NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs

Enyan Dai
The Pennsylvania State University
emd5759@psu.edu

Charu Aggarwal
IBM
charu@us.ibm.com

Suhang Wang
The Pennsylvania State University
szw494@psu.edu

ABSTRACT

Graph Neural Networks (GNNs) have achieved promising results for semi-supervised learning tasks on graphs. Despite the great success of GNNs, real-world graphs are often sparsely and noisily labeled, which can significantly degrade the performance of GNNs, as the noisy information can propagate to unlabeled nodes via graph structure. Thus, it is crucial to develop a label noise-resistant GNN for node classification. Though extensive studies have been conducted to deal with noisy labels, they mostly focus on independent and identically distributed data and assume a large number of noisy labels are available, which are not applicable for GNNs. Thus, we investigate a novel problem of learning a robust GNN with noisy and limited labels. To alleviate the negative effects of label noise, we propose to link the unlabeled nodes with labeled nodes of high feature similarity to bring more clean label information. Furthermore, accurate pseudo labels could be obtained by this strategy to provide more supervision and further reduce the effects of label noise. Our theoretical analysis and extensive experiments on real-world datasets verify the effectiveness of our proposed method.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; **Neural networks**.

KEYWORDS

Noise Labels; Robustness; Graph Neural Network

ACM Reference Format:
Enyan Dai, Charu Aggarwal, and Suhang Wang. 2021. NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3447548.3467364

1 INTRODUCTION

Graph structured data is very pervasive in real-world, such as social networks [9], financial transaction networks [37] and traffic networks [40]. Graph Neural Networks (GNNs) have shown great ability in modeling graph structured data and are attracting increasing attention [1, 9, 15, 38]. Generally, GNNs adopt the message-passing process to update node representations by aggregating the information from their neighbors [15, 36]. One of the most important and popular tasks that benefits from this message-passing mechanism is node classification in a semi-supervised manner. With this mechanism, labeled nodes can propagate their information to unlabeled nodes [9, 35], thus resulting in superior performance of GNNs.

Despite the great performance of GNNs for semi-supervised node classification, the majority of existing methods assume the training labels are clean; while for many real-world graphs and applications, the collected labels could be noisy and limited. For instance, for the geo-location prediction in social networks, only a small portion of users will fill in the geo-location; and the provided locations can be noisy because users randomly fill in wrong locations to protect their privacy or users have moved to new locations but forget to update them in social networks [21]. Similarly, for bot detection in social media, the labeling process can be tedious, costly, and error-prone, which can end up with limited noisily labeled nodes [17].

The graph with noisy and limited labels could significantly degrade the performance of GNNs for semi-supervised node classification. *First*, recent work has shown that neural networks will overfit to the noisy labels and results in poor generalization performance [31, 42]. As a generalization of neural networks for graphs, GNNs are also likely to have poor performance trained on noisy labels. *Second*, for graphs, the noisy information can propagate through the network topology. Falsely labeled nodes will negatively affect their unlabeled neighbors. Since the graph is sparsely labeled, neighbors of falsely labeled nodes are unlikely to accept the information from nodes with true labels to correct the representations. In addition, many unlabeled nodes will only be able to aggregate information from unlabeled nodes when the labels are limited. Thus, the performance of GNNs trained on noisily and sparsely labeled graph would be poor.

Though extensive approaches have been proposed for learning with noisy labels such as loss correction [7, 31] and sample selection [10, 13, 19, 23, 41], they are not directly applicable for learning GNNs with limited noisy labels. *First*, generally, these methods assume a large amount of noisy labels are available for learning noise distribution or for sampling correct labels. They are challenged by the small label size. *Second*, the majority of existing work for noisy labels [10, 19, 23, 31] focus on independent and identically distributed (i.i.d) data such as images, which cannot handle the information propagation of noisy labels on graphs. The work on learning a robust GNN with noisy and limited labels is rather limited [8, 43]. Therefore, it is important to develop a robust GNN that could deal with noisy and limited labels.

Since the labeled nodes can propagate its information to the unlabeled nodes, it is promising to correct the predictions of unlabeled

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467364

nodes affected by falsely labeled nodes by linking them with nodes of clean labels. However, in practice, we do not know which labels are clean. Alternatively, for an unlabeled node v_i, we propose to link v_i with labeled nodes of high feature similarity with v_i to make it robust to label noise and facilitate the message passing of GNNs. The basic idea is if two nodes have high feature similarity, they are more likely to have the same label. Thus, if the probability that labeled nodes having correct labels is higher than that of having incorrect labels, by connecting v_i with more labeled nodes of high feature similarity with v_i, we can potentially bring more correct label information to v_i. Our theoretical and empirical analysis in Sec 3.4 verify the effectiveness of linking unlabeled nodes with noisily labeled nodes under mild conditions. In addition, with this strategy, we can first train a classifier to obtain accurate pseudo labels to ease the problem of learning with noisy and limited labels. By extending the label set with pseudo labels, more supervision could be utilized to make predictions for unlabeled nodes. Linking unlabeled nodes with similar nodes of accurate pseudo labels could further reduce the issue of label noise, which is verified in Sec 3.5. Though promising, there are no existing work exploring these strategies for learning GNNs with noisy and limited labels.

Therefore, in this paper, we investigate a novel problem of learning *N*oise-*R*esistant GNNs on sparsely and noisily labeled graphs. In essence, we are faced with two challenges: (i) How to effectively link unlabeled nodes with labeled nodes to alleviate the effects of label noise and benefit the prediction? (ii) Given the graph with noisy and limited labels, how can we obtain accurate pseudo labels? To solve these challenges, we proposed a novel framework named noise-resistant GNN (NRGNN)[1]. NRGNN adopts a GNN-based edge predictor to predict edges to benefit the classification on graphs with noisy and limited labels. Since the existing edges in the graph generally link nodes in similar attributes [24], these edges could provide supervision to train a good edge predictor. The graph densified by linking unlabeled nodes with similar noisily labeled nodes is utilized to obtain accurate pseudo labels, which extends the label set to provide more supervision for node classification. NRGNN also adopts the edge predictor to link unlabeled with similar extended labeled nodes to further reduce the effects of label noise. In summary, our main contributions are:

- We investigate a novel problem of learning noise-resistant GNNs on graphs with noisy and limited labels;
- We propose a new framework which can generate accurate pseudo labels and assign high-quality edges between unlabeled nodes and (pseudo) labeled nodes to alleviate label noise issue;
- Theoretical and empirical analysis are conducted to verify the effectiveness of the proposed strategies against label noise;
- Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed NRGNN in node classification on graphs with noisy and limited labels.

2 RELATED WORK

2.1 Graph Neural Networks

Graph neural networks (GNNs) have shown great ability in modeling graph structured data. They have achieved remarkable success in various applications such as social networks [3, 9], financial transaction networks [37] and traffic networks [40, 45]. Based on the definition of graph convolution, GNNs can be generally divided into two categories, i.e., spectral-based [1, 4, 12, 15, 18] and spatial-based [2, 9, 34, 36, 39, 46]. Bruna et al. [1] first explored spectral-based GNNs by utilizing the spectral filter on the local spectral space. Since then, various spectral-based methods are developed for further improvements [4, 12, 15, 18]. For instance, Kipf and Welling [15] propose Graph Convolutional Network (GCN) which simplifies the graph convolution. Spatial-based graph convolution directly updates the node representation by aggregating its neighborhoods' representations [6, 9, 28, 39]. For example, graph attention network (GAT) [36] applies the self-attention mechanism into the aggregation of spatial graph convolution. Graph Isomorphism Network (GIN) [38] is proposed to learn more powerful representations of the graph structures. Moreover, various spatial-based methods are investigated to solve the scalability issue of GNNs [2, 9].

