Leveraging Peer-Informed Label Consistency for Robust Graph Neural Networks with Noisy Labels

Kailai Li[1,3], **Jiawei Sun**[1,3], **Jiong Lou**[1,3,5,*], **Zhanbo Feng**[1,3], **Hefeng Zhou**[1,3], **Chentao Wu**[1,4,5], **Guangtao Xue**[1], **Wei Zhao**[2] and **Jie Li**[1,4,5,*]

[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Shenzhen University of Advanced Technology
[3]Shanghai Jiao Tong University (Wuxi) Blockchain Advanced Research Center
[4]Yancheng Blockchain Research Institute
[5]Shanghai Key Laboratory of Trusted Data Circulation and Governance and Web3
{kailai_li,noelsjw,lj1994,zhanbofeng,zadeji1}@sjtu.edu.cn, wuct@cs.sjtu.edu.cn, gt_xue@sjtu.edu.cn, weizhao86@outlook.com, lijiecs@sjtu.edu.cn

Abstract

Graph Neural Networks (GNNs) excel in many applications but struggle when trained with noisy labels, especially as noise can propagate through the graph structure. Despite recent progress in developing robust GNNs, few methods exploit the intrinsic properties of graph data to filter out noise. In this paper, we introduce ProCon, a novel framework that identifies mislabeled nodes by measuring label consistency among semantically similar peers, which are determined by feature similarity and graph adjacency. Mislabeled nodes typically exhibit lower consistency with these peers, a signal we measure using pseudo-labels derived from representational prototypes. A Gaussian Mixture Model is fitted to the consistency distribution to identify clean samples, which refine prototype quality in an iterative feedback loop. Experiments on multiple datasets demonstrate that ProCon significantly outperforms state-of-the-art methods, effectively mitigating label noise and enhancing GNN robustness.

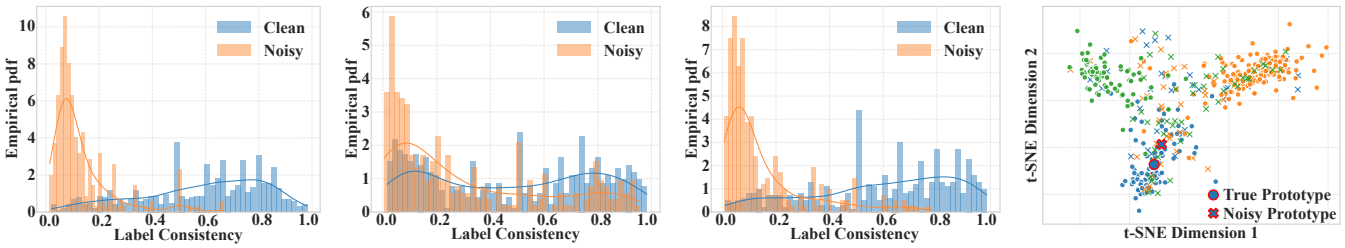
1 Introduction

Graph Neural Networks (GNNs) have demonstrated excellent performance in various applications. Training a GNN requires data, but the labels of data can be noisy. This noise can originate from human annotators or automatic label extraction tools, such as crowd-sourcing and web crawling. Existing research indicates that GNNs tend to memorize label noise [Zhang *et al.*, 2021], and noise can propagate through the graph structure [Dai *et al.*, 2021; Qian *et al.*, 2023], degrading the performance. Therefore, training a robust GNN to resist label noise is crucial. As label noise problems may appear in any context, such robustness enhances GNN reliability in many applications, including recommender systems [Wu *et al.*, 2022], traffic forecasting [Sun *et al.*, 2022] and molecular property prediction [Wieder *et al.*, 2020].

Current research presents various solutions to mitigate the issue of label noise. One prevalent approach involves identifying accurately labeled samples. A commonly adopted strategy is to consider samples with small loss values as clean. This strategy is based on the observation that, in the early stages of training, loss values of clean samples are smaller than those of mislabeled samples [Han *et al.*, 2018; Yu *et al.*, 2019]. Some studies also leverage representation space [Wu *et al.*, 2020] or training dynamics [Pleiss *et al.*, 2020] to identify noisy data. To enhance the robustness of GNNs against label noise, existing methods utilize graph structure. These approaches include graph structure refinement [Dai *et al.*, 2021; Dai *et al.*, 2022], topology-based curriculum learning [Wu *et al.*, 2024], and introduction of additional supervision signals [Qian *et al.*, 2023; Du *et al.*, 2023; Yuan *et al.*, 2023a; Chen *et al.*, 2024]. Despite these advancements, existing approaches for identifying clean samples remain dependent on model outputs. However, noisy supervision during training can compromise the reliability of outputs. Consequently, sample selection that relies on model outputs can be misled by noisy labels, potentially resulting in confirmation bias [Tarvainen and Valpola, 2017] and suboptimal performance. The intrinsic properties of graph data are not adequately exploited, thereby reducing the effectiveness in dealing with label noise.

To address this issue, we propose a method that leverages the label consistency of semantically similar nodes for sample selection. For each target node, a set of peers is identified, comprising nodes with either similar features or structural adjacency within the graph. Label consistency is defined as the proportion of peers sharing the target node's assigned label, serving as a metric to evaluate its correctness. This approach is grounded in two general properties of graph data: (1) nodes with similar features are more likely to share the same label [Zhu *et al.*, 2022], and (2) graph homophily [McPherson *et al.*, 2001] and monophily [Altenburger and Ugander, 2018] suggest that nodes often belong to the same semantic class as their one-hop or two-hop neighbors. Consequently, correctly

*Corresponding Author



(a) Ground truth (b) GNN classifier (c) ProCon (d) Representations

Figure 1: Peer-Informed Label Consistency distributions and node representations from WikiCS dataset. Subfigures (a-c) depict label consistency distributions based on (a) ground truth labels, (b) GNN classifier predictions, and (c) ProCon leveraging prototypes. Subfigure (d) shows a t-SNE visualization of node representations obtained by training a GNN on noisily labeled WikiCS. Colors indicate noisy labels. Markers '×' represent nodes with incorrect labels. Prototypes for the blue class, derived from both ground truth and noisy labels, are highlighted.

labeled nodes typically demonstrate higher label consistency with peers, while mislabeled nodes exhibit lower consistency. This discrepancy, as illustrated in Figure 1a, underpins the identification of mislabeled nodes through the analysis of peer-informed label consistency.

The challenge arises from the unavailability of ground truth labels, which prevents the calculation of true label consistency. One workaround is to use the predictions from a neural network classifier as a proxy. However, neural networks are prone to memorizing and overfitting noisy labels [Arpit *et al.*, 2017; Zhang *et al.*, 2021]. Additionally, the message-passing mechanism of GNNs inherently smooths outputs across neighboring nodes [Chen *et al.*, 2020]. Consequently, classifier predictions for the peers of mislabeled nodes tend to align with incorrect labels, making label consistency between mislabeled and correctly labeled nodes indistinguishable, as shown in Figure 1b.

