Noisy Node Classification by Bi-level Optimization Based Multi-Teacher Distillation

Yujing Liu[1,2], Zongqian Wu[3], Zhengyu Lu[1,2,4] Ci Nie[1,2], Guoqiu Wen[1,2], Yonghua Zhu[1,2,5,*], Xiaofeng Zhu[1,2,3]

[1]School of Computer Science and Engineering, Guangxi Normal University, Guilin, China
[2]Guangxi Key Lab of Multisource Information Mining and Security, Guangxi Normal University, Guilin, China
[3]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[4]School of Computer Science and Information Engineering, Anyang Institute of Technology, Anyang, China
[5]Information Systems Technology Design Pillar, Singapore University of Technology and Design, Singapore
liuyujingcn@gmail.com, yhzhu9423@gmail.com

Abstract

Previous graph neural networks (GNNs) usually assume that the graph data is with clean labels for representation learning, but it is not true in real applications. In this paper, we propose a new multi-teacher distillation method based on bi-level optimization (namely BO-NNC), to conduct noisy node classification on the graph data. Specifically, we first employ multiple self-supervised learning methods to train diverse teacher models, and then aggregate their predictions through a teacher weight matrix. Furthermore, we design a new bi-level optimization strategy to dynamically adjust the teacher weight matrix based on the training progress of the student model. Finally, we design a label improvement module to improve the label quality. Extensive experimental results on real datasets show that our method achieves the best results compared to state-of-the-art methods.

Introduction

Node classification usually relies on correct supervised information to predict unlabelled nodes on the graph data. However, real-world graph data often contain incorrect supervised information, *i.e.,* noisy labels, which easily misleads the training model and in turn degrades the node classification performance. To address this issue, noisy node classification (NNC) has been widely proposed to train graph neural networks (GNNs) on the graph data with noisy labels (Li, Yin, and Chen 2021), so NNC has garnered growing interest in practical applications.

A number of methods have been proposed to handle noisy labels. For example, NRGNN (Dai, Aggarwal, and Wang 2021) trains a pseudo-label miner to select pseudo-labels for augmenting the supervision information. UnionNET (Li, Yin, and Chen 2021) first trains a GNN using the original labels and then performs label correction based on the node embeddings learned by the obtained GNN. However, these methods excessively depend on the performance of individual model and struggle to handle datasets with high noise rates. For instance, in the case of high noise rates, NRGNN will select incorrect pseudo-labels and UnionNET will contaminate correct labels with label correction. One of the key reasons is that previous methods design a single model only to handle with noisy label to result in error accumulation.

To alleviate the above issue, the multi-teacher distillation method is proposed to transfer knowledge from multiple models to deal with noisy labels. For instance, MTS-GNN (Liu et al. 2023a) uses models saved from earlier iterations to guide model training and label correction for later iterations. However, previous multi-teacher distillation methods still have limitations to be addressed. First, the constructed multi-teacher models across nearby iterations have few differences, and thus resulting in the lack of diversity. Second, previous methods do not consider the complementary information among teacher models and could not fully exploit the knowledge of teacher models.

Addressing the above issues of multi-teacher distillation methods is challenging. First, since the labelled data is sparse and with noise, it is difficult to train diverse teacher models by existing methods (*e.g.,* randomly sampling training subsets from the training set). Second, previous methods usually require a large number of correct labels to learn the weights of the teacher models, which is not practical for NNC tasks. For example, MTS-GNN sets the weights of teacher models as hyper-parameters, resulting in expensive time cost and difficult to reasonably set the weights.

In this paper, we propose a new multi-teacher distillation method based on bi-level optimization (namely BO-NNC, shown in Figure 1) to address the above issues. To do this, we first employ diverse self-supervised graph methods (*e.g.,* (Veličković et al. 2018; Zhu et al. 2021; Mo et al. 2022)) to construct multiple teacher models, solving the first issue of previous methods, and then construct a teacher weight matrix to integrate the predictions of the teacher models as the soft label matrix, which is regarded as the ground truth of the student model. Moreover, we propose a new bi-level optimization strategy to iteratively train the teacher weight matrix and the student model. This allows the teacher weight matrix to be dynamically adjusted based on the training progress of the student model without requiring a large number of correct labels to learn the weights of the teacher models. In particular, the proposed bi-level optimization algorithm helps to learn the complementary information among teacher models. After the bi-level optimization, we further

*Corresponding author.

