Contrastive Adversarial Training for Robust Graph Neural Networks under Label Poisoning

Manshika Charvi Bissessur[*1,3] manshika.bissessur@dtc.ox.ac.uk
Melis Ilayda Bal[*2,3] mbal@tuebingen.mpg.de

Michael Muehlebach[3,4]
michaelm@tuebingen.mpg.de
[1]University of Oxford [2]LIONS – École Polytechnique Fédérale de Lausanne
[3]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[4]Vienna University of Economics and Business

Abstract

Graph Neural Networks (GNNs) are effective for modeling relational data but are vulnerable to label poisoning, where a small number of corrupted training labels can propagate errors across the graph via message passing. Despite this risk, defenses against label poisoning remain underexplored: existing methods are primarily designed for label noise and often rely on robust losses or heuristic data cleaning that fail to distinguish adversarial poisoning from naturally hard examples. In this paper, we propose **CoLAT** (**Co**ntrastive **L**abel-flipping for **A**dversarial **T**raining), a novel contrastive adversarial training framework tailored for robust node classification on graphs. The core of our approach is a structure-aware detection model that uses contrastive learning to identify *label–structure inconsistencies*. Unlike prior contrastive methods that focus on representation learning, CoLAT leverages contrastive embeddings to select high-risk nodes that guide adversarial label-flipping during training. This alternating optimization not only performs structure-aware adversarial training but also implicitly sanitizes some corrupted labels, improving robustness against diverse attacks from the literature. Extensive experiments on multiple benchmarks show that CoLAT consistently outperforms existing defenses under various poisoning intensities while scaling efficiently to large graphs.

1 Introduction

Graph Neural Networks (GNNs) have become the dominant paradigm for learning on graph-structured data, with strong results across social network analysis, molecular property prediction, and knowledge graph reasoning (Wu et al., 2020). However, GNNs are notoriously susceptible to adversarial perturbations (Kipf and Welling, 2016; Sun et al., 2022). While extensive research has focused on structural attacks, in which the graph topology is subtly altered (Zügner et al., 2018), the vulnerability of GNNs to label poisoning attacks (Biggio et al., 2012) remains a critical yet relatively underexplored threat. This vulnerability is particularly pronounced in semi-supervised settings. Since GNNs propagate information through message passing, a small number of corrupted labels can influence large portions of the graph, amplifying the effect of the attack (Lingam et al., 2024). As a result, even limited label poisoning can significantly degrade model performance.

Defenses against label poisoning in GNNs remain scarce. Aside from a regularization-based method (Zhang et al., 2020), most prior work addresses label noise rather than adversarial cor-

[*]Equal Contribution. Code is available at https://github.com/ManshikaBissessur/CoLAT

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

ruption (Qian et al., 2023), and thus often fails to distinguish adversarial outliers from hard clean examples, particularly in complex graph structures. Adversarial training has been explored as a defense in graph learning, but prior work primarily focuses on structural perturbations (Gosch et al., 2023), rather than label poisoning. In parallel, contrastive learning has been shown to improve robustness by producing representations that are less sensitive to label noise (You et al., 2020; Yuan et al., 2023), but recent defenses against label noise (Du et al., 2023; Li et al., 2024b) use contrastive objectives mainly as auxiliary regularization. Consequently, existing approaches fail to address label poisoning and leave a critical gap in exploiting graph structure to identify or defend against adversarial label perturbations.

In this paper, we propose **CoLAT** (**Co**ntrastive **L**abel-flipping for **A**dversarial **T**raining), a contrastive-adversarial training framework for robust node classification under label poisoning attacks. We model the problem through a bilevel formulation and derive a tractable alternating optimization scheme. Our key idea is to exploit graph structure to guide adversarial perturbations. In particular, we leverage the unique property of graphs that local neighborhood provides a natural prior for label consistency. Building on this, CoLAT introduces a *contrastive detection model* that learns representations aligned with this structural signal, to identify nodes whose labels are incongruent with their local context and thus susceptible to adversarial manipulation. Conditioned on these high-risk nodes, CoLAT constructs targeted label perturbations by assigning labels that maximize the classifier's loss. The model is then trained on these adversarial labels, yielding an iterative procedure that focuses on structurally inconsistent regions of the graph as a guide for adversarial learning. By tightly coupling *contrastive detection with adversarial optimization*, CoLAT effectively regularizes the classifier against label corruption, improving robustness while preserving performance on clean data.

