GraphCleaner: Detecting Mislabelled Samples in Popular Graph Learning Benchmarks

Yuwen Li [1] **Xiong Miao** [2] **Bryan Hooi** [1 2]

Abstract

Label errors have been found to be prevalent in popular text, vision, and audio datasets, which heavily influence the safe development and evaluation of machine learning algorithms. Despite increasing efforts towards improving the quality of generic data types, such as images and texts, the problem of mislabel detection in graph data remains underexplored. To bridge the gap, we explore mislabelling issues in popular real-world graph datasets and propose GRAPHCLEANER, a post-hoc method to detect and correct these mislabelled nodes in graph datasets. GRAPHCLEANER combines the novel ideas of 1) *Synthetic Mislabel Dataset Generation*, which seeks to generate realistic mislabels; and 2) *Neighborhood-Aware Mislabel Detection*, where neighborhood dependency is exploited in both labels and base classifier predictions. Empirical evaluations on 6 datasets and 6 experimental settings demonstrate that GRAPHCLEANER outperforms the closest baseline, with an average improvement of 0.14 in F1 score, and 0.16 in MCC. On real-data case studies, GRAPHCLEANER detects real and previously unknown mislabels in popular graph benchmarks: `PubMed`, `Cora`, `CiteSeer` and `OGB-arxiv`; we find that at least 6.91% of `PubMed` data is mislabelled or ambiguous, and simply removing these mislabelled data can boost evaluation performance from 86.71% to 89.11%[1].

1. Introduction

Data is the primary input to any AI system, both for learning and evaluation. Hence, recognizing *data quality* as a critically important factor for the success of AI systems, *data-centric AI* has rapidly emerged in recent years. The vision of data-centric AI is to ensure clean and high-quality data in all phases of the project life-cycle and to develop tools and processes for monitoring and improving data quality. In particular, in supervised learning contexts, correcting label noise, including mislabelled and ambiguously labelled data, is of high priority due to the key importance of labels in the training and evaluation process.

Recent work (Northcutt et al., 2021b) has shown that label errors are prevalent across machine learning benchmarks: they find a label error rate of 3.4% on average, across 10 of the most popular natural language processing, computer vision, and audio datasets. However, label errors remain unexplored in the *graph* setting. Graphs are widely used for representing entities and the relationships between them, with numerous applications such as molecules, financial networks, and social networks (Wu et al., 2020). In this paper, we focus on the node classification setting, one of the most common graph learning tasks, and aim to answer the questions: *are mislabels also common in this setting*, and *can they be automatically detected and corrected*?

Existing solutions for mislabel detection (Arazo et al., 2019; Pleiss et al., 2020; Northcutt et al., 2021a; Zhu et al., 2021) are designed for machine learning on generic data, such as images, audio, and text, where instances are largely seen as being independent of one another. However, this means that when used in a graph setting, these methods do not effectively exploit the close *neighbor-dependence* between nodes and their neighbors, which is a key characteristic of graph data. In contrast, a key idea of our approach is that strong violations in the dependency patterns between a node and its neighbors act as an important signal that the node is more likely to be mislabelled.

Hence, in this work, we propose GRAPHCLEANER, a post-hoc framework for detecting and correcting mislabelled nodes on graph datasets. To do this, GRAPHCLEANER introduces the novel ideas of *1) Synthetic Mislabel Dataset Generation*, where we first estimate a 'mislabel transition matrix' describing patterns of how samples of different classes tend to be mislabelled, then use this transition matrix to synthesize realistically mislabelled samples in a class-

[1]School of Computing, National University of Singapore, Singapore [2]Institute of Data Science, National University of Singapore, Singapore. Correspondence to: Yuwen Li <yuwenli@u.nus.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

[1]Corrected datasets and code are available at https://github.com/lywww/GraphCleaner/tree/master.

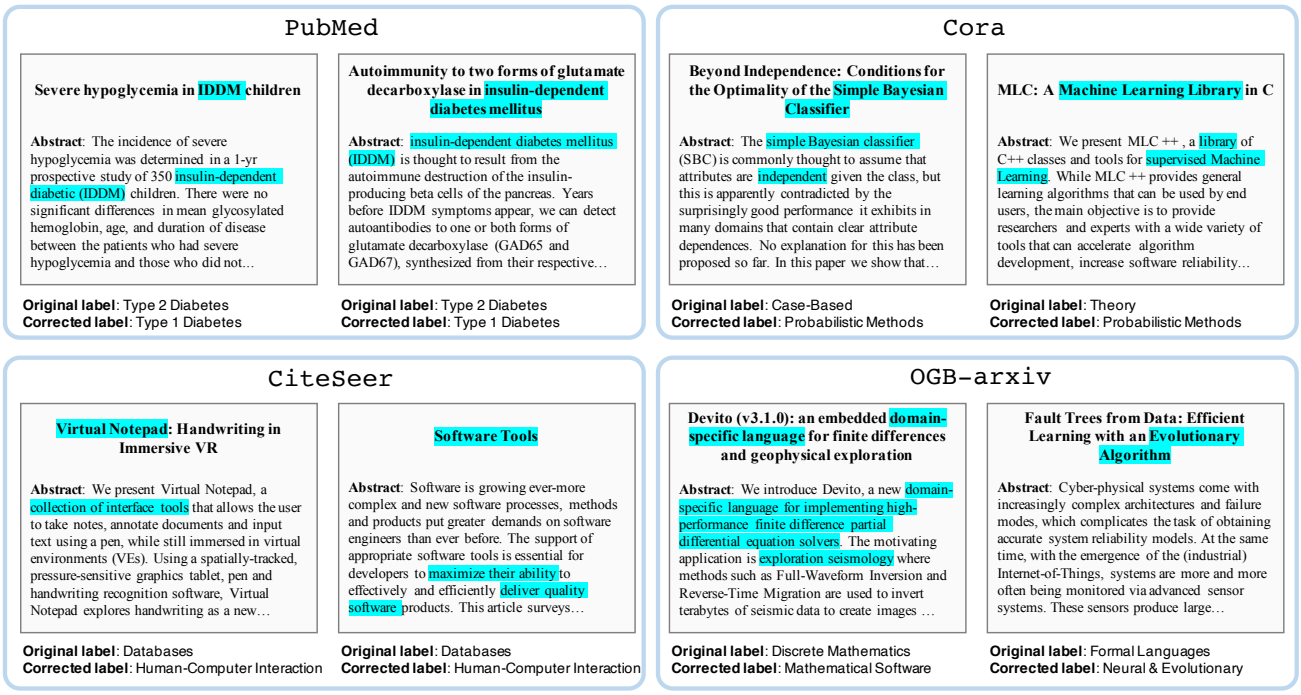

Figure 1. Examples of mislabelled samples detected by our approach in PubMed, Cora, CiteSeer, and OGB-arxiv respectively. Below each case, we give the original label given in the dataset, along with the corrected label suggested by our algorithm. We verify that all these cases are indeed mislabels, and highlight in blue the text that provides evidence for this. For PubMed, note that Insulin-Dependent Diabetes Mellitus (or IDDM) is synonymous with Type 1 Diabetes.

dependent way. Next, these synthesized mislabelled samples are used as negative samples to train our *2) Neighborhood-Aware Mislabel Detector* component, which is a binary classifier taking as input both the observed labels and the base classifier predictions, in a node's neighborhood. In this way, GRAPHCLEANER exploits neighbor-dependence in graphs by learning to distinguish between the neighborhoods of correctly labelled and mislabelled nodes.

To evaluate its real-world utility and understand the implications of label errors, we conduct detailed case studies. On PubMed, we find that at least 6.91% of the data is mislabelled or ambiguous, and simply removing these mislabelled data boosts evaluation performance from 86.71% to 89.11%. This validates the importance of data quality for performance and evaluation, and suggests that correcting mislabels has scope for significant value. Then, as shown in Figure 1, we apply our method to four widely used datasets to detect mislabels. To show the labels suggested by our algorithm are indeed correct, we verify in each case and provide evidence as highlighted in blue. Automatically detecting such samples can greatly speed up the efficiency of human manual checking in finding and correcting mislabels, and hence is our main goal in this work.

Our contributions can be summarized as follows:

- We propose GRAPHCLEANER to detect mislabels in graph data, and prove the theoretical guarantees regarding the mislabel score threshold. Unlike existing approaches, GRAPHCLEANER exploits graph's *neighbor-dependence* patterns, by introducing a neighborhood-aware mislabel detector and the novel *synthetic mislabel dataset generation* for better training the detector.
- Extensive experiments on 6 graph datasets across 6 experimental settings show that GRAPHCLEANER consistently outperforms state-of-the-art methods by an average margin of 0.14 in F1 score, and 0.16 in MCC.
- Detailed case studies show real and previously unknown mislabels in four datasets, verifying GRAPHCLEANER's effectiveness in real-world applications. At least 6.91% of PubMed data is found to be mislabelled or ambiguous, with significant implications for algorithm evaluation.
- We publicly release 2 improved variants of PubMed dataset: PubMedCleaned and PubMedMulti for more accurate evaluation.

2. Related Work

We first review related explorations in graph neural networks, learning with noisy labels and confidence calibration.