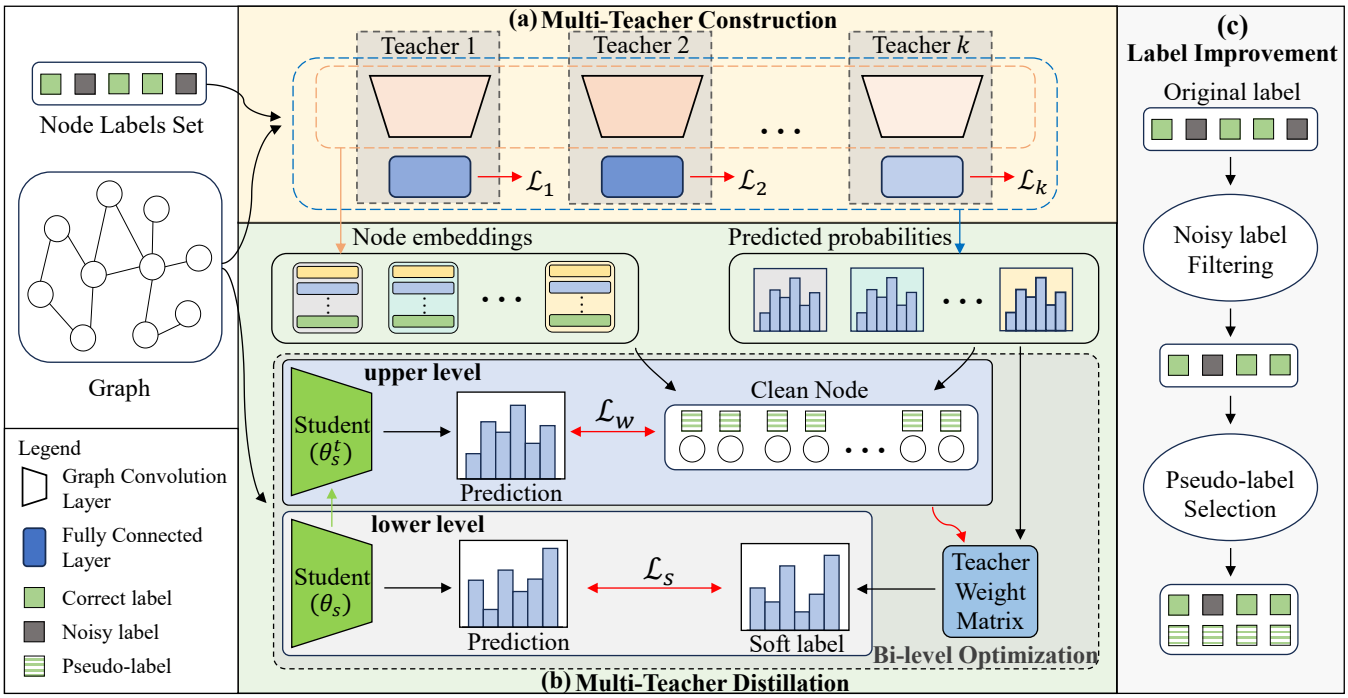


Figure 1: The framework of the proposed BO-NNC, consisting of three modules. **Multi-teacher Construction** employs multiple self-supervised learning methods to obtain diverse teacher models. **Multi-teacher Distillation** transfers the knowledge from the teacher models to the student model through multi-teacher distillation based on bi-level optimization. **Label Improvement** uses both the student model and the teacher models to first detect noisy labels and then select pseudo-labels.

design a label improvement module to improve the label quality of the dataset, which further enhances the learning of multiple teacher models and the student model.

Compared with previous NNC methods, the main contributions of our proposed method are summarized as follows:

- We propose a new method to deal with noisy node classification, which solves the limitations of previous methods by multi-teacher distillations, and designs a label improvement module to gradually improve the quality of labels.
- We design a new distillation method based on bi-level optimization, which automatically adjust the teacher weight matrix, thus more fully exploring the complementary information among multiple teacher models.
- We conduct extensive experiments on five real datasets. Experimental results demonstrate that our method outperforms existing SOTA methods.

Related Work

Noisy Node Classification

Recently, graph data has been widely used for its ability of expressing complex relationships and structures in the real world, *e.g.,* ecological networks (Windsor et al. 2023), traffic networks (Guo et al. 2019), and social networks (Dai and Wang 2021). Simultaneously, Graph Neural Networks (GNNs) have attracted more and more attention due to their powerful ability in process and analyze graph data (Wei et al. 2024; Fu et al. 2023). The success of GNNs is that it is able to leverage topological information between nodes through neighborhood aggregation. For example, GCN (Kipf and Welling 2016) implements aggregation and propagation of node features by approximating first-order Chebyshev polynomials. GAT (Veličković et al. 2017) introduces self-attention mechanism to enable different weights to different neighboring nodes.

However, existing researches (Arpit et al. 2017; Zhang et al. 2021) has demonstrated that deep neural networks are prone to overfitting noisy labels, leading to poor generalization. Recently, some methods have been proposed to deal with the problem of noisy labels in graph data. For example, NRGNN (Dai, Aggarwal, and Wang 2021) links unlabelled nodes with labelled nodes that have high feature similarity and trains a pseudo-label miner to select pseudo-labels to increase supervisory information. MTS-GNN (Liu et al. 2023a) mitigates the impact of noisy labels by using models saved in previous iterations to guide model training and label correction in subsequent iterations. However, existing methods are difficult to deal with the case of high noise rates.

Bi-Level Optimization

Bi-level Optimization (Liu et al. 2021a) initially originated in the domains of economics and game theory, and has subsequently been widely used in several scientific and engineering fields. The core idea of bi-level optimization is to solve two levels of optimization problems in a nested manner: the objective function of the upper-level optimization

problem depends on the optimal solution of the lower-level problem. Therefore, optimizing the upper-level problem requires repeatedly solving the lower-level problem. The advantage of this approach is that its ability to capture interdependencies in multi-level decision-making processes more accurately, leading to better optimization results. In this paper, we design a bi-level optimization approach to solve for both the teacher weight matrix and the student model parameters, enabling the student model to more effectively integrate the knowledge from different teacher models.

Methodology

Denoting $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as a graph, where $\mathcal{V} = \{v_1, \cdots, v_n\}$ represents the node set with n nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represents the edge set, we denote $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \in \mathbb{R}^{n \times d}$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$, respectively, as the feature matrix of all nodes and the adjacency matrix of the graph $\mathcal{G}$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector of the i-th node. The labels in the datasets are denoted as $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_b\}$, where b is the number of labelled nodes, $\mathbf{y}_i \in \mathbb{R}^c$ is one-hot encoding of the i-th label and c is the number of classes.

Multi-Teacher Construction

Existing studies (Wu et al. 2022) demonstrate that different representation learning methods usually output diverse node representations. Motivated by such observation, we employ existing graph representation learning algorithms to construct multiple teacher models.