Contributions. Our main contributions are the following.

- We propose CoLAT, a contrastive-adversarial training framework that explicitly models label poisoning as an adversarial process to improve robustness on graph settings and approximates the minimax formulation via a tractable, structure-aware bilevel procedure.
- Structurally-guided inconsistency scoring: We introduce a contrastive detection model that leverages graph structure to identify nodes whose labels are inconsistent with their local neighborhoods. Unlike prior work that focuses on representation improvement, we employ contrastive learning for node selection to generate adversarial examples in our proposed adversarial training defense.
- We develop an alternating optimization scheme that decomposes an intractable inner maximization into node selection and loss-maximizing label assignment, enabling *scalable adversarial training* without combinatorial search.
- We empirically demonstrate that CoLAT significantly improves robustness to label poisoning across multiple benchmarks and attack types from the literature, including varying attack budgets and large-scale graphs, while maintaining competitive performance on clean data.

2 Problem Formulation and Background

We consider robust semi-supervised node classification on graphs under label poisoning. Let $\mathcal{G} = (V, A, X)$ denote an undirected, unweighted graph with node set $V = \{1, \dots, n\}$, adjacency matrix $A \in \{0,1\}^{n \times n}$, and node features $X \in \mathbb{R}^{n \times d}$, where $n = |V|$ and d is the feature dimension. An edge $(i, j) \in \mathcal{E}$ exists between nodes i and j if and only if $A_{ij} = 1$. We denote the neighborhood of node i by $\mathcal{N}(i) = \{j \in V : A_{ij} = 1\}$ and its degree by $\deg(i) = |\mathcal{N}(i)|$. A subset $V_L \subset V$ of nodes carries labels, while the remaining $V_U = V \setminus V_L$ is unlabeled. Let $C > 1$ be the number of classes and denote the clean one-hot labels as $\mathbf{y}_i \in \{0,1\}^C$ for $i \in V_L$, collected in $Y = \{\mathbf{y}_i : i \in V_L\}$.

We consider a Graph Neural Network (GNN) classifier f_θ parameterized by θ, which maps node features and graph structure to class predictions by iteratively aggregating information from the local neighborhood of nodes. At layer l, the representation of node i is updated as:

$$\mathbf{a}_i^{(l)} = \textsc{Aggregate}^{(l)}(\{\mathbf{h}_j^{(l-1)} : j \in \mathcal{N}(i)\}), \quad \mathbf{h}_i^{(l)} = \textsc{Combine}^{(l)}(\mathbf{h}_i^{(l-1)}, \mathbf{a}_i^{(l)}),$$

where $\mathbf{h}_i^{(0)} = \mathbf{x}_i \in \mathbb{R}^d$. After L layers, $\mathbf{h}_i^{(L)}$ encodes structural and feature information within the L-hop neighborhood of node i. Different GNN architectures instantiate AGGREGATE and COMBINE differently; e.g., Graph Convolutional Network (GCN) (Kipf and Welling, 2016) computes:

$\mathbf{h}_i^{(l)} = \mathrm{ReLU}(W^{(l)} \cdot \mathrm{Mean}\{\mathbf{h}_j^{(l-1)} : j \in \mathcal{N}(i) \cup \{i\}\})$, where $W^{(l)}$ is a learnable weight matrix and the self-loop $\{i\}$ implements residual aggregation.