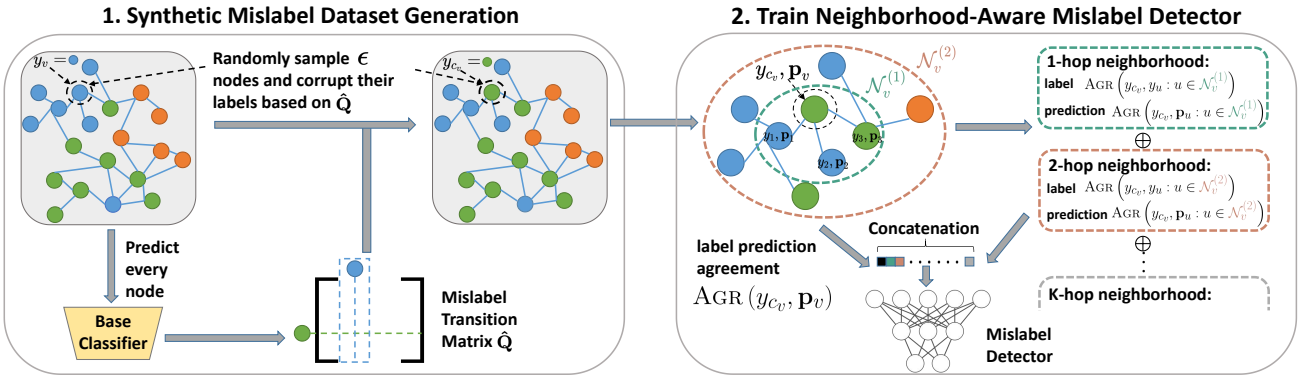

Figure 2. The framework of GRAPHCLEANER. Different colors indicate different class labels. To train a mislabel detector, we first estimate the mislabel transition matrix $\hat{\mathbf{Q}}$ from the predictions of a 'base classifier', sample ϵ-ratio of nodes and flip their classes j to another class i based on the probability of $\hat{\mathbf{Q}}_{\tilde{y}=i|y^*=j}$. Then for every node v, we measure the agreement between v and its neighborhood within K hops. The detailed design for 1-hop neighborhood $\mathcal{N}_v^{(1)}$ and 2-hop neighborhood $\mathcal{N}_v^{(2)}$ is illustrated.

Then we summarize methods for mislabel detection.

Graph Neural Networks Graph neural networks (GNN) have been a powerful tool for graph learning tasks. One line of research similar to ours utilizes label information, such as label propagation (Wang & Leskovec, 2020; Jia & Benson, 2020) and integrating labels with features (Shi et al., 2020). Another line is post-processing methods. Among them, Wang et al. (2021) tackles the graph-based calibration problem and Huang et al. (2020) proposes the C&S procedure. Due to the goal of mislabel detection, our framework differs, e.g., in its use of mislabel generation and neighbor-dependence assumption.

Confidence Calibration In machine learning, reliable uncertainty estimates are crucial for safe decision-making. However, Guo et al. (2017) shows that modern neural networks suffer from *over-confidence* issue, which *Confidence Calibration* aims to address. A well-calibrated model should align the output confidence score with the ground truth accuracy (Lakshminarayanan et al., 2016). Popular methods include temperature scaling (Guo et al., 2017), Dirichlet calibration (Kull et al., 2019), and Gaussian processes (Wenger et al., 2020), etc. Despite the shared interest in making decision-making safe, calibration algorithms adopt the model-centric view, while we focus on improving data quality, and we do not expect a straightforward mapping between confidence score and ground truth accuracy.

Learning with Noisy Labels Training reliable models in the presence of label noise (Natarajan et al., 2013; Algan & Ulusoy, 2021) is a closely related topic to mislabel detection, usually via modified training procedures (Hoang et al., 2019; Dai et al., 2021). Some also explore neighborhood similarity (Zhu et al., 2021b;a), or utilize synthetic mislabels (Xia et al., 2019; Jiang et al., 2021). Our GRAPHCLEANER differs from them in its focus on non-i.i.d graph data, and how it obtains and use transition probabilities. Please refer to Appendix A for specific differences.

Mislabel Detection Existing solutions for mislabel detection are mostly designed for generic data and typically come from three perspectives: training dynamics-based (Arazo et al., 2019; Pleiss et al., 2020), joint distribution estimation of noisy and true labels (Northcutt et al., 2021a), and neighborhood similarity (Zhu et al., 2021). Our method draws inspiration from Northcutt et al. (2021a) when estimating the mislabel transition matrix, but focuses on the graph setting and utilizes the neighborhood dependence property of graphs to deliver a post-hoc, plug-and-play solution.

3. Proposed Framework

3.1. Problem Definition

We aim to detect mislabelled nodes in a graph $G = (\mathcal{V}, \mathbf{X}, \mathbf{A})$, with node set $\mathcal{V}$ containing n nodes, node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ and adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$. We assume that G is undirected for simplicity, but our method can generalize straightforwardly to directed graphs. The unknown true label of node v is denoted by $y_v^* \in [c]$, where $[c] := \{1, 2, \cdots, c\}$ is the class label set; the observed noisy label is denoted by $\tilde{y}_v \in [c]$. The node v is said to be mislabelled if $y_v^* \neq \tilde{y}_v$. The node set $\mathcal{V}$ is partitioned into training, validation, and test sets, denoted by $\mathcal{V}_{\text{train}}$, $\mathcal{V}_{\text{val}}$, and $\mathcal{V}_{\text{test}}$.

Given a graph G, we aim to answer the following two fundamental questions:

- **Identify**: Which nodes are mislabelled?
- **Correct**: What are their unknown true labels y^*?

Post-hoc Setting Our work focuses on the post-hoc setting, where we are given a *pretrained base classifier* $f_\theta(\mathbf{X}, \mathbf{A})$, e.g., any graph neural network used as a node classifier, which is trained on the training set $\mathcal{V}_{\text{train}}$ with the observed noisy labels $\tilde{y}$.

3.2. GRAPHCLEANER: Overview

In order to determine if a sample is mislabelled, we need a good mislabel detector that is able to capture the behavior of mislabels. We specify this detector as a neural network, which leads to the question of how to construct a training dataset representative of the real world. Unlike other problems, such as out-of-distribution detection, where the outlier distribution is unknown, mislabel detection has the advantage that the outlier distribution is well-defined and can be easily simulated by flipping the labels. Motivated by this, we tackle the mislabel detection task through two steps: *generating synthetic mislabel dataset* and *training neighborhood-aware mislabel detector*. The framework is illustrated in Figure 2.

To obtain mislabelled data that captures the characteristics of real mislabels, we first estimate the mislabel transition matrix $\hat{\mathbf{Q}}$ using a base classifier. The mislabel transition matrix is then utilized to generate the balanced corrupted dataset. Next, the *Mislabel Detector* is trained on the corrupted dataset by capturing neighborhood-aware features.

3.3. Synthetic Mislabel Dataset Generation

There is no ground truth about if a sample is mislabelled. To train a mislabel detector, we first need to synthesize mislabelled samples to obtain ground truth labels. Our approach generates synthetic mislabels in a *class-aware* way: the label noise is class-dependent, meaning that the probability of mislabelling a node as class i is dependent on the node's actual class j. For example, an image of a little tiger is more easily mislabelled as a cat than a dog. The overall procedure is summarized in Algorithm 1.

Learning Mislabel Transition Matrix To generate class-dependent label noise, we first learn a *mislabel transition matrix* $\hat{\mathbf{Q}}$ to capture the probability of mislabelling one class label to another class. Specifically, $\hat{\mathbf{Q}}_{\tilde{y}=i|y^*=j}$ denotes the probability of a sample with unknown true class j being mislabelled with an observed class i.

We estimate the mislabel transition matrix using a pretrained base classifier evaluated on validation set data, following the same assumption outlined in Northcutt et al. (2021a). The underlying assumption is that predictions given by the base classifier with *high confidence* are very likely to match the true labels. Therefore, we consider predictions with confidence scores no less than a threshold (see Appendix 5 Equation 4) to be correct. Under this assumption, we count the number of samples with true label j and observed noisy label i as $\mathbf{C}_{\tilde{y}=i,y^*=j}$. This joint distribution can then be transformed into an estimate of the mislabel transition matrix $\hat{\mathbf{Q}}_{\tilde{y}=i|y^*=j}$, by an application of Bayes' theorem. We refer you to Appendix 5 for more details and formal definitions of this process.

Sampling Mislabels To generate a balanced training set for training our mislabel detector, we uniformly sample half of the nodes in the validation set to be synthetic mislabels. We denote the set of sampled nodes as $\mathcal{V}_{\text{synth}} \subseteq \mathcal{V}_{\text{val}}$.

Then, nodes in $\mathcal{V}_{\text{synth}}$ will have their labels randomly flipped according to the mislabel transition matrix: specifically, the probability of changing a node's given label from j to a different label i is $\hat{\mathbf{Q}}_{\tilde{y}=i|y^*=j}$. Effectively, this process simulates realistic label noise, using our best estimate of the distribution of label noise in the original data. We denote the resulting *corrupted label matrix* as $\mathbf{Y}_c \in \mathbb{R}^{n\times c}$, which is the one-hot label matrix after nodes in $\mathcal{V}_{\text{synth}}$ have been flipped according to the above process. For further discussion, see Appendix C.

Algorithm 1 Synthetic Mislabel Dataset Generation

Input: graph $G=(\mathcal{V}, \mathbf{X}, \mathbf{A})$ with $\mathcal{V}=(\mathcal{V}_{train}, \mathcal{V}_{val}, \mathcal{V}_{test})$, base classifier f_θ, sample ratio ϵ.
Output: corrupted graph $G_c = (\mathcal{V}_c, \mathbf{X}, \mathbf{A})$ with nodes in $\mathcal{V}_{\text{synth}}$ having flipped labels
Notations: confident joint $\mathbf{C}_{\tilde{y},y^*}$, threshold t_c for class c, noise transition matrix $\hat{\mathbf{Q}}_{\tilde{y}|y^*}$.