To address this challenge, we employ representations and prototypes to assign pseudo-labels to nodes and estimate true label consistency. A prototype is calculated as the average representation of samples in a class, serving as a representative point in the representation space. Intuitively, samples with the same ground truth label tend to cluster together in the representation space [Papyan *et al.*, 2020; Zhu *et al.*, 2021], with clean samples dominating the prototype calculation, as shown in Figure 1d. Consequently, prototypes are more robust and less prone to memorizing noisy labels compared to neural network classifiers. Each node is assigned the label of its nearest prototype, and the resulting label consistency is referred to as *prototypical consistency*. A Gaussian Mixture Model is then fitted to the prototypical consistency distribution of the training set to distinguish between correctly labeled and mislabeled nodes. The correctly labeled nodes identified in this process refine the prototypes, leading to more accurate label consistency estimation and enhanced reliability of label identification in subsequent iterations, forming a positive feedback loop. Since noisy labels can degrade the quality of node representations [Wang *et al.*, 2024; Wu *et al.*, 2020], we incorporate a graph self-supervised module to maintain high-quality node representations even in the presence of noisy labels. Our contributions are as follows:

1) We introduce a novel perspective, suggesting that the intrinsic properties of graph data can be exploited to identify mislabeled nodes.

2) We propose ProCon, a novel graph learning framework for noisy labels, incorporating the peer-informed label consistency and node representations to identify mislabeled nodes.

3) Extensive experiments conducted on multiple datasets demonstrate that ProCon outperforms current state-of-the-art methods, demonstrating its superior performance in handling noisy-labeled graph data.

2 Related Work

2.1 Learning with Noisy Labels

Approaches to learning with noisy labels can be primarily divided into robust algorithms and sample selection strategies. Robust algorithms are designed to train networks on noisy datasets while mitigating performance degradation. This approach includes modifications in network architectures [Lee *et al.*, 2019; Goldberger and Ben-Reuven, 2017], loss functions [Zhang and Sabuncu, 2018], and robust regularization [Wei *et al.*, 2021; Menon *et al.*, 2020].

Sample selection methods focus on identifying clean samples within noisy datasets. Motivated by the memorization effect of deep neural networks (DNNs) [Arpit *et al.*, 2017], early studies typically employ peer networks and the small-loss trick, exemplified by methods such as Decoupling [Malach and Shalev-Shwartz, 2017], Co-teaching [Han *et al.*, 2018], and Co-teaching+ [Yu *et al.*, 2019]. Another line of research aims to establish new sample selection criteria. For instance, AUM [Pleiss *et al.*, 2020] identifies mislabeled data by monitoring its prediction margin during training, while TopoFilter [Wu *et al.*, 2020] identifies isolated data in the representation space as mislabeled. Upon obtaining clean data, a straightforward strategy is to train the network solely on clean data or to re-weight the data. Some studies [Li *et al.*, 2020; Wang *et al.*, 2024] treat selected noisy data as unlabeled and apply semi-supervised or contrastive learning techniques. Others [Xiao *et al.*, 2015; Yi and Wu, 2019] develop label-correction modules to pseudo-label noisy data for training.

2.2 Robust Graph Neural Networks

Graph Neural Networks (GNNs) excel in tasks such as node classification, link prediction, and graph classification by leveraging graph structure to propagate information and capture both local and global patterns [Sun *et al.*, 2024]. Nev-

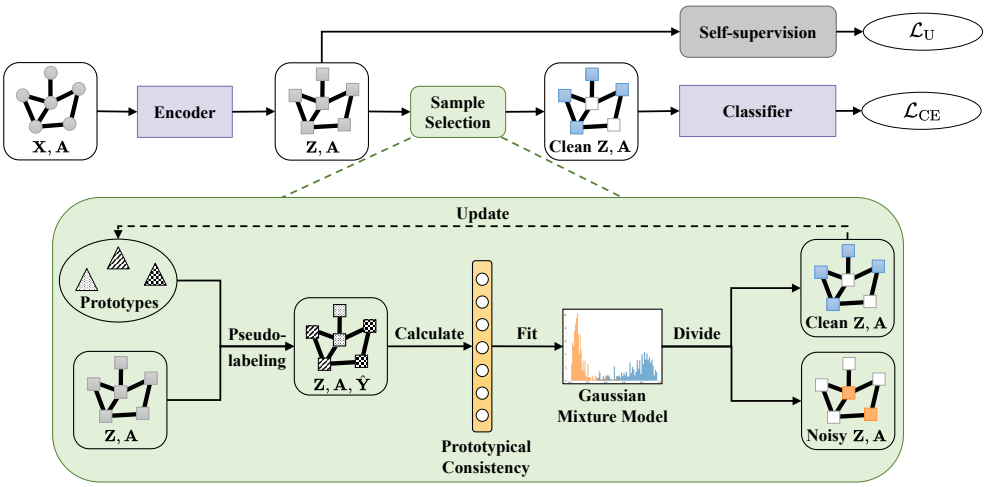


Figure 2: Pipeline of ProCon. The prototypical consistency is determined based on pseudo-labels from peers. A Gaussian Mixture Model is then fitted to identify clean nodes. Subsequently, the identified clean nodes are utilized to refine the prototypes. ProCon effectively identifies clean nodes by exploiting the discrepancy of the Peer-Informed Prototypical Consistency between mislabeled and correctly labeled nodes.

ertheless, existing work [Dai *et al.*, 2021; Du *et al.*, 2023; Qian *et al.*, 2023] reveals that the performance of GNNs can be substantially compromised by noisy labels.

Recent studies have introduced various strategies to enhance the robustness of GNNs against noisy labels. Several approaches enhance message passing in GNNs to counteract the effects of label noise. For instance, NRGNN [Dai *et al.*, 2021] connects unlabeled nodes with similar labeled counterparts to integrate clean label information, while RS-GNN [Dai *et al.*, 2022] diminishes noisy edges by learning a denoised graph.

Other approaches introduce supplementary and reliable supervision, including self-reinforcement and consistency regularization [Qian *et al.*, 2023], principles guided by information theory [Zhou *et al.*, 2023], and pairwise interactions [Du *et al.*, 2023]. Additionally, graph contrastive learning has been utilized to address label noise, as evidenced by RNCGLN [Zhu *et al.*, 2024], ALEX [Yuan *et al.*, 2023a], and CGNN [Yuan *et al.*, 2023b]. In addition, UnionNet [Li *et al.*, 2021] employs label aggregation to facilitate sample reweighting and label correction. TSS [Wu *et al.*, 2024] introduces a Class-conditional Betweenness Centrality measure to optimize sample selection using topological information for curriculum learning. ERASE [Chen *et al.*, 2024] adopts label propagation and maximize coding rate reduction. Furthermore, GNN Cleaner [Xia *et al.*, 2024] presuppose the availability of a clean set.