For node classification, the final-layer representation $\mathbf{h}_i^{(L)}$ is passed through a linear head and softmax to produce the class-probability vector: $\mathbf{p}_i^\theta = \mathrm{softmax}(f_\theta(\mathcal{G})_i) \in \Delta^{C-1}$, where $C > 1$ is the number of classes and Δ^{C-1} is the $(C-1)$-dimensional probability simplex. Under clean labels, The model is trained by minimizing the empirical risk over labeled nodes: $\min_\theta \ \mathcal{L}_{\mathrm{CE}}(\theta; Y) = \frac{1}{|V_L|} \sum_{i \in V_L} \ell(\mathbf{p}_i^\theta, \mathbf{y}_i)$, where $\ell(\cdot, \cdot)$ is the cross-entropy loss. In realistic settings, labels are scarce ($|V_L| \ll n$) and potentially corrupted: an adversary can poison labels in Y within some budget, yielding a perturbed label set $\tilde{Y} = \{\tilde{\mathbf{y}}_i : i \in V_L\}$. Due to message passing in GNNs, such perturbations can propagate through the graph and influence many unlabeled nodes, making label poisoning particularly harmful (Zügner and Günnemann, 2019b). Our goal is to learn a robust classifier θ that generalizes well to V_U despite training on $\tilde{Y}$; we formalize this in Sec. 3.

3 Methodology

3.1 Overview

We study robust node classification under label poisoning, where an adversary perturbs a limited number of training labels, e.g., with budget k, to degrade model performance. Given a graph $\mathcal{G}$ and a GNN f_θ as defined in Sec. 2, the adversary constructs a perturbed label set $\tilde{Y} = \{\tilde{\mathbf{y}}_i : i \in V_L\}$ that maximizes the model's test classification error under a budget constraint. A straightforward formulation of this interaction is the minimax problem (Goodfellow et al., 2015; Madry et al., 2017):

$$\min_\theta \ \max_{\tilde{Y}} \frac{1}{|V_L|} \sum_{i \in V_L} \ell\big(\mathbf{p}_i^\theta, \tilde{\mathbf{y}}_i\big) \quad \text{s.t.} \quad \sum_{i \in V_L} \mathbf{1}[\mathbf{y}_i \neq \tilde{\mathbf{y}}_i] \leq k, \tag{1}$$

where $\mathbf{p}_i^\theta$ is the predicted class distribution for node i, and k is the perturbation budget. However, solving the inner maximization is intractable due to the combinatorial nature of the label space. We thus view Eq. (1) through a bilevel adversarial-training perspective (Robey et al., 2024; Bal et al., 2025), which leads to the key observation that the inner maximization decomposes into two sub-problems: *which* nodes to perturb, and *what* labels to assign them. We decompose the inner maximization into two steps:

1. **Node selection:** A contrastive detection model g_ϕ, parameterized by ϕ, is trained with a contrastive loss, $\mathcal{L}_{\mathrm{contrast}}$ to score each node by its local label inconsistency with its neighbors (Oord et al., 2018). The k most inconsistent nodes form the attack set $\mathcal{S}$.
2. **Label assignment:** Given $\mathcal{S}$, the adversary assigns each selected node the label that maximizes the classification loss.

This decomposition yields the bilevel program illustrated below.

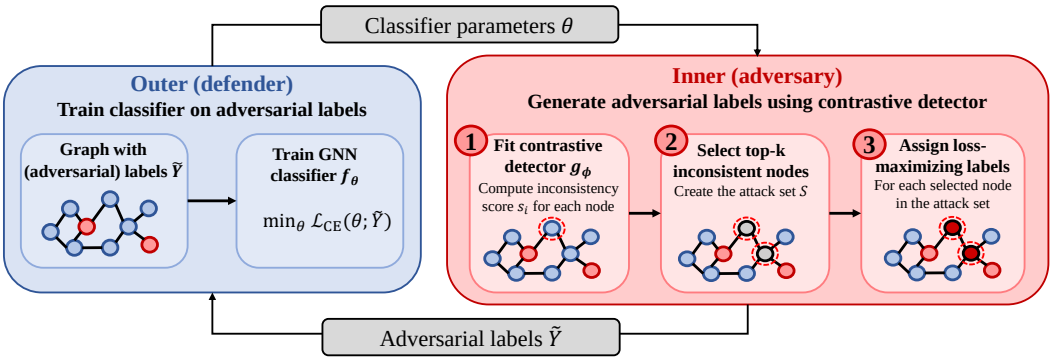


Figure 1: The bilevel optimization framework underlying CoLAT, which creates a feedback loop between the outer (defender) and inner (adversary) problems.