However, as a generalization of neural networks on graph structured data, GNNs are also vulnerable to noisy labels [29, 44]. In addition, due to the message passing mechanism of GNNs, the noisy label information will pass to the unlabeled nodes, which severely degrades the performances of GNNs. For example, [29] shows that the performance of GNNs will drop significantly when noises are added to the training labels. However, very few efforts are taken to address the problem of learning GNNs on graphs with noisy labels [29, 31]. D-GNN [29] applied the backward loss correction [31] to reduce the effects of noisy labels. Zhang et al. [44] avoid the overfitting of the noisy labels by adding a regularization which encourages the learned representations well predict the community labels. Our proposed framework is inherently different from aforementioned methods. We investigate a novel framework which could achieve robustness towards noisy labels in graphs by carefully connecting unlabeled nodes with (pseudoly) labeled nodes.

2.2 Deep Learning with Noisy Labels

It is shown in [42] that a standard deep neural network will overfit to the noisy labels and results in poor generalization performance. Extensive studies have been investigated to address this problem on i.i.d data such as images, which can be generally categorized into two groups: loss correction [7, 22, 31, 32] and sample selection [10, 13, 19, 23, 41]. The loss correction methods correct the loss of training samples with noisy labels. For example, Goldberger and Ben-Reuven [7] propose a noise adaptation layer to automatically learn the noise transition matrix and sequentially apply it to correct the loss in S-model. Patrini et al. [31] estimate the label corruption matrix and propose two ways of correcting the loss, i.e., forward and backward correction. Bootstrap [32] handles noisy labels by augmenting the prediction objective with a notion of consistency. The sample selection methods aim to find the clean samples during the training process. For example, Decoupling [23] deploys two networks to select clean samples and update the two networks with the clean samples obtained from each other. MentorNet [13] pre-trains a teacher network to reweight the samples during the training process of the student network. Coteaching [10] also employs two networks and selects the small-loss samples as clean

[1] https://github.com/EnyanDai/NRGNN

samples for each other. Moreover, Coteaching+[41] incorporates additional rule of updating when disagreement to improve the performance of Coteaching. Recently, methods that utilize the data points that are not selected as clean samples by semi-supervised learning methods are also investigated [19, 27].

However, the aforementioned approaches are dedicated to i.i.d data, which may not be directly applicable to GNNs for handing noisy labels because the noisy information can propagate via message passing of GNNs. Therefore, we propose a novel approach NRGNN to handle the label corruption on the graph-structured data. Furthermore, we address the challenge of learning with labels that are often noisy and limited in graphs.

3 PRELIMINARIES

In this section, we firstly introduce the basic design of GNNs. Next, two strategies of addressing the problem of learning on noisily and sparsely labeled graphs are analyzed theoretically and empirically.

3.1 Notation

We use $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ to denote a graph, where $\mathcal{V} = \{v_1, ..., v_N\}$ is the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{A}_{ij} = 1$ if nodes v_i and v_j are connected, otherwise $\mathbf{A}_{ij} = 0$. $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ is the set of node attributes with $\mathbf{x}_i$ being the node attributes of node v_i. $\mathcal{V}_L = \{v_1, ..., v_l\}$ is a set of labeled nodes. $\mathcal{V}_U = \mathcal{V} - \mathcal{V}_L$ is a set of unlabeled nodes. The provided labels of $\mathcal{V}_L$ are corrupted by noise, which are denoted as $\mathcal{Y}_N = \{y_1^n, ..., y_l^n\}$. And $\mathcal{Y}_T = \{y_1^t, ..., y_l^t\}$ is used to represent the true labels.

3.2 Preliminaries about GNN

Graph neural networks (GNNs) utilize the node features and the graph structures to learn presentations for prediction. Specifically, each layer of GNNs will update the representations of the nodes using the representations of the neighborhood nodes. Thus, the representations after k layers' aggregation would capture the information of the k-hop network neighborhoods, which would benefit the node classification. Generally, the updating process of the k-th layer in GNN is formally stated as:

$$\begin{aligned} \mathbf{a}_v^{(k)} &= \text{AGGREGATE}^{(k-1)}(\{\mathbf{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}), \\ \mathbf{h}_v^{(k)} &= \text{COMBINE}^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{a}^{(k)}), \end{aligned} \tag{1}$$

where $\mathbf{h}_v^{(k)}$ is the representation vector of the node $v \in \mathcal{V}$ at k-th layer and $\mathcal{N}(v)$ is a set of neighborhoods of v. GCN is one of the most popular GNN structures, which could be viewed as a special case of Eq.(1). Each layer of GCN can be written as:

$$\mathbf{H}^{(k+1)} = \sigma(\tilde{\mathbf{A}}\mathbf{H}^{(k)}\mathbf{W}^{(k)}), \tag{2}$$

where $\mathbf{H}^{(k)}$ is the representation matrix of the output of the k-th layer; $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + \mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix and $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_i A_{ij}$. $\mathbf{I}$ is the identity matrix and σ is an activation function such as ReLU.

3.3 Problem Definition

Given the notation in Sec 3.1, the problem of learning a robust GNN with noisy and limited labels is formally defined as:

Problem 1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with a small set of nodes $\mathcal{V}_L \in \mathcal{V}$ provided with noisy labels $\mathcal{Y}_N$, we aim to learn a robust GNN which predicts the true labels of the unlabeled nodes, i.e.,*

$$f(\mathcal{G}, \mathcal{Y}_N) \rightarrow \hat{\mathcal{Y}}_U \tag{3}$$

where f is the function we aim to learn and $\hat{\mathcal{Y}}_U$ is the set of predicted labels for unlabeled nodes.

3.4 How the Size of Noisily Labeled Neighbors Affect the Node Classification

For a trained K-layer GNN with a set of learned parameters $\theta = \{\mathbf{W}^{(1)}, ..., \mathbf{W}^{(K)}\}$, it makes predictions by $\mathbf{Y} = \tilde{\mathbf{A}}\mathbf{H}^{(K)}\mathbf{W}^{(K)}$. Since the parameters θ are well trained for node classification, the K-th latent representations, i.e., $\mathbf{S} = \mathbf{H}^{(k)}\mathbf{W}^{(K)}$ could be treated as predictions of the nodes [5]. And the final predictions are obtained by the aggregation of $\mathbf{S}$, i.e., $\mathbf{Y} = \tilde{\mathbf{A}}\mathbf{S}$. Let's treat s_{ik} as the predicted probability that node v_i belongs to class k. For an unlabeled $v_u \in \mathcal{V}_U$ belonging to class c, we consider three types of neighbors: (i) an unlabeled node $v_a \in \mathcal{V}_U$; (ii) a node $v_b \in \mathcal{V}_L$ labeled as c; and (iii) a node $v_d \in \mathcal{V}_L$ labeled to a class other than c. Since the GCN is optimized to make s_{bc} close to 1 and s_{dc} close to 0, generally we could have $\mathbb{E}(s_{bc}) > \mathbb{E}(s_{ac}) > \mathbb{E}(s_{dc})$. To simplify the analysis, we assume that nodes with high feature similarity belong to the same class. Then, we can have the following theorem which indicates that linking an unlabeled node with similar labeled nodes could increase the robustness against label noise.

Theorem 3.1. *We consider an unlabeled node $v_u \in \mathcal{V}_U$ which belongs to class c. It is linked with n unlabeled nodes and m labeled nodes. The ratio of intra-class edges is h. Assume that:*

(1) For labeled nodes, a node belonging to class c is more likely to be labeled as c than a node not belonging to class c;
(2) The probability p_t that a node belonging to class c is labeled as c meets this constraint: $p_t > \frac{\mathbb{E}(s_{ac}) - \mathbb{E}(s_{dc})}{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{dc})}$.