Train f_θ on $\mathcal{V}_{train}$
Calculate $t_c = \frac{1}{|\mathbf{X}_{\tilde{y}=c}|} \sum_{x\in\mathbf{X}_{\tilde{y}=c}} \hat{p}\left(\tilde{y}=c; x, \theta\right)$, where $\hat{p}$ is f_θ's predictions
for v **in** $\mathcal{V}_{val}$ **do**
  $y_v \leftarrow f_\theta(v)$
  $P_v \leftarrow f_\theta$'s softmax prediction on v
  if $y_v \neq \tilde{y}_v$ **and** $P_v(y_v) \geq t_{y_v}$ **then**
    $\mathbf{C}_{\tilde{y}=\tilde{y}_v, y^*=y_v}$ += 1
  end if
end for
$\hat{\mathbf{Q}}_{\tilde{y},y^*} \leftarrow$ normalized $\mathbf{C}_{\tilde{y},y^*}$ ▷ *joint distribution*
$\hat{\mathbf{Q}}_{\tilde{y}|y^*} \leftarrow \hat{\mathbf{Q}}_{\tilde{y},y^*} / \hat{\mathbf{Q}}_{y^*}$ ▷ *conditional probability*
$\mathcal{V}_{\text{synth}} \leftarrow$ sampled data from $\mathcal{V}_{val}$ by ϵ
for v **in** $\mathcal{V}_{\text{synth}}$ **do**
  Flip v's label according to the transition probability distribution $\hat{\mathbf{Q}}_{\tilde{y}|y^*=\tilde{y}_v}$
end for

3.4. Neighborhood-Aware Mislabel Detector

To better detect mislabelled data, we leverage a useful and commonly-held assumption of *neighbor-dependence*: the ground truth label of one node tends to agree with the labels

of its neighbors. That is, if a node strongly disagrees with its neighborhood in terms of labels, the node has a relatively high risk of being mislabelled. In addition, the base classifier's softmax predictions also carry important information about the unknown true labels.

Motivated by these two factors, our mislabel detector intuitively focuses on the *agreement between a sample's label, and the labels and base classifier predictions in its neighborhood*, to decide if the sample is mislabelled. The procedure is summarized in Algorithm 2, and more details can be found in Appendix D.

Algorithm 2 Neighborhood-Aware Mislabel Detector

Input: corrupted graph $G_c = (\mathcal{V}_c, \mathbf{X}, \mathbf{A})$ with $\mathbf{D}$ as its diagonal degree matrix, softmax prediction matrix $\mathbf{P}$, original label matrix $\mathbf{Y}$, corrupted label matrix $\mathbf{Y}_c$, number of hops K.
Output: the trained mislabel detector
Notations: neighborhood agreement features $\mathbf{Z}$

$\tilde{\mathbf{A}} \leftarrow \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
$\mathbf{Z} \leftarrow \textsc{Agr}(\mathbf{Y}_c, \mathbf{P})$
for $k = 1 \rightarrow K$ **do**
  $\mathbf{S}_k \leftarrow \mathsf{zero}(\tilde{\mathbf{A}}^k)$
  $\mathbf{Y}^{(k)} \leftarrow \mathbf{S}_k \cdot \mathbf{Y}$
  $\mathbf{P}^{(k)} \leftarrow \mathbf{S}_k \cdot \mathbf{P}$
  $\mathbf{Z} = \mathbf{Z} \oplus \mathbf{Y}^{(k)} \oplus \mathbf{P}^{(k)}$
end for
Train the mislabel detector with $\mathbf{Z}_{val}$, $\mathbf{Y}_{val_c}$
Apply the mislabel detector on $\mathbf{Z}_{test}$

Neighborhood Extraction We now describe how we extract the labels and base classifier predictions from the neighborhood of each node. Define the normalized adjacency matrix as $\tilde{\mathbf{A}} := \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ where $\mathbf{D}$ is the diagonal matrix of node degrees.

Definition 3.1. The k-hop propagation matrix is defined as

$$\mathbf{S}_k := \mathsf{zero}(\tilde{\mathbf{A}}^k) \in \mathbb{R}^{n \times n}, \tag{1}$$

where the $\mathsf{zero}(\cdot)$ operation indicates zeroing-out all the diagonal entries of the matrix.

Multiplying any signal by $\mathbf{S}_k$ propagates it over k-hop neighborhoods. The $\mathsf{zero}(\cdot)$ operation is essential for avoiding *label leakage*, where information about the target labels influences the input to a classifier.

Then, to extract label information in each node's k-hop neighborhood, we start with the label matrix $\mathbf{Y} \in \mathbb{R}^{n \times c}$, and propagate it over k hops by computing $\mathbf{Y}^{(k)} := \mathbf{S}_k \cdot \mathbf{Y} \in \mathbb{R}^{n \times c}$ for any k. In order to extract the base classifier's predictions in such neighborhoods, we do the same propagation steps on the softmax prediction matrix $\mathbf{P} \in \mathbb{R}^{n \times c}$, obtaining $\mathbf{P}^{(k)} := \mathbf{S}_k \cdot \mathbf{P} \in \mathbb{R}^{n \times c}$. Intuitively, $\mathbf{Y}$ and $\mathbf{P}$ are smoothed by information in k-hop neighborhood, yielding $\mathbf{Y}^{(k)}$ and $\mathbf{P}^{(k)}$, which have greater expression capability by encoding neighborhood information.

Neighborhood Agreement Features Recall that $\mathbf{Y}$ contains the original observed labels, while $\mathbf{Y}_c$ contains the corrupted labels where we flip the labels of nodes in $\mathcal{V}_{\text{synth}}$. To better exploit neighbor-dependence, we employ an agreement operator $\textsc{Agr} : \mathbb{R}^{n \times c} \times \mathbb{R}^{n \times c} \mapsto \mathbb{R}^{n \times c}$, which we will use to measure the *agreement* between the (possibly corrupted) node's own label $\mathbf{Y}_c$, and three quantities from the above Neighborhood Extraction process: 1) the model predictions $\mathbf{P}$; 2) the neighborhood-propagated predictions $\mathbf{P}^{(k)}$; and 3) the neighborhood-propagated labels $\mathbf{Y}^{(k)}$, for every hop $k \in [K]$.

There are various ways to gauge such agreement and a simple way is to take *dot products* $\ominus$ as the agreement measure $\textsc{Agr}$. Accordingly, define the *row-wise dot product*, denoted $\mathbf{U} \ominus \mathbf{V}$ for any same-sized matrices $\mathbf{U}$ and $\mathbf{V}$, as the column vector whose i-th entry is the dot product between the i-th rows of $\mathbf{U}$ and $\mathbf{V}$. In this way, we construct the input feature matrix $\mathbf{Z}$ to our mislabel detector by concatenating such agreement terms[2]:

$$\mathbf{Z} = \Big[\mathbf{Y}_c \ominus \mathbf{P}, \underbrace{\mathbf{Y}_c \ominus \mathbf{P}^{(1)}, \cdots, \mathbf{Y}_c \ominus \mathbf{P}^{(K)}}_{K}, \underbrace{\mathbf{Y}_c \ominus \mathbf{Y}^{(1)}, \cdots, \mathbf{Y}_c \ominus \mathbf{Y}^{(K)}}_{K}\Big]$$

Mislabel Detector We train a Multi-Layer Perceptron (MLP) model on the validation set as the mislabel detector, with input features $\mathbf{Z}$, and output as a binary variable indicating whether the node is a synthetic mislabelled node. L1 loss $L(x, y) = \frac{1}{N}\sum_{i=1}^{n} |x_i - y_i|$ is adopted for training, motivated by the finding in Hu et al. (2022) that L1 loss has stronger robustness and smaller calibration error than the commonly used cross entropy loss. Mislabel detection is closely related to calibration, since both tasks expect the softmax probability associated with the predicted class label to reflect its ground truth correctness likelihood.

Inference For the target graph $G = (\mathcal{V}, \mathbf{X}, \mathbf{A})$ with label matrix $\mathbf{Y}$, we construct the softmax matrix $\mathbf{P}$ using a base classifier and do K-hop neighborhood propagation to obtain $\mathbf{P}^{(k)}$ and $\mathbf{Y}^{(k)}$ for $k \in [K]$. Note that the corrupted label matrix $\mathbf{Y}_c$ is only needed at training time; at test time, the observed label matrix $\mathbf{Y}$ is used in its place. For every mislabelled node, its unknown true label y^* is estimated using the base classifier's prediction $\hat{y}^* = \text{argmax}_i\, p(y =$

[2]Using both original and corrupted label matrices ($\mathbf{Y}$ and $\mathbf{Y}_c$) in generating the neighborhood agreement features is an important design choice to ensure that synthetic mislabels only corrupt their own features, not the features of other nodes; see Appendix D.

$i \mid \mathbf{x}$). For the time complexity of our GRAPHCLEANER, see Appendix E.

Remark Both our mislabel transition matrix and mislabel detector are estimated on $\mathcal{V}_{\text{val}}$. There are three advantages in doing so. 1) $\mathcal{V}_{\text{val}}$ is more representative of $\mathcal{V}_{\text{test}}$, avoiding overfitting on $\mathcal{V}_{\text{train}}$; 2) We can easily adapt to distribution shift away from the training distribution; 3) We only need a small amount of data to train the mislabel detector, which requires fewer parameters and improves efficiency.

3.5. Theoretical Guarantees

In order to assist users in selecting appropriate thresholds for converting mislabel scores into binary mislabel predictions, we propose an algorithm that is accompanied by theoretical guarantees on false positive and false negative rates, derived from the conformal prediction framework (Vovk et al., 2005; Balasubramanian et al., 2014). That is, if we choose the threshold based on the following propositions, the probability of a false positive (i.e. mistakenly classifying a correctly labelled sample as mislabelled) and a false negative (i.e. mistakenly classifying a mislabelled sample as correctly labelled) can be bounded by the user-defined confidence level α.

