Robust Node Classification on Graph Data with Graph and Label Noise

Yonghua Zhu^{1,2}, Lei Feng^{3}, Zhenyun Deng^{4}, Yang Chen^{1,2}, Robert Amor^{2}, Michael Witbrock^{1,2}

^{1}NAOInstitute, University of Auckland, NZ
^{2}School of Computer Science, University of Auckland, NZ
^{3}School of Computer Science and Engineering, Nanyang Technological University, Singapore
^{4}Department of Computer Science, University of Cambridge, UK
yzhu970@aucklanduni.ac.nz, lfengqaq@gmail.com, zd302@cam.ac.uk, {yang.chen, r.amor, m.witbrock}@auckland.ac.nz

Abstract

Current research for node classification focuses on dealing with either graph noise or label noise, but few studies consider both of them. In this paper, we propose a new robust node classification method to simultaneously deal with graph noise and label noise. To do this, we design a graph contrastive loss to conduct local graph learning and employ self-attention to conduct global graph learning. They enable us to improve the expressiveness of node representation by using comprehensive information among nodes. We also utilize pseudo graphs and pseudo labels to deal with graph noise and label noise, respectively. Furthermore, We numerically validate the superiority of our method in terms of robust node classification compared with all comparison methods.

Introduction

Node classification is a classic learning task in graph data and has been widely used in real applications, such as citation networks, traffic networks, social networks, *etc.* (Wu et al. 2020). In many node classification methods, node representation is updated by aggregating neighbors' information, assuming that both graph and label information are clean. As a result, these methods are vulnerable to either graph noise (Fox and Rajamanickam 2019) or label noise (Li, Yin, and Chen 2021).

Graph noise points to the noise in the graph, including missing-edges and noisy-edges. With the missing-edges, models cannot aggregate enough intra-class neighbors, easily resulting in limited generalization; with noisy-edges, models easily aggregate inter-class neighbors to lose their discrimination (Bojchevski and Günnemann 2019). Current approaches for dealing with graph noise include three categories, *i.e.,* structure-learning method, weight-learning method, and robust-constraint method. The structure-learning method attempts to reconstruct a denoised graph for enhancing the effectiveness of the message-passing, so that similar nodes' representation can be aggregated to improve their expressiveness (Dai et al. 2022). The weight-learning method specifies different weights to edges to assign noisy edges with small weights in updating node representations (Ye and Ji 2021). Different from the above two methods denoising for the message-passing mechanism, the robust-constraint method attempts to preserve inherent graph properties, such as low rank (Cheng, Miao, and Qiu 2020), sparsity (Ye and Ji 2021), and Laplacian (Runwal, Kumar et al. 2022), to improve the denoising ability of node representation. However, these methods focus on conducting local graph learning (Runwal, Kumar et al. 2022) to update node representation. Obviously, they don't capture the global information among nodes, which has been proven to effectively improve the robustness of node classification via introducing more edges beyond the primitive graph (Huang et al. 2023). Moreover, these methods are parameterized by back-propagation involving label information, leading to a sensitivity to label noise.

It is critical to train reliable deep-learning models under label noise because accurate label annotation is time-consuming and requires expert knowledge. It has been validated that deep learning models easily overfit to noisy labels with any ratio of corrupted labels (Zhang et al. 2021). To address this issue, many approaches have been proposed by training robust models with label noise, such as sample selection method (Sukhbaatar et al. 2014), robust loss design method (Song, Kim, and Lee 2019), and robust regularization method (Srivastava et al. 2014). However, these methods cannot be directly extended for node classification in the graph data because of issues such as sparse labeling (Dai, Aggarwal, and Wang 2021) and severe scarcity of label information caused by label noise. Recently, studies have attempted to introduce more supervision information based on the graph, *e.g.,* using label propagation over graphs (Dai, Aggarwal, and Wang 2021). However, with the message-passing mechanism, label noise is easily back-propagated to disturb the representation learning of other nodes. As a result, this makes these methods sensitive to the graph quality.

Studies have demonstrated that graph noise and label noise widely exist in graph data simultaneously (Zhong et al. 2019). Most existing studies concentrate on dealing with either graph noise or label noise. Few works attempt to tackle both in a unified framework. The primary challenges arise due to the limited number of available labels for node classification, and both graph noise and label noise further intensify the scarcity of labels. Specifically, previous node classification methods first aggregate neighbor information in the graph to make updated node representation more distinguishing, followed by the back-propagation process merely