Then, linking v_u with more similar noisily labeled nodes $v_l \in \mathcal{V}_L$ can on average improve its predicted probability of belonging to class c, i.e., improve $y_{uc} = \frac{1}{d_u}\sum_{j \in \mathcal{N}(v_i)} s_{jc}$.

The proof of this theorem is presented in Appendix B. When a graph network network for multi-class classification is corrupted with label noise, the predicted probability of an unlabeled node $v_a \in \mathcal{V}_U$ would be much smaller that the probability that a node labeled as c. Therefore, the assumption that $p_t > \frac{\mathbb{E}(s_{ac}) - \mathbb{E}(s_{dc})}{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{dc})}$ could be generally satisfied.

Empirical analysis: According to the Theorem 3.1, if we could link the unlabeled nodes with more similar labeled nodes belonging to the same class, we will have a more robust model. On the contrary, linking unlabeled nodes may not be useful. To empirically verify this, we utilize the cosine similarity scores of the raw features to identify similar nodes. Then, edges could be added based on the similarity scores. More specifically, we compare the results of the following methods:

- **Initial $\mathcal{G}$**: We train a GCN on the initial graph structure with noisy labels as the baseline.

Table 1: Accuracy(%) of node classification with noisy labels.

Dataset	Noise Rate	Initial $\mathcal{G}$	Link $\mathcal{V}_U$	Link $\mathcal{V}_L$
Cora	0.1	77.9 ±0.3	77.8 ±0.5	**78.7 ±0.4**
	0.2	72.8 ±1.8	72.8 ±1.0	**74.0 ±0.9**
	0.3	65.6 ±0.8	65.8 ±1.7	**68.5 ±1.4**
Citeseer	0.1	68.1 ±0.8	68.1 ±0.6	**69.0 ±1.0**
	0.2	64.9 ±1.7	65.2 ±0.8	**66.4 ±1.5**
	0.3	60.4 ±2.5	61.8 ±1.0	**62.7 ±1.0**

Table 2: Accuracy(%) of node classification with noisy labels.

Dataset	Initial $\mathcal{G}$	Link $\mathcal{V}_L$	Link $\mathcal{V}_L$ (Retrain)	Link $\mathcal{V}_A$
Cora	72.8 ±1.8	74.0 ±0.9	75.4 ±1.0	**77.1 ±1.3**
Citeseer	64.9 ±1.7	66.4 ±1.5	66.5 ±1.5	**68.0 ±1.4**

- **Link** $\mathcal{V}_L$: For $v_u \in \mathcal{V}_U$ and $v_l \in \mathbf{V}_L$, if their raw feature cosine similarity is larger than t, we add a link between them. Then, a GCN trained on $\mathcal{G}$ will make predictions with the new graph.
- **Link** $\mathcal{V}_U$: Unlabeled nodes will be linked with other unlabeled nodes if they have high cosine similarity of features. Similarly, a GCN trained on $\mathcal{G}$ will make predictions with the new graph.

We conduct experiments on widely used benchmark Cora and Citeseer [33]. In both datasets, we randomly sample 5% nodes as labeled nodes. And labels are corrupted by randomly flipping the true labels to other class with a probability of p. More specifically, we vary the noise rate. i.e. the probability that given labels is wrong, from 10% to 30% with a step of 10%. The thresholds of cosine similarity are selected based on the validation set. We report average results of 5 runs in Table 1. We could have the following observations:

- Linking the unlabeled nodes with unlabeled nodes shows no difference from the results of training on initial graph.
- Even a simple strategy based on raw feature cosine similarity to link unlabeled nodes with labeled nodes could benefit node classification trained on noisy labels significantly.
- When the noise rate is raised to 0.3, linking unlabeled nodes with similar labeled nodes still shows its effectiveness.

3.5 A Strategy On Graphs with Small Amount of Noisy Labels

With the analysis in Sec. 3.4, we find that linking more existing noisily labeled nodes with the unlabeled nodes could make more robust predictions. However, the size of noisily labeled nodes are often very small in graph-structure data. And an unlabeled node may have small node similarity with the labeled nodes. In this situation, the benefits from the strategy described in Sec 3.4, would be largely limited. A strategy to address this problem is to obtain accurate pseudo labels $\mathcal{V}_P$. As a result, we could have an extended label set $\mathcal{V}_A = \mathcal{V}_L \cup \mathcal{V}_P$. Then an unlabeled node can have more similar nodes in $\mathcal{V}_A$ to have a more robust model. In addition, more supervision from pseudo labels can be utilized. Let s_{pc} denotes the predicted probability that node $v_p \in \mathcal{V}_P$ belongs to class c based on the K-th latent representations, i.e., $\mathbf{S} = \mathbf{H}^{(k)}\mathbf{W}^{(K)}$. The following theorem verifies the effectiveness of this strategy when s_{pc} meets a mild constraint.

Theorem 3.2. *We consider an unlabeled node $v_u \in \mathcal{V}_U$ which belongs to class c. It is linked with n unlabeled nodes and m labeled nodes. Let p denotes the probability that the existing linked labeled nodes is labeled as c. For a node $v_p \in \mathcal{V}_P$ which is provided with pseudo label, if $\mathbb{E}(s_{pc}) > max(\mathbb{E}(s_{ac}), p\mathbb{E}(s_{bc}) + (1-p)\mathbb{E}(s_{dc}))$, then, linking v_u with v_p can improve its predicted probability of belonging to class c, i.e., $y_{uc} = \frac{1}{d_u}\sum_{j\in\mathcal{N}(v_i)} s_{jc}$.*

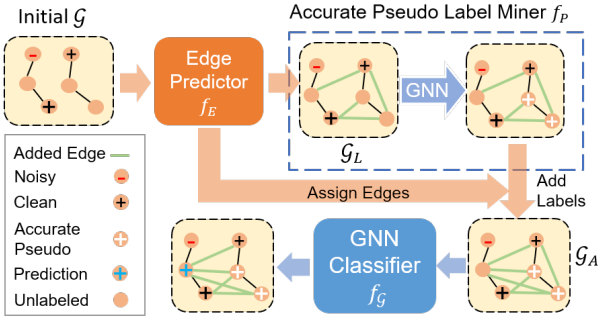


Figure 1: The overall framework of our method.

The details of the proof is listed in Appendix C. To obtain pseudo labels which meet the assumption to benefit the predictions under label noise, we could utilize the strategy described in Sec 3.4 to give better predictions. Furthermore, we can select the predictions whose confidence scores are high.

Empirical analysis: To show the effectiveness of the strategy of utilizing accurate pseudo labels, we conduct experiments with the following process: 1) obtain a GNN classifier using the strategy of linking unlabeled nodes and labeled nodes based on cosine similarity; 2) select the predictions of unlabeled nodes whose confidence scores are high as pseudo labels $\mathcal{V}_P$ to compose extended label set $\mathcal{V}_A = \mathcal{V}_L \cup \mathcal{V}_P$; 3) link $\mathcal{V}_U$ and $\mathcal{V}_A$ based on cosine similarity of raw features, and train a GCN with the new graph with accurate pseudo labels and noisy labels. This process is named as **Link** $\mathcal{V}_A$. And the results of 5 runs on Cora and Citeseer with 20% uniform noise are presented in Table 2. To make a fair comparison, we also retrain the GCN on the graph densified by linking unlabeled nodes with similar labeled nodes. From the Table 1, we could find that with the strategy of utilizing accurate pseudo labels, the model become more robust to the label noise.