Specifically, given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with mislabelling rate p (i.e. the total number of mislabeled samples is Np), we compute every sample's corresponding mislabel scores $\{s^{(i)}\}_{i=1}^{N}$ and sort them in non-decreasing order and denote them as $(s_{(1)}, \ldots, s_{(N)})$. We can have following theoretical guarantees:

Proposition 3.2 (False Positive Guarantee)**.** *For any given confidence level $\alpha \in (\frac{1}{N+1}, 1)$, define the threshold as $\lambda_\alpha := s_{(B_\alpha)}$, where $B_\alpha = \lceil (N(1-p)+1)(1-\alpha) + Np \rceil$, with probability at least $1-\alpha$ over the random choice of a correctly labelled sample $(\tilde{\mathbf{x}}, \tilde{y})$, we have:*

$$\tilde{s} \leq \lambda_\alpha,$$

Proposition 3.3 (False Negative Guarantee)**.** *Define the modified score function $s' := (1-s) \cdot \mathbb{1}_{\{s>0.5\}}$. For any given confidence level $\alpha \in (\frac{1}{N+1}, 1)$, define the threshold as $\lambda_\alpha := s'_{(B_\alpha)}$, where $B_\alpha = \lceil (Np+1)(1-\alpha) + N(1-p) \rceil$, with probability at least $1-\alpha$ over the random choice of a mislabelled sample $(\tilde{\mathbf{x}}, \tilde{y})$ and with $\tilde{s}'$ as its modified score, we have:*

$$\tilde{s}' = (1-\tilde{s}) \cdot \mathbb{1}_{\{\tilde{s}>0.5\}} \leq \lambda_\alpha,$$

The detailed proof is in Appendix F.

Remark These two propositions show that by sorting the mislabel scores in non-decreasing order and selecting a threshold based on the user-specified confidence level α, we can ensure that the probability of falsely classifying a correctly labelled sample as mislabelled or a mislabelled sample as correctly labelled is bounded by α. This allows users to have confidence in the algorithm's ability to effectively detect mislabelled samples for their specific use case.

4. Experiments

In this section, we conduct experiments to answer the following research questions:

- **RQ1 (Effectiveness):** Does our method outperform other state-of-the-art methods in detecting mislabels across multiple settings, datasets, and base classifiers?
- **RQ2 (Ablation and Robustness):** How do different components and hyperparameters of our method contribute to the performance?
- **RQ3 (Case Studies):** Does our method detect real, previously unknown mislabels in existing popular graph learning datasets such as `PubMed`, `Cora`, `CiteSeer`, and `OGB-arxiv`?

Due to space limitation, we refer experiments about hyperparameters and additional discussion on case studies to Appendix H and I.

4.1. RQ1. Effectiveness

Experimental Setup There is no ground truth dataset for the mislabel detection task. In order to derive datasets with ground truth labels indicating whether a sample is mislabelled, we follow the practice of related works (e.g., DYB, AUM, CL) and randomly introduce mislabels at ϵ-fraction of the nodes. Similarly to INCV (Chen et al., 2019), we introduce two mislabel types in our experiment, *symmetric* and *asymmetric*. In the symmetric setting, the probability of shift from one class to another is equal. In the asymmetric setting, we change class i to class $(i+1) \mod c$.

We use three mislabel rates, $\epsilon = 0.1, 0.05, 0.025$, for realistic concern. Mislabel rates are typically very low in real-world datasets. In Northcutt et al. (2021b), the averaged test set mislabel rate is about 3.4%, and the highest mislabel rate is 10.12% on ImageNet. Though many other methods use larger mislabel rates, we choose 2.5%, 5% and 10% to mimic real-world settings. We refer to Appendix G for more details about experimental setup.

Methods in Comparison We compare GRAPHCLEANER with four other methods: AUM (Pleiss et al., 2020), DYB (Arazo et al., 2019), CL (Northcutt et al., 2021a) and a simple baseline that treats samples whose argmax predictions differ from given labels as mislabels. To the best of our knowledge, AUM, DYB and CL are the current state-

Table 1. Mislabel detection accuracy of our GRAPHCLEANER and other methods across 6 graph benchmarks and 6 noise settings. The percentage in the first column indicates the mislabel ratio ϵ, while 'sym' and 'asym' refer to symmetric and asymmetric noise. The best results are emphasized in bold, and * indicates a statistically significant ($p < 0.01$) difference between the best and the second best result according to T-test. The last row gives the average improvement of GRAPHCLEANER's result over the second best. If GRAPHCLEANER is not the best, it is compared to the best method.

	Method	Cora			CiteSeer			PubMed			Computers			Photo			OGB-arxiv		
		F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T
10% sym	baseline	0.269	0.361	0.158	0.287	0.289	0.195	0.505	0.465	0.441	0.211	0.325	0.118	0.213	0.322	0.120	0.047	0.138	0.024
	DYB	0.455	0.467	0.321	0.297	0.270	0.167	0.197	0.045	0.179	0.755	0.745	**0.843**	0.777	0.770	0.884	0.500	0.496	0.705
	AUM	0.235	0.199	0.620	0.202	0.121	0.424	0.183	0.046	**0.678***	0.181	0.026	0.762	0.179	0.040	0.839	0.366	0.365	0.425
	CL	0.560	0.513	0.570	0.432	0.386	0.459	0.447	0.388	0.483	0.657	0.619	0.692	0.776	0.755	0.733	0.303	0.292	0.383
	Ours	**0.790***	**0.773***	**0.803***	**0.560***	**0.535***	**0.504***	**0.616***	**0.576***	0.627	**0.844***	**0.830***	0.840	**0.902***	**0.893***	**0.897**	**0.760***	**0.744***	**0.809***
10% asym	baseline	0.141	0.219	0.078	0.153	0.170	0.094	0.366	0.324	0.296	0.006	0.046	0.003	0.000	-0.009	0.000	0.003	0.025	0.002
	DYB	0.404	0.397	0.283	0.289	0.246	0.174	0.278	0.234	0.242	0.625	0.626	0.625	0.678	0.675	0.744	0.310	0.251	0.320
	AUM	0.223	0.169	0.577	0.199	0.111	0.385	0.183	0.048	**0.657***	0.182	0.033	**0.726***	0.179	0.037	0.782	0.252	0.145	0.357
	CL	0.556	0.511	0.542	0.419	0.365	**0.431***	0.456	0.402	0.431	0.671	0.635	0.693	0.760	0.735	0.737	0.274	0.232	0.322
	Ours	**0.669***	**0.633***	**0.647***	**0.475***	**0.424***	0.398	**0.565***	**0.516***	0.545	**0.778***	**0.757***	0.692	**0.832***	**0.817***	**0.797***	**0.477***	**0.415***	**0.452***
5% sym	baseline	0.185	0.262	0.108	0.256	0.278	0.171	0.443	0.414	0.453	0.199	0.317	0.111	0.147	0.261	0.081	0.053	0.152	0.028
	DYB	0.276	0.340	0.257	0.190	0.229	0.062	0.129	0.100	0.151	0.585	0.611	0.699	0.644	0.670	0.760	0.275	0.332	0.612
	AUM	0.132	0.157	0.532	0.108	0.093	0.381	0.095	0.032	**0.557**	0.093	0.015	0.616	0.091	0.030	0.760	0.212	0.266	0.302
	CL	0.477	0.456	0.506	0.324	0.340	0.273	0.310	0.303	0.320	0.572	0.560	0.628	0.736	0.723	0.743	0.165	0.198	0.264
	Ours	**0.661***	**0.671***	**0.719***	**0.406***	**0.425***	**0.396**	**0.477**	**0.483***	0.551	**0.724***	**0.728***	**0.754***	**0.841***	**0.843***	**0.818***	**0.596***	**0.615***	**0.747***
5% asym	baseline	0.157	0.235	0.089	0.232	0.257	0.152	0.252	0.213	0.251	0.000	0.000	0.000	0.000	-0.006	0.000	0.005	0.032	0.003
	DYB	0.270	0.329	0.266	0.174	0.203	0.062	0.158	0.173	0.159	0.509	0.553	0.545	0.538	0.578	0.639	0.216	0.245	0.304
	AUM	0.125	0.132	0.500	0.110	0.100	0.323	0.095	0.032	**0.541***	0.129	0.066	0.599	0.091	0.028	0.742	0.095	0.004	0.266
	CL	0.512	0.489	0.517	0.306	0.325	0.223	0.311	0.315	0.206	0.576	0.567	0.597	0.755	0.743	**0.757**	0.154	0.168	0.223
	Ours	**0.586***	**0.590***	**0.617***	**0.364***	**0.372***	**0.350**	**0.454***	**0.449***	0.486	**0.695***	**0.700***	**0.608**	**0.762**	**0.758**	0.733	**0.457***	**0.446***	**0.435***
2.5% sym	baseline	0.264	0.327	0.167	0.223	0.215	0.191	**0.393**	0.392	0.463	0.163	0.272	0.091	0.122	0.210	0.068	0.051	0.143	0.026
	DYB	0.133	0.220	0.162	0.090	0.147	0.000	0.086	0.140	0.204	0.321	0.404	0.382	0.349	0.440	0.441	0.137	0.221	0.440
	AUM	0.061	0.103	0.448	0.054	0.078	**0.250**	0.048	0.021	**0.596**	0.046	0.012	0.489	0.044	0.018	0.747	0.114	0.189	0.203
	CL	0.402	0.409	0.476	0.166	0.216	0.186	0.221	0.274	0.300	0.424	0.446	0.511	0.698	0.699	0.738	0.085	0.138	0.169
	Ours	**0.445***	**0.492***	**0.586***	**0.237**	**0.303***	0.245	0.385	**0.452***	0.562	**0.555***	**0.594***	**0.658***	**0.709**	**0.729**	**0.753**	**0.417***	**0.484***	**0.626***
2.5% asym	baseline	0.136	0.183	0.081	0.112	0.108	0.086	0.237	0.226	0.225	0.000	0.000	0.000	0.000	-0.004	0.000	0.005	0.032	0.003
	DYB	0.139	0.234	0.167	0.070	0.099	0.000	0.083	0.125	0.125	0.328	0.421	0.377	0.320	0.409	0.344	0.127	0.197	0.250
	AUM	0.063	0.104	0.424	0.053	0.073	**0.218**	0.063	0.049	**0.583***	0.069	0.048	0.488	0.057	0.045	0.709	0.055	0.021	0.184
	CL	0.411	0.418	0.438	0.162	0.207	0.145	0.225	0.289	0.208	0.428	0.456	0.471	**0.698***	**0.700***	**0.712***	0.083	0.121	0.146
	Ours	**0.502***	**0.566***	**0.591***	**0.214***	**0.262***	0.173	**0.334***	**0.384***	0.417	**0.572***	**0.621***	**0.558***	0.669	0.683	0.600	**0.365***	**0.412***	**0.406***
improvement		+0.122	+0.155	+0.134	+0.065	+0.080	+0.001	+0.081	0.097	-0.071	+0.121	+0.128	+0.041	+0.049	+0.059	-0.007	+0.251	+0.229	+0.135

of-the-art approaches for mislabel detection on non-graph data. We apply them to graphs by using a graph-based base classifier. Moreover, we only compare with the label noise modeling part of DYB as our goal is to detect mislabels.

Datasets and Evaluation Metrics We use 6 datasets, namely, `Cora`, `CiteSeer` and `PubMed` (Yang et al., 2016), `Computers` and `Photo` (Shchur et al., 2018), `OGB-arxiv` (Hu et al., 2020) and 3 commonly used metrics: F1, Matthews Correlation Coefficient (MCC) and P@T[3] , where 'T' represents the number of artificially mislabelled nodes, which varies in different experimental settings.

Findings Our GRAPHCLEANER outperforms other methods under almost all metrics across 6 datasets and 6 noise settings, especially in F1 and MCC. We obtain an average margin of 0.14 in F1 and 0.16 in MCC compared to the second best method as shown in Table 1. Table 2 shows that our GRAPHCLEANER generalizes well to different base classifiers and consistently outperforms other methods.

[3] https://en.wikipedia.org/wiki/Evaluation_measures_(information_retrieval)#Precision_at_k

Table 2. Performance on two different base GNN classifiers: GIN and GraphUNet (referred as 'GUN'). We refer to Table 1 for the full name of abbreviations. Experiments are done under the 10% symmetric setting.

	Method	Cora			Computers			OGB-arxiv		
		F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T
GIN	baseline	0.161	0.083	0.144	0.176	0.194	0.108	0.045	0.140	0.023
	DYB	0.199	0.104	0.142	0.441	0.438	0.603	0.374	0.364	0.548
	AUM	0.209	0.147	0.449	0.408	0.409	0.644	0.267	0.237	0.441
	CL	0.339	0.258	0.326	0.411	0.344	0.531	0.234	0.134	0.353
	Ours	**0.784**	**0.762**	**0.756**	**0.855**	**0.840**	**0.851**	**0.687**	**0.669**	**0.693**
GUN	baseline	0.300	0.355	0.186	0.169	0.277	0.093	0.051	0.141	0.026
	DYB	0.269	0.254	0.227	0.516	0.513	**0.831**	0.232	0.185	0.404
	AUM	0.308	0.307	0.732	0.555	0.555	0.684	0.331	0.318	0.365
	CL	0.520	0.471	0.546	0.555	0.514	0.607	0.258	0.230	0.358
	Ours	**0.804**	**0.788**	**0.845**	**0.846**	**0.832**	**0.831**	**0.691**	**0.668**	**0.645**

4.2. RQ2. Ablation and Robustness

Ablation To show the effectiveness of each component, we compare GRAPHCLEANER with three ablated variants:

- *L only*: only the agreement between a sample's label and the labels of its neighbors is used;
- *P only*: only the agreement between a sample's label and the softmax predictions of its neighbors is used;
- *No CL*: instead of following the mislabel transition matrix, we randomly mislabel sampled nodes to some other classes to generate synthetic mislabels.

Table 3. Ablation study. We refer to Section 4.2 for the description of different variants: 'No CL', 'L only' and 'P only'. Experiments are conducted using GCN with 10% noise.

	Method	Cora			Computers			OGB-arxiv		
		F1	MCC	P@T	F1	MCC	P@T	F1	MCC	P@T
sym	*L only*	0.798	0.780	0.801	0.840	0.826	0.845	0.750	0.733	0.808
	P only	0.676	0.654	0.680	0.844	0.829	0.838	0.736	0.718	0.805
	No CL	**0.802**	**0.782**	**0.808**	**0.865**	**0.851**	**0.859**	**0.819**	**0.799**	**0.813**
	Ours	0.790	0.773	0.803	0.844	0.830	0.840	0.760	0.744	0.809
asym	*L only*	0.658	0.624	0.642	**0.781**	**0.761**	0.685	0.460	0.397	0.438
	P only	0.544	0.508	0.554	0.759	0.740	0.661	**0.481**	**0.420**	**0.455**
	No CL	0.635	0.596	0.621	0.474	0.449	0.382	0.295	0.288	0.203
	Ours	**0.669**	**0.633**	**0.647**	0.778	0.757	**0.692**	0.477	0.415	0.452

Table 3 shows that our GRAPHCLEANER consistently outperform *No CL* version with a large margin in asymmetric settings, and is slightly worse in symmetric settings. This is unsurprising and reasonable considering that asymmetric mislabelling noise follows a class-wise pattern that can be dealt with more effectively using the mislabel transition matrix, while symmetric setting follows a class-irrelevant noise pattern. Overall, this suggests that our default framework with mislabel transition matrix is more robust to different environments, e.g. symmetric and asymmetric. Moreover, when compared to *L only* and *P only*, our default GRAPHCLEANER always outperforms the inferior method, and in most cases outperforms both, indicating that our method is adaptive and can generalize better.

4.3. RQ3. Case Studies

Overview In this subsection, we aim to show that our method is able to detect and correct real, previously unknown mislabelled samples in a variety of popular real-world graph datasets: `PubMed`, `CiteSeer`, `Cora`, and `OGB-arxiv`. Then, to understand the implications of these findings, we conduct a more detailed analysis on the `PubMed` dataset, where some auxiliary information allows us to loosely estimate the number and impact of mislabels.

Experimental Procedure We follow the same experimental settings in Section 4.1, except applied to the original datasets without synthetic mislabels. For each dataset, we extract the top 30 samples which our algorithm assigns the highest mislabel scores, i.e., that it regards as most likely to be mislabelled. We then manually inspect each of these top 30 samples from each dataset by accessing the original text of each paper, to determine whether our algorithm is correct. In particular, we manually categorize each sample into 5 possible categories: 'clear mislabel', 'likely mislabel', 'ambiguous', 'likely non-mislabel', and 'clear non-mislabel'.

Findings The results are shown in Figure 3. We have a number of key findings:

- A substantial fraction of these samples are mislabelled: averaged over the 4 datasets, 39% of the samples are likely or clear mislabels. We do not expect this fraction to be close to 100%, since the fraction of mislabels overall is expected to be low (Northcutt et al., 2021b); moreover, our base classifiers are not close to 100% accurate. Still, the results indicate that our method can greatly improve the efficiency of manual corrections, by identifying likely mislabels for humans to check.
- There is a fairly large variation between datasets, with the largest fraction of mislabels in `PubMed`. This may be due to several factors: differences in the fraction of mislabels in each dataset, different accuracy of the base classifiers, and different levels of label ambiguity.
- In practice, there can also be value in identifying the 'ambiguous' or even 'likely non-mislabel' samples, as such samples carry a significant amount of *label noise*. Many such samples span multiple categories , where algorithms for handling label noise could be employed.

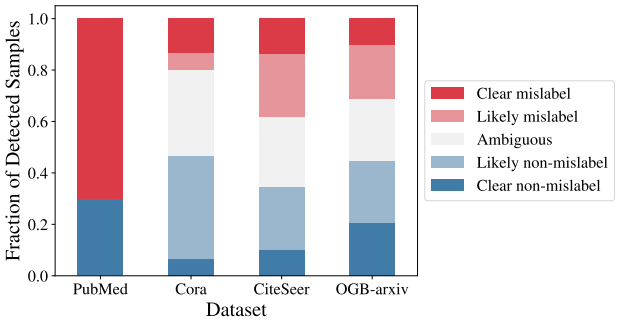

Figure 3. Fraction of each mislabel type among the top 30 detected samples in each dataset: namely, `PubMed`, `Cora`, `CiteSeer`, and `OGB-arxiv`.

4.3.1. MISLABEL RATE ESTIMATION ON PUBMED

Can we loosely estimate the total amount of label noise in `PubMed`? What are the implications of this label noise? How do they affect the evaluation performance? To answer these questions, we focus on `PubMed` due to the presence of some helpful auxiliary information.

By querying PubMed API, we find that 71 papers of `PubMed` are assigned 0 labels (i.e., MeSH terms), 1256 papers are assigned 2 labels, and 39 papers are assigned 3 labels. However, the `PubMed` dataset was created by simply selecting the first label in alphabetical order rather than the 'most correct' class. Consequently, this injects a form of label noise or ambiguity to at least 6.91% of the entire dataset. More details about the estimation can be found in Appendix I. Note that this is only a lower bound for the true amount of label noise, as it excludes mislabels from annotations and other sources, such as those in Figure 1.

Implication of Label Noise What are the implications of these 6.91% label noise samples? To study the effect, we compare the accuracy of a simple GCN+mixup baseline before and after removing the above-mentioned noisy-labelled samples in the test set. The accuracy improves from 86.71% to 89.11%, showing that label noise has significant effects on performance evaluation. This validates the importance of data quality (e.g., label noise) for performance and evaluation, and suggests that correcting mislabels has scope for significant value.

In addition, recognizing the importance of correcting this label noise, we publicly release two new variants of `PubMed` dataset: 1) `PubMedCleaned`, which removes[4] these noisy-labelled samples, and also corrects all detected mislabels; 2) `PubMedMulti`, which keeps multi-labelled samples but explicitly assigns them multiple labels, for users to develop algorithms which can handle the multi-labelling scenario.

5. Conclusions

Data quality is crucial to the success of AI systems. Our case studies, however, demonstrate that label noise is prevalent in popular graph datasets, and that performance on PUBMED changes from 86.71% to 89.11% by simply removing some label noise, highlighting the importance of detecting and correcting mislabels. To address the issue, we propose GRAPHCLEANER that utilizes the *neighborhood-dependence* pattern of graphs to detect mislabelled nodes in a graph. Extensive experiments on 6 datasets across 6 noise settings verify the effectiveness and robustness of our method. The case studies further validate its practicality in real-world applications.

The current study utilizes the neighborhood-dependence property to detect mislabels and improve data quality. Besides that, it would be interesting to explore what other characteristics are associated with label and data quality. Going beyond the current task, we believe this framework is also promising for providing calibrated confidence and misclassification detection on graph data.

Acknowledgements

This work was supported by NUS-NCS Joint Laboratory (A-0008542-00-00).

[4]We would like to clarify that we do not actually remove detected mislabels from the graph by deleting their nodes, or recommend doing so. Instead, we recommend keeping the nodes themselves, but only removing their labels (or equivalently, removing their node index from the set of node indices constituting the training or test sets, without removing them from the graph). Most graph neural networks can be trained in a semi-supervised manner, so we can easily avoid computing the loss on these mislabelled nodes, while preserving the structure and feature information of these nodes, e.g. to avoid bridge nodes from being removed.

References

Algan, G. and Ulusoy, I. Image classification with deep learning in the presence of noisy labels: A survey. *Knowledge-Based Systems*, 215:106771, 2021.

Arazo, E., Ortego, D., Albert, P., O'Connor, N., and McGuinness, K. Unsupervised label noise modeling and loss correction. In *International conference on machine learning*, pp. 312–321. PMLR, 2019.

Balasubramanian, V., Ho, S.-S., and Vovk, V. *Conformal prediction for reliable machine learning: theory, adaptations and applications*. Newnes, 2014.

Chen, P., Liao, B. B., Chen, G., and Zhang, S. Understanding and utilizing deep neural networks trained with noisy labels. In *International Conference on Machine Learning*, pp. 1062–1070. PMLR, 2019.

Dai, E., Aggarwal, C., and Wang, S. Nrgnn: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 227–236, 2021.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Hoang, N., Choong, J. J., and Murata, T. Learning graph neural networks with noisy labels. 2019.

Hu, T., Wang, J., Wang, W., and Li, Z. Understanding square loss in training overparametrized neural network classifiers, 2022. URL https://openreview.net/forum?id=N3KYKkSvciP.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

Huang, Q., He, H., Singh, A., Lim, S.-N., and Benson, A. Combining label propagation and simple models outperforms graph neural networks. In *International Conference on Learning Representations*, 2020.

Jia, J. and Benson, A. R. Residual correlation in graph neural network regression. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pp. 588–598, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/

3394486.3403101. URL https://doi.org/10.1145/3394486.3403101.

Jiang, Z., Zhou, K., Liu, Z., Li, L., Chen, R., Choi, S.-H., and Hu, X. An information fusion approach to learning with instance-dependent label noise. In *International Conference on Learning Representations*, 2021.

Kull, M., Perello Nieto, M., Kängsepp, M., Silva Filho, T., Song, H., and Flach, P. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. *Advances in neural information processing systems*, 32, 2019.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. 12 2016.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

Northcutt, C., Jiang, L., and Chuang, I. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021a.

Northcutt, C. G., Athalye, A., and Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. 2021b. doi: 10.48550/ARXIV.2103.14749. URL https://arxiv.org/abs/2103.14749.

Pleiss, G., Zhang, T., Elenberg, E., and Weinberger, K. Q. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.

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

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., and Sun, Y. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

Vovk, V., Gammerman, A., and Shafer, G. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

Wang, H. and Leskovec, J. Unifying graph convolutional neural networks and label propagation. *arXiv preprint arXiv:2002.06755*, 2020.

Wang, X., Liu, H., Shi, C., and Yang, C. Be confident! towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, 34:23768–23779, 2021.

Wenger, J., Kjellström, H., and Triebel, R. Non-parametric calibration for classification. In Chiappa, S. and Calandra, R. (eds.), *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pp. 178–190. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/wenger20a.html.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Xia, X., Liu, T., Wang, N., Han, B., Gong, C., Niu, G., and Sugiyama, M. Are anchor points really indispensable in label-noise learning? *Advances in Neural Information Processing Systems*, 32, 2019.

Xiong, M., Li, S., Feng, W., Deng, A., Zhang, J., and Hooi, B. Birds of a feather trust together: Knowing when to trust a classifier via adaptive neighborhood aggregation. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=p5V8P2J61u.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Zhu, Z., Dong, Z., and Liu, Y. Detecting Corrupted Labels Without Training a Model to Predict. *arXiv e-prints*, art. arXiv:2110.06283, October 2021.

Zhu, Z., Liu, T., and Liu, Y. A second-order approach to learning with instance-dependent label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10113–10123, 2021a.

Zhu, Z., Song, Y., and Liu, Y. Clusterability as an alternative to anchor points when learning with noisy labels. In *International Conference on Machine Learning*, pp. 12912–12923. PMLR, 2021b.

A. Comparison to Learning with Noisy Labels Methods

Our GRAPHCLEANER and some learning-with-noise methods all explore neighborhood similarity (Zhu et al., 2021b;a). But they focus on i.i.d. image data, while we focus on non-i.i.d. graph data, which differs from i.i.d settings, as our edge relations are 1) of primary importance; and 2) can involve more complex relations other than just similarity. Hence, we presents a flexible approach that learns the patterns of mislabels from data, primarily based on graph structure information.

Synthetic mislabel generation based on transition probabilities is applied in Xia et al. (2019); Jiang et al. (2021) to help training model with noisy data. But their focus is on how to estimate and predefine a noise transition matrix, while our noise transition matrix is learned and utilized to generate synthetic mislabels in a post-hoc way, which is more flexible and can easily adapt to different data.

B. Learning Mislabel Transition Matrix

To generate class-dependent label noise, we learn a *mislabel transition matrix* to capture the probability of mislabelling one class to some other class.

Following the definition in Northcutt et al. (2021a), we first calculate confident joint $\mathbf{C}_{\tilde{y},y^*}$ based on the model predictions on validation set, which estimates the number of samples with noisy label $\tilde{y}$ and actual true label y^*:

$$\mathbf{C}_{\tilde{y},y^*}[i][j] \coloneqq \left|\hat{\mathbf{X}}_{\tilde{y}=i,y^*=j}\right|, \tag{2}$$

where $\hat{\mathbf{X}}_{\tilde{y}=i,y^*=j}$ is the estimated set of samples with observed noisy label i and unknown true label j. Its formal definition is:

$$\begin{aligned}\hat{\mathbf{X}}_{\tilde{y}=i,y^*=j} \coloneqq &\{x \in \mathbf{X}_{\tilde{y}=i} : \hat{p}\left(\tilde{y}=j;x,\theta\right) \geq t_j, \\ &j = \underset{l\in[c]:\hat{p}(\tilde{y}=j;x,\theta)\geq t_l}{\arg\max}\hat{p}\left(\tilde{y}=j;x,\theta\right)\},\end{aligned} \tag{3}$$

where $\hat{p}\left(\tilde{y}=i;x,\theta\right)$ is the predicted probability of label $\tilde{y}=i$ for sample x and model parameters θ, and $\mathbf{X}_{\tilde{y}=i}$ is the set of samples with observed noisy label i. The threshold t_j is the average self-confidence for class j:

$$t_j = \frac{1}{|\mathbf{X}_{\tilde{y}=j}|}\sum_{x\in\mathbf{X}_{\tilde{y}=j}}\hat{p}\left(\tilde{y}=j;x,\theta\right). \tag{4}$$

As shown in the above formulas, samples in $\hat{\mathbf{X}}_{\tilde{y}=i,y^*=j}$ should satisfy three conditions. First, their observed noisy label $\tilde{y}$ should be i. Second, their predicted probability of belonging to class j should be higher than class j's average self-confidence. Third, among all the classes whose predicted probability is larger than their corresponding self-confidence, class j's predicted probability is the highest.

Now we can estimate the joint distribution of noisy label and true label based on the confident joint:

$$\hat{\mathbf{Q}}_{\tilde{y}=i,y^*=j} = \frac{\frac{\mathbf{C}_{\tilde{y}=i,y^*=j}}{\sum_{j\in[c]}\mathbf{C}_{\tilde{y}=i,y^*=j}}\cdot|\mathbf{X}_{\tilde{y}=i}|}{\sum\limits_{i\in[c],j\in[c]}\left(\frac{\mathbf{C}_{\tilde{y}=i,y^*=j}}{\sum_{j'\in[c]}\mathbf{C}_{\tilde{y}=i,y^*=j'}}\cdot|\mathbf{X}_{\tilde{y}=i}|\right)}. \tag{5}$$

Finally, the mislabel transition matrix can be calculated following the conditional probability formula $\hat{\mathbf{Q}}_{\tilde{y}=i|y^*=j} \coloneqq \hat{\mathbf{Q}}_{\tilde{y}=i,y^*=j}/\hat{\mathbf{Q}}_{y^*=j}$.

C. Discussion on Sampling Mislabels

We assume that $\mathcal{V}_{\text{train}}$, $\mathcal{V}_{\text{val}}$, $\mathcal{V}_{\text{test}}$ are all noisy, which means mislabels exist in the raw dataset. But according to Northcutt et al. (2021a), raw mislabels only account for a minor proportion of a dataset. Some of the raw mislabels may be chosen as synthesized mislabels. There is a slight chance that the labels of these chosen raw mislabels will be flipped to the correct ones. Nevertheless, the raw mislabels whose labels remain wrong after *Synthetic Mislabel Dataset Generation* are still a very small portion, which serve as acceptable noises contributing to the robustness of our mislabel detector.

D. Design of Neighborhood Agreement Features

Using both the corrupted and original label matrices ($\mathbf{Y}$ and $\mathbf{Y}_c$) is an important design choice to ensure that *synthetic mislabels only corrupt their own features, not the features of other nodes*.

Specifically, when we flip the label of a node $v \in \mathcal{V}_{\text{synth}}$, this affects $\mathbf{Y}_c$ but not $\mathbf{Y}$ or $\mathbf{P}$; hence, the features at node v are affected, but those at other nodes are not. This is important as when training the mislabel detector, the fraction of synthetic mislabels is high (due to our use of a balanced training set), so we do not want to allow the mislabels at corrupted nodes to mislead the training of the mislabel detector.

E. Time Complexity

In this section, we estimate the time complexity of our GRAPHCLEANER. Given that there are m edges and s epochs, it takes $O(n+c^2)$ to estimate the mislabel transition

matrix, $O(mKc)$ to generate the neighborhood agreement features, and $O(nKs)$ to train the mislabel detector. Test time complexity is just $O(K)$ per test sample. Overall, training time is linear and testing time is constant per test sample, which is highly efficient.

F. Theoretical Guarantees

Overview In this section, we show how we can set mislabel score thresholds to obtain guarantees on the false positive and false negative probabilities, using the approach of conformal prediction. Such guarantees can be valuable in many practical settings, where are interested in having guarantees on the reliability of our model's decisions.

Conformal prediction (Vovk et al., 2005; Balasubramanian et al., 2014) is a simple, distribution-free approach for obtaining confidence guarantees. Importantly, conformal prediction does not require the assumption that the mislabel scores are i.i.d., but only that they are *exchangeable*; i.e., their distribution does not change under any permutation of the sample indices. This is especially suitable for the graph setting, where the samples are in fact not i.i.d.: e.g., samples which are nearby along the graph tend to be correlated. However, the data is still exchangeable, since permutations to the indices do not affect the data distribution.

Comparison to prior work Conformal prediction (Vovk et al., 2005) is typically applied to the standard setting where all the samples are exchangeable. Most closely related to our approach is Xiong et al. (2022), which extended this to the mislabel detection setting, where a small number of samples can be mislabelled. However, the approach in Xiong et al. (2022) only allows for practical false positive guarantees, so they do not prove false negative guarantees, while our approach allows for both false positive and negative guarantees due to our use of modified scores, which tightens the bounds by exploiting the accuracy of the mislabel classifier.

Let $\{\cdots\}$ denote a set and $(\cdots)$ denote an ordered tuple: e.g., sorting a set $\{3, 1, 2\}$ yields an ordered tuple $(1, 2, 3)$.

Before focusing on the mislabel detection setting, we first consider a more general setting where we are given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ with any set of scores $\{s^{(i)}\}_{i=1}^{N}$; we further assume that the dataset comes from a mixture of two distributions: specifically, N_U of the the samples come from some distribution $\mathcal{U}$, i.e., $(\mathbf{x}, y) \sim \mathcal{U}$, while the remaining $N_V = N - N_U$ samples come from the distribution $\mathcal{V}$, i.e., $(\mathbf{x}, y) \sim \mathcal{V}$. Let $(s_{(i)})_{i=1}^{N}$ denote the score of these samples in non-decreasing order.

Theorem F.1 (Conformal Prediction for Mixtures)**.** *For any given confidence level $\alpha \in (\frac{1}{N+1}, 1)$, define the threshold as*

$$\lambda_\alpha := s_{(B_\alpha)}, \text{ where } B_\alpha = \lceil (N_U + 1)(1 - \alpha) + N_V \rceil .$$

Now consider a newly drawn sample (e.g., from the test set):

$$(\tilde{\mathbf{x}}, \tilde{y}) \sim \mathcal{U}.$$

Then, with probability at least $1 - \alpha$ over the random choice of $(\tilde{\mathbf{x}}, \tilde{y})$, we have:

$$\tilde{s} \leq \lambda_\alpha,$$

where $\tilde{s}$ is the score of $\tilde{\mathbf{x}}$.

Proof. Let $(s^{\mathcal{U}}_{(i)})_{i=1}^{N_U}$ denote the scores of the samples from $\mathcal{U}$ in non-decreasing order. The probability that the score of $\tilde{\mathbf{x}}$ exceeds the threshold λ_α is:

$$\begin{aligned}
\mathbb{P}(\tilde{s} > \lambda_\alpha) &= \mathbb{P}\left(\tilde{s} > s_{(B_\alpha)}\right) && (6)\\
&\leq \mathbb{P}\left(\tilde{s} > s^{\mathcal{U}}_{(B_\alpha - N_V)}\right) && (7)\\
&\leq \frac{N_U + 1 - (B_\alpha - N_V)}{N_U + 1} && (8)\\
&= \frac{N + 1 - B_\alpha}{N_U + 1} && (9)\\
&= \frac{N + 1 - \lceil (N_U + 1)(1 - \alpha) + N_V \rceil}{N_U + 1} && (10)\\
&\leq \frac{N + 1 - (N_U + 1)(1 - \alpha) - N_V}{N_U + 1} && (11)\\
&= \frac{N_U + 1 - (N_U + 1)(1 - \alpha)}{N_U + 1} && (12)\\
&= \frac{(N_U + 1)(\alpha)}{N_U + 1} && (13)\\
&= \alpha && (14)
\end{aligned}$$

Therefore, with probability at least $1 - \alpha$, the score $\tilde{s}$ lies below the threshold, i.e.

$$\mathbb{P}(\tilde{s} \leq \lambda_\alpha) \geq 1 - \alpha. \quad (15)$$

Note that the step (8) comes from the fact that $\tilde{\mathbf{x}}$ is exchangeable with the other samples from $\mathcal{U}$ (Balasubramanian et al., 2014). □

Next, we show how Theorem F.1 can be used to obtain both false positive and negative guarantees, for appropriately selected thresholds.

Given a dataset $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$, and let the mislabel scores computed by our GRAPHCLEANER algorithm be $\{s^{(i)}\}_{i=1}^{N}$. Let p be the fraction of these samples which are mislabelled.

Proposition F.2 (False Positive Guarantee)**.** *Letting* $\lambda_\alpha := s_{(B_\alpha)}$*, where* $B_\alpha = \lceil (N(1-p)+1)(1-\alpha) + Np \rceil$*, with probability at least* $1-\alpha$ *over the random choice of a new sample* $(\tilde{\mathbf{x}}, \tilde{y}) \sim \mathcal{U}$*, we have:*

$$\tilde{s} \leq \lambda_\alpha,$$

Proof. We apply Theorem F.1 to the score function s, with $\mathcal{U}$ representing the distribution of correctly labelled samples, and $\mathcal{V}$ representing the distribution of mislabelled samples. □

Discussion This result allows us to set the threshold λ_α in a principled way which provides guarantees on the probability of a false positive (i.e. mistakenly classifying a correctly labelled sample as mislabelled).

Proposition F.3 (False Negative Guarantee)**.** *Define the modified score function* $s' := (1-s) \cdot \mathbb{1}_{\{s>0.5\}}$*. Then letting* $\lambda_\alpha := s'_{(B_\alpha)}$*, where* $B_\alpha = \lceil (Np+1)(1-\alpha) + N(1-p) \rceil$*, with probability at least* $1-\alpha$ *over the random choice of a new sample* $(\tilde{\mathbf{x}}, \tilde{y}) \sim \mathcal{V}$ *and with* $\tilde{s}'$ *as its modified score, we have:*

$$\tilde{s}' = (1-\tilde{s}) \cdot \mathbb{1}_{\{\tilde{s}>0.5\}} \leq \lambda_\alpha,$$

Proof. In this case, we similarly apply Theorem F.1 to the modified score function s', but now with $\mathcal{U}$ representing the distribution of mislabelled samples, and $\mathcal{V}$ representing the distribution of correctly labelled samples. □

Discussion One difference between s' and s is that the direction of s' is reversed compared to s; this is minor and just for ease of interpretation, by making higher scores typically more unlikely. The main difference between them is that s' is nonzero only when the mislabel classifier predicts the sample to be mislabelled. Using this modified score makes sense since we are primarily interested in setting an appropriate threshold for mislabelled samples, using the distribution of mislabelled samples. Meanwhile, note that it is impractical to achieve false negative bounds using the original scores s, due to the large majority of correctly labelled samples which would be expected to have low mislabel scores. In contrast, s' is able to map these samples to 0 as long as they are correctly classified by the mislabel classifier, allowing a tighter bound.

Remark To show the effectiveness of our theoretical guarantees, we calculate the actual false positive rate of our GRAPHCLEANER's predictions on `OGB-arxiv` and compare it to the theoretical bounds in Figure 4.

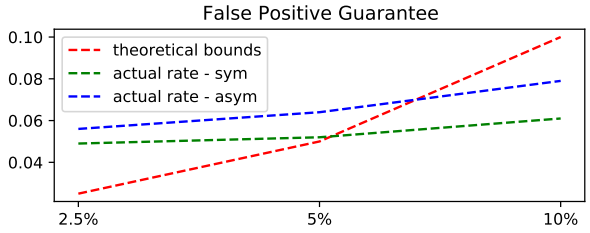

Figure 4. The plot of the theoretical and actual false positive rate under both symmetric and asymmetric noise scenarios. The actual rate is calculated based on the experiments on `OGB-arxiv`.