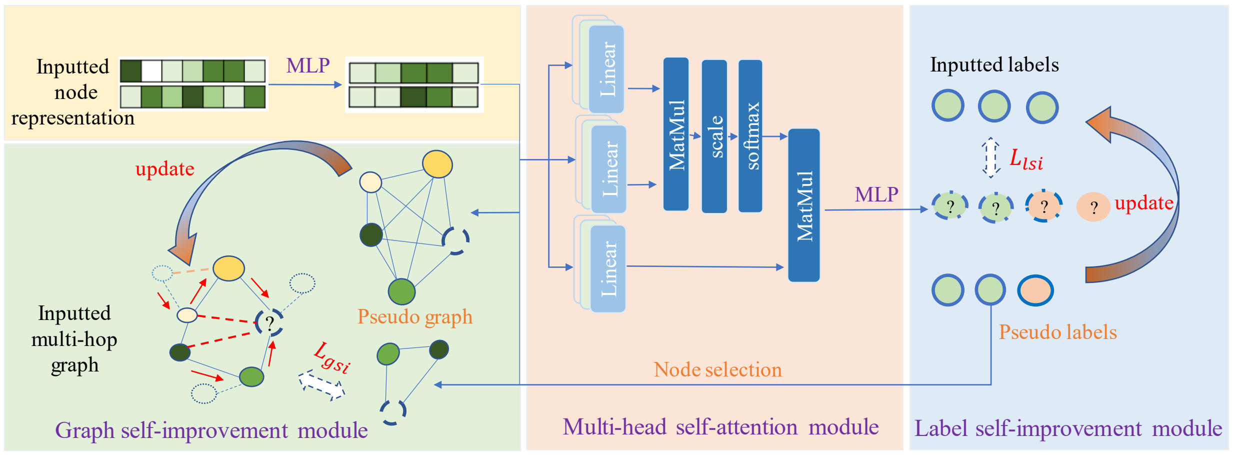


Figure 1: The framework of the proposed method RNCGLN. The feature matrix $\mathbf{X}$ is first fed into an MLP, and then a graph contrastive loss is designed to capture local information in the multi-hop graph, *i.e.,* local graph learning, is also used to deal with graph noise. The output node representation is fed into the multi-head self-attention module to consider the global information beyond the primitive graph, *i.e.,* global graph learning. In the label self-improvement module, the node representation and the original labels are used to construct a classifier, which is further used for predicting all labeled and unlabeled nodes. The original labels with low predictive confidence are replaced because they are considered noise.

involving label information. As a result, graph noise can easily produce incorrect label prediction, and label noise is back-propagated along the noisy graph to impact the representation learning of other nodes. Therefore, node classification is sensitive to these two kinds of noise. Moreover, their coexistence makes node classification worse.

To tackle the above issues, we propose a new method RNCGLN shown in Figure 1, namely **R**obust **N**ode **C**lassification under **G**raph and **L**abel **N**oise, which consists of three modules, *i.e.,* a graph self-improvement module, a multi-head self-attention module, and a label self-improvement module. Specifically, in the graph self-improvement module, we design a graph contrastive loss to conduct local graph learning and deal with graph noise. The output representation is then fed into the multi-head self-attention module to amplify its discriminative ability by conducting global graph learning. In the label self-improvement module, the node representation learned by the first two modules and labels is used to construct a classifier, which is further used to detect the label noise, *i.e.,* the labels with low predictive confidence by the classifier.

Compared with previous node classification methods, the main contributions of our method are listed as follows:

- Our method focuses on conducting node classification by simultaneously exploring graph and label noise issues. On the contrary, previous methods only consider either for node classification.
- Different from many previous node classification methods based on graph neural networks (GNNs) (Wu et al. 2020), we employ multi-head self-attention as the backbone network and design a graph contrastive loss for considering both global and local information among the nodes. Furthermore, we numerically prove the effectiveness of our proposed framework over GNNs.

Approach

Motivations

Given a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the node set with n nodes representation $\mathbf{X} \in \mathbb{R}^{n \times d}$, and $\mathbf{E} \in \mathbb{R}^{|\mathbf{V}| \times |\mathbf{V}|}$ is the edge set. Generally, a graph can be denoted as an adjacency matrix $\mathbf{A} \in [0,1]^{n \times n}$, where $a_{ij} = 1$ denotes node v_i connected with node v_j. Given $\mathbf{X}$, $\mathbf{A}$, and a small set of nodes labels $\mathbf{Y}_L$, the goal of node classification is training a model $f(\cdot)$ to minimize $\mathcal{L}(f(\mathbf{X}, \mathbf{A}), \mathbf{Y}_L)$, such that $f(\cdot)$ can accurately predict unlabelled nodes $\mathbf{Y}_U$.

In the literature on node classification, the message-passing mechanism is used to update the node representation via aggregating the neighbor's information. The updating scheme is formulated as follows:

$$\mathbf{Z}^{(l+1)} = \sigma(\mathcal{F}(\mathbf{A})\mathbf{Z}^{(l)}\mathbf{W}) \tag{1}$$

where $\sigma(\cdot)$ and $\mathbf{W}$ denote an activation function and a learnable weight matrix, respectively, and $\mathbf{Z}^{(0)} = \mathbf{X}$. $\mathcal{F}(\mathbf{A})$ is a graph filter (Balcilar et al. 2021) which distinguishes different GNN methods. For example, $\mathcal{F}(\mathbf{A}) = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{\frac{1}{2}}$ in GCN where $\mathbf{D}$ is the degree matrix (Kipf and Welling 2016). After conducting representation learning by Eq. (1), the cross-entropy loss function is used to update networks:

$$\mathcal{L}_{\text{ce}} = -\sum\nolimits_{i \in \mathbf{Y}_L} \sum\nolimits_{j=1}^{C} y_{ij} \log z_{ij} \tag{2}$$

Eq. (1) easily degrades the effectiveness of the message-passing mechanism with graph noise in the graph $\mathbf{A}$. Existing solutions generally focus on conducting local graph learning, *i.e.,* aggregating local neighbor information, to improve the robustness of $\mathcal{F}(\mathbf{A})$, limiting the expressiveness by overlooking crucial neighbors that do not initially present in the graph.

Label noise also impacts the performance of Eq. (2) since it produces error gradient in the back-propagation process. Since only a small set of labels is available, and label noise may reduce the available label information, many existing label-robust methods cannot be directly employed for node classification under noise (Dai, Aggarwal, and Wang 2021).

Considering that real-world graph data generally contain both graph noise and label noise, in this paper, we focus on conducting node classification in the graph data with graph noise and label noise, as shown in Figure 1. To do this, we regard the multi-head self-attention as the backbone to consider the global information among nodes and propose a graph contrastive loss to consider the local information for representation learning. Moreover, the graph self-improvement is designed to deal with graph noise. Furthermore, the output node representation is used to construct a classifier for noisy labels.

