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

Enyan Dai  
The Pennsylvania State University  
em5759@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 such as node classification. Despite the great success of GNNs, many real-world graphs are often sparsely and noisily labeled, which could significantly degrade the performance of GNNs, as the noisy information could propagate to unlabeled nodes via graph structure. Thus, it is important to develop a label noise-resistant GNN for semi-supervised node classification. Though extensive studies have been conducted to learn neural networks 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 directly 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 and empirical analysis verify the effectiveness of these two strategies under mild conditions. Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed method in learning a robust GNN with noisy and limited labels.

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 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 \( v_j \) 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 \( v_j \), 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 Noise-Resistant 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). 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

In this section, we present the related literature of graph neural networks and deep learning with noisy labels.

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 data [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 samples for each other. Moreover, Coteaching+ [41] incorporates

https://github.com/EnyanDai/NRGNN
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 noisy and sparsely labeled graphs are analyzed theoretically and empirically.

3.1 Notation

We use $G = (\mathcal{V}, \mathcal{E})$ 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 $G$, where $A_{ij} = 1$ if nodes $v_i$ and $v_j$ are connected, otherwise $A_{ij} = 0$. $\mathbf{X} = \{x_1, ..., x_N\}$ is the set of node attributes with $x_i$ being the node attributes of node $v_i$. $\mathcal{V}_k = \{v_1, ..., v_k\}$ is a set of labeled nodes. $\mathcal{V}_l = \mathcal{V} - \mathcal{V}_k$ is a set of unlabeled nodes. The provided labels of $\mathcal{V}_l$ are corrupted by noise, which are denoted as $\mathcal{Y}_N = \{y_1', ..., y_k'\}$. And $\mathcal{Y}_l = \{y_{k+1}', ..., y_l'\}$ 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:

$$a^{(k)}_u = \text{AGGREGATE}^{(k-1)}(\{h^{(k-1)}_w : u \in \mathcal{N}(v)\}),$$

$$h^{(k)}_v = \text{COMBINE}^{(k)}(h^{(k-1)}_v, a^{(k)}_u),$$

where $h^{(k)}_v$ is the representation vector of the node $v \in \mathcal{V}$ at the $k$-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:

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

where $H^{(k)}$ is the representation matrix of the output of the $k$-th layer; $\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A} + 1)\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix and $\mathbf{D}$ is a diagonal matrix with $D_{ii} = \sum_j A_{ij}$. $\mathbf{I}$ is the identity matrix and $\sigma$ 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 $G = (\mathcal{V}, \mathcal{E}, X)$ 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.,

$$f(G, \mathcal{Y}_N) \rightarrow \tilde{\mathcal{Y}}_l$$

where $f$ is the function we aim to learn and $\tilde{\mathcal{Y}}_l$ 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 $\mathbf{Y} = \tilde{\mathbf{H}}^{(K)}\mathbf{W}^{(K)}$. Since the parameters $\theta$ are well trained for node classification, the $K$-th latent representations, i.e., $\mathbf{S} = \tilde{\mathbf{H}}^{(K)}\mathbf{W}^{(K)}$ could be treated as predictions of the nodes [5]. And the final predictions are obtained by the aggregation of $\mathbf{S}$, i.e., $\mathbf{Y} = \mathbf{A}S$. Let’s treat $s_{bk}$ as the predicted probability that node $v_i$ belongs to class $k$. For an unlabeled $v_i \in \mathcal{V}_l$ belonging to class $c$, we consider three types of neighbors: (i) an unlabeled node $v_j \in \mathcal{V}_l$; (ii) a node $v_j \in \mathcal{V}_l$ labeled as $c$; and (iii) a node $v_j \in \mathcal{V}_l$ 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 $v_a \in \mathcal{V}_l$ 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 $p_b$ that a node belonging to class $c$ is labeled as $c$ meets this constraint: $p_b > \frac{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{ac})}{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{dc})}$.

Then, linking $v_a$ with more similarly labeled nodes $v_j \in \mathcal{V}_l$ can on average improve its predicted probability of belonging to class $c$, i.e., improve $y_{ac} = \frac{1}{d_a} \sum_{j \in \mathcal{N}(v_a)} y_{cj}$.