4 METHODOLOGY

In this section, we present the details of the proposed framework NRGNN. As shown in Sec 3, carefully linking unlabeled nodes with nodes with noisy labels or accurate pseudo labels could benefit the learning of GNNs on noisily and sparsely labeled graphs. However, there are two main challenges: (i) How to accurately add edges between unlabeled nodes and extended labeled nodes of the same class to benefit the prediction? and (ii) Given the graph with limited noisy labels, how to obtain accurate pseudo labels? To solve these two challenges, we propose to learn a GNN-based edge predictor using node attributes to assign high-quality edges. To obtain more high-quality pseudo labels, we utilize the graph densified by linking similar unlabeled nodes and noisily labeled nodes with the GNN-based edge predictor. An illustration of the proposed framework is shown in Fig. 1, which is composed of an edge predictor f_E, a pseudo label miner f_P, and a GNN classifier $f_{\mathcal{G}}$. The edge predictor takes the initial graph $\mathcal{G}$ as input to predict edges. f_E will first link similar nodes between $\mathcal{V}_U$ and $\mathcal{V}_L$ to obtain $\mathcal{G}_L$, which could benefit the pseudo label miner. The pseudo label miner f_P adopts a GNN classifier trained on $\mathcal{G}_L$ to collect nodes with high-confident pseudo

labels, denoted as $\mathcal{V}_P$. With the extended label set, the edge predictor f_E further connects unlabeled nodes V_U with similar nodes in $\mathcal{V}_L \cup \mathcal{V}_P$ to form a new graph $\mathcal{G}_A$, which helps to propagate the information from $\mathcal{V}_L \cup \mathcal{V}_P$ to unlabeled nodes. The final GNN classifier $f_{\mathcal{G}}$ takes $\mathcal{G}_A$ with the label set $\mathcal{V}_L \cup \mathcal{V}_P$ for robust prediction. Next, we introduce each component in detail.

4.1 Edge Prediction

In many real-world networks, linked nodes generally have the same labels or similar features [24]. For instance, papers are more likely to cite papers belonging to the same research field, and friends tend to share similar interests [26]. In addition, many real-world graphs are very sparse and contains lots of missing links. For example, a social media user may miss a lot of potential friends sharing same interests and only follow a small number of people due to time limitation in exploring friends online. Thus, link prediction algorithms can learn from the attributed graph to predict the missing links, which provides one direction for us to link nodes.

Our preliminary analysis in Sec. 3 has shown that by simply using the node feature similarity based link prediction to connect unlabeled nodes with labeled nodes could help to improve the performance of GNNs with noisy labels. However, the simple approach only considers node features to measure the similarity. While for graphs, the local graph structure also provides another perspective for measuring similarities. To better predict the missing links, we propose to use a GNN-based edge predictor. Instead of simply relying on node features, the GNN-based edge predictor learns node representations capturing both node features and local graph structure, and predict links based on the learned representations, which could improve the link prediction performance. Following [16], our edge predictor adopts the GCN to learn node representations as:

$$\mathbf{Z} = GCN(\mathbf{A}, \mathbf{X}). \tag{4}$$

Let $\mathbf{z}_i$ and $\mathbf{z}_j$ denote the representations of node v_i and v_j, respectively. The closer $\mathbf{z}_i$ and $\mathbf{z}_j$ are, the more likely v_i and v_j are linked. Thus, the probability that v_i and v_j are linked can be calculated as

$$\mathbf{S}_{ij} = \sigma(\mathbf{z}_i \mathbf{z}_j^T), \tag{5}$$

where $\sigma(\cdot)$ is the activation function. Because the learned weights $\mathbf{S}$ would be fed into other modules and trained end-to-end, we use ReLU as the activation function to avoid the gradient vanishing [11].

If the edge predictor could well reconstruct the adjacency matrix $\mathbf{A}$, then it would be good at predicting missing links. Thus, following existing work [16], we use the adjacency matrix reconstruction as the loss. However, the majority of the elements in $\mathbf{A}$ are 0's, which could dominate the loss function and result in edge predictor f_E simply outputting 0's. To avoid this, we apply negative sampling [25], i.e., for each positive sample $A_{ij} = 1$, we randomly sample K nodes which are not connected with node j as negative samples. With the negative sampling, the loss function could be formally written as:

$$\min_{\theta_E} \mathcal{L}_E = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} \left((\mathbf{S}_{ij} - 1)^2 + \sum_{n=1}^{K} \mathbb{E}_{v_n \sim P_n(v_i)} (\mathbf{S}_{in} - 0)^2 \right) \tag{6}$$

where θ_E is the set of parameters of f_E, $\mathcal{N}(v_i)$ represents the neighbors of node v_i, and $P_n(v_i)$ is the distribution of the nodes which have no connections with node v_i in the graph. With the GNN-based edge predictor trained with Eq.(6), we could predict useful missing edges to link unlabeled nodes and labeled nodes to benefit the robust classification with noisy and limited labels.

4.2 Accurate Pseudo Label Prediction

According to the analysis in Sec. 3.5, more accurate pseudo labels would better facilitate the training of GNNs with noisy and limited labels. Thus, in this subsection, we describe how to obtain accurate pseudo labels. Since connecting unlabeled nodes with labeled nodes by link prediction can help improve the node classification of a GCN on graph with noisy labels, we propose to first predict the missing edges between $\mathcal{V}_U$ and $\mathcal{V}_L$ with the edge predictor f_E. Then, we could obtain a densified graph $\mathcal{G}_L$ to train a more accurate pseudo label miner. Specifically, for $v_i \in \mathcal{V}_U$ and $v_j \in \mathcal{V}_L$, if S_{ij} is larger than a threshold t, then v_i and v_j are more likely to have the same label and we would connect them. If $S_{ij} < t$, then the probability that v_i and v_j having the same label is small and we don't want to include such links. Thus, the process of obtaining the adjacency matrix of $\mathcal{G}_L$ could be formally stated as:

$$\mathbf{S}^L_{ij} = \begin{cases} 1 & \text{if } v_j \in \mathcal{N}(v_i); \\ \mathbf{S}_{ij} & \text{else if } \mathbf{S}_{ij} > t, v_i \in \mathcal{V}_U \text{ and } v_j \in \mathcal{V}_L; \\ 0 & \text{else,} \end{cases} \tag{7}$$

where $\mathbf{S}^L_{ij}$ indicates the weight of edges between node v_i and v_j in $\mathcal{G}_L$, and t is the threshold to filter out edges with small weights. With $\mathbf{S}^L$, we can train a GNN classifier as pseudo label miner f_P. The pseudo labels of nodes $\mathcal{V}$ is predicted as:

$$\hat{\mathbf{Y}}^P = GNN(\mathbf{S}^L, \mathbf{X}), \tag{8}$$

where GNN is flexible to various models such as GCN [15] and GIN [38]. Its training objective function can be written as:

$$\min_{\theta_P} \mathcal{L}_P = \sum_{v_i \in \mathcal{V}_L} l(\hat{y}^P_i, y_i), \tag{9}$$

where θ_P is the parameters of the pseudo label miner f_P, $\hat{y}^P_i$ is the prediction of node v_i from f_P, and $l(\cdot)$ is the cross entropy loss. With $\mathbf{S}^L$, we can reduce the negative effects of label noise and have more reliable pseudo labels for the unlabeled nodes. Intuitively, the pseudo label whose confidence score is high should be more likely to be correct. Let $\hat{y}^P_{ic}$ denotes the predicted probability that node v_i belongs to the class c. Then the accurate pseudo labels is obtained by the following process:

$$\mathcal{Y}_P = \{\hat{y}^P_i \in \hat{\mathcal{Y}}^P_U; \hat{y}^P_{ic} > T_p\}, \tag{10}$$

where $\hat{\mathcal{Y}}^P_U$ is the set of predictions from the pseudo label miner for unlabeled nodes, and T_p is the threshold to select the accurate pseudo labels.