Multi-Head Self-Attention Module

Existing robust methods often design structure learning and weight learning (Chen, Wu, and Zaki 2021; Tang et al. 2019) to make minor modifications to the initial graph. However, they predominantly involve local graph learning without comprehensively considering the information among nodes. To address this issue, we adopt multi-head self-attention (Vaswani et al. 2017) as the backbone network, which is supported by the facts that self-attention can be seen as a learnable fully-connected weighting graph and multi-head attention could capture diverse correlations among nodes (Min et al. 2022). In particular, graph noise includes missing-edges and noisy-edges, and multi-head self-attention allows all edges to exist and utilizes task-driven weight learning to deal with both missing-edges and noisy-edges. The multi-head self-attention in the l-th layer is formulated as follows:

$$\begin{cases}\mathbf{Z}^{(1+1)} = \text{Concat}(\text{head}_1, ..., \text{head}_H)\mathbf{W}^c \\ \text{head}_h = \text{softmax}(\frac{(\mathbf{Z}_h^{(l)}\mathbf{W}_h^q)(\mathbf{Z}_h^{(l)}\mathbf{W}_h^k)^T}{\sqrt{d_k}})\mathbf{Z}_h^{(l)}\mathbf{W}_h^v,\end{cases} \tag{3}$$

where $\text{Concat}(\cdot)$ and d_k denote the concatenation of node representation and a scaling parameter, respectively. $\mathbf{W}_h^q$, $\mathbf{W}_h^k$, and $\mathbf{W}_h^v$ are three learnable parameter matrices in the h-th head. Eq. (3) measures the similarities of all nodes via $\text{softmax}(\frac{(\mathbf{Z}_h^{(l)}\mathbf{W}_h^q)(\mathbf{Z}_h^{(l)}\mathbf{W}_i^k)^T}{\sqrt{d_k}})$ without using graph information, similar to Eq. (1).

The standard self-attention has the complexity of $O(n^2d)$, in the implementation, we adopt the efficient attention in (Shen et al. 2021) with the complexity of $O(d^2n)$. The formulation is listed as follows:

$$\text{head}_h = \phi_q(\mathbf{Z}_h^{(l)}\mathbf{W}_h^q)(\phi_k(\mathbf{Z}_h^{(l)}\mathbf{W}_h^k)^T\mathbf{Z}_h^{(l)}\mathbf{W}_h^v) \tag{4}$$

where ϕ_q and ϕ_k are two normalization functions. After updating the node representation by Eq. (4), as what Transformer do (Vaswani et al. 2017), we merge information across heads and layers to output the final representation:

$$\begin{cases}\mathbf{Z}^{(l+1)} = \mathbf{Z}^{(l)} + \mathbf{Z}^{(l+1)} \\ \mathbf{Z}^{(l+1)} = \mathbf{W}_2\text{ReLU}(\mathbf{W}_1\text{LN}(\mathbf{Z}^{(l+1)})) + \mathbf{Z}^{(l+1)},\end{cases} \tag{5}$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ denote two trainable parameter matrices, and $\text{LN}(\cdot)$ is the layer normalization operation.

Compared with previous methods, Eq. (3) has the following advantages. Firstly, self-attention integrates structure learning with weight learning to mitigate the influence of graph noise. Specifically, Eq. (3) reconstructs a fully-connected graph, which is independent of the primitive graph, to conduct structure learning (Dai et al. 2022) for addressing the issue of graph noise. Moreover, Eq. (3) learns the similarities between arbitrary node pairs, resulting in weight learning (Ye and Ji 2021). Secondly, Eq. (3) captures global information among nodes by a fully-connected graph. In contrast, previous methods modify the primitive graph to capture the local information among nodes.

Extensive studies have demonstrated that local and global information provide complementary information to each other (Zhu et al. 2017). However, self-attention conducts global graph learning for representation learning by ignoring local graph learning.

Graph Self-Improvement Module

In this part, local graph learning is used to capture the local information among nodes and graph self-improvement is designed to deal with graph noise.

Local Graph Learning We design a graph contrastive loss to capture the local graph information among nodes for learning the node representation, which is then used as the input of the multi-head self-attention module.

Given the feature matrix $\mathbf{X}$, we first use the multi-layer perceptron (MLP) with the parameter $\mathbf{W}$ to learn the new representation of $\mathbf{X}$, *i.e.,*

$$\mathbf{Z}^{(0)} = \text{Dropout}(\sigma(\mathbf{XW})) \tag{6}$$

We then design a graph contrastive loss to capture the local information among nodes for learning $\mathbf{Z}^{(0)}$, which preserves the consistency between the similarity matrix of $\mathbf{Z}^{(0)}$ and the multi-hop graph $\hat{\mathbf{G}}$, which will be defined later. To do this, considering that enlarging the receptive field for message-passing can empower nodes' expressiveness (Wang et al. 2021), we first construct a multi-hop graph and then design a graph contrastive loss. Given a binary adjacency matrix $\mathbf{A}$, we construct the multi-hop graph $\hat{\mathbf{G}}$ via:

$$\hat{g}_{ij} = \begin{cases}\hat{a}_{i,j}, \text{if node } j \text{ is the } r\text{-hop neighbor of node } i \\ 0, \text{if node } j \text{ is not the } r\text{-hop neighbor of node } i\end{cases} \tag{7}$$

where $\hat{a}_{i,j}$ is a weight normalized by the degree of the graph. We then employ the graph contrastive loss in (Hu et al. 2021) to capture the local information of the multi-hop graph $\hat{\mathbf{G}}$ for representation learning, *i.e.,*

$$\mathcal{L}_{\text{gsi}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{I}_i^c(-log\frac{\sum_{j=1}^{n}\mathbf{1}_{[j\neq i]}^g\hat{g}_{ij}\exp(\text{sim}(\mathbf{z}_i^{(0)},\mathbf{z}_j^{(0)})/\tau)}{\sum_{k=1}^{n}\mathbf{1}_{[k\neq i]}^g\exp(\text{sim}(\mathbf{z}_i^{(0)},\mathbf{z}_k^{(0)})/\tau)}) \tag{8}$$

where τ and $\text{sim}(\cdot)$ represent a temperature parameter and the similarity measurement, respectively. Additionally, the matrix $\mathbf{I}^g$ indicates the presence of edges in the multi-hop graph $\hat{\mathbf{G}}$, and the vector $\mathbf{I}^c$ indicates the nodes' reliability in terms of their connection, being used to conduct edge selection and node selection respectively. To minimize this graph contrastive loss, if there is an edge connecting node i with node j in the multi-hop graph $\hat{\mathbf{G}}$, $\mathbf{z}_i^{(0)}$ should be similar to $\mathbf{z}_j^{(0)}$, *i.e.,* maximizing the $\mathrm{sim}(\mathbf{z}_i^{(0)}, \mathbf{z}_j^{(0)})$. As a result, the local information of $\hat{\mathbf{G}}$ is passed to the node representation.