Specifically, we first train k encoders by k different unsupervised graph representation learning methods and then use them to obtain node embeddings:

$$\mathbf{H}_i = \mathbf{F}_i(\mathbf{X}, \mathbf{A}), \tag{1}$$

where $i \in [1, ..., k]$ and $\mathbf{F}_i$ denotes the i-th encoder. We then use individual embedding as inputs to train k classifiers by the loss function as follows:

$$\mathcal{L}_i = -\sum_{j \in \mathcal{S}} \mathbf{y}_j \log \mathbf{p}_j^i, \tag{2}$$

where $\mathcal{S}$ denotes the labelled training set, $\mathbf{p}_j^i \in \mathbb{R}^c$ denotes the prediction of the i-th classifier for node v_j, and $\mathbf{y}_j$ denotes the label of node v_j. In the end, we obtain k teacher models consisting of k encoders and k classifiers to achieve model diversity.

The literature (*e.g.,* (Hinton, Vinyals, and Dean 2015; Kim et al. 2021)) demonstrate that the knowledge in the teacher model can be transferred to the student model by making the output of the student model mimic the output of the teacher model. Motivated by this, we consider using the predictions of the teacher models as soft labels to train the student model. This may enable the student model to explore the complementary information among teacher models, and thus addressing the issue of the excessive dependence on the performance of a single model in previous methods.

Multi-Teacher Distillation

After constructing multiple teacher models on labelled nodes, in this section, diverse knowledge will be transferred to the student model. To do this, we first employ multiple teachers to generate the node embeddings $\mathbf{H} = \{\mathbf{H}_1, ..., \mathbf{H}_k\}$ and prediction probability matrices $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_k\}$ for all nodes (including labelled nodes and unlabelled nodes), which are then used to guide the student model.

Second, the student model are generated by randomly initialized parameters on all nodes. That is, the node embedding of all nodes $\mathbf{Z}^{(i)}$ in the i-th layer of the student model and the probability matrix $\mathbf{P}^S$ of the student model are obtained by:

$$\begin{cases} \mathbf{Z}^{(i)} = \sigma(\hat{\mathbf{A}}\mathbf{Z}^{(i-1)}\mathbf{W}^{(i-1)}), \\ \mathbf{P}^S = \mathrm{softmax}(\sigma(\hat{\mathbf{A}}\mathbf{Z}^{(i)}\mathbf{W}^{(i)})), \end{cases} \tag{3}$$

where $\mathbf{W}^{(i)}$ is the trainable parameters of the i-th layer and σ is the activation function.

After that, we fuse k prediction probability matrices to a probability matrix, *i.e.,* soft label matrix, which is the ground truth of the student model. The simple strategy for mapping k prediction probability matrices to the soft label matrix is the mean method. However, such a method equivalently treats every teacher model. A good alternative is co-attention method (Gao, Parcollet, and Lane 2021). Specifically, we learn the weight matrix $\mathcal{W} \in \mathbb{R}^{k \times c}$ (c denotes the number of classes) to map k prediction probability matrices to the soft label matrix. However, previous co-attention methods usually require a large number of correct labels to learn the weights of the teacher models, which is not feasible in NNC tasks.

To address the above issue, we investigate clean nodes to effectively learn the weight matrix, resulting in a bi-level optimization. Specifically, after t iterations of the student model, we calculate the loss among clean nodes to update the teacher weight matrix by the back-propagation (Details in Section Bi-level Optimization). As a result, information obtained from the student model can be used to improve the teacher weight matrix. The updated teacher weight matrix guarantees the effectiveness of soft labels even with low-quality teacher models due to their limited number of labels.

In our method, clean nodes come from unlabelled nodes and labelled nodes. We select clean nodes from unlabelled nodes by first considering the prediction results of all teacher models and then filtering the selected clean nodes by node embedding similarity. Specifically, we first chose unlabelled nodes with consistent prediction among all teacher models to form the candidate set Can:

$$Can = \{v_i \mid \forall j \in (1, k), \arg\max \mathbf{p}_i^j = \arg\max \mathbf{p}_i^k\}, \tag{4}$$

where $\mathbf{p}_i^j$ denotes the prediction of the j-th teacher model for node v_i. We then filter clean nodes in Can based on the node embedding similarity between every node in Can and its β_1 (β_1 is a hyper-parameter) closest nodes in the same class, *i.e.,*

$$\xi_i = \sum_{j=1}^{k} \sum_{v_u \in C_i} d(h_i^j, h_u^j), \tag{5}$$

where C_i denotes the set consisting of β_1 closest nodes of the same class to node v_i, h_i^j denotes the embedding vector

of the j-th teacher for node v_i, and $d(\cdot,\cdot)$ denotes Euclidean distance. Finally, we select β_2 (β_2 is a hyper-parameter) nodes with the smallest ξ (*i.e.,* with the highest confidence) in each class as the first part of clean nodes.

In order to increase the number of clean nodes, we consider selecting the second part of clean nodes from labelled nodes, based on the loss values of the teacher models. Specifically, we first use the teacher models to calculate loss values for all labelled nodes:

$$l_i^T = -\sum_{j=1}^{k} \mathbf{y}_i \log \mathbf{p}_i^j, \tag{6}$$

where l_i^T denotes the loss value of node v_i. After that, we select $\alpha\%$ (α is a hyper-parameter) nodes with the lowest loss values (*i.e.,* with the highest confidence) in each class to form the second part of clean nodes.

Finally, we combine two parts of clean nodes (*i.e.,* with the highest confidence out of all nodes) to obtain the clean node set Cle, which is used to calculate the loss of the upper level, in the bi-level optimization. Moreover, the loss in the upper level is then used to update the teacher weight matrix. In the lower level, the teacher weight matrix is used to generate the soft label matrix, which is further used to update the parameters of the student model. In particular, the bi-level optimization strategy makes the teacher weight matrix be dynamically adjusted based on the feedback from the student model. This allows the updated teacher weight matrix to effectively fuse multiple prediction matrices, *i.e.,* extracting their common information across all models and complementary information in individual models.