4.3 Robust Classification with Edge Predictor and Accurate Pseudo Labels

The accurate pseudo labels $\mathcal{Y}_P$ could facilitate the classification with noisy and limited labels in two folds: (i) accurate pseudo labels could provide more supervision for node classification; and (ii) edges linking unlabeled nodes and accurate pseudo labeled nodes could be added to reduce the effects of label noise. To fully utilize the pseudo labels, we adopt the edge predictor f_E to assign missing links

between unlabeled nodes $\mathcal{V}_U$ and extended labeled nodes $\mathcal{V}_A = \mathcal{V}_L \cup \mathcal{V}_P$, where $\mathcal{V}_P$ is the node set with accurate pseudo labels $\mathcal{Y}_P$. Similar to the construction of S^L, we use the same threshold t to select links. This process is written as:

$$\mathrm{S}^A_{ij} = \begin{cases} 1 & \text{if } v_j \in \mathcal{N}(v_i); \\ \mathrm{S}_{ij} & \text{else if } \mathrm{S}_{ij} > t, v_i \in \mathcal{V}_U \text{ and } v_j \in \mathcal{V}_A; \\ 0 & \text{else,} \end{cases} \tag{11}$$

where S^A_{ij} denotes the weight of edge linking node v_i and v_j. With the extended label set $\mathcal{V}_A$ providing more label information, and the new adjacency matrix facilitating the information propagation from $\mathcal{V}_A$ to $\mathcal{V}_U$, we can train a more robust GNN classifier against the noisy for label prediction as

$$\hat{\mathbf{Y}} = f_{\mathcal{G}}(\mathrm{S}^A, \mathbf{X}) \tag{12}$$

where $\hat{\mathbf{Y}}$ is the final label prediction. Similar to the accurate pseudo label miner, the GNN classifier $f_{\mathcal{G}}$ is flexible to various GNNs such as GCN [15] and GIN [38]. The training of $f_{\mathcal{G}}$ utilizes the supervision from both noisy labels and accurate pseudo labels. The loss function can be written as:

$$\mathcal{L}_{\mathcal{G}} = \sum_{v_i \in \mathcal{V}_A} l(\hat{y}_i, y_i), \tag{13}$$

where y_i denotes the noisy label or accurate pseudo label of the node $v_i \in \mathcal{V}_A$ and $\hat{y}_i$ denotes the prediction of node $v_i \in \mathcal{V}_A$.

4.4 Final Objective Function

With edge predictor adding links for facilitating the information propagation, pseudo label miner providing more labels and the GNN classifier predicting the labels, the overall loss function can be written as:

$$\arg\min_{\theta_E, \theta_P, \theta_{\mathcal{G}}} \mathcal{L}_{\mathcal{G}} + \alpha \mathcal{L}_E + \beta \mathcal{L}_P, \tag{14}$$

where θ_E, θ_P, and $\theta_{\mathcal{G}}$ are the parameters of edge predictor f_E, accurate label miner f_P and GNN classifier $f_{\mathcal{G}}$, respectively. α and β are hyperparameters to balance the contributions of adjacency matrix reconstruction loss of f_E and the loss of pseudo label miner. $f_{\mathcal{G}}$, f_E and f_P are jointly trained together with Eq.(14). The details of the *training algorithm* is presented in Appendix A.

5 EXPERIMENTS

In this section, we conduct experiments on real-world datasets to show the effectiveness of the proposed framework. In particular, we aim to answer the following research questions:

- **RQ1** Is the proposed framework NRGNN robust to different types and levels of label noise?
- **RQ2** Is the proposed framework effective under different sizes of noisy labels and graph sparsity?
- **RQ3** Is NRGNN flexible to various GNN backbones and how do the edge predictor and pseudo label miner contribute to NRGNN?

5.1 Experimental Settings

5.1.1 Datasets. We conduct experiments on four widely used benchmark datasets, i.e., Cora, Citeseer, Pubmed [33] and DBLP [30]. The statistics of the datasets are presented in Table 4 in Appendix. The validation and test sets are kept the same as the cited papers to keep consistency. As for the training set, we randomly sample 5% nodes for Cora and Citeseer. For large datasets, i.e., Pubmed and DBLP, we sample 1% nodes to compose the training set. All the training set has no overlap with validation and test sets. Since the labels of these datasets are clean, following [31, 32, 41], we corrupt the labels of training and validation set with two types of label noises:

- **Uniform Noise**: The labels have a probability of p to be uniformly flipped to other classes.
- **Pair Noise**: Labelers are assumed to make mistakes only within the most similar pair classes. More specifically, labels have a probability of p to flip to their pair class.

5.1.2 Implementation Details. We report the average results with standard deviations of 5 runs for all experiments. A two-layer GCN whose hidden dimension is 16 is deployed as the backbone of the edge predictor. Similarly, the pseudo label miner and GNN classifier also uses two-layer GCNs as backbones, respectively. Note that our framework is flexible to use various GNNs, which is demonstrated by the experimental results in Sec 5.5. All hyper-parameters are tuned based on the validation set. We vary α and β among $\{0.001, 0.01, 0.1, 1, 10\}$ and $\{0.001, 0.01, 0.1, 1, 10, 100\}$, respectively. As for t and T_p, we fix them as 0.1 and 0.8 for all the datasets. And the number of negative samples K is set as 50.

5.1.3 Baselines. We compare NRGNN with representative and state-of-the-art GNNs and methods of learning with noisy labels:

- **GCN** [15]: GCN is a popular graph convolutional network based on spectral theory.
- **GIN** [38]: Compared with GCN, GIN could learn more powerful representations of graph structures by using multi-layer perception to process the information aggregated from the neighbors.
- **Self-Training** [20]: It first trains a GCN then picks the most confident pseudo labels of GCN and puts it into the labeled node set to improve the performance of GCN.
- **Forward** [31]: This is a loss correction method. It revises predictions to obtain unbiased loss on noisy training samples.
- **Coteaching+** [41]: This method maintains two networks to select clean samples for each other. More specifically, the small-loss samples that obtain different predictions are selected for training.
- **D-GNN** [29]: It obtains a robust GNNs with backward loss correction [31] which estimates the unbiased loss on clean labels.
- **CP** [44]: Community labels obtained by clustering node embeddings are added to train GCN. It encourages the GCN capture community information to avoid the overfitting to noisy labels.

We use GCN in Self-Training, D-GNN, CP and NRGNN to give predictions. Forward, and Coteaching+ are proposed for i.i.d data. To make a fair comparison, GCN is also adopted as backbone in these methods.

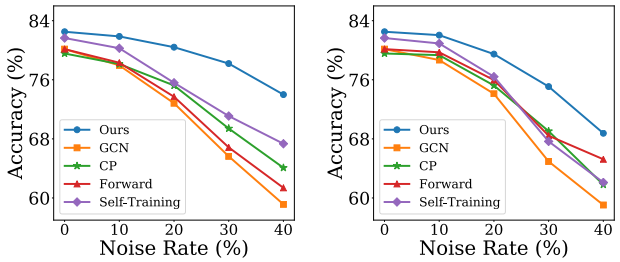
5.2 Node Classification with Noisy Labels

To answer **RQ1**, we compare the proposed framework with baselines on graphs containing two types of label noise. In addition, we conduct node classification on graphs corrupted by different levels of label noise to demonstrate the effectiveness of our method.

5.2.1 Comparisons with Baselines. Two types of label noise, i.e., uniform and pair noise, are considered for all datasets. The noise rate, i.e., the probability that a provided label is not correct, is set

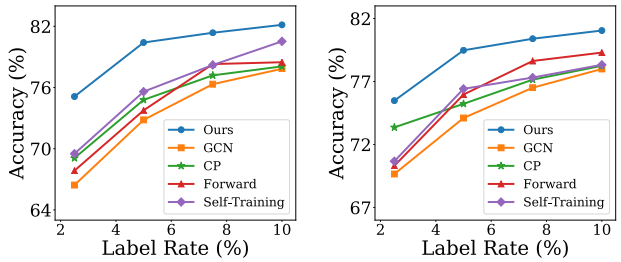
Table 3: Node classification performance (Accuracy (%)±Std) under various types of noise.