Graph Self-Improvement After conducting the propagation process several times, we assume the networks in the current epoch are better than in the former epochs. In this scenario, the pseudo graph constructed by the node representation in the current epoch should be better than the original multi-hop graph $\hat{\mathbf{G}}$ and the graph in the former epochs. In particular, the pseudo graph actually conducts edge prediction, where edges with high predictive confidence are kept and otherwise discarded. $\hat{\mathbf{G}}$ in Eq. (7) is updated by:

$$\hat{g}_{ij} = \begin{cases} 1, \text{ if } \mathrm{sim}(\mathbf{z}_i^{(0)}, \mathbf{z}_j^{(0)}) \geq \tau_{g1}, \\ \mathrm{sim}(\mathbf{z}_i^{(0)}, \mathbf{z}_j^{(0)}), \text{ if } \tau_{g2} \leq \mathrm{sim}(\mathbf{z}_i^{(0)}, \mathbf{z}_j^{(0)}) < \tau_{g1}, \\ 0, \text{otherwise}. \end{cases} \tag{9}$$

where τ_{g1} and τ_{g2} are two hyper-parameters. Correct node classification usually has reliable connectivities. It is difficult to find all potentially crucial edges in a noisy graph. This motivates us to preserve crucial nodes with high predictive probabilities in node classification by:

$$\mathbf{I}_i^c = \begin{cases} 1, \text{ if } z_{ij}^* \geq \tau_p, \\ 0, \text{ otherwise}. \end{cases} \tag{10}$$

where $\mathbf{Z}^*$ denotes the predictive probability matrix, and $\mathbf{I}^c$ is an indicator vector in Eq. (8), aiming at conducting node selection to preserve nodes with reliable neighbors.

Compared with previous methods in Eq. (1), the graph self-improvement module achieves the following advantages. Firstly, the pseudo graph automatically improves the quality of the primitive graph, tackling the issue of graph noise. Secondly, it and the multi-head self-attention module capture local and global graph information among nodes, effectively using their complementary information for representation learning.

Label Self-Improvement Module

In semi-supervised node classification, the pseudo label technique makes label information approximate to ground truth as well as obtaining more supervision information from unlabeled data. Such a process is formulated by:

$$y_{ij} = \begin{cases} 1, \text{ if } z_{ij}^* \geq \tau_{p1}, \\ z_{ij}^*, \text{ if } \tau_{p2} \leq z_{ij}^* < \tau_{p1}, \\ 0, \text{ else}. \end{cases} \tag{11}$$

where τ_{p1} and τ_{p2} are two adjustable hyper-parameters. If the predicted probability of node i exceeds a set threshold, it is deemed a reliable candidate. Consequently, the associated predicted probability is deemed trustworthy and serves as pseudo labels, acting as truthful labels in the subsequent training. This approach effectively addresses challenges posed by the issues of label noise and sparse labeling of node classification.

Algorithm 1: The pseudo-code of our RNCGLN method.

```
Input: the feature matrix $\mathbf{X}$, the noisy graph $\mathbf{A}$, the noisy
    labels $\mathbf{Y}$, and the hyper-parameter $\alpha$;
Output: predicted labels $\mathbf{Z}^*$, loss $\mathcal{L}$, and the well trained
    RNCGLN;
 1: while epochs do
 2:   obtain $\mathbf{Z}^{(0)}$ with MLP on $\mathbf{X}$;
 3:   update $\mathcal{L}_{\mathrm{gsi}}$ based on $\hat{\mathbf{G}}$ and $\mathbf{Z}$ via Eq. (8);
 4:   update $\mathbf{Z}$ with self-attention layers on $\mathbf{Z}^{(0)}$;
 5:   $\mathbf{Z}^* = \mathrm{softmax}(\mathrm{MLP}(\mathbf{Z}))$;
 6:   update $\mathcal{L}_{\mathrm{lsi}}$ based on $\mathbf{Y}$ and $\mathbf{Z}^*$ via Eq. (12);
 7:   calculate $\mathcal{L} = \mathcal{L}_{\mathrm{lsi}} + \alpha\mathcal{L}_{\mathrm{gsi}}$ via Eq. (13);
 8:   back-propagate $\mathcal{L}$;
 9:   if epochs > warm-up then
10:     update labels via Eq. (11);
11:     select nodes with reliable neighbors via Eq. (10);
12:     update the graph via Eq. (9);
13:   end if
14: end while
```

We select the pseudo-labeling technique for the following considerations. Motivated by the fact that models tend to fit clean labels and subsequently overfit noisy labels (Van Engelen and Hoos 2020), our method leverages self-training and pseudo labels to facilitate a self-improvement process within an end-to-end learning framework.