Label Improvement

After conducting both multi-teacher construction and multi-teacher distillation, we consider gradually improve the label quality of the dataset through two stages of noise label filtering and pseudo-label selection during training.

Noisy Label Filtering Noisy label filtering is designed to filter a portion of noisy labels from the labelled data, thereby reducing incorrect supervision information in the dataset. Existing literature (Gui, Wang, and Tian 2021) shows that the node with higher loss has higher probability to be a noisy node than the node with lower loss. Moreover, the loss in one class is different from that in other classes. Hence, we calculate the loss in the j-th class between the node labels and the predictions of the student model by:

$$L_j = \{l_i^S \mid \mathbf{y}_i = j\}, \text{ where } l_i^S = -\mathbf{y}_i \log \mathbf{p}_i^S, \tag{7}$$

where $j \in (1, c)$ denotes the j-th class. We then set a threshold $\boldsymbol{\varphi}$ for each class to select noisy nodes:

$$\boldsymbol{\varphi}_i = sort(L_i)_{\lceil \delta_i r \rceil}, \tag{8}$$

where δ_i denotes the number of labelled nodes of class i, r is a hyper-parameter that denotes the percentage of selected noisy nodes in the total labelled nodes. Finally, nodes with loss values larger than threshold in each class are regarded as noisy nodes and they are regarded as unlabelled nodes in the following part of the label improvement module.

Pseudo-Label Selection After removing noisy labels, pseudo-label selection investigates to assign pseudo-labels to unlabelled nodes, aim at increasing supervision information. We follow the literature (Li, Han, and Wu 2018) to separately select pseudo-labels based on the prediction confidence of both the student model and the teacher models. Specifically, we first use the student model to predict all unlabelled nodes and then record the confidence of the student model by:

$$\delta_i^S = \max \mathbf{p}_i^S, \tag{9}$$

where δ_i^S denotes the prediction confidence of the student model for node v_i. Given the prediction confidence, we select the top ρ (ρ is a hyper-parameter) nodes with the highest δ^S in each class as the first part of pseudo-labelled nodes.

We then sum the predictions of the teacher models as teacher predictions and calculate the confidence of the teacher models by:

$$\delta_i^T = \max(\sum_{j=1}^{k} \mathbf{p}_i^j), \tag{10}$$

where δ_i^T denotes the prediction confidence of the teacher models for node v_i. Similarly, we select the top ρ nodes with the highest δ^T in each class as the second part of pseudo-labelled nodes.

Finally, we assign all pseudo-labelled nodes with pseudo-labels and further add them to the labelled set, which will be used to adaptively update both the multi-teacher construction module and the multi-teacher distillation module, whose robustness are improved on the data with corrected labels and pseudo-labels with high quality.

Bi-Level Optimization

Optimization Process

Previous works often train the teacher weight matrix and the student model separately. This ignores the potentiality of the student model to provide information for improving the teacher weight matrix. In this paper, we propose to dynamically adjust the teacher weight matrix based on the training progress of the student model, aiming at reducing the dependence of the learning the teacher weight matrix on correct labels. As a result, the optimization of both the teacher weight matrix and the student model results in a bi-level optimization problem, (Liu et al. 2021a; Chen et al. 2022). Considering that the bi-level optimization usually includes two levels of optimization tasks, where the solution of the upper optimization task depends on the results of the lower optimization task. In this paper, we treat the optimization of the teacher weight matrix as the upper level task and the training of the student model as the lower level task, to have the following objective function:

$$\underbrace{\arg\min_{\theta_w} \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)}_{\text{upper level task}} \quad s.t. \quad \theta_s^t = \underbrace{\arg\min_{\theta_s} \mathbf{\Gamma}_{\text{low}}(\theta_w, \theta_s)}_{\text{lower level task}}, \tag{11}$$

where $\mathbf{\Gamma}_{\text{up}}$ and $\mathbf{\Gamma}_{\text{low}}$, respectively, denote the loss functions for the teacher weight matrix and the student model. θ_w and θ_s,

respectively, denote the trainable parameters of the teacher weight matrix $\mathcal{W}$ and the student model. θ_s^t denotes the parameters of the student model in the t-th iteration.

In the lower-level optimization, we train the student model using soft labels. Specifically, we first construct a teacher weight matrix $\mathcal{W} \in \mathbb{R}^{k\times c}$ to fuse the predictions of k teacher models as a soft label matrix, *i.e.,*

$$\widetilde{\mathbf{y}}_i = \text{softmax}(\sum_{j=1}^{k} \mathcal{W}_j \odot \mathbf{p}_i^j), \tag{12}$$

where "$\odot$" denotes the Hadamard product. We then calculate the loss of the lower level optimization by:

$$\mathbf{\Gamma}_{\text{low}}(\theta_w, \theta_s) = \mathcal{L}_s, \text{ where } \mathcal{L}_s = -\sum_{i\in\mathcal{T}} \widetilde{\mathbf{y}}_i \log \mathbf{p}_i^S, \tag{13}$$

where $\mathcal{T}$ denotes the training set, and $\mathbf{p}_i^S$ denote the prediction of the student model for node v_i.

After calculating $\mathbf{\Gamma}_{\text{low}}$, we update the parameters θ_s of the student model by the gradient descent:

$$\theta_s^{\mu} = \theta_s^{\mu-1} - \eta_\mu \mathbf{d}_{\Gamma_{\text{low}}}(\theta_s^{\mu-1}; \theta_w), \tag{14}$$

where η_μ denotes the learning rate and $\mathbf{d}_{\Gamma_{\text{low}}}(\theta_s^{\mu-1}; \theta_w)$ denotes the derivative of $\mathcal{L}_s$ with respect to $\theta_s^{\mu-1}$. As a result, in the lower level optimization, we transfer the knowledge from the teacher models to the student model by having the prediction of the student model mimic the soft labels.