While previous studies have effectively enhanced the robustness of Graph Neural Networks (GNNs) against label noise, few have leveraged the intrinsic properties of graph data to filter out noisy samples. In contrast, our method leverages peer-informed label consistency by selecting semantically similar peers based on feature similarity and graph adjacency. This intrinsic exploitation allows ProCon to accurately identify mislabeled nodes, thereby significantly improving the robustness and effectiveness of GNNs.

3 Method

3.1 Preliminaries

We focus on the node classification task with label noise. For an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with n nodes, the node set is denoted as $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, and the edge set is denoted as $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix of $\mathcal{G}$. If nodes v_i and v_j are connected, then $\mathbf{A}_{ij} = 1$, otherwise $\mathbf{A}_{ij} = 0$. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ denote the node feature matrix, where $\mathbf{X}_i \in \mathbb{R}^d$ represents the feature vector of node v_i. Let $\mathbf{Y} = (y_1, y_2, \ldots, y_n)$ denote the node label vector, where $y_i \in [L] = \{1, 2, \ldots, L\}$ represents the label of node v_i. The training node set is denoted as $\mathcal{V}_{\mathrm{tr}} \subset \mathcal{V}$. The labels of the training set, denoted as $\mathbf{Y}_{\mathrm{tr}}$, are corrupted by noise. Our objective is to train a GNN encoder f and a classifier g from the noisy training set such that $g \circ f : (\mathbf{X}, \mathbf{A}) \to \mathbf{Y}$.

3.2 Divide by Peer-Informed Prototypical Consistency

Our goal is to identify correctly labeled nodes and estimate their probabilities of being correct. The key to our method lies in harnessing the disparity in peer-informed label consistency between correctly labeled and mislabeled nodes.

We first outline the process for identifying the peers of each node. The objective is to ensure that the labels of the selected peers are likely to align with the label of the target node. Peer selection is driven by two main factors: (1) feature similarity, as nodes exhibiting similar features are more prone to sharing the same label, and (2) graph adjacency, since nodes are more likely to share labels with their first- or second-order neighbors [McPherson *et al.*, 2001; Altenburger and Ugander, 2018]. To quantify feature similarity, we construct a similarity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where $\mathbf{S}_{i,j}$ represents the cosine similarity between nodes v_i and v_j, computed as follows:

$$\mathbf{S}_{i,j} = \frac{\mathbf{X}_i \cdot \mathbf{X}_j}{\|\mathbf{X}_i\| \|\mathbf{X}_j\|} \tag{1}$$

Subsequently, the similarity matrix $\mathbf{S}$ is refined by incorporating graph adjacency information:

$$\tilde{\mathbf{S}} = \mathbf{S} \odot \left(\alpha \mathbf{1}\mathbf{1}^\top + (1-\alpha)\mathbf{A}^{[2]}\right) \tag{2}$$

Here, $\odot$ represents the Hadamard product, and $\mathbf{1}$ is a column vector of ones. The matrix $\mathbf{A}^{[2]} \in \mathbb{R}^{n\times n}$ represents the two-hop reachability matrix of graph $\mathcal{G}$. If nodes v_i and v_j are reachable within two hops, then $\mathbf{A}^{[2]}_{i,j}$ equals 1; otherwise, it is 0. The hyperparameter $\alpha \in (0,1)$ controls the influence of graph structure on the similarity matrix. A smaller value of α emphasizes selecting peers from local neighbors, while a larger value places more weight on global feature similarity. The vector $\tilde{\mathbf{S}}_i$ represents the topology-aware similarity of all nodes with respect to node v_i. For a target node v_i, let the peer set be denoted as $\tilde{\mathcal{V}}^{(i)}$. Peers are selected based on the top-k largest values in $\tilde{\mathbf{S}}_i$:

$$\tilde{\mathcal{V}}^{(i)} = \{v_j | v_j \in \text{Top}_k(\tilde{\mathbf{S}}_i)\} \tag{3}$$

where $\text{Top}_k(\cdot)$ represents the set of nodes corresponding to the top-k largest entries in the input.

To evaluate label consistency between training nodes and peers, pseudo-labels are assigned to unlabeled peers using a prototype-based method. We define the node representation matrix as $\mathbf{Z}$, with $\mathbf{Z}_i$ denoting the representation of node v_i. Node representations are derived from a GNN encoder f, parameterized by $\boldsymbol{\theta}$, as $\mathbf{Z} = f(\mathbf{X}, \mathbf{A}; \boldsymbol{\theta})$. For each class ℓ, a prototype representation $\mathbf{c}_\ell$ is maintained as its representative vector. The pseudo-label $\hat{y}_i$ for node v_i corresponds to the class of the closest prototype:

$$\hat{y}_i = \arg\min_{\ell \in [L]} \left(1 - \frac{\mathbf{z}_i \cdot \mathbf{c}_\ell}{\|\mathbf{z}_i\| \|\mathbf{c}_\ell\|}\right). \tag{4}$$

We use cosine distance to measure the closeness between representations. The prototypical pseudo-label $\hat{y}_i$ indicates the class to which node v_i belongs in the representation space.

Subsequently, label consistency for training nodes is computed by given labels and prototypical pseudo-labels of peers. We term the label consistency value Peer-Informed Prototypical Consistency.

Definition 1 (Peer-Informed Prototypical Consistency)**.** *Peer-Informed Prototypical Consistency for a node v_t is defined as the proportion of its peers whose prototypical pseudo-labels match the given label:*

$$h_t = \frac{\left|\left\{v_j \mid v_j \in \tilde{\mathcal{V}}^{(t)} \wedge y_t = \hat{y_j}\right\}\right|}{|\tilde{\mathcal{V}}^{(t)}|}, \tag{5}$$

where y_t is the given label of node v_t, $\hat{y}_j$ is the prototypical pseudo-label of node v_j, and $\tilde{\mathcal{V}}^{(t)}$ represents the peers of v_t.

The prototypical pseudo-labels of peers represent the expected labels for the target node, derived from representations, features, and graph structure. Prototypical consistency measures this expectation, with lower values reflecting increased conflict between the given label and the expected label, thereby implying a higher likelihood of label error.