In Figure 1, $\tilde{Y}$ denotes the adversarial label set; g_ϕ is the contrastive detection model; s_i is the neighborhood inconsistency score of node i, computed by the contrastive detector; $\mathcal{S}$ is the resulting

Algorithm 1 CoLAT: **Co**ntrastive **L**abel-flipping for **A**dversarial **T**raining

Input: Graph $\mathcal{G} = (V, A, X)$, initial (potentially poisoned) labels $\tilde{Y}^{(0)} = \{\tilde{\mathbf{y}}_i^{(0)}\}_{i \in V_L}$, classifier f_θ, detection model g_ϕ, training epochs T, flip budget k, margin γ.

for $t = 0, \ldots, T$ **do**
  ▷ *Outer step: update classifier on current adversarial labels*
  $\theta^{(t)} \leftarrow$ Update θ by minimizing $\mathcal{L}_{\text{CE}}(f_\theta(\mathcal{G}), \tilde{Y}^{(t)})$ ▷ Eq. (2)
  ▷ *Inner step (i): fit detection model and compute inconsistency scores*
  $g_{\phi^{(t)}} \leftarrow$ Update $\phi^{(t)}$ by minimizing $\mathcal{L}_{\text{contrast}}(\phi; \tilde{Y}^{(t)})$ ▷ Eq. (5)
  Compute embeddings $\mathbf{z}_i^{\phi^{(t)}} \leftarrow g_{\phi^{(t)}}(\mathcal{G})_i$ for all $i \in V_L$
  Compute inconsistency scores $s_i(\phi^{(t)})$ ▷ Eq. (6)
  ▷ *Inner step (ii): select nodes and assign worst-case labels*
  Select attack set: $\mathcal{S}^{(t)} \leftarrow$ Top-$k\{s_i(\phi^{(t)})\}_{i \in V_L}$ ▷ Eq. (7)
  $\tilde{\mathbf{y}}_i^{(t+1)} \leftarrow \arg\max_{\bar{\mathbf{y}}_i \neq \tilde{\mathbf{y}}_i^{(t)}} \ell(\mathbf{p}_i^{\theta^{(t)}}, \bar{\mathbf{y}}_i),\ \forall i \in \mathcal{S}^{(t)}$ ▷ worst-case label assignment
end for

Output: Robust classifier $f_{\theta^{(T)}}$.

attack set formed with the top-k inconsistent nodes of the graph. In practice, the feedback loop between outer and inner problems forms *alternating updates*: we iterate between re-training the contrastive detection model g_ϕ on the current adversarial labels $\tilde{Y}$ and computing the new adversarially labeled dataset using the updated contrastive detector, as detailed in Sec. 3.2.

3.2 CoLAT Approach

CoLAT (see Algorithm 1) is a contrastive adversarial training framework that approximates the intractable minimax problem in Eq. (1) via alternating between updating the classifier (outer step) and constructing adversarial labels (inner step) for robust node classification. For notational simplicity, we drop the iteration index (t) in the following and describe a generic update step.

Classifier update. Given the adversarial labels $\tilde{Y} = \{\tilde{\mathbf{y}}_i\}_{i \in V_L}$, we train f_θ by minimizing the cross-entropy loss:

$$\min_\theta \ \mathcal{L}_{\text{CE}}\Big(f_\theta(\mathcal{G}),\, \tilde{Y}\Big) = -\frac{1}{|V_L|} \sum_{i \in V_L} \tilde{\mathbf{y}}_i^\top \log \mathbf{p}_i^\theta, \tag{2}$$

via stochastic gradient descent. Training on adversarially perturbed labels encourages the model to be robust to worst-case label poisoning (Bal et al., 2025).