Dataset	Noise	GCN	GIN	Self-Training	Forward	Coteaching+	D-GNN	CP	Ours
Cora	Uniform	72.8 ±1.8	72.3 ±0.9	75.6 ±1.8	73.7 ±0.7	73.6 ±1.7	72.4 ±1.8	74.8 ±1.3	**80.4 ±0.5**
	Pair	74.1 ±0.7	74.7 ±1.4	76.4 ±1.4	76.0 ±0.7	73.8 ±1.4	73.5 ±1.6	75.2 ±1.4	**79.5 ±0.4**
Citeseer	Uniform	64.9 ±1.7	65.7 ±2.1	67.8 ±1.4	65.0 ±1.5	66.4 ±1.3	64.9 ±1.3	66.0 ±1.6	**70.1 ±1.3**
	Pair	60.3 ±1.0	61.6 ±1.0	62.0 ±1.6	61.6 ±0.4	65.1 ±2.1	62.3 ±1.2	62.0 ±1.0	**67.8 ±1.3**
Pubmed	Uniform	77.3 ±0.9	77.4 ±0.5	78.2 ±0.4	77.5 ±0.4	78.6 ±0.4	77.6 ±0.3	78.6 ±0.3	**80.0 ±0.2**
	Pair	78.0 ±0.4	78.1 ±0.6	78.9 ±0.8	79.6 ±0.2	78.5 ±0.1	79.4 ±0.4	77.9 ±0.3	**80.0 ±0.3**
DBLP	Uniform	71.0 ±1.5	72.4 ±0.7	74.9 ±0.7	73.1 ±0.3	73.5 ±1.3	72.8 ±1.2	74.2 ±0.5	**80.8 ±0.4**
	Pair	72.5 ±1.2	73.4 ±2.1	76.3 ±1.6	74.4 ±0.5	72.7 ±1.2	75.4 ±0.9	73.6 ±1.0	**81.1 ±0.3**



(a) Uniform Noise **(b) Pair Noise**

Figure 2: Accuracy on Cora with various levels of label noise.



(a) Uniform Noise **(b) Pair Noise**

Figure 3: Accuracy on Cora with various noisy label sizes.

as 20% for both types of label noise. The size of the noisy labels is the same as the description in Sec 5.1.1. The average results and standard deviations of 5 runs are reported in Table 3. From this table, we have the following observations:

- Both GCN and GIN perform poorly on graph with noisy and limited labels; while methods utilizing pseudo labels such as Self-Training have significantly better performance. This implies pseudo labels are helpful to alleviate the issue of learning with noisy and limited labels.
- Compared with Self-Training and CP which also utilize pseudo labels, the proposed NRGNN achieve higher performance under various scenarios, which is because NRGNN adopts edge predictor to add missing links between unlabeled nodes and nodes with noisy labels or pseudo labels to reduce the negative effects of the label noise. Meanwhile, these added links also help to obtain pseudo labels in higher quality .
- The loss correction or sample selection based methods such as Coteaching+ and D-GNN bring limited improvements, which is due to the small training set in semi-supervised learning setting. By contrast, the proposed NRGNN outperforms these baselines by a large margin, which is because NRGNN adopts a pseudo label miner to extend the size of labeled nodes and mitigates the effects of label noise by linking unlabeled nodes and extended labeled nodes.

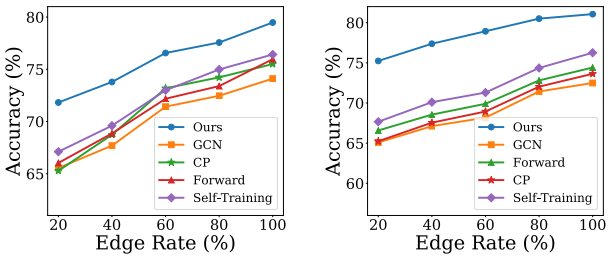
5.2.2 Performance under Different Levels of Label Noise. To demonstrate the effectiveness of the proposed NRGNN under different levels of label noise, we vary the noise rate as $\{0\%, 10\%, \ldots, 40\%\}$. The most effective baselines in Table 3 are implemented for comparisons. We only report the results on Cora, because we have similar observations for other datasets. As mentioned in Sec 5.1.1, 5% nodes are randomly sampled to compose the training set. The average performance of 5 runs is shown in Figure 2. From the figure, we have the following observations:

- As the label noise level increases, the performance of all baselines drop dramatically. Though the performance of NRGNN also drops, it is more resistant to the label noise. The performance gap between NRGNN and the baselines increases when more noise exists in the labels. This implies the effectiveness of handling noisy and limited labels by extending the label set with accurate pseudo labels and adding missing links between the unlabeled and extended labeled node set.
- When there is little or no label noise, our proposed method still outperforms GCN and methods utilizing pseudo labels such as self-training. This is because adding high-quality edges between unlabeled nodes and extended labeled nodes could facilitate the message passing of GNNs.

5.3 Impacts of Noisy Label Size

In this subsection, we investigate how the size of noisy labels would affect NRGNN to answer **RQ2**. We vary the label rate, i.e., the training size, as $\{2.5\%, 5\%, 7.5\%, 10\%\}$. The noise rates of both uniform and pair noise are set as 0.2. We only report the results on Cora in Figure 3 as we have similar observations on other datasets. Each experiment is run 5 times. From Figure 3, we observe:

- Our proposed method brings the most significant performance improvements when the label rate is as small as 2.5%. It indicates the effectiveness of mining accurate pseudo labels to have more supervision and benefit more from adding missing links between the unlabeled nodes and nodes with accurate pseudo labels.
- With the increase of label size, the gap between our method and the baselines only decrease slightly but is still large. This is because accurate pseudo labels play a less important role when the provided noisy labels are sufficient. Though the noisy labels are sufficient, corrupted labels still degrade the performance of GNNs. The proposed NRGNN leverages the edge predictor to link more unlabeled nodes and labeled nodes to alleviate the negative effects of label noise. Thus, it can still outperform the baselines when the size of labeled nodes are large.



(a) Cora (b) DBLP

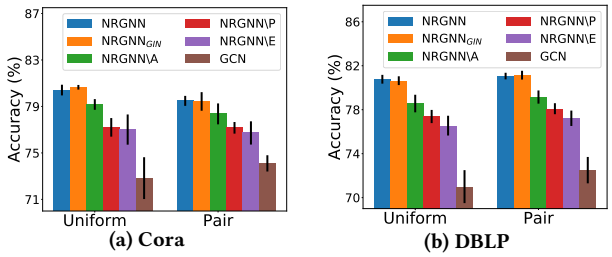
Figure 4: Performance on graphs with different densities.

5.4 Impacts of the Graph Sparsity

The proposed NRGNN relies on an edge predictor to predict the missing links between the unlabeled nodes and labeled nodes to alleviate the effects of noisy labels. And the supervision from the adjacency matrix is utilized to have a good edge predictor. A natural question is whether NRGNN is effective when the graph is very sparse. Thus, to demonstrate that the edge predictor could learn to add useful links for robust node classification with very sparse graph, we train our model on sparse graphs obtained by randomly selecting a subset of edges in original graphs. More specifically, we vary the edge rate. i.e., the ratio of the selected edges, from 20% to 100% with a step of 20%. We only report the results on Cora and DBLP corrupted by pair noise. The noise rate is set as 20%. Average results of 5 runs are shown in Fig. 4. From the figure, we can observe that our proposed model consistently outperforms the baselines by a large margin on graphs of different sparse levels. This indicates that even with a very sparse graph, the learned edge predictor still could predict useful links between unlabeled nodes and nodes with noisy labels or pseudo labels to benefit the accurate pseudo label mining and alleviate the effects of label noise.