The prototypical consistency can vary across different nodes and graphs. Therefore, rather than setting a fixed threshold for sample selection, we analyze the empirical distribution of prototypical consistency value. Let the consistency values of training nodes be $\mathcal{H} = \{h_t : v_t \in \mathcal{V}_{\text{tr}}\}$. To distinguish between mislabeled and correctly labeled training nodes, a two-component Gaussian Mixture Model (GMM) is fitted to $\mathcal{H}$ using the expectation-maximization algorithm. For any training node v_t, the GMM outputs the posterior probability $p(m \mid h_t)$ of its label being clean, where m represents the Gaussian component with the larger mean. For simplicity, we refer to $p(\ell \mid h_t)$ as w_t. Let $\mathcal{W} = \{w_t : v_t \in \mathcal{V}_{\text{tr}}\}$ be the set of clean probabilities of training nodes. We set a threshold τ on clean probabilities, and consider nodes satisfying $w_t > \tau$ as clean. The set of all selected clean nodes is denoted as $\mathcal{V}_{\text{c}} = \{v_t : v_t \in \mathcal{V}_{\text{tr}} \wedge w_t > \tau\}$. Note that $\mathcal{V}_{\text{c}} \subset \mathcal{V}_{\text{tr}}$. Using GMM to fit the distribution allows for a more flexible and accurate selection process than setting a fixed threshold, as it can adapt to different prototypical consistency distributions and provide the probability of each sample being clean.

3.3 Weighted Prototype Updating

The canonical way for updating prototypes is to take the simple average of representations from the same class after each training iteration. However, mislabeled samples may lead to the inclusion of representations that are not truly from the same class, thereby compromising prototype quality.

To achieve robust and accurate prototypes, we propose using a weighted average of selected clean sample representations. Specifically, the prototype updates are performed as follows:

$$\mathbf{c}_\ell \leftarrow \gamma \mathbf{c}_\ell + (1-\gamma)\frac{\sum_{v_j \in \mathcal{V}_{\text{c}}^\ell} w_j \mathbf{z}_j}{\sum_{v_j \in \mathcal{V}_{\text{c}}^\ell} w_j}, \tag{6}$$

where $\mathcal{V}_{\text{c}}^\ell = \{v_i : v_i \in \mathcal{V}_{\text{c}} \wedge y_i = \ell\}$ denotes the set of identified clean training nodes with label ℓ. w_i represents the clean probability of node v_i, and $\gamma \in (0,1)$ is the prototype momentum factor. For each class, the average of filtered clean samples, weighted by clean probabilities, is computed. Subsequently, prototypes are updated using the exponential moving average method. Both the representations and the updated prototypes are normalized to the unit sphere.

Updating prototypes by computing the weighted average of selected clean node representations mitigates the impact of mislabeled nodes, thereby enhancing prototype quality. This enhanced prototype quality subsequently improves the accuracy of pseudo-labeling and label consistency estimation. As label consistency estimation becomes more precise, the GMM more effectively identifies noisy labels, resulting in cleaner samples for future prototype updates and establishing a positive feedback loop.

3.4 Robust Representation Learning

Noisy labels can compromise the quality of representations learned by the network [Wang *et al.*, 2024; Yi *et al.*, 2022; Li *et al.*, 2022], resulting in suboptimal prototypes and inaccurate sample selection. To mitigate the impact of noisy labels and achieve robust node representations, ProCon incorporates a self-supervised learning module.

Drawing inspiration from bootstrap self-supervised learning techniques [Grill *et al.*, 2020; Thakoor *et al.*, 2022], this module enhances node representations by predicting alternative augmentations of the input. Two encoders, namely an online encoder and a target encoder, are employed to obtain node representations from different augmentations. Specifically, for the graph $\mathcal{G}$ with node feature matrix $\mathbf{X}$ and adjacency matrix $\mathbf{A}$, two independent random graph augmentations are applied to $\mathcal{G}$, generating two views $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$. The augmented feature matrices are denoted as $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(2)}$, with the corresponding adjacency matrices $\mathbf{A}^{(1)}$ and $\mathbf{A}^{(2)}$. Besides the encoder $f(\cdot,\cdot;\boldsymbol{\theta})$ serving as the online encoder, this module introduces a target encoder $\tilde{f}(\cdot,\cdot;\tilde{\boldsymbol{\theta}})$ sharing the same architecture. The two different views $\mathcal{G}^{(1)}$ and $\mathcal{G}^{(2)}$ are processed through the online encoder and target encoder, respectively, yielding node representation matrices node representation matrices $\mathbf{Z}^{(1)} = f(\mathbf{X}^{(1)}, \mathbf{A}^{(1)}; \boldsymbol{\theta})$ and $\mathbf{Z}^{(2)} = \tilde{f}(\mathbf{X}^{(2)}, \mathbf{A}^{(2)}; \tilde{\boldsymbol{\theta}})$. The online representation $\mathbf{Z}^{(1)}$ is subsequently fed into a predictor network f_p, producing predictions $\mathbf{H}^{(p)} = f_p(\mathbf{Z}^{(1)}; \boldsymbol{\theta}_p)$ for the target representation $\mathbf{Z}^{(2)}$. The loss function for updating the online encoder and predictor is defined as the mean negative cosine similarity between $\mathbf{Z}^{(2)}$ and $\mathbf{H}^{(p)}$ for each node:

$$\mathcal{L}_{\mathrm{U}} = -\frac{1}{|\mathcal{V}|} \sum_{v_i \in \mathcal{V}} \frac{\mathbf{H}_i^{(p)} \cdot \mathbf{Z}_i^{(2)}}{\|\mathbf{H}_i^{(p)}\| \|\mathbf{Z}_i^{(2)}\|}, \tag{7}$$

where vector $\mathbf{H}_i^{(p)}$ and $\mathbf{Z}_i^{(2)}$ represent the i-th components of the matrices $\mathbf{H}^{(p)}$ and $\mathbf{Z}^{(2)}$, respectively. The target encoder parameters $\tilde{\boldsymbol{\theta}}$ are updated as the exponential moving average of the online encoder parameters $\boldsymbol{\theta}$ with a decay rate μ:

$$\tilde{\boldsymbol{\theta}} \leftarrow \mu \tilde{\boldsymbol{\theta}} + (1-\mu)\boldsymbol{\theta}. \tag{8}$$

The self-supervised learning module functions independently of noisy labels during training, thus mitigating the detrimental effects of noisy labels on node representations. Additionally, this module utilizes unlabeled nodes, facilitating the encoder in learning superior node representations.

3.5 Model Training

The training regime within our framework is partitioned into two modules. The first module entails supervised learning with only the selected clean training nodes, whereas the second module involves self-supervised learning encompassing all nodes. Both modules use the same GNN encoder f. For the supervised learning module, the cross entropy loss function is used, defined as follows:

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{|\mathcal{V}_{\mathrm{c}}|} \sum_{v_i \in \mathcal{V}_{\mathrm{c}}} \sum_{\ell \in [L]} y_i^{\ell} \log \frac{e^{g^{\ell}(\mathbf{z}_i;\boldsymbol{\phi})}}{\sum_{j=1}^{L} e^{g^{j}(\mathbf{z}_i;\boldsymbol{\phi})}}, \tag{9}$$

where $g(\cdot;\boldsymbol{\phi})$ is a linear classifier parameterized by $\boldsymbol{\phi}$, which takes a node representation as input and produces a logit vector. The term y_i^{ℓ} represents the ℓ-th component of the one-hot encoded label y_i, while $g^{\ell}(\mathbf{z}_i;\boldsymbol{\phi})$ indicates the ℓ-th component of vector $g(\mathbf{z}_i;\boldsymbol{\phi})$.