The proof of this theorem is presented in Appendix B. When a graph network framework for multi-class classification is corrupted with label noise, the predicted probability of an unlabeled node $v_a \in \mathcal{V}_l$ would be much smaller that the probability that a node labeled as $c$. Therefore, the assumption that $p_b > \frac{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{ac})}{\mathbb{E}(s_{bc}) - \mathbb{E}(s_{dc})}$ 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 $G$:** We train a GCN on the initial graph structure with noisy labels as the baseline.
With the analysis in Sec. 3.4, we find that linking more existing nodes is labeled as \( c \) nodes. Let \( p_c \) denotes the probability that the existing linked labeled nodes are in 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 \( v_p \) which is provided with noisily labeled nodes, 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 \( f_G \). The edge predictor takes the initial graph \( G \) as input to predict edges. \( f_E \) will first link similar nodes between \( \mathcal{V}_L \) 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, if \( \mathbb{E}(s_{pc}) > \max(\mathbb{E}(s_{ac}), p\mathbb{E}(s_{hc})+(1-p)\mathbb{E}(s_{lc})) \), then, linking \( v_u \) with \( v_p \) can improve its predicted probability of belonging to class \( c \), i.e., \( y_{uc} = \frac{1}{d_c} \sum_{j \in N(v_i)} s_{jc} \).

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: (i) 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 \( V_P \) to compose extended label set \( \mathcal{V}_A = \mathcal{V}_L \cup \mathcal{V}_P \); 3) link \( \mathcal{V}_L \) and \( \mathcal{V}_A \) based on cosine similarity of raw features, and train a GNN 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 GNN 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.

### 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 noisy 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 \( v_p \) belongs to class \( c \) based on the \( K \)-th latent representations, i.e., \( S = \mathbf{W}^{(K)} \mathbf{W}^{(K)} \). The following theorem verifies the effectiveness of this strategy when \( s_{pc} \) meets a mild constraint.

**Theorem 3.2.** We consider an unlabeled node \( v_u \) 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 \( v_p \) which is provided with

| Dataset | Noise Rate | Initial \( G \) | Link \( \mathcal{V}_L \) | Link \( \mathcal{V}_L \) (Retrain) | Link \( \mathcal{V}_A \) |
|---------|------------|----------------|----------------|-----------------------------|----------------|
| 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        | 66.1 ± 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 \( G \) | 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     |
labels, denoted as $\mathcal{V}_P$. With the extended label set, the edge predictor $f_E$ further connects unlabeled nodes $\mathcal{V}_U$ 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 $f_G$ 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 nodes 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:

$$Z = \text{GCN}(A, X).$$

Let $z_i$ and $z_j$ denote the representations of node $v_i$ and $v_j$, respectively. The closer $z_i$ and $z_j$ 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

$$S_{ij} = \sigma(z_i^T z_j^T),$$

where $\sigma(\cdot)$ is the activation function. Because the learned weights $S$ 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 $A$, 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 $A$ 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:

$$\min_{f_E} \mathcal{L}_E = \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} ((S_{ij} - 1)^2 + \sum_{n=1}^{K} E_{n} \cdot P_n(v_i) (S_{in} - 0)^2),$$

where $\mathcal{N}(v_i)$ 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:

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

where $S^L_{ij}$ 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 $S^L$, we can train a GNN classifier as pseudo label miner $f_P$. The pseudo labels of nodes $\mathcal{V}$ is predicted as:

$$\hat{\mathcal{Y}}^P = \text{GNN}(S^L, X),$$

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

$$\min_{\theta_P} \mathcal{L}_P = \sum_{v_i \in \mathcal{V}_U} l(\hat{y}^P_{i}, y_i),$$

where $\theta_P$ 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 $S^L$, 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 $\hat{y}^L_{i}$ denotes the predicted probability that node $v_i$ belongs to the class $c$. Then the accurate pseudo labels is obtained by the following process:

$$\mathcal{Y}_P = \{\hat{y}^L_{i} \in \hat{\mathcal{Y}}^P_U ; \hat{y}^L_{i} > T_p\},$$

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 $\hat{L}_p$. Similar to the construction of $S^p_l$, we use the same threshold $t$ to select links. This process is written as:

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

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

$$\hat{Y} = f_G(S^A, X)$$

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

$$L_G = \sum_{v_i \in \mathcal{V}_A} l(\hat{y}_i, y_i),$$

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:

$$\arg\min_{\theta_E, \theta_P, \theta_G} L_G + \alpha L_E + \beta L_P,$$

where $\theta_E$, $\theta_P$, and $\theta_G$ are the parameters of edge predictor $f_E$, accurate label miner $f_p$ and GNN classifier $f_G$, respectively. $\alpha$ and $\beta$ are hyperparameters to balance the contributions of adjacency matrix reconstruction loss of $f_E$ and the loss of pseudo label miner. $f_G$, $f_p$, and $f_E$ 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 Citepeer. 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

As for the training set, we randomly sample 5% 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 $\alpha$ and $\beta$ 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 to 0.001, 0.01, 0.1, 1, 10, 0.001, 0.01, 0.1, 1, 10, 100, respectively. For all the datasets, 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.
Table 3: Node classification performance (Accuracy (%)$\pm$ 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 |

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

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

As the label noise level increases, the performance of all baselines drop dramatically. Though the performance of NRGGNN also drops, it is more resistant to the label noise. The performance gap between NRGGNN 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 NRGGNN 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 NRGGNN 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.

5.2.2 Performance under Different Levels of Label Noise. To demonstrate the effectiveness of the proposed NRGGNN under different levels of label noise, we vary the noise rate as {0%, 10%, ..., 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:
which does not utilize pseudo labels. Secondly, to investigate how
RQ3

The proposed NRGNN relies on an edge predictor to predict the
vary the edge rate. i.e., the ratio of the selected edges, from 20% to
noise rate is set the same as the description in Sec 5.1.1. The noise
datasets, we only report the performance on Cora and
described in Sec 5.1.2. Since we have similar observations in other
per parameters of these variants are tuned following the process
initial graph to obtain a variant named as NRGNN
pseudo label miner, we analyze it from two aspects. Firstly, to show

demonstrate the effectiveness of the GNN-based edge predictor, we
predictor and the pseudo label miner. To investigate whether vari-
bility of our proposed NRGNN and the contributions of the edge
predictor reconstructs the initial graph, and
edge predictor is trained on the graph densified by linking similar unlabeled
missing links between the unlabeled nodes and labeled nodes to
node classification of GNN on sparsely and noisily labeled graphs.

5.5 Ablation Study
To answer RQ3, we conduct ablation study to investigate the flexi-
ibility of our proposed NRGNN and the contributions of the edge
predictor and the pseudo label miner. To investigate whether vari-
ous GNNs could be benefited from NRGNN, we replace the GCN
classifier with a GIN classifier. More specifically, the GIN classi-
fier 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\textit{GIN}. To
demonstrate the effectiveness of the GNN-based edge predictor, we
train a variant NRGNN\textit{E} by replacing the edge predictor with co-
sine 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\textit{P}
which does not utilize pseudo labels. Secondly, to investigate how
the quality of pseudo labels will influence the final results, we re-
place the accurate pseudo label miner with a GCN trained on the
initial graph to obtain a variant named as NRGNN\textit{A}. All the hy-
perparameters 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:

5.6 Hyperparameter Sensitivity Analysis
In this subsection, we investigate how the hyperparameters \(\alpha\) and \(\beta\) affect the performance of NRGNN. \(\alpha\) controls how well the
dge predictor reconstructs the initial graph, and \(\beta\) controls the
learning of the pseudo label miner and its impact to the edge pre-
dictor. To explore the parameter sensitivity, we alter \(\alpha\) and \(\beta\) 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 con-
ducted 5 times and the average results are shown in Figure 6. From
the figure, we observe (i) Generally, with the increasing of \(\alpha\), the
performance tends to first increase and then decrease. A too small \(\alpha\)
may dominate to a weak edge predictor while a large \(\alpha\) may dominate
the whole loss of NRGNN. The performance is relatively good and
stable when \(\alpha\) is between 0.01 and 0.1 , which eases the parameter
selection for NRGNN. (ii) Similarly, with the increment of \(\beta\), the
performance tends to first increase and then decrease. When \(\beta\) 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 noisy 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 Caio 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 labels. In WWW.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional networks.
[5] Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 Caio 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 labels. In WWW.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional networks.
[5] Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 Caio 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 labels. In WWW.
[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional networks.
[5] Jacob Goldberger and Ehud Ben-Reuven. 2016. Training deep neural-networks 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 NRGGNN 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 NRGGNN 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 NRGGNN 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 NRGGNN 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,
Algorithm 1: Training Algorithm of NRGNN.