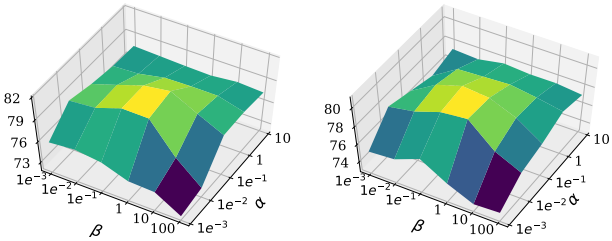
5.5 Ablation Study

To answer **RQ3**, we conduct ablation study to investigate the flexibility of our proposed NRGNN and the contributions of the edge predictor and the pseudo label miner. To investigate whether various GNNs could be benefited from NRGNN, we replace the GCN classifier with a GIN classifier. More specifically, the GIN classifier is trained on the graph densified by linking similar unlabeled nodes and extended labeled nodes with the accurate pseudo labels produced by NRGNN. This variant is named as NRGNN$_{GIN}$. To demonstrate the effectiveness of the GNN-based edge predictor, we train a variant NRGNN\E by replacing the edge predictor with cosine similarity scores of raw features. To show the importance of the pseudo label miner, we analyze it from two aspects. Firstly, to show the contributions of pseudo labels, we train a variant NRGNN\P which does not utilize pseudo labels. Secondly, to investigate how the quality of pseudo labels will influence the final results, we replace the accurate pseudo label miner with a GCN trained on the initial graph to obtain a variant named as NRGNN\A. All the hyperparameters of these variants are tuned following the process described in Sec 5.1.2. Since we have similar observations in other datasets, we only report the performance on Cora and DBLP. The label rate is set the same as the description in Sec 5.1.1. The noise rate is set as 20%. The results of 5 runs are reported in Figure 5. From this figure, we can observe:



(a) Cora (b) DBLP

Figure 5: Comparisons between NRGNN and its variants.



(a) Uniform (b) Pair

Figure 6: Parameter sensitivity analysis on Cora.

- NRGNN$_{GIN}$ achieves comparable results with NRGNN, which indicates that NRGNN is flexible to various GNN backbones.
- The performance of NRGNN\E is significantly worse than that of NRGNN, which shows the necessity of learning a high quality edge predictor to predict the missing links between unlabeled nodes and extended labeled nodes.
- The performance of NRGNN is better than that of NRGNN\A and NRGNN\P, which implies that pseudo label miner is helpful for learning a robust GNN with noisy and limited labels and high quality pseudo labels can bring more benefits.
- NRGNN\P outperforms GCN by a large margin, which demonstrates that linking unlabeled nodes with labeled nodes can alleviate the effects of label noise.

5.6 Hyperparameter Sensitivity Analysis

In this subsection, we investigate how the hyperparameters α and β affect the performance of NRGNN. α controls how well the edge predictor reconstructs the initial graph, and β controls the learning of the pseudo label miner and its impact to the edge predictor. To explore the parameter sensitivity, we alter α and β as $\{0.001, 0.01, 0.1, 1, 10\}$ and $\{0.001, 0.01, 0.1, 1, 10, 100\}$, respectively. We report the results on the Cora graph corrupted by uniform and pair noise with noise rate set as 20%. The experiments are conducted 5 times and the average results are shown in Figure 6. From the figure, we observe (i) Generally, with the increasing of α, the performance tends to first increase and then decrease. A too small α would lead to a weak edge predictor while a large α may dominate the whole loss of NRGNN. The performance is relatively good and stable when α is between 0.01 and 0.1 , which eases the parameter selection for NRGNN. (ii) Similarly, with the increment of β, the performance tends to first increase and then decrease. When β is between 0.1 and 10, the performance is relatively good.

6 CONCLUSION

In this paper, we investigate a novel problem of semi-supervised node classification of GNN on sparsely and noisily labeled graphs.

We theoretically and empirically verify the effectiveness of linking unlabeled nodes with noisily labeled nodes under mild conditions. We also show that pseudo labels could help to alleviate the limited label issue. Based on the analysis, we propose a novel framework NRGNN which utilizes an edge predictor to predict missing links for connecting unlabeled nodes with labeled nodes, and a pseudo label miner to expand the label set. With the new graph and the extended label set, a more robust GNN is trained for node classification. Experimental results on real-world datasets show the effectiveness of the proposed NRGNN on graphs with various types and levels of noise and different label and graph sparsity. Further experiments are conducted to understand the parameter sensitivity. There are several interesting directions need further investigation. First, in this paper, we mainly evaluate NRGNN under two types of noises. In practice, an adversary might on purposely attack the graph by flipping some labels to reduce the performance of GNN. We will investigate the robustness of NRGNN under adversarial label-flipping. Second, for some applications, the edges and node attributes of the given graph can also be noisy, which might affect the edge prediction. Thus, we will study how to extend NRGNN on noisy graphs with noisy labels.

7 ACKNOWLEDGEMENTS

This material is based upon work supported by, or in part by, the National Science Foundation (NSF) under grant #IIS-1909702, #IIS1955851, and Army Research Office (ARO) under grant #W911NF-21-1-0198. The findings and conclusions in this paper do not necessarily reflect the view of the funding agency.

REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. *ICLR* (2014).
[2] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *ICLR* (2018).
[3] Enyan Dai and Suhang Wang. 2021. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information. In *WSDM*. 680–688.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*. 3844–3852.
[5] Hande Dong, Jiawei Chen, Fuli Feng, Xiangnan He, Shuxian Bi, Zhaolin Ding, and Peng Cui. 2020. On the Equivalence of Decoupled Graph Convolution Network and Label Propagation. *arXiv preprint arXiv:2010.12408* (2020).
[6] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. *ICML* (2017).
[7] Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks using a noise adaptation layer. (2016).
[8] Chen Gong, Hengmin Zhang, Jian Yang, and Dacheng Tao. 2017. Learning with inadequate and incorrect supervision. In *ICDM*. IEEE, 889–894.
[9] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.
[10] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *arXiv preprint arXiv:1804.06872* (2018).
[11] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[12] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).
[13] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*. PMLR, 2304–2313.
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[16] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).
[17] Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467 (2018), 312–322.
[18] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.
[19] Junnan Li, Richard Socher, and Steven CH Hoi. 2020. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394* (2020).
[20] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI* (2018).
[21] Rui Li, Shengjie Wang, and Kevin Chen-Chuan Chang. 2012. Multiple location profiling for users and relationships from social network and content. *arXiv preprint arXiv:1208.0288* (2012).
[22] Xingjun Ma, Yisen Wang, Michael E Houle, Shuo Zhou, Sarah Erfani, Shutao Xia, Sudanthi Wijewickrema, and James Bailey. 2018. Dimensionality-driven learning with noisy labels. In *ICML*. PMLR, 3355–3364.
[23] Eran Malach and Shai Shalev-Shwartz. 2017. Decoupling" when to update" from" how to update". *arXiv preprint arXiv:1706.02613* (2017).
[24] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.
[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.
[26] Mark Newman. 2018. *Networks*. Oxford university press.
[27] Duc Tam Nguyen, Chaithanya Kumar Mummadi, Thi Phuong Nhung Ngo, Thi Hoai Phuong Nguyen, Laura Beggel, and Thomas Brox. 2019. Self: Learning to filter noisy labels with self-ensembling. *arXiv preprint arXiv:1910.01842* (2019).
[28] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.
[29] Hoang NT, Choong Jun Jin, and Tsuyoshi Murata. 2019. Learning graph neural networks with noisy labels. *arXiv preprint arXiv:1905.01591* (2019).
[30] Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. 2016. Tri-party deep network representation. *Network* 11, 9 (2016), 12.
[31] Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*. 1944–1952.
[32] Scott Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2014. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596* (2014).
[33] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.
[34] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring Robustness for Graph Neural Network Against Poisoning Attacks. In *WSDM*. 600–608.
[35] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and Mitigating Degree-Related Biases in Graph Convoltuional Networks. In *CIKM*. 1435–1444.
[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *ICLR* (2017).
[37] Daixin Wang, Jianbin Lin, Peng Cui, Quanhui Jia, Zhen Wang, Yanming Fang, Quan Yu, Jun Zhou, Shuang Yang, and Yuan Qi. 2019. A semi-supervised graph attentive network for financial fraud detection. *ICDM* (2019).
[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[39] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *SIGKDD*. 974–983.
[40] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2017. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875* (2017).
[41] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. 2019. How does disagreement help generalization against label corruption?. In *International Conference on Machine Learning*. PMLR, 7164–7173.
[42] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).
[43] Huan Zhang, Zhao Zhang, Mingbo Zhao, Qiaolin Ye, Min Zhang, and Meng Wang. 2020. Robust triple-matrix-recovery-based auto-weighted label propagation for classification. *IEEE TNNLS* 31, 11 (2020), 4538–4552.
[44] Mengmei Zhang, Chuan Shi, Linmei Hu, and Xiao Wang. 2020. Adversarial Label-Flipping Attack and Defense for Graph Neural Networks. *ICDM* (2020).
[45] Tianxiang Zhao, Xianfeng Tang, Xiang Zhang, and Suhang Wang. 2020. Semi-Supervised Graph-to-Graph Translation. In *CIKM*. 1863–1872.
[46] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. In *WSDM*. 833–841.