For the self-supervised learning module, the loss function $\mathcal{L}_{\mathrm{U}}$ is defined in Eq. (7). The overall loss function integrates the cross entropy loss and the self-supervised loss as a weighted average:

$$\mathcal{L} = \lambda \mathcal{L}_{\mathrm{CE}} + (1-\lambda)\mathcal{L}_{\mathrm{U}}. \tag{10}$$

The parameter $\lambda \in (0,1)$ operates as the loss balance factor, regulating the trade-off between representation learning and classification. The pseudo code of ProCon is presented in the Appendix A, with an overview depicted in Figure 2.

4 Experiments

4.1 Setup

Datasets and Settings. We conduct experiments on Cora [McCallum *et al.*, 2000], CiteSeer [Giles *et al.*, 1998], and WikiCS [Mernyei and Cangea, 2020] datasets. More experimental results from additional datasets and dataset statistics are detailed in Appendix D and B. For all datasets, 10% of the nodes are randomly selected for training, 10% for validation, and the rest for testing. Following previous works [Dai *et al.*, 2021; Du *et al.*, 2023], labels of training and validation sets are randomly corrupted in the following two ways:

1) Symmetric-p (Sym-p): Labels have a probability p of being uniformly flipped to any other class.

2) Asymmetric-p (Asym-p): Labels have a probability p of being non-uniformly flipped to specific other classes, reflecting common error patterns where some classes are more likely mislabeled as similar ones.

Details of the noise synthesis process are provided in Appendix C. The probability p is also referred to as the 'noise rate'. We set $p \in \{0.2, 0.4, 0.6\}$ for symmetric noise and $p \in \{0.2, 0.3, 0.4\}$ for asymmetric noise.

Implementation details including model architectures and hyperparameters are provided in Appendix E. All hyperparameters are tuned on validation sets. Each experiment is repeated 10 times with different random seeds. The means and standard deviations are reported.

Baselines. We compare ProCon with several state-of-the-art noisy label learning methods to evaluate its effectiveness. General noisy label learning methods compared include Me-momentum [Bai and Liu, 2021], Co-teaching+ [Yu *et al.*, 2019] and ProMix [Xiao *et al.*, 2023], which use the memorization effect, AUM [Pleiss *et al.*, 2020] based on training dynamics, and TopoFilter [Wu *et al.*, 2020] leveraging representation space. Additionally, ProCon is compared against noisy label learning approaches for graph data, including NRGNN [Dai *et al.*, 2021], PI-GNN [Du *et al.*, 2023], RTGNN [Qian *et al.*, 2023] and TSS [Wu *et al.*, 2024].

4.2 Main Results

Table 1 reports the experimental results under different noise settings for the Cora, CiteSeer, and WikiCS datasets. ProCon consistently exhibites superior performance across diverse datasets and noise levels. For the Cora dataset, ProCon achieves an average improvement of 3.25% under symmetric noise and 3.51% under asymmetric noise compared to the best baseline. On the WikiCS dataset, ProCon shows an average improvement of 3.63% under symmetric noise and 4.88%

Dataset	Method	Symmetric			Asymmetric		
		$p = 0.2$	$p = 0.4$	$p = 0.6$	$p = 0.2$	$p = 0.3$	$p = 0.4$
Cora	Cross Entropy	76.35±1.25	69.41±1.26	61.86±2.31	75.41±1.63	68.43±3.20	59.20±6.87
	Me-momentum	78.79±0.97	70.86±1.32	62.38±2.53	74.76±0.88	67.12±4.10	60.74±7.33
	Co-teaching+	76.30±0.90	69.42±1.02	61.75±2.01	77.87±1.53	67.80±3.35	58.52±8.04
	TopoFilter	76.55±0.84	71.55±1.58	63.28±2.89	72.33±2.06	66.62±4.26	59.34±6.60
	ProMix	75.71±1.67	69.52±0.93	60.14±2.13	74.81±1.97	65.48±3.02	60.32±6.17
	NRGNN	77.76±0.73	73.22±1.30	63.57±3.02	75.59±1.30	70.49±3.57	63.26±6.72
	RTGNN	78.37±0.93	71.49±1.93	63.33±3.47	77.93±0.71	71.55±4.04	64.29±5.23
	PI-GNN	77.16±1.20	72.10±1.00	65.65±3.18	76.30±0.92	68.42±1.51	62.39±7.01
	TSS	79.24±1.02	73.08±1.82	64.90±2.94	77.81±1.28	70.27±5.07	63.08±7.87
	ProCon	**79.38±1.46**	**77.93±1.97**	**70.55±3.55**	**78.78±1.83**	**76.61±4.35**	**68.91±7.24**
CiteSeer	Cross Entropy	65.62±1.93	59.05±1.67	52.83±2.15	61.53±1.65	58.44±4.56	52.29±1.03
	Me-momentum	67.12±1.31	60.73±2.24	53.85±2.83	63.57±2.03	58.01±4.73	48.70±2.61
	Co-teaching+	66.91±1.90	58.43±2.21	51.73±2.75	62.64±2.07	61.54±5.67	54.58±1.31
	TopoFilter	65.98±0.91	60.33±1.74	52.69±2.80	58.08±2.18	59.33±3.32	54.37±1.65
	ProMix	65.01±1.51	60.84±1.13	53.75±3.51	62.45±1.75	57.65±4.23	52.55±2.87
	NRGNN	67.09±1.86	60.19±1.62	54.51±1.87	63.37±1.77	57.36±2.89	53.02±2.31
	RTGNN	66.93±1.38	63.13±2.85	53.71±3.27	64.56±0.98	59.94±3.77	55.86±2.56
	PI-GNN	66.45±1.95	61.91±3.56	55.52±3.34	66.41±2.00	62.19±4.26	56.45±2.13
	TSS	65.55±1.97	62.07±2.54	57.66±3.58	62.87±2.18	60.56±3.37	56.13±1.80
	ProCon	**68.56±1.29**	**67.40±1.59**	**62.84±3.27**	**69.00±1.34**	**66.23±4.30**	**64.46±2.86**
WikiCS	Cross Entropy	76.24±1.04	71.29±1.08	60.32±1.74	74.11±1.95	67.28±2.92	56.07±5.54
	Me-momentum	75.72±0.78	73.17±0.97	59.01±1.43	74.61±1.31	66.37±1.17	56.99±4.04
	Co-teaching+	76.06±1.22	71.58±1.63	61.53±1.00	73.08±0.85	66.93±2.11	57.42±5.17
	TopoFilter	76.60±0.26	71.28±1.02	60.39±0.96	72.85±1.12	68.41±1.86	56.71±5.05
	ProMix	75.34±1.43	71.21±0.85	59.75±1.53	72.95±1.72	66.51±1.79	56.64±6.01
	NRGNN	75.90±1.11	74.72±0.76	63.53±1.67	74.96±1.67	70.31±2.31	59.79±6.65
	RTGNN	76.02±0.64	73.26±0.05	63.83±2.33	75.68±0.84	71.64±1.49	59.24±5.10
	PI-GNN	77.46±1.78	73.47±0.22	61.20±1.87	74.37±1.29	69.72±1.23	58.85±6.51
	TSS	77.73±0.66	74.12±0.37	63.77±1.28	75.26±0.75	70.14±2.84	59.15±5.70
	ProCon	**78.48±0.58**	**75.74±1.10**	**70.33±1.79**	**76.75±1.58**	**72.26±2.34**	**64.81±5.23**