**Input:** \(\mathcal{G} = (\mathcal{V}, E, X), \mathcal{Y}, K, t, T_p, \alpha \) and \(\beta\).

**Output:** \(f_G, f_F\) and \(f_E\)

1. Pretrain \(f_P\) and \(f_F\) with Eq.(6) and Eq.(9)
2. repeat
3. Obtain the graph \(S^t\) with \(f_E\) by Eq.(7).
4. Feed \(S^t\) to \(f_P\) to obtain pseudo labels \(Y_P\) by Eq.(10)
5. Generate the graph \(S^t\) for \(f_G\) with \(f_E\) by Eq.(11)
6. Jointly optimize the parameters of \(f_G, f_P\) and \(f_E\) by Eq.(14)
7. until convergence
8. return \(f_G, f_P\) 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 \(S^t\) for \(f_P\) with \(f_E\). Then, the accurate pseudo labels could be obtained. In line 5, the graph \(S^t\) which linking nodes with similar extended labeled nodes is obtained for \(f_G\) to make robust predictions. Finally, \(f_G, f_E\) 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:

\[
y_{uc} = \frac{1}{m+n} \sum_{v_u \in \mathcal{V}_u} \sum_{v_l \in \mathcal{V}_l} s_{ac} + \sum_{v_l \in \mathcal{V}_l} s_{lc},
\]

(15)

where \(\mathcal{V}_u\) denotes the unlabeled neighbors of \(v_u\), \(\mathcal{V}_l\) 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:

\[
\mathbb{E}(y_{uc}) = \frac{nh_{p_u} + (1-h_{p_u})p_f m + n}{m + n} \mathbb{E}(s_{bc}) + \frac{h(1-p_t) + (1-h)(1-p_f) m + n}{m + n} \mathbb{E}(s_{dc}),
\]

(16)

where \(s_{ac}\) corresponds to the unlabeled node \(v_u \in \mathcal{V}_u\), \(s_{bc}\) corresponds to the labeled node \(v_l \in \mathcal{V}_l\) 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 \(p = (h_{p_u} + (1-h_{p_u})p_f) < p_t\). And Eq.(16) could be rewritten to:

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

(17)

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

\[
\mathbb{E}(y_{uc}^k) = \frac{m + n}{m + n + k} \mathbb{E}(y_{uc}) + \frac{k p_t \mathbb{E}(s_{bc}) + k(1-p_t) \mathbb{E}(s_{dc})}{m + n + k},
\]

(18)

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

\[
p_t \mathbb{E}(s_{bc}) + (1-p_t) \mathbb{E}(s_{dc}) > p \mathbb{E}(s_{bc}) + (1-p) \mathbb{E}(s_{dc}).
\]

(19)

\[\text{Table 4: Statistics of datasets.}\]

|                  | 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

\[
p_t \mathbb{E}(s_{bc}) + (1-p_t) \mathbb{E}(s_{dc}) > \mathbb{E}(s_{ac}).
\]

(20)

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

\[
p_t \mathbb{E}(s_{bc}) + (1-p_t) \mathbb{E}(s_{dc}) > \mathbb{E}(y_{uc}).
\]

(21)

Therefore, we could conclude \(\mathbb{E}(y_{uc}^k) > \mathbb{E}(y_{uc})\). 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:

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

(22)

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

\[
\mathbb{E}(y_{uc}^k) = \frac{m + n}{m + n + k} \mathbb{E}(y_{uc}) + \frac{k}{m + n + k} \mathbb{E}(s_{bc}).
\]

(23)

Since \(\mathbb{E}(s_{bc}) > \mathbb{E}(y_{uc})\), we could conclude that with the increasing of \(k\), \(\mathbb{E}(y_{uc}^k)\) will be higher.

\[\square\]