Loss Function

Our pseudo label method belongs to self-training (Lee et al. 2013). After a warm-up, *i.e.,* some epochs, confident pseudo labels (via the label self-improvement module) and pseudo graphs (via the graph self-improvement module) are updated to replace the previous ones, which are then used in the next epoch. Therefore, our method uses the cross-entropy loss for node classification via:

$$\mathcal{L}_{\mathrm{lsi}} = -\sum\nolimits_{i \in \mathbf{Y}_L} \sum\nolimits_{j=1}^{C} y_{ij} \log z_{ij}^* \tag{12}$$

where C denotes the number of labels and y_{ij} is the initial label information or the pseudo labels after conducting the label self-improvement. To supplement label information, we also use Eq. (8) to update networks, where $\hat{g}_{ij}$ is the initial graph and then is the pseudo graph from Eq. (9) after conducting the graph self-improvement. Therefore, our final loss function is formulated as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{lsi}} + \alpha\mathcal{L}_{\mathrm{gsi}}, \tag{13}$$

where α is a trade-off hyper-parameter to balance two terms of the loss function. Our method focuses more on classification effectiveness under graph noise (*i.e.,* with a small α). Reliable graph information results in correctly predicting labels under label noise (*i.e.,* with a high α). Consequently, labels and the graph can complement each other, mitigating the supervision scarcity caused by noise. Using two self-improvement modules, the self-attention module converges to self-consistency, ensuring similar neighbors share labels and dissimilar nodes differ in labels. The pseudo-code of our method is listed in Algorithm 1.

Datasets	$\{gr, lr\}$	GCN	GAT	JLGCN	IDGL	NRGNN	LAFAK	RNCGLN
Cora	$\{0,0\}$	82.72 ±0.78	83.20±0.31	83.83±0.16	83.85±0.25	82.30±0.22	81.58±0.92	**84.46±0.35**
	$\{.1,.3\}$	70.58±1.21	74.48±1.00	69.01±0.30	76.83±2.57	70.90±3.88	79.34±0.54	**81.73±0.33**
	$\{.3,.1\}$	72.94±1.58	73.70±1.29	72.40±1.70	80.53±1.70	72.43±1.98	74.34±1.34	**80.60±1.06**
Citeseer	$\{0,0\}$	71.18±1.06	71.76±0.58	73.60±0.28	72.10±0.20	70.78±0.39	71.08±0.34	**73.96±0.30**
	$\{.1,.3\}$	61.70±0.89	64.28±0.76	64.65±3.79	68.03±3.65	66.88±2.31	69.14±1.28	**71.83±0.33**
	$\{.3,.1\}$	65.30±1.21	67.74±1.24	63.21±1.54	69.71±1.3	64.46±3.23	66.42±1.04	**70.20±1.41**
Pubmed	$\{0,0\}$	79.04±0.30	79.16±0.28	79.50±0.29	80.80±0.39	79.58± 0.41	78.28±0.46	**81.16±0.64**
	$\{.1,.3\}$	68.60±1.63	67.88±1.72	67.81±2.81	77.12±1.61	75.67±2.05	76.74±0.39	**77.60±1.20**
	$\{.3,.1\}$	70.76±1.91	71.74±1.54	75.61±1.54	74.61±1.66	71.05±2.84	73.61±1.63	**79.23±1.03**
Photo	$\{0,0\}$	90.84±0.70	91.15±0.35	91.98±0.88	91.00±0.51	90.21±0.23	91.10±0.66	**92.21±0.54**
	$\{.1,.3\}$	82.18±0.69	83.73±0.51	85.57±3.10	75.15±2.00	81.54±2.27	89.01±0.79	**90.27±0.39**
	$\{.3,.1\}$	81.86±0.73	80.05±0.68	85.83±0.34	84.65±1.79	82.54±2.26	85.60±0.81	**90.72±0.08**

Table 1: The results of node classification (ACC ± STD) under different ratios of graph noise (*gr*) and label noise (*lr*).

Complexity Analysis

The multi-head self-attention module is the main complexity of our method. We employ efficient attention (Shen et al. 2021) to have linear complexity, *i.e.,* $O(nd^2)$ for computation cost and $O(d^2 + dn)$ for memory cost. Besides, Eq. (6) and Eq. (5) need $O(nd^2)$ and $O(d^2 + dn)$, respectively. Hence, the overall complexity of our method is $O(nd^2)$ for computation cost and $O(d^2 + dn)$ for memory cost.

Experiments

Datasets and Comparison Methods

We evaluate the robustness of our proposed method[1] on four popular datasets, including three citation datasets (*i.e.,* Cora, Citedeer, Pubmed) (Sen et al. 2008) and one amazon sale dataset (*i.e.,* Photo) (Shchur et al. 2018).

The comparison methods include two conventional GNN methods that consider neither graph noise nor label noise (*i.e.,* GCN (Kipf and Welling 2016) and GAT (Veličković et al. 2017)), two graph-noise robust methods (*i.e.,* JLGCN (Tang et al. 2019) and IDGL (Chen, Wu, and Zaki 2021)); two label-noise robust methods (*i.e.,* NRGNN (Dai, Aggarwal, and Wang 2021) and LAFAK (Zhang et al. 2020)).

Result Analysis of Node Classification With Noise

We compare our method with all comparison methods regarding node classification under two kinds of noise on all four datasets. We take two situations into account in terms of different ratios of graph noise (*gr* for short) and label noise (*lr* for short), *i.e.,* $\{gr, lr\} = \{0.3, 0.1\}$ and $\{gr, lr\} = \{0.1, 0.3\}$. For an intuitive comparison, we also list the results when the graph and the labels are both clean, *i.e.,* $\{gr, lr\} = \{0, 0\}$. All results are reported in Table 1.

First, our method achieves the best performance in node classification under different ratios, followed by KAFAK, IDGL, NRGNN, JLGCN, GAT, and GCN. Compared with the best comparison method (*i.e.,* KAFAK), our method, on average, improves by 3.50% on all four datasets. Compared with the best baseline method, no considering noise issues (*i.e.,* GAT), our method shows a significant improvement of 7.32% on all datasets. The reason is that our method considers both graph noise and label noise, while comparison methods consider either or neither.