After the lower level optimization, we first pass the result θ_s^t of the lower level optimization to the upper level optimization, and then conduct the upper-level optimization. To do this, we update the teacher weight matrix using both the student model (after t iterations) and clean nodes. Specifically, the loss function of the upper level optimization is designed as the cross-entropy loss between the prediction of the clean nodes in the student model and the pseudo-labels of clean nodes:

$$\mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t) = \mathcal{L}_w, \text{ where } \mathcal{L}_w = -\sum_{i\in Cle} \hat{\mathbf{y}}_i \log \mathbf{p}_i^{s_t}, \tag{15}$$

where Cle denotes the clean node set, $\hat{\mathbf{y}}_i$ denotes the pseudo-label of v_i, and $\mathbf{p}_i^{s_t}$ denotes the prediction of the student model (after t iterations) for node v_i.

Obviously, θ_w is not directly involved in the calculation of $\mathcal{L}_w$, thus we cannot follow the ordinary approach to directly calculate the gradient of $\mathcal{L}_w$ with respect to θ_w and update θ_w. To address this issue, we follow the literature (Liu et al. 2021a,b) to perform derivations based on specific situation.

Specifically, we denote the outer optimization objective as $\min_{\theta_w} \boldsymbol{\varphi}(\theta_w) := \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)$, where θ_s^t is a variable dependent on θ_w, so we have:

$$\frac{\partial \boldsymbol{\varphi}(\theta_w)}{\partial \theta_w} = \underbrace{\frac{\partial \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)}{\partial \theta_w}}_{\text{direct grad}} + \underbrace{\frac{\partial \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)}{\partial \theta_s^t}\left(\frac{\partial \theta_s^t}{\partial \theta_w}\right)}_{\text{indirect grad}}. \tag{16}$$

For the Eq.16, on the one hand, θ_w is not directly involved in the calculation of $\mathcal{L}_w$, thus the "direct grad" is zero. On the other hand, the "indirect grad" consists of two parts, *i.e.,* $\frac{\partial \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)}{\partial \theta_s^t}$ and $\frac{\partial \theta_s^t}{\partial \theta_w}$. Since the former can be directly calculated by the derivation of $\mathcal{L}_w$ with respect to θ_s^t, the main challenging lies in the calculation both $\frac{\partial \theta_s^t}{\partial \theta_w}$ and $\frac{\partial \boldsymbol{\varphi}(\theta_w)}{\partial \theta_w}$.

In this paper, we first dynamically unfold θ_s^t based on Eq.14 to calculate $\frac{\partial \theta_s^t}{\partial \theta_w}$ by:

$$\theta_s^t = \boldsymbol{\psi}_t(\theta_s^{t-1}; \theta_w), \text{ where} \tag{17}$$
$$\boldsymbol{\psi}_i(\theta_s^{i-1}; \theta_w) = \theta_s^{i-1} - \eta_i \mathbf{d}_{\mathbf{\Gamma}_{\text{low}}}(\theta_s^{i-1}; \theta_w),\ i = 1, \cdots, t.$$

Combining $\frac{\partial \theta_s^t}{\partial \theta_w}$ and Eq.17, we have:

$$\frac{\partial \theta_s^t}{\partial \theta_w} = \frac{\partial \boldsymbol{\psi}_t(\theta_s^{t-1}; \theta_w)}{\partial \theta_s^{t-1}} \frac{\partial \theta_s^{t-1}}{\partial \theta_w} + \frac{\partial \boldsymbol{\psi}_t(\theta_s^{t-1}; \theta_w)}{\partial \theta_w}. \tag{18}$$

To simplify the notation, we denote:

$$\mathbf{Z}_t = \frac{\partial \theta_s^t}{\partial \theta_w},\ \mathbf{A}_t = \frac{\partial \boldsymbol{\psi}_t(\theta_s^{t-1}; \theta_w)}{\partial \theta_s^{t-1}},\ \mathbf{B}_t = \frac{\partial \boldsymbol{\psi}_t(\theta_s^{t-1}; \theta_w)}{\partial \theta_w}. \tag{19}$$

We then rewrite Eq.18 to:

$$\frac{\partial \theta_s^t}{\partial \theta_w} = \mathbf{Z}_t = \sum_{i=0}^{t} (\prod_{j=0}^{t} \mathbf{A}_j)\mathbf{B}_i, \tag{20}$$

where $\mathbf{A}_0 = \mathbf{B}_0 = 0$. $\mathbf{A}_i$ and $\mathbf{B}_i$ ($i \neq 0$), respectively, can be obtained by calculating the derivative of $\mathbf{d}_{\mathbf{\Gamma}_{\text{low}}}(\theta_s^{i-1}; \theta_w)$ with respect to θ_s^{i-1} and θ_w.

Based on the above derivation, it is obviously that we can iteratively calculate $\frac{\partial \theta_s^t}{\partial \theta_w}$ during the optimization of the lower level optimization. Finally, we calculate $\frac{\partial \boldsymbol{\varphi}(\theta_w)}{\partial \theta_w} = \frac{\partial \mathbf{\Gamma}_{\text{up}}(\theta_w, \theta_s^t)}{\partial \theta_s^t}\left(\frac{\partial \theta_s^t}{\partial \theta_w}\right)$ and update the parameters θ_w of the teacher weight matrix by the gradient descent method:

$$\theta_w^{\lambda} = \theta_w^{\lambda-1} - \eta_\lambda \mathbf{d}_{\boldsymbol{\varphi}(\theta_w)}(\theta_s^t; \theta_w^{\lambda-1}), \tag{21}$$

where η_λ denotes the learning rate and $\mathbf{d}_{\boldsymbol{\varphi}(\theta_w)}(\theta_s^t; \theta_w^{\lambda-1})$ denotes the derivative of $\mathcal{L}_w$ with respect to $\theta_w^{\lambda-1}$.