Table 1: Node classification accuracy (%, mean ± std) under symmetric and asymmetric noise on Cora, CiteSeer and WikiCS datasets. p denotes the noise rate of each noise setting. Best in **bold**.

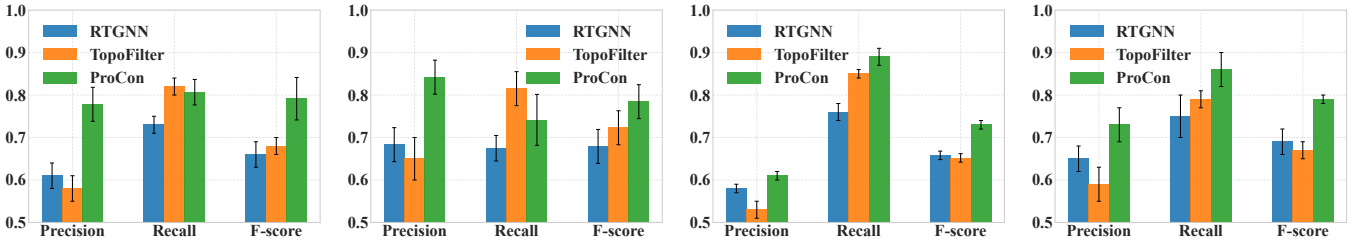
under asymmetric noise settings compared to the best baseline. Under high noise scenarios, our method demonstrates remarkable performance gains over other baseline methods. For example, with an asymmetric noise rate of 0.4, ProCon surpasses the vanilla Cross Entropy method by 8.74% on the WikiCS dataset, while the best baseline improves by only 3.72%. Additionally, graph-specific baseline methods consistently outperforms general ones, emphasizing the advantage of utilizing graph properties in noisy label learning. This partly elucidates the superior performance of ProCon, as it harnesses graph structure to identify mislabeled nodes.

Sample Selection Performance. Figure 3 illustrates the sample selection efficacy of ProCon. We compare ProCon with two alternative baselines: 1) RTGNN [Qian *et al.*, 2023], a graph learning approach that utilizes loss distributions for sample selection, and 2) TopoFilter [Wu *et al.*, 2020], a general sample selection method that leverages representational patterns. ProCon integrates label consistency with feature similarity, representation, and graph adjacency for sample selection. Performance is evaluated using three metrics: Precision, Recall, and F-score, where the F-score is calculated as the harmonic mean of Precision and Recall. Overall, ProCon demonstrates a consistent improvement over the other baselines. For example, with a Sym-0.6 noise on the Cora dataset, the F-score of ProCon surpasses that of TopoFilter by 11%.

4.3 Ablation and Analysis

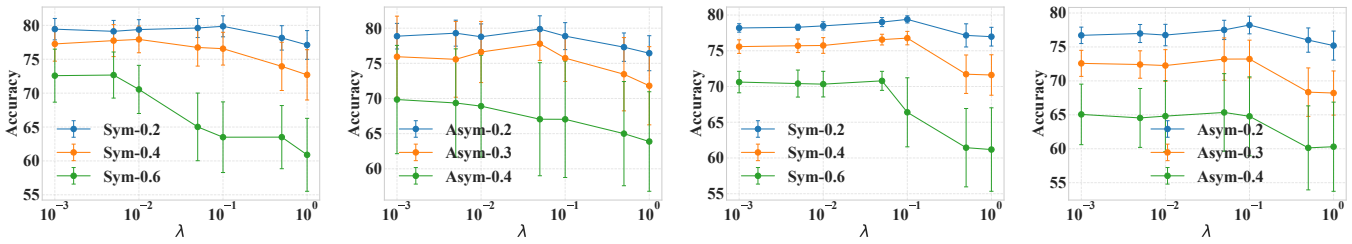
Effect of Sample Selection. We conduct ablation studies on the sample selection module of ProCon. Specifically, ProCon is compared with two variants: (1) No Selection: ProCon without the sample selection module, where the entire training dataset is used for model updating. (2) ProCon-C: Employing labels predicted by the classifier to determine label consistency distribution. Figure 5 illustrates the accuracy comparison between ProCon and two variants on Cora and WikiCS dataset. Both ProCon-C and ProCon outperform the No Selection variant. This indicates that merely enhancing node representations through the self-supervised module is insufficient for robustness, and sample selection is crucial. Additionally, ProCon outperforms ProCon-C substantially (e.g., +7.8% on WikiCS with an asymmetric noise rate of 0.4), highlighting the effectiveness of using representations for sample selection.

Effect of Self-Supervised Learning and Loss Balance Factor λ. Figure 4 illustrates the performance of ProCon across the Cora and WikiCS datasets as the loss balance factor λ is varied. We evaluated the accuracy of ProCon for $\lambda \in \{1\text{e}{-3}, 5\text{e}{-3}, 1\text{e}{-2}, 5\text{e}{-2}, 1\text{e}{-1}, 5\text{e}{-1}, 1\}$. A larger λ signifies an increased emphasis on supervised learning with noisy labels. It is noteworthy that $\lambda = 1$ excludes the self-supervised module from ProCon. As λ increases, the per-



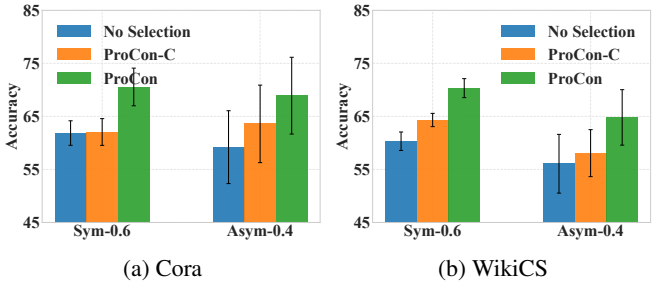
(a) Cora, Sym-0.6 (b) Cora, Asym-0.4 (c) WikiCS, Sym-0.6 (d) WikiCS, Asym-0.4

Figure 3: Sample selection performace on Cora and WikiCS.



(a) Cora, Symmetric (b) Cora, Asymmetric (c) WikiCS, Symmetric (d) WikiCS, Asymmetric

Figure 4: Ablation on loss balance factor λ with a logarithmic x-axis.



(a) Cora (b) WikiCS

Figure 5: Ablation on sample selection.

Dataset	Variant	Symmetric		Asymmetric
		$p = 0.4$	$p = 0.6$	$p = 0.4$
Cora	$\mathcal{V}_{\text{tr}}$ S-AVG	73.81±1.59	64.27±2.89	65.38±7.46
	$\mathcal{V}_{\text{c}}$ S-AVG	74.73±1.68	68.92±3.47	68.06±7.97
	$\mathcal{V}_{\text{c}}$ W-AVG	77.93±1.97	70.55±3.55	68.91±7.24
WikiCS	$\mathcal{V}_{\text{tr}}$ S-AVG	74.97±1.65	68.86±1.57	60.89±6.42
	$\mathcal{V}_{\text{c}}$ S-AVG	74.73±1.15	69.64±1.86	63.38±5.08
	$\mathcal{V}_{\text{c}}$ W-AVG	75.74±1.10	70.33±1.79	64.81±5.23

Table 2: Ablation on prototype updating. Node classification accuracy is reported under different prototype updating strategies.

formance of ProCon generally deteriorates, with marked declines observed at $\lambda = 5\text{e}{-1}$ and 1. This trend highlights the detrimental impact of noisy labels and accentuates the necessity of self-supervised learning. Furthermore, for $\lambda < 1\text{e}{-2}$, ProCon's performance remains stable, indicating that smaller λ values do not significantly affect the results, thereby simplifying hyperparameter tuning. Consequently, we fix λ at $1\text{e}{-2}$ across all datasets and noise settings.

Effect of Weighted Prototype Updating. We validate the effectiveness of weighted prototype updating. The following variants are considered: 1) $\mathcal{V}_{\text{tr}}$ S-AVG, which applies the simple average of all training samples, equivalent to setting $w_j = 1, \tau < 0$, and $\mathcal{V}_c^k = \mathcal{V}_{\text{tr}}^k$ in Eq. (6). 2) $\mathcal{V}_{\text{c}}$ S-AVG, which utilizes the simple average of samples in the clean node set, equivalent to setting $w_j = 1$ in Eq. (6). 3) $\mathcal{V}_{\text{tr}}$ W-AVG, the prototype updating method of ProCon. Table 2 summarizes the results on Cora and WikiCS dataset. The $\mathcal{V}_{\text{c}}$ S-AVG variant notably outperforms the $\mathcal{V}_{\text{tr}}$ S-AVG variant (e.g. +2.7% on Cora and +2.5% on WikiCS under asymmetric noise with a noise ratio of 0.4), highlighting the effectiveness of prototype updates utilizing clean nodes. Furthermore, the $\mathcal{V}_{\text{tr}}$ W-AVG variant improves upon the $\mathcal{V}_{\text{c}}$ S-AVG variant, demonstrating the benefits of weighting samples.

5 Conclusion

In this paper, we introduce ProCon, a novel framework designed to enhance the robustness of Graph Neural Networks (GNNs) against label noise. ProCon identifies mislabeled nodes by leveraging peer-informed label consistency, selecting semantically similar peers based on feature similarity and graph adjacency. We assess label consistency using pseudo-labels derived from representational prototypes and employ a Gaussian Mixture Model to distinguish clean samples from mislabeled ones. Extensive experiments across multiple datasets demonstrate that ProCon consistently outperforms state-of-the-art methods, effectively mitigating label noise and strengthening GNN robustness. Future work will explore applying this method to other areas, such as fairness, partial label learning, and graph anomaly detection.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 62402315, 62232011, and 61932014, and the Shanghai Science and Technology Innovation Action Plan Grant 24BC3201200.

References

[Altenburger and Ugander, 2018] Kristen M. Altenburger and Johan Ugander. Monophily in social networks introduces similarity among friends-of-friends. *Nature Human Behaviour*, pages 284–290, 2018.

[Arpit *et al.*, 2017] Devansh Arpit, Stanislaw Jastrzkebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *Proc. of ICML*, pages 233–242, 2017.

[Bai and Liu, 2021] Yingbin Bai and Tongliang Liu. Me-momentum: Extracting hard confident examples from noisily labeled data. In *Proc. of CVPR*, pages 9292–9301, 2021.

[Chen *et al.*, 2020] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proc. of AAAI*, pages 3438–3445, 2020.

[Chen *et al.*, 2024] Ling-Hao Chen, Yuanshuo Zhang, Taohua Huang, Liangcai Su, Zeyi Lin, Xi Xiao, Xiaobo Xia, and Tongliang Liu. Erase: Error-resilient representation learning on graphs for label noise tolerance. In *Proc. of CIKM*, pages 270–280, 2024.

[Dai *et al.*, 2021] Enyan Dai, Charu Aggarwal, and Suhang Wang. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proc. of KDD*, pages 227–236, 2021.

[Dai *et al.*, 2022] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proc. of WSDM*, pages 181–191, 2022.

[Du *et al.*, 2023] Xuefeng Du, Tian Bian, Yu Rong, Bo Han, Tongliang Liu, Tingyang Xu, Wenbing Huang, Yixuan Li, and Junzhou Huang. Noise-robust graph learning by estimating and leveraging pairwise interactions. *Transactions on Machine Learning Research*, 2023.

[Giles *et al.*, 1998] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. Citeseer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries*, pages 89–98, 1998.

[Goldberger and Ben-Reuven, 2017] Jacob Goldberger and Ehud Ben-Reuven. Training deep neural-networks using a noise adaptation layer. In *ICLR*, 2017.

[Grill *et al.*, 2020] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *Proc. of NeurIPS*, pages 21271–21284, 2020.

[Han *et al.*, 2018] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor W. Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proc. of NeurIPS*, pages 8536–8546, 2018.

[Lee *et al.*, 2019] Kimin Lee, Sukmin Yun, Kibok Lee, Honglak Lee, Bo Li, and Jinwoo Shin. Robust inference via generative classifiers for handling noisy labels. In *Proc. of ICML*, pages 3763–3772, 2019.

[Li *et al.*, 2020] Junnan Li, Richard Socher, and Steven C. H. Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *ICLR*, 2020.

[Li *et al.*, 2021] Yayong Li, Jie Yin, and Ling Chen. Unified robust training for graph neural networks against label noise. In *Proc. of KDD*, pages 528–540, 2021.