Second, our method can produce better performance independently, focusing on one kind of noise. For example, compared with the best label-noise robust method (*i.e.,* KAFAK) under $\{gr, lr\} = \{0.1, 0.3\}$, our method improves by 1.80% on all datasets. Compared with the best graph-noise robust method (*i.e.,* IDGL) under $\{gr, lr\} = \{0.3, 0.1\}$, our method improves by 2.81% on all datasets. This validates the necessity of taking into account both graph noise and label noise for robust node classification. By contrast, previous robust methods perform unstably when processing noise beyond their focus. For example, on dataset Photo, IDGL robust to graph noise performs better under $\{gr, lr\} = \{0.3, 0.1\}$ while worse under $\{gr, lr\} = \{0.1, 0.3\}$, compared with the GCN without considering any kinds of noise. This is because these methods need to rely heavily on label information to purify the graph structure. When label noise exists, the errors from noisy labels could severely impact the models. Differently, our method takes two kinds of noises into account. When there is severe graph noise, our model depends more on label information and vice versa. This is why our method consistently outperforms previous robust models under different scenarios.

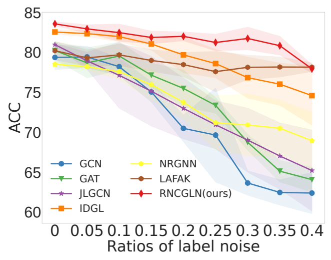
Third, our proposed method shows more powerful expressiveness than previous GNNs. Compared with two baseline methods (*i.e.,* GCN and GAT) no concerning noise issues, our method improves by 2.00% and 1.63%, respectively, on clean data, *i.e.,* $\{gr, lr\} = \{0, 0\}$, and improves by 8.53% and 7.32%, respectively, on noisy data, *i.e.,* $\{gr, lr\} \neq \{0, 0\}$. This indicates our method has stronger robustness and also a more powerful expressiveness. We attribute superiority to conducting both local and global graph learning.

Result Analysis of Noise Robustness

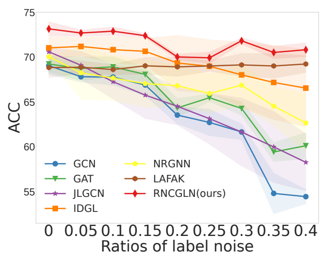
We investigate the noise effects of all methods by exploring their sensitivity under different noise ratios. To do this, we fix one noise as 0.1 and change the ratio of another noise to $[0, 0.05, 0.1, ..., 0.4]$. All results are listed in Figure 2.

On the upper four figures considering the variation of la-

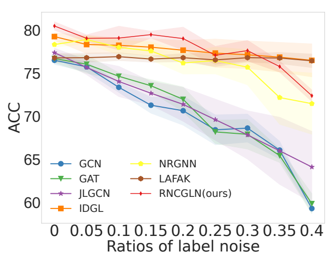
[1]Our code and comprehensive theoretical version are available at: https://github.com/yhzhu66/RNCGLN



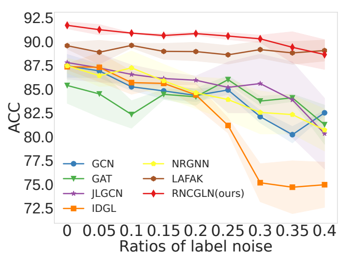
(a) Cora



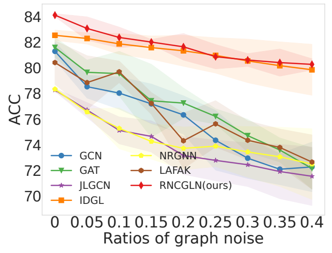
(b) Citeseer



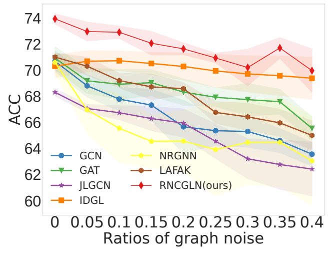
(c) Pubmed



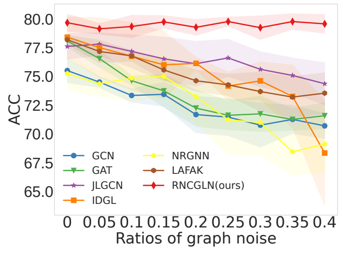
(d) Photo



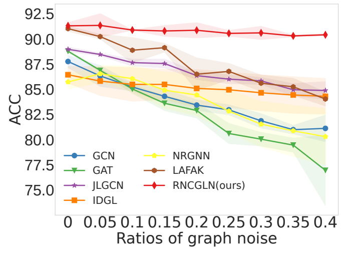
(e) Cora



(f) Citeseer



(g) Pubmed



(h) Photo

Figure 2: The sensitivities of all methods to label noise and graph noise under different ratios, where the upper four plots show sensitivities in terms of label noise and the lower four plots show sensitivities in terms of graph noise.

bel noise, all methods show a decreasing trend with the increase of ratios of label noise, but our method consistently outperforms comparison methods. For example, on dataset Photo under varying label noise (*i.e.,* Figure (2-d)), our method on average improves by 1.40%, compared with the best comparison method (*i.e.,* LAFAK) on different ratios of label noise. Besides, two methods (*i.e.,* RNCGLN (ours) and LAFAK) both leverage a graph-related loss to supplement label information and thus output stable performance. This implies that using graph contrastive loss is effective in handling label noise via supplementing more supervision information. Differently, our method leverages the pseudo graph and the pseudo label to achieve better performance by improving the quality of two kinds of supervision information.