In the upper level optimization, we use the feedback from the lower level optimization (*i.e.,* the student model) to update the teacher weight matrix, and thus rationally adjust the weights of the teacher models. After updating the teacher weight matrix, we map the predictions of k teacher models to soft labels through Eq.12 and then train the student models again (*i.e.,* the lower level optimization). By iteratively updating the upper level optimization and lower level optimization, we fully transfer knowledge from the teacher models to the student model. As a result, through bi-level optimization strategy, we successfully addressed the issue that previous methods require a large number of correct labels to learn the weights of the teacher models.

Experiments

Experimental Settings

Datasets The benchmark datasets in our experiments include three citation datasets (*i.e.,* DBLP (Bojchevski and Günnemann 2017), Cora and Citeseer (Yang, Cohen, and Salakhudinov 2016)), and two business datasets (*i.e.,* Computers and Photo (Shchur et al. 2018)).

Datasets	Methods	Uniform			Pair	
		20%	40%	60%	20%	40%
Cora	GCN	74.74 (0.6)	69.40 (1.6)	41.76 (1.9)	72.58 (2.1)	51.82 (0.4)
	DGI	81.56 (0.2)	79.92 (0.4)	48.94 (0.3)	81.16 (0.3)	61.60 (0.5)
	GREET	80.88 (0.8)	78.22 (1.3)	44.70 (2.8)	77.96 (1.7)	56.42 (1.8)
	JoCoR	75.58 (0.8)	71.14 (1.1)	42.82 (1.7)	74.46 (1.1)	52.54 (1.2)
	NRGNN	79.14 (0.7)	78.68 (1.0)	52.02 (2.1)	78.66 (1.8)	56.30 (5.9)
	MTS-GNN	81.50 (0.8)	79.26 (0.7)	63.34 (3.1)	80.68 (1.8)	62.16 (2.2)
	Proposed	**82.64 (1.1)**	**82.20 (0.9)**	**73.16 (1.0)**	**81.64 (1.6)**	**73.46 (3.2)**
Citeseer	GCN	63.18 (0.6)	56.04 (2.0)	37.74 (0.7)	66.60 (0.9)	40.68 (2.7)
	DGI	69.24 (0.2)	63.20 (0.1)	42.14 (0.3)	69.24 (0.3)	45.14 (0.2)
	GREET	68.36 (0.9)	62.46 (1.2)	44.50 (1.7)	70.74 (1.5)	46.88 (1.0)
	JoCoR	69.06 (1.2)	57.18 (1.4)	38.64 (1.0)	69.54 (0.6)	46.82 (2.4)
	NRGNN	66.22 (1.2)	59.66 (1.5)	45.82 (3.2)	67.86 (1.8)	50.10 (2.1)
	MTS-GNN	71.94 (1.6)	69.74 (1.1)	53.64 (2.1)	72.34 (1.1)	59.84 (2.8)
	Proposed	**72.14 (0.7)**	**70.80 (0.8)**	**58.18 (3.9)**	**72.90 (0.5)**	**63.36 (3.9)**
DBLP	GCN	80.32 (0.2)	77.54 (1.1)	65.05 (0.6)	80.19 (0.5)	61.56 (1.2)
	DGI	79.86 (0.2)	79.28 (0.2)	67.59 (0.2)	78.99 (0.1)	70.29 (0.5)
	GREET	80.62 (0.3)	79.83 (0.5)	69.10 (1.0)	81.16 (0.2)	71.41 (1.8)
	JoCoR	80.77 (0.3)	78.52 (0.9)	67.32 (1.1)	80.26 (0.3)	64.71 (3.3)
	NRGNN	81.89 (0.4)	82.20 (0.3)	76.13 (1.6)	82.01 (0.4)	72.80 (1.2)
	MTS-GNN	82.76 (0.2)	82.49 (0.3)	78.82 (1.5)	82.68 (2.4)	70.35 (4.5)
	Proposed	**83.93 (0.2)**	**83.87 (0.2)**	**81.18 (0.3)**	**83.56 (1.7)**	**77.32 (0.8)**
Photo	GCN	88.15 (0.6)	84.51 (0.7)	73.27 (5.9)	88.50 (0.8)	66.20 (1.3)
	DGI	87.83 (0.4)	86.78 (0.5)	78.61 (0.9)	85.67 (0.1)	70.71 (0.2)
	GREET	88.77 (0.4)	86.39 (0.5)	74.71 (1.2)	88.06 (0.4)	65.66 (0.7)
	JoCoR	88.34 (1.1)	84.17 (0.8)	79.22 (2.1)	88.78 (0.5)	70.02 (2.9)
	NRGNN	88.30 (0.5)	86.17 (0.6)	77.25 (3.5)	88.61 (0.6)	66.20 (2.4)
	MTS-GNN	91.01 (1.4)	88.10 (1.2)	85.12 (1.7)	**91.74 (0.3)**	81.26 (4.6)
	Proposed	**91.75 (0.1)**	**90.06 (1.0)**	**87.01 (0.9)**	91.26 (0.3)	**86.96 (2.3)**
Computers	GCN	86.66 (0.3)	83.18 (0.5)	78.07 (0.4)	84.60 (0.1)	72.91 (0.8)
	DGI	83.12 (0.1)	77.71 (0.1)	75.89 (0.3)	81.73 (0.2)	77.71 (0.1)
	GREET	84.16 (0.2)	80.75 (0.8)	72.19 (1.2)	83.05 (0.1)	64.35 (0.9)
	JoCoR	**86.96 (0.4)**	83.46 (0.4)	77.99 (0.5)	84.92 (0.4)	73.93 (0.9)
	NRGNN	86.51 (0.5)	83.63 (0.5)	77.37 (0.8)	84.34 (0.5)	74.22 (1.1)
	MTS-GNN	86.43 (0.2)	84.29 (0.2)	80.65 (0.1)	**86.36 (0.4)**	76.55 (2.1)
	Proposed	85.81 (0.2)	**84.89 (0.1)**	**80.85 (0.3)**	86.04 (0.2)	**82.38 (1.2)**