Node selection via contrastive scoring. The inner maximization requires identifying which k nodes, when relabeled, cause the greatest damage to f_θ. A naive approach would score nodes by their current training loss $\ell(\mathbf{p}_\mathbf{i}^\theta, \tilde{\mathbf{y}}_i)$. However, this is unreliable: high-loss nodes may simply be hard examples rather than corrupted ones, and scores shift with updates to θ, making selection unstable. Instead, we exploit the *graph structure*. Under homophily (McPherson et al., 2001), connected nodes tend to share the same class, hence, a node whose label is inconsistent with its neighborhood is a strong candidate for adversarial corruption. Measuring this inconsistency requires a representation that captures label-neighborhood agreement, which is precisely what a *contrastive detection model* provides. Representations trained with contrastive objectives are inherently more robust to label noise than those trained with cross-entropy (Xue et al., 2022). Contrastive losses operate on *pairs* of nodes (e.g., edges), capturing relationships between neighbors. As a result, a corrupted label induces inconsistencies across its local neighborhood, amplifying its effect in the contrastive signal (Khosla et al., 2020). This noise-resilience has been exploited for robustness in the vision domain (Fang et al., 2022) and, more recently, on graphs (Li et al., 2024b; Yuan et al., 2023; Yin et al., 2023). We build on this insight but repurpose it: rather than using contrastive representations to directly train a robust classifier, we use them to *identify candidate poisoned nodes* as a guide for adversarial training.

To this end, we introduce a detection model $g_\phi \colon \mathbb{R}^d \times \{0,1\}^{n \times n} \to \mathbb{R}^{n \times d_h}$ with the same architecture as f_θ but separate parameters, mapping nodes to ℓ_2-normalised embeddings via contrastive learning:

$$\mathbf{Z}^\phi = g_\phi(\mathcal{G}) \in \mathbb{R}^{n \times d_h}, \quad \|\mathbf{z}_i^\phi\|_2 = 1 \quad \forall\, i \in V, \tag{3}$$

where $\mathbf{z}_i^\phi \in \mathbb{R}^{d_h}$ denotes the embedding of node i. The model is trained separately from f_θ so that node selection is not confounded by the classifier's current state.

Contrastive loss. We train g_ϕ with a margin-based supervised contrastive loss defined over edges (Khosla et al., 2020; Tian et al., 2020):

$$\mathcal{L}_{\text{contrast}}(\phi;\, \tilde{Y}) = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} \Big[\mathbf{1}[\tilde{\mathbf{y}}_i = \tilde{\mathbf{y}}_j]\big(1 - \mathbf{z}_i^{\phi\top}\mathbf{z}_j^\phi\big) + \mathbf{1}[\tilde{\mathbf{y}}_i \neq \tilde{\mathbf{y}}_j]\,\text{ReLU}(\mathbf{z}_i^{\phi\top}\mathbf{z}_j^\phi - \gamma)\Big], \quad (4)$$

where $\gamma \in (0, 1)$ is a margin hyperparameter. The loss enforces two geometric properties in the embedding space: *(i)* same-label neighbors are pulled together ($\mathbf{z}_i^{\phi\top}\mathbf{z}_j^\phi \to 1$); *(ii)* different-label neighbors are pushed apart beyond margin γ. *Poisoned* nodes violate this structure: their assigned label conflicts with their neighborhood, causing them to incur higher contrastive penalties. Consequently, their embeddings become poorly aligned with neighbors, leading to higher inconsistency scores. This is the geometric intuition behind using $\mathcal{L}_{\text{contrast}}$ as a natural proxy for adversarial corruption. Overall, the detection model g_ϕ is trained and updated given adversarial labels $\tilde{Y}$ via:

$$\phi^* = \arg\min_\phi \mathcal{L}_{\text{contrast}}(\phi;\, \tilde{Y}). \quad (5)$$