On the lower four figures considering the variation of graph noise, all methods show a decreasing trend with the increase of ratios of graph noise, but our method consistently outperforms comparison methods and shows more stable performance. For example, on dataset Pubmed under varying graph noise (*i.e.,* Figure (2-g)), the performance of our method shows very slight fluctuation in different ratios, while the best comparison method (*i.e.,* JLGCN) reports a decrease of 3.24% from 0 to 0.4 in terms of graph noise. Moreover, our method, on average, improves by 3.17% on different ratios. Besides using the pseudo graph, it can also attribute the superiority to using a fully connected weighted graph of self-attention, which can simultaneously relieve the issues of the missing-edges and the noisy-edges.

Overall, our method could output stable performance under different scenarios for both graph noise and label noise. This is because graph information and label information can complement each other in our method, while others rely highly on one kind of information to purify another. Therefore, previous robust GNNs may not consistently output satisfactory performance when other noise is present. For example, JLGCN on dataset Cora under $\{gr, lr\} = \{0.3, 0.1\}$ (*i.e.,* Figure 2-e), and NRGNN on dataset Photo under $\{gr, lr\} = \{0.1, 0.3\}$ (*i.e.,* Figure 2-d), drop under the baseline methods considering any noise. This indicates when there is another kind of noise (*e.g.,* label noise to JLGCN and graph noise to NRGNN.), the previous robust GNNs become vulnerable.

Ablation Study

To evaluate the effectiveness of our proposed method, we conduct the ablation study from two aspects, *i.e.,* the key components and the pseudo techniques. In the key components, we explore the necessity of graph contrastive loss, *i.e.,* Eq. (8) for local graph learning, Gloss for short, and node selection, *i.e.,* Eq. (10) for selecting nodes with reliable neighbors, NS for short. In the pseudo techniques, we evaluate the influence of the pseudo graph and pseudo label. All results are reported in Table 2.

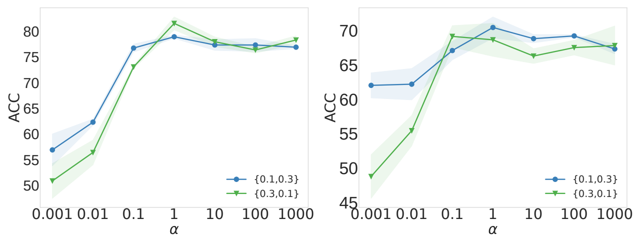
From the upper table, the performance of our model dramatically declines and slightly declines, respectively, disregarding the Gloss and the NS. For example, our standard model decreases by 19.95% and 0.53% than the two ablation models on different ratios. Graph contrastive loss is essential to our method to capture local information from the primitive graph; otherwise, self-attention cannot accurately capture global information. Moreover, node selection can filter out nodes with error neighbors by using predictive confidence to construct the pseudo graph.

From the lower table, the pseudo graph and the pseudo label can both improve the performance of node classification under two kinds of noise. Our model improves by

Component		Cora		Citeseer		Pubmed		Photo	
Gloss	NS	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$
	✓	53.93±0.66	56.80±3.34	41.73±2.04	60.33±1.19	60.26±1.21	69.63±1.76	60.29±1.60	79.77±0.90
✓		81.56±1.56	79.80±0.48	70.93±1.43	71.40±0.21	76.80±1.75	**79.66±0.90**	88.10±0.59	89.49±1.25
✓	✓	**81.73±1.42**	**80.60±0.50**	**71.73±0.33**	**71.53±0.17**	**77.60±1.20**	79.23±1.03	**89.38±2.34**	**90.59±0.66**

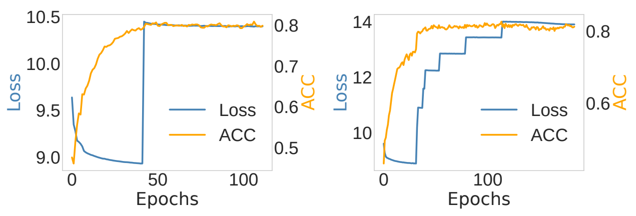
Pseudo		Cora		Citeseer		Pubmed		Photo	
Graph	Label	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$	$\{0.1, 0.3\}$	$\{0.3, 0.1\}$
		75.90±1.01	79.53±0.41	65.30±1.33	67.93±0.49	76.36±0.67	77.93±0.87	86.25±0.36	87.56±0.59
✓		78.36±0.60	79.20±0.45	65.50±1.14	70.50±0.08	76.50±0.58	78.10±0.43	86.93±0.29	87.59±0.55
	✓	80.80±0.90	79.93±0.52	68.46±1.85	70.83±0.26	77.30±1.80	78.90±0.43	87.71±0.86	88.57±1.35
✓	✓	**81.73±1.42**	**80.60±0.50**	**71.73±0.33**	**71.53±0.17**	**77.60±1.20**	**79.23±1.03**	**89.38±2.34**	**90.59±0.66**

Table 2: Ablation study. $\{\cdot, \cdot\}$ stands for $\{gr, lr\}$ in terms of key components and the use of pseudo graph and pseudo label.



(a) Cora (b) Citeseer

Figure 3: The sensitivity of the hyper-parameter α in Eq. (13), where the legend is the $\{gr, lr\}$.



(a) $\{gr, lr\} = \{0.3, 0.1\}$ (b) $\{gr, lr\} = \{0.1, 0.3\}$

Figure 4: Convergence on the datasets Cora under the joint graph and label noise.

1.18%, 2.41%, and 3.20%, respectively, compared with the models using pseudo labels only, using the pseudo graph only, and using neither. This indicates that our proposed self-improvement modules effectively improve the quality of supervision information, via not only correcting error information but also supplementing more supervision information from unlabeled nodes and missing edges.

Parameter Sensitivity Analysis

Our method has only one hyper-parameter in the loss function, *i.e.,* α. Figure 3 shows how performance varies with different α within the ranges of $\{10^{-3}, 10^{-2}, ..., 10^{3}\}$.

Based on the results, our methods are sensitive to the setting of α under two scenarios *i.e.,* $\{gr, lr\} = \{0.3, 0.1\}$ and $\{gr, lr\} = \{0.1, 0.3\}$. Both lines first dramatically increase to a peak and then slightly decline. When α is small, the local graph information cannot be fully captured and thus the model performs worse. After which, the values of α can balance two kinds of supervision information in terms of the graph and labels, based on their individual quality. Therefore, by tuning the hyper-parameter α, our method can flexibly handle two kinds of noise.

Convergence

Figure 4 reports the convergence of our method, including both variations in terms of the ACCs (*i.e.,* accuracies) and losses with epochs.

All losses first smoothly converge and continue converging after some jumps, which differs from the ACCs that keep increasing to peaks before keeping stable. A representative example is shown in Figure 4-(a), where the loss decreases from 9.5 to 9.0 before suddenly jumping to 10.5, after which it keeps decreasing until stopping. Actually, before ending with the warm-up, our model continually fits the clean data, after which the pseudo graph and the pseudo label correct noise information and bring in more supervision information from unlabeled nodes and missing edges, such that losses suddenly increase. Two processes, *i.e.,* fitting clean data and purifying supervision information, are iteratively updated until the model achieves convergence when similar nodes share the same label.

Conclusion

In this paper, we proposed a new method designed for robust node classification with graph noise and label noise. Our method simultaneously conducts local and global graph learning through graph contrastive learning and the self-attention mechanisms to enhance the discriminative capacity of node representation. We also employ two self-improvement modules to make graph information and label information provide complementary information to each other. Extensive experiments demonstrated that our method outperformed all comparison methods in node classification.

Acknowledgments

This work was supported by a grant from the New Zealand Tertiary Education Commission and by the Strong AI Lab at the University of Auckland. The first author is supported by a PhD scholarship from the China Scholarship Council.

References

Balcilar, M.; Guillaume, R.; Héroux, P.; Gaüzère, B.; Adam, S.; and Honeine, P. 2021. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*.

Bojchevski, A.; and Günnemann, S. 2019. Certifiable Robustness to Graph Perturbations. In *NeurIPS*.

Chen, Y.; Wu, L.; and Zaki, M. 2021. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *NeurIPS*, 19314–19326.

Cheng, X.; Miao, Z.; and Qiu, Q. 2020. Graph convolution with low-rank learnable local filters. *arXiv preprint arXiv:2008.01818*.

Dai, E.; Aggarwal, C.; and Wang, S. 2021. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *SIGKDD*, 227–236.

Dai, E.; Jin, W.; Liu, H.; and Wang, S. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*, 181–191.

Fox, J.; and Rajamanickam, S. 2019. How robust are graph neural networks to structural noise? *arXiv preprint arXiv:1912.10206*.

Hu, Y.; You, H.; Wang, Z.; Wang, Z.; Zhou, E.; and Gao, Y. 2021. Graph-MLP: node classification without message passing in graph. *arXiv preprint arXiv:2106.04051*.

Huang, J.; Du, L.; Chen, X.; Fu, Q.; Han, S.; and Zhang, D. 2023. Robust Mid-Pass Filtering Graph Convolutional Networks. In *WWW*, 328–338.

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

Lee, D.-H.; et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop on Challenges in Representation Learning*, 896.

Li, Y.; Yin, J.; and Chen, L. 2021. Unified robust training for graph neural networks against label noise. In *PAKDD*, 528–540.

Min, E.; Chen, R.; Bian, Y.; Xu, T.; Zhao, K.; Huang, W.; Zhao, P.; Huang, J.; Ananiadou, S.; and Rong, Y. 2022. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*.

Runwal, B.; Kumar, S.; et al. 2022. Robust Graph Neural Networks using Weighted Graph Laplacian. *arXiv preprint arXiv:2208.01853*.

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

Shchur, O.; Mumme, M.; Bojchevski, A.; and Günnemann, S. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*.

Shen, Z.; Zhang, M.; Zhao, H.; Yi, S.; and Li, H. 2021. Efficient attention: Attention with linear complexities. In *CVPR*, 3531–3539.

Song, H.; Kim, M.; and Lee, J.-G. 2019. Selfie: Refurbishing unclean samples for robust deep learning. In *ICML*, 5907–5915.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958.

Sukhbaatar, S.; Bruna, J.; Paluri, M.; Bourdev, L.; and Fergus, R. 2014. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Tang, J.; Hu, W.; Gao, X.; and Guo, Z. 2019. Joint learning of graph representation and node features in graph convolutional neural networks. *arXiv preprint arXiv:1909.04931*.

Van Engelen, J. E.; and Hoos, H. H. 2020. A survey on semi-supervised learning. *Machine learning*, 109(2): 373–440.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, G.; Ying, R.; Huang, J.; and Leskovec, J. 2021. Multi-hop Attention Graph Neural Networks. In *IJCAI*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24.

Ye, Y.; and Ji, S. 2021. Sparse graph attention networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(1): 905–916.

Zhang, C.; Bengio, S.; Hardt, M.; Recht, B.; and Vinyals, O. 2021. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115.

Zhang, M.; Hu, L.; Shi, C.; and Wang, X. 2020. Adversarial label-flipping attack and defense for graph neural networks. In *ICDM*, 791–800.

Zhong, J.-X.; Li, N.; Kong, W.; Liu, S.; Li, T. H.; and Li, G. 2019. Graph convolutional label noise cleaner: Train a plug-and-play action classifier for anomaly detection. In *CVPR*, 1237–1246.

Zhu, X.; Zhang, S.; Hu, R.; Zhu, Y.; et al. 2017. Local and global structure preservation for robust unsupervised spectral feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 30(3): 517–529.