Table 1: Classification accuracy (average accuracy (%) and standard deviation) of all methods with different noise rates on all datasets. Note that, the bold number represents the best results in the whole column.

Comparison Methods The comparison methods include a baseline method (*i.e.,* GCN (Kipf and Welling 2016)), two unsupervised graph representation learning method (*i.e.,* DGI (Veličković et al. 2018) and GREET (Liu et al. 2023b)), and three noisy label methods (*i.e.,* JoCoR (Wei et al. 2020), NRGNN (Dai, Aggarwal, and Wang 2021), and MTS-GNN (Liu et al. 2023a)).

Setting-Up We partition each dataset into three non-overlapping subsets, *i.e.,* training set, validation set, and test sets. Furthermore, we follow the literature (Han et al. 2018; Jiang et al. 2018) to add noise into the training datasets.

To reduce randomness of experimental results, we conduct five experiments with different random seeds and report the average results and corresponding standard deviations. In our experiments, we train three encoders by employing GCA (Zhu et al. 2021), DGI (Veličković et al. 2018) and SUGRL (Mo et al. 2022), plus single layer MLP for every encoder to obtain three teacher classifiers. For a fair comparison, the methods, including JoCoR, NRGNN, MTS-GNN, and the student model of our method use a two-layer GCN as the backbone. In addition, we obtain the source code of all comparison methods from the authors and set the parameters of all comparison methods according to the original literature so that they output the best performance on all datasets.

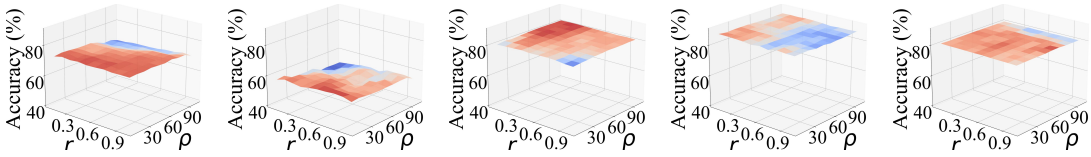
Result Analysis

We compare our proposed BO-NNC with all comparison methods on five datsets in terms of node classification tasks with different noise rates and report the results in Table 1.

First, the proposed BO-NNC consistently achieves the best results on all datasets, followed by MTS-GNN, NRGNN, DGI, GREET, JoCoR, and GCN. The reason is

Uniform (60%)							
C1	C2	C3	Cora	Citeseer	DBLP	Photo	Computers
✓			55.22 (1.0)	44.74 (2.4)	77.57 (0.8)	83.74 (1.3)	79.54 (0.4)
		✓	72.50 (1.3)	53.24 (3.0)	80.54 (0.4)	86.23 (1.1)	80.45 (0.6)
✓	✓		58.56 (5.0)	45.28 (2.1)	76.67 (0.6)	84.04 (1.0)	79.30 (2.0)
✓		✓	71.26 (1.4)	56.78 (3.5)	81.25 (0.3)	86.10 (0.7)	80.36 (0.7)
✓	✓	✓	**73.16 (1.0)**	**58.18 (3.9)**	**81.95 (0.3)**	**87.01 (0.9)**	**80.85 (0.3)**

Table 2: Classification accuracy (average accuracy (%) and standard deviation) of the proposed BO-NNC with different components on all datasets at the highest noise rates. Note that the bold number represents the best results in the whole column.



(a) Cora (b) Citeseer (c) DBLP (d) Photo (e) Computers

Figure 2: Parameter sensitivity analysis of r and ρ in our method at 60% noise rate.

that our BO-NNC enables the student model to obtain the knowledge from multiple teacher models, thus solving the limitations of the single model to effectively deal with noisy labels. Second, our BO-NNC outperforms GCN, DGI, and GREET by a large margin, because the proposed method provides more correct supervised information for the model training by noisy label filtering and pseudo-label selection. For example, the proposed BO-NNC averagely improves by 2.93%, 4.99%, and 13.44% respectively, compared with the DGI that performs best in there three methods, in uniform noise rates of 20%, 40%, and 60% on all datasets. Third, our BO-NNC achieves promising improvements, compared with the methods that deal with the noisy label problem, *i.e.,* JoCoR, NRGNN, and MTS-GNN. This indicates that our method fully leverages the complementary information among teacher models, and thus enabling the student model to achieve better generalization performance.

Ablation Study

Our method contains three important components, *i.e.,* using the student model to integrate knowledge from multiple teacher models (C1 for short), multi-teacher knowledge distillation based on bi-level optimization (C2 for short), and label improvement (C3 for short). To demonstrate the effectiveness of each component, we report the node classification performance of different component combinations on all datasets at the highest noise rates in Table 2.

Based on the experimental results, our method achieves the best performance with all components included, which demonstrates the feasibility of our proposed method. Second, the overall effectiveness of the method considering the student model is significantly superior to the one without considering the student model, because the student model can explore the complementary information from the teacher models. Last, bi-level optimization needs enough clean node to achieve effectiveness, while label improvement may provide enough clean data. Hence, it is reasonable to simultaneously consider bi-level optimization based multi-teacher distillation and label improvement for dealing with noisy labels.

Parameter Sensitivity Analysis

Our method has two key hyper-parameters, *i.e.,* ρ for pseudo-label selection and r for noisy label selection. We investigate the sensitivity of our method to these hyper-parameters, and report the results in Figure 2.

Based on the experimental results, our method is sensitive to the parameters r and ρ. Specifically, if the value of r is less than 0.4, the performance of the model gradually increases with the increase of r. If the value of r is larger than 0.6, the performance of the model gradually decreases with the increase of r. The reason is that too many correct labels were removed. If the value of ρ is less than 70, the variation of ρ has little impact on the model performance. In addition, if the value of ρ is larger than 70, the performance of the model gradually decreases with the increase of ρ. The reason is that the incorrect pseudo-label will be inevitably selected.

Conclusion

In this paper, we proposed a new NNC method to address the limitations in previous methods. Specifically, multi-teacher distillations transfer diverse information to the learning of the student model. The bi-level optimization strategy fully explores the complementary information among multiple teacher models for the learning of the student model so that requiring less number of correct labels to achieve a robust student model. Experimental results showed that our method achieves supreme performance, compared to the state-of-the-art methods, in terms of noisy node classification tasks.

Acknowledgments

This work was supported in part by National Key Research and Development Program of China under Grant 2022YFA1004100 and the Guangxi Natural Science Foundation 2023GXNSFBA026010.

References

Arpit, D.; Jastrzebski, S.; Ballas, N.; Krueger, D.; Bengio, E.; Kanwal, M. S.; Maharaj, T.; Fischer, A.; Courville, A.; Bengio, Y.; et al. 2017. A closer look at memorization in deep networks. In *ICML*, 233–242. PMLR.

Bojchevski, A.; and Günnemann, S. 2017. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. *arXiv preprint arXiv:1707.03815*.

Chen, C.; Chen, X.; Ma, C.; Liu, Z.; and Liu, X. 2022. Gradient-based bi-level optimization for deep learning: A survey. *arXiv preprint arXiv:2207.11719*.

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

Dai, E.; and Wang, S. 2021. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *WSDM*, 680–688.

Fu, X.; Wei, Y.; Sun, Q.; Yuan, H.; Wu, J.; Peng, H.; and Li, J. 2023. Hyperbolic geometric graph representation learning for hierarchy-imbalance node classification. In *ACM WWW*, 460–468.

Gao, Y.; Parcollet, T.; and Lane, N. D. 2021. Distilling knowledge from ensembles of acoustic models for joint ctc-attention end-to-end speech recognition. In *ASRU*, 138–145. IEEE.

Gui, X.-J.; Wang, W.; and Tian, Z.-H. 2021. Towards understanding deep learning from noisy labels with small-loss criterion. *arXiv preprint arXiv:2106.09291*.

Guo, S.; Lin, Y.; Feng, N.; Song, C.; and Wan, H. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, volume 33, 922–929.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. *NeurIPS*, 31.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jiang, L.; Zhou, Z.; Leung, T.; Li, L.-J.; and Fei-Fei, L. 2018. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2304–2313. PMLR.

Kim, K.; Ji, B.; Yoon, D.; and Hwang, S. 2021. Self-knowledge distillation with progressive refinement of targets. In *ICCV*, 6567–6576.

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

Li, Q.; Han, Z.; and Wu, X.-M. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, volume 32.

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

Liu, R.; Gao, J.; Zhang, J.; Meng, D.; and Lin, Z. 2021a. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *TPAMI*, 44(12): 10045–10067.

Liu, R.; Liu, X.; Yuan, X.; Zeng, S.; and Zhang, J. 2021b. A value-function-based interior-point method for non-convex bi-level optimization. In *ICML*, 6882–6892. PMLR.

Liu, Y.; Wu, Z.; Lu, Z.; Wen, G.; Ma, J.; Lu, G.; and Zhu, X. 2023a. Multi-teacher Self-training for Semi-supervised Node Classification with Noisy Labels. In *ACM MM*, 2946–2954.

Liu, Y.; Zheng, Y.; Zhang, D.; Lee, V. C.; and Pan, S. 2023b. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *AAAI*, volume 37, 4516–4524.

Mo, Y.; Peng, L.; Xu, J.; Shi, X.; and Zhu, X. 2022. Simple unsupervised graph representation learning. In *AAAI*, volume 36, 7797–7805.

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

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

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2018. Deep graph infomax. *arXiv preprint arXiv:1809.10341*.

Wei, H.; Feng, L.; Chen, X.; and An, B. 2020. Combating noisy labels by agreement: A joint training method with co-regularization. In *CVPR*, 13726–13735.

Wei, Y.; Yuan, H.; Fu, X.; Sun, Q.; Peng, H.; Li, X.; and Hu, C. 2024. Poincaré Differential Privacy for Hierarchy-aware Graph Embedding. In *AAAI*, volume 38, 9160–9168.

Windsor, F. M.; van den Hoogen, J.; Crowther, T. W.; and Evans, D. M. 2023. Using ecological networks to answer questions in global biogeography and ecology. *J Biogeogr*, 50(1): 57–69.

Wu, L.; Huang, Y.; Lin, H.; Liu, Z.; Fan, T.; and Li, S. Z. 2022. Automated Graph Self-supervised Learning via Multi-teacher Knowledge Distillation. *arXiv preprint arXiv:2210.02099*.

Yang, Z.; Cohen, W.; and Salakhudinov, R. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 40–48. PMLR.

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

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, 2069–2080.