G. Experimental Setup

Experimental Setup In order to derive datasets with ground truth labels indicating whether a sample is mislabelled, we randomly introduce artificial noise to ϵ fraction of the training, validation and test set. We then further assume that the 'actual' mislabel ratio of the corrupted dataset is ϵ. We follow the practice of INCV (Chen et al., 2019) to introduce two mislabelling types in our experiment: symmetric and asymmetric setting, where the probability of changing one class to any other class is equal or different. In the asymmetric setting, we simply change class i to class $(i+1)$ mod c. Specifically, if mislabel ratio ϵ is 0.1, then for n_i nodes belonging to class i, $0.9n_i$ will remain class i, while the rest $0.1n_i$ will be mislabelled as class $(i+1) \mod c$. Illustration for these two mislabelling types is in Figure 5.

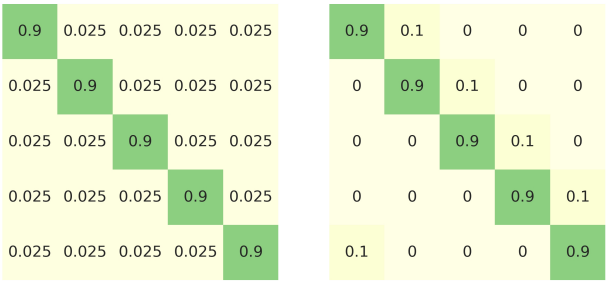

Figure 5. Examples of symmetric (left) and asymmetric (right) mislabelling types (taking 5 classes and mislabel ratio ϵ 0.1 as an example).

We test three mislabel rates ϵ to show that our GRAPHCLEANER can tackle various mislabel severities, each with two mislabel types to show that GRAPHCLEANER consistently performs well under different mislabel patterns. Specifically, ϵ is set as 0.1, 0.05, 0.025, because the max test set error of different benchmarks reported in Northcutt et al. (2021b) is 10.12%. We do admit that there can be some unknown noisy nodes in $\mathcal{V}_{\text{test}}$. But it is safe to assume that this unknown mislabel rate is minor. In experiments, we compare all methods on the same manually corrupted

test set, which is fair.

Threshold Selection Our mislabel detector is a binary classifier. Usually, 0.5 is used as the threshold for binary classifiers' output. But in our case, the training stage is based on a balanced dataset, while the test stage is conducted under an imbalanced setting, which is expected to have only a few percent of mislabelled data. In consideration of this, we adjust the threshold according to Bayes rule. Let $\mathbf{P}_s$ and $\mathbf{P}_t$ denote the binary classifier's predictions on training and test set, then we can have the following equations:

$$\mathbf{P}_s(y|x) = \frac{\mathbf{P}_s(x|y)\mathbf{P}_s(y)}{\mathbf{P}_s(x)}, \mathbf{P}_t(y|x) = \frac{\mathbf{P}_t(x|y)\mathbf{P}_t(y)}{\mathbf{P}_t(x)}. \tag{16}$$

We assume that the only change between training and test set is the class (mislabelled or not) distribution, then we have $\mathbf{P}_s(x|y) = \mathbf{P}_t(x|y)$. Dividing the above two equations and plugging $\mathbf{P}_s(x|y) = \mathbf{P}_t(x|y)$ in will yield:

$$\mathbf{P}_t(y|x) = \mathbf{P}_s(y|x) \cdot \frac{\mathbf{P}_t(y)}{\mathbf{P}_s(y)} \cdot \frac{\mathbf{P}_s(x)}{\mathbf{P}_t(x)}, \tag{17}$$

where predictions on test set depend on training set data distribution $\mathbf{P}_s(y|x)$, some class-dependent scaling factor $\frac{\mathbf{P}_t(y)}{\mathbf{P}_s(y)}$, and one constant value $\frac{\mathbf{P}_s(x)}{\mathbf{P}_t(x)}$ which only replies on x. This indicates that we can adjust the threshold by $\frac{\mathbf{P}_t(y)}{\mathbf{P}_s(y)}$, an expected value of mislabel proportion. Since the average label error reported in Northcutt et al. (2021b) is 3.4%, we simply set the threshold as 0.97. All our experiments and case studies use this threshold.

Dataset 6 datasets for node classificatoin are chosen: `Cora`, `CiteSeer` and `PubMed` (Yang et al., 2016), `Computers` and `Photo` (Shchur et al., 2018), `OGB-arxiv` (Hu et al., 2020). `Cora`, `CiteSeer`, `PubMed` and `OGB-arxiv` are citation networks where nodes represent documents and edges represent citation links. `Computers` and `Photo` are Amazon co-purchase networks where nodes represent goods and edges represent that two goods are frequently bought together.

H. Hyperparameters

The maximum neighborhood size K determines the range of neighborhood we consider. To investigate the robustness of GraphCleaner to K, we vary K from 1 to 5 with other parameters fixed.

Figure 6 shows that our GraphCleaner is insensitive to the hyperparameter K and even setting $K = 1$ yields decent performance, suggesting that using the information of direct neighbors is sufficient in most cases.

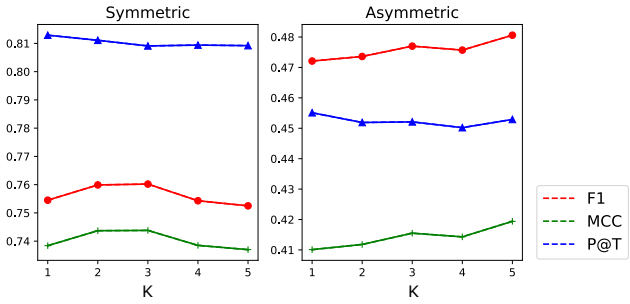

Figure 6. Sensitivity of hyperparameter K. Experiments are performed using GCN on `OGB-arxiv` with 10% noise.

I. Mislabel Case Studies

Supplement to Findings Different datasets have different levels of label ambiguity: e.g., the classes in `PubMed` are more objective and precisely defined than those in `OGB-arxiv`.

Many 'ambiguous' and 'likely non-mislabel' samples span multiple categories: e.g., a paper predicting financial indicators using methods from natural language processing could reasonably belong to the 'finance' or 'language' categories.

Limitations While most samples were straightforward to categorize, we acknowledge that some manual judgments[5] involved unavoidable subjectivity. Even so, our goals in this section are mainly to understand broad overall trends and differences, and our overall findings are relatively robust to small variations. Such subjectivity could be reduced by employing a larger number of manual raters, but this process is fairly labor-intensive. Crowdsourcing could be employed, but is challenging as the task requires some expertise in order to read and categorize papers. Another point is about the inference stage. The focus of our GraphCleaner is not correcting mislabels. We check the accuracy of the inferred labels of clearly mislabelled samples in `Cora`, `CiteSeer` and `OGB-arxiv`, getting results of 75%, 100% and 66.67%, which is acceptable but still has room for improvement.

Mislabel Rate Estimation on `PubMed` Can we loosely estimate the total amount of label noise in `PubMed`? How do these samples affect the use of `PubMed` for evaluating algorithm performance? We focus on `PubMed` due to the presence of some auxiliary information that helps us to answer these questions.

Since `PubMed` contains 19717 samples, estimating the number of mislabels via manual checking is clearly infeasible.

[5]For transparency, we include all manual judgments in our attached code repository.

Fortunately, it is possible to exactly count the number of samples with label noise of a different kind, as we will explain. Concretely, we use the PubMed API[6], which allows us to query the raw 'keywords', known as 'MeSH terms'[7] associated with each paper. The 3 labels used in the PubMed dataset were assigned based on 3 MeSH terms: 'Diabetes Mellitus, Experimental', 'Diabetes Mellitus, Type 1', and 'Diabetes Mellitus, Type 2'. Thus, querying the PubMed API tells us which papers are assigned with 2 or more of these MeSH terms. For clarity, we refer to these samples as 'multi-labelled'. For such cases, it turns out that the PubMed dataset was generated by discarding all but one of such labels for each sample. We note that there is no possibility that the single label kept is somehow indicative of the 'most correct' class, since the label to be kept is simply chosen alphabetically. This is a form of label noise or ambiguity, and the number of such samples can be counted exactly via the PubMed API.

We find that 71 papers are assigned 0 labels (i.e., MeSH terms), 18351 papers are assigned 1 label, 1256 papers are assigned 2 labels, and 39 papers are assigned 3 labels. The latter two categories are multi-labels, while the 0-label papers occur due to updates in the MeSH terms assigned to some papers over time. In summary, we have 1366 papers with some form of label noise, or 6.91% of the entire dataset. Note that this is only a lower bound for the true amount of label noise, as it does not include mislabels such as those in Figure 1.

Effect of Correcting Label Noise What are the implications of these 6.91% label noise samples? The top reported leaderboard scores on PubMed[8] are close to 90% accuracy, suggesting that a significant fraction of the remaining 'missing accuracy' could be attributed to label noise. This supports a 'data-centric' view, which recognizes data quality (e.g., label noise) as an important factor affecting performance and evaluation, and suggests that correcting mislabels has scope for significant value.

Finally, we study the effect of correcting label noise, by comparing the accuracy of a simple GCN+mixup baseline before and after removing the above-mentioned noisy-labelled samples in the test set. The accuracy improves from 86.71% to 89.11%, suggesting that label noise has significant effects on performance evaluation. Thus, recognizing the importance of correcting this label noise, we publicly release two new variants of the PubMed dataset: 1) PubMedCleaned, which removes these noisy-labelled samples, and also corrects all mislabels we have detected; 2) PubMedMulti, which keeps multi-labelled samples but explicitly assigns them multiple labels, for users to develop algorithms which can handle the multi-labelling scenario. Please refer to our code link for these two new datasets and the manual judgement files for case studies.

[6] https://www.ncbi.nlm.nih.gov/home/develop/api/

[7] MeSH terms are a 'vocabulary' or ontology used in PubMed.

[8] https://paperswithcode.com/dataset/pubmed