjacency matrix without using the primary module to learn discriminative node embeddings from the original graph and labeled nodes. Overall the results in Table 5 demonstrate that both the primary and auxiliary modules make essential contributions to the effective performance of the joint Dual GNN learning framework with a very limited number of labeled nodes.

The comparison results with different edge corruption rates are reported in Table 6. From the table, we can see that all variants have inferior performance to the full model except the Primary+Cluster on Chameleon with a very low edge corruption rate of 0.25, while the performance gap between the variants and the full model increases as the edge corruption rate of the graph structure increases. The variant of Primary+Cluster has similar performance as the base model at low edge corruption rates while obtaining better performance than the base model at higher edge corruption rates on the citation datasets. This suggests that the fined-grained clustering loss is able to enhance node embedding learning at high graph structure corruption rates, improving the performance of the base model. However, the performance of Primary+Cluster cannot match that of the full Dual framework, which highlights the importance of the auxiliary module. This suggests constructing a new graph structure from the fine-grained clustering is crucial for promoting message propagation across the graph and countering the effect of edge deletion attacks on the input graph structure. As for the Auxiliary+Cluster variant, its performance is significantly inferior to that of the full model, similar to the case with few labeled nodes, which indicates using the primary module to induce good node embeddings is an important foundation for building the auxiliary module. These results in Table 6 again validate the contribution of each module in the proposed framework and highlight the importance of the joint Dual learning framework.

5 CONCLUSION

In this paper, we proposed a novel Dual GNN learning framework to address the drawbacks of standard GNN models on handling scarce labeled nodes and noisy graph structures for semi-supervised node classification. The proposed framework consists of two modules. The primary GNN module works on the original input graph, while the auxiliary module employs a new adjacency matrix constructed using fine-grained spectral clustering to facilitate message propagation across the graph. The two modules and the spectral clustering are learned under a joint optimization framework. This dual learning framework can be applied on many existing GNN baselines. We conducted experiments with four GNN baseline models, and the experimental results demonstrated that the proposed framework is robust to scarce labels and noisy graph structures. Moreover, the proposed Dual GNN learning framework significantly improves the GNN baselines and outperforms many state-of-the-art methods on benchmark graph datasets, which validates the efficacy of the Dual GNN design with limited supervision.

References

Calder, J., Cook, B., Thorpe, M., and Slepcev, D. (2020). Poisson learning: Graph based semi-supervised learning at very low label rates. In *International Conference on Machine Learning (ICML)*.

Chen, Y., Wu, L., and Zaki, M. (2020). Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Du, J., Zhang, S., Wu, G., Moura, J. M., and Kar, S. (2017). Topology adaptive graph convolutional networks. *arXiv preprint arXiv:1710.10370*.

Elinas, P., Bonilla, E. V., and Tiao, L. C. (2020). Variational inference for graph convolutional networks in the absence of graph data and adversarial settings. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. (2020). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the International Conference on Web Search and Data Mining*.

Fey, M. (2019). Just jump: Dynamic neighborhood aggregation in graph neural networks. *arXiv preprint arXiv:1904.04849*.

Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.

Geisler, S., Zügner, D., and Günnemann, S. (2020). Reliable graph neural networks via robust aggregation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. (2020). Graph structure learning for robust graph neural networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Li, Q., Han, Z., and Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Lin, W., Gao, Z., and Li, B. (2020). Shoestring: Graph-based semi-supervised classification with severely limited labeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Luan, S., Zhao, M., Chang, X.-W., and Precup, D. (2019). Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Min, Y., Wenkel, F., and Wolf, G. (2020). Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Niepert, M., Ahmed, M., and Kutzkov, K. (2016). Learning convolutional neural networks for graphs. In *International conference on machine learning (ICML)*.

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., and Yang, B. (2019). Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

Rong, Y., Huang, W., Xu, T., and Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations (ICLR)*.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine 29(3)*, pages 93–93.

Shi, C., Xu, M., Guo, H., Zhang, M., and Tang, J. (2020). A graph to graphs framework for retrosynthesis prediction. In *International Conference on Machine Learning (ICML)*.

Sun, K., Lin, Z., and Zhu, Z. (2020). Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations (ICLR)*.

Wan, S., Zhan, Y., Liu, L., Yu, B., Pan, S., and Gong, C. (2021). Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wang, H., Zhou, C., Chen, X., Wu, J., Pan, S., and Wang, J. (2020). Graph stochastic neural networks for semi-supervised learning. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *International conference on machine learning (ICML)*.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. (2019b). Adversarial examples for graph data: Deep insights into attack and defense. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*.

Yang, Z., Cohen, W., and Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning (ICML)*.

You, Y., Chen, T., Wang, Z., and Shen, Y. (2020). When does self-supervision help graph convolutional networks? In *International Conference on Machine Learning (ICML)*.

Zhang, X. and Zitnik, M. (2020). Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems (NeurIPS)*.

Zhou, Z., Zhang, S., and Huang, Z. (2019). Dynamic self-training framework for graph convolutional networks. *arXiv preprint arXiv:1910.02684*.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zitnik, M. and Leskovec, J. (2017). Predicting multicellular function through multi-layer tissue networks. *Bioinformatics 33(14)*, pages 190–198.