[Li *et al.*, 2022] Shikun Li, Xiaobo Xia, Shiming Ge, and Tongliang Liu. Selective-supervised contrastive learning with noisy labels. In *Proc. of CVPR*, pages 316–325, 2022.

[Malach and Shalev-Shwartz, 2017] Eran Malach and Shai Shalev-Shwartz. Decoupling "when to update" from "how to update". In *Proc. of NeurIPS*, pages 960–970, 2017.

[McCallum *et al.*, 2000] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, pages 127–163, 2000.

[McPherson *et al.*, 2001] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, pages 415–444, 2001.

[Menon *et al.*, 2020] Aditya Krishna Menon, Ankit Singh Rawat, Sashank J. Reddi, and Sanjiv Kumar. Can gradient clipping mitigate label noise? In *ICLR*, 2020.

[Mernyei and Cangea, 2020] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901*, 2020.

[Papyan *et al.*, 2020] Vardan Papyan, X. Y. Han, and David L. Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proc. of the National Academy of Sciences*, pages 24652–24663, 2020.

[Pleiss *et al.*, 2020] Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q. Weinberger. Identifying mislabeled data using the area under the margin ranking. In *Proc. of NeurIPS*, pages 17044–17056, 2020.

[Qian *et al.*, 2023] Siyi Qian, Haochao Ying, Renjun Hu, Jingbo Zhou, Jintai Chen, Danny Z. Chen, and Jian Wu. Robust training of graph neural networks via noise governance. In *Proc. of WSDM*, pages 607–615, 2023.

[Sun *et al.*, 2022] Jiawei Sun, Jie Li, Chentao Wu, Zili Tang, and Celimuge Wu. Ada-stnet: A dynamic adaboost spatio-temporal network for traffic flow prediction. In *Proc. of ICASSP*, pages 5478–5482, 2022.

[Sun *et al.*, 2024] Jiawei Sun, Kailai Li, Ruoxin Chen, Jie LI, Chentao Wu, Yue Ding, and Junchi Yan. InterpGNN: Understand and improve generalization ability of transdutive GNNs through the lens of interplay between train and test nodes. In *ICLR*, 2024.

[Tarvainen and Valpola, 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proc. of NeurIPS*, pages 1195–1204, 2017.

[Thakoor *et al.*, 2022] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *ICLR*, 2022.

[Wang *et al.*, 2024] Haobo Wang, Ruixuan Xiao, Yixuan Li, Lei Feng, Gang Niu, Gang Chen, and Junbo Zhao. Pico+: Contrastive label disambiguation for robust partial label learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 3183–3198, 2024.

[Wei *et al.*, 2021] Hongxin Wei, Lue Tao, Renchunzi Xie, and Bo An. Open-set label noise can improve robustness against inherent label noise. In *Proc. of NeurIPS*, pages 7978–7992, 2021.

[Wieder *et al.*, 2020] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, pages 1–12, 2020.

[Wu *et al.*, 2020] Pengxiang Wu, Songzhu Zheng, Mayank Goswami, Dimitris Metaxas, and Chao Chen. A topological filter for learning with label noise. In *Proc. of NeurIPS*, pages 21382–21393, 2020.

[Wu *et al.*, 2022] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, pages 1–37, 2022.

[Wu *et al.*, 2024] Yuhao Wu, Jiangchao Yao, Xiaobo Xia, Jun Yu, Ruxin Wang, Bo Han, and Tongliang Liu. Mitigating label noise on graphs via topological sample selection. In *Proc. of ICML*, pages 53944–53972, 2024.

[Xia *et al.*, 2024] Jun Xia, Haitao Lin, Yongjie Xu, Cheng Tan, Lirong Wu, Siyuan Li, and Stan Z. Li. Gnn cleaner: Label cleaner for graph structured data. *IEEE Transactions on Knowledge and Data Engineering*, pages 640–651, 2024.

[Xiao *et al.*, 2015] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *Proc. of CVPR*, pages 2691–2699, 2015.

[Xiao *et al.*, 2023] Ruixuan Xiao, Yiwen Dong, Haobo Wang, Lei Feng, Runze Wu, Gang Chen, and Junbo Zhao. Promix: combating label noise via maximizing clean sample utility. In *Proc. of IJCAI*, pages 4442–4450, 2023.

[Yi and Wu, 2019] Kun Yi and Jianxin Wu. Probabilistic end-to-end noise correction for learning with noisy labels. In *Proc. of CVPR*, pages 7017–7025, 2019.

[Yi *et al.*, 2022] Li Yi, Sheng Liu, Qi She, A. Ian McLeod, and Boyu Wang. On learning contrastive representations for learning with noisy labels. In *Proc. of CVPR*, pages 16682–16691, 2022.

[Yu *et al.*, 2019] Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor W. Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *Proc. of ICML*, pages 7164–7173, 2019.

[Yuan *et al.*, 2023a] Jingyang Yuan, Xiao Luo, Yifang Qin, Zhengyang Mao, Wei Ju, and Ming Zhang. Alex: Towards effective graph transfer learning with noisy labels. In *Proc. of MM*, pages 3647–3656, 2023.

[Yuan *et al.*, 2023b] Jingyang Yuan, Xiao Luo, Yifang Qin, Yusheng Zhao, Wei Ju, and Ming Zhang. Learning on graphs under label noise. In *Proc. of ICASSP*, pages 1–5, 2023.

[Zhang and Sabuncu, 2018] Zhilu Zhang and Mert R. Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Proc. of NeurIPS*, pages 8792–8802, 2018.

[Zhang *et al.*, 2021] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, pages 107–115, 2021.

[Zhou *et al.*, 2023] Zhanke Zhou, Jiangchao Yao, Jiaxu Liu, Xiawei Guo, Quanming Yao, Li He, Liang Wang, Bo Zheng, and Bo Han. Combating bilateral edge noise for robust link prediction. In *Proc. of NeurIPS*, pages 21368–21414, 2023.

[Zhu *et al.*, 2021] Zhihui Zhu, Tianyu Ding, Jinxin Zhou, Xiao Li, Chong You, Jeremias Sulam, and Qing Qu. A geometric analysis of neural collapse with unconstrained features. In *Proc. of NeurIPS*, pages 29820–29834, 2021.

[Zhu *et al.*, 2022] Zhaowei Zhu, Zihao Dong, and Yang Liu. Detecting corrupted labels without training a model to predict. In *Proc. of ICML*, pages 27412–27427, 2022.

[Zhu *et al.*, 2024] Yonghua Zhu, Lei Feng, Zhenyun Deng, Yang Chen, Robert Amor, and Michael Witbrock. Robust node classification on graph data with graph and label noise. In *Proc. of AAAI*, pages 17220–17227, 2024.