Algorithm 1: Training Algorithm of NRGNN.

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, $\mathcal{Y}$, K, t, T_p, α and β.
Output: $f_{\mathcal{G}}$, f_P and f_E
1: Pretrain f_P and f_E with Eq.(6) and Eq.(9)
2: **repeat**
3: Obtain the graph S^L with f_E by Eq.(7).
4: Feed S^L to f_P to obtain pseudo labels $\mathcal{Y}_P$ by Eq.(10)
5: Generate the graph S^A for $f_{\mathcal{G}}$ with f_E by Eq.(11)
6: Jointly optimize the parameters of $f_{\mathcal{G}}$, f_P and f_E by Eq.(14)
7: **until** convergence
8: **return** $f_{\mathcal{G}}$, f_P and f_E

A TRAINING ALGORITHM

The training algorithm of NRGNN is shown in Algorithm 1. In line 1, edge predictor f_E and accurate pseudo label miner f_P will be pretrained with Eq.(6) and Eq.(9). In line 2, we generate S^P for f_P with f_E. Then, the accurate pseudo labels could be obtained. In line 5, the graph S^A which linking nodes with similar extended labeled nodes is obtained for $f_{\mathcal{G}}$ to make robust predictions. Finally, $f_{\mathcal{G}}$, f_E and f_P will be jointly trained with an Adam optimizer [14] with the learning rate set as 0.001.

B PROOF OF THEOREM 3.1

Proof. The predicted probability that node v_u belongs to the class c could be rewritten to the following format:

$$y_{uc} = \frac{1}{m+n}\Big(\sum_{v_a \in \mathcal{V}_a} s_{ac} + \sum_{v_l \in \mathcal{V}_n} s_{lc}\Big), \tag{15}$$

where $\mathcal{V}_a$ denotes the unlabeled neighbors of v_u, $\mathcal{V}_n$ denotes the linked nodes with noisy labels. Let p_t denotes the probability that a node belonging to class c is assigned to label c, and p_f denotes the probability that a node not belonging to class c is assigned to label c. Then average value of y_{uc} would be:

$$\begin{aligned}\mathbb{E}(y_{uc}) =& \frac{n}{m+n}\mathbb{E}(s_{ac}) + \frac{(hp_t + (1-h)p_f))m}{m+n}\mathbb{E}(s_{bc}) \\ &+ \frac{(h(1-p_t) + (1-h)(1-p_f))m}{m+n}\mathbb{E}(s_{dc}),\end{aligned} \tag{16}$$

where s_{ac} corresponds to the unlabeled node $v_a \in \mathcal{V}_U$, s_{bc} corresponds to the labeled node $v_b \in \mathcal{V}_L$ whose provided label is c, and s_{dc} corresponds to the labeled node $v_d \in \mathcal{V}_L$ whose provided label is not c. Since $p_t > p_f$, we could have $p = (hp_t + (1-h)p_f) < p_t$. And Eq.(16) could be rewritten to:

$$\mathbb{E}(y_{uc}) = \frac{n\mathbb{E}(s_{ac}) + pm\mathbb{E}(s_{bc}) + (1-p)m\mathbb{E}(s_{dc})}{m+n}. \tag{17}$$

If we further link v_u with k labeled nodes which belong to c. Then we could obtain the corresponding predicted probability y^k_{uc}. The expectation of y^k_{uc} can be written as:

$$\mathbb{E}(y^k_{uc}) = \frac{m+n}{m+n+k}\mathbb{E}(y_{uc}) + \frac{kp_t\mathbb{E}(s_{bc}) + k(1-p_t)\mathbb{E}(s_{dc})}{m+n+k}. \tag{18}$$

Since $p < p_t$ and $\mathbb{E}(s_{bc}) > \mathbb{E}(s_{ac}) > \mathbb{E}(s_{dc})$, we can derive that

$$p_t\mathbb{E}(s_{bc}) + (1-p_t)\mathbb{E}(s_{dc}) > p\mathbb{E}(s_{bc}) + (1-p)\mathbb{E}(s_{dc}). \tag{19}$$

Table 4: Statistics of datasets.

	Cora	Citeseer	Pubmed	DBLP
# of nodes	2,485	2,110	19,717	17,716
# of edges	5,068	3,668	44,338	52,867
# of features	1,433	3,703	500	1,639
# of classes	7	6	3	4

When $p_t > \frac{\mathbb{E}(s_{ac}) - \mathbb{E}(s_{dc})}{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{dc})}$, we could have

$$p_t\mathbb{E}(s_{bc}) + (1-p_t)\mathbb{E}(s_{dc}) > \mathbb{E}(s_{ac}). \tag{20}$$

Combining Eq.(19) and Eq.(20), we can derive

$$p_t\mathbb{E}(s_{bc}) + (1-p_t)\mathbb{E}(s_{dc}) > \mathbb{E}(y_{uc}). \tag{21}$$

Therefore, we could conclude $\mathbb{E}(y^k_{uc}) > \mathbb{E}(y_{uc})$. And with the increasing of k, the predicted probability that node v_u belonging to class c would increase. □

C PROOF OF THEOREM 3.2

Proof. The average value of y_{uc} could be written as:

$$\mathbb{E}(y_{uc}) = \frac{n\mathbb{E}(s_{ac}) + pm\mathbb{E}(s_{bc}) + (1-p)m\mathbb{E}(s_{dc})}{m+n}, \tag{22}$$

Since $\mathbb{E}(s_{pc}) > \mathbb{E}(s_{ac})$ and $\mathbb{E}(s_{pc}) > p\mathbb{E}(s_{bc}) + (1-p)\mathbb{E}(s_{dc}))$, then we could have $\mathbb{E}(s_{pc}) > \mathbb{E}(y_{uc})$. Therefore, the expectation of y_{uc} after linking k nodes with pseudo labels would be:

$$\mathbb{E}(y^k_{uc}) = \frac{m+n}{m+n+k}\mathbb{E}(y_{uc}) + \frac{k}{m+n+k}\mathbb{E}(s_{pc}). \tag{23}$$

Since $\mathbb{E}(s_{pc}) > \mathbb{E}(y_{uc})$, we could conclude that with the increasing of k, $\mathbb{E}(y^k_{uc})$ would be higher. □