Inconsistency (noise) score. Given the optimal detection model parameters ϕ, which minimizes Eq. (4), the inconsistency score of node i is the average contrastive distance to its neighbors:

$$s_i(\phi) = \frac{1}{\deg(i)} \sum_{j\in\mathcal{N}(i)} \ell_{\text{contrast}}(\mathbf{z}_i^\phi,\, \mathbf{z}_j^\phi), \quad (6)$$

where $\ell_{\text{contrast}}(\mathbf{z}_i, \mathbf{z}_j)$ is the contribution of edge (i, j) to the contrastive loss in Eq. (4). Normalizing by $\deg(i)$ ensures the score is comparable across nodes with different degrees (Hamilton et al., 2017). We further apply min-max normalization across V_L to obtain scores in $[0, 1]$ with $\varepsilon > 0$ for stability.

The attack and training. The k nodes with the highest scores are selected as the attack set:

$$\mathcal{S} = \text{Top-}k\{s_i(\phi)\}_{i\in V_L} \subseteq V_L, \quad |\mathcal{S}| = k. \quad (7)$$

Given the attack set $\mathcal{S}$, we assign each selected node the label that maximally increases the current classification loss. Specifically, for each $i \in \mathcal{S}$:

$$\tilde{\mathbf{y}}_i \leftarrow \arg\max_{\bar{\mathbf{y}}_i \neq \tilde{\mathbf{y}}_i} \ell(\mathbf{p}_i^\theta, \bar{\mathbf{y}}_i). \quad (8)$$

The constraint $\bar{\mathbf{y}}_i \neq \tilde{\mathbf{y}}_i$, with $\bar{\mathbf{y}}_i$ restricted to one-hot vectors, ensures the assignment is a *genuine perturbation* away from the current label, not merely a re-flip of an already-perturbed label. These steps are interleaved in an alternating optimisation loop, summarised in Algorithm 1. The alternating scheme resolves the dependency between ϕ and $\tilde{Y}$ and provides a tractable approximation to the intractable inner maximization.

4 Related Work

A substantial body of work studies adversarial attacks on GNNs, but primarily focuses on evasion and poisoning via *topological perturbations* (Alom et al., 2025; Zang et al., 2020; Zügner and Günnemann, 2019a; Zügner et al., 2018; Dai et al., 2018). In contrast, *label poisoning attacks* remain underexplored. These training-time attacks corrupt a small subset of node labels to degrade performance, and are typically addressed via gradient-based methods (Liu et al., 2022) or by approximating GNNs using linearization or neural tangent kernel (NTK) formulations to make the resulting bilevel optimization tractable (Lingam et al., 2024; Zhang et al., 2020). Accordingly, a wide range of defense strategies have been proposed, predominantly targeting structural perturbations (Yuan et al., 2024; Li et al., 2024a; Entezari et al., 2020; Zhang and Zitnik, 2020; Jin et al., 2020; You et al., 2020). Only a limited subset of works explicitly considers robustness to label poisoning (Zhang et al., 2020) and certification approaches (Sabanayagam et al., 2025). While adversarial training for label poisoning has been studied in standard classification settings (Bal et al., 2025; Biggio et al., 2012), it remains largely unexplored in the graph setting. A related line of work studies *learning under label noise*,

where labels are corrupted randomly (e.g., via uniform flips) rather than adversarially. In contrast, adversarial label poisoning is inherently *data-dependent*, with label flips optimized to maximally degrade model performance. We refer the reader to the comprehensive survey in Wang et al. (2024). Existing approaches for noisy labels typically rely on loss regularization (Du et al., 2023; Zhang et al., 2020), graph structure augmentation (Dai et al., 2021; Qian et al., 2023), or contrastive learning (Xue et al., 2022; Yuan et al., 2023). While contrastive learning has shown strong robustness to label noise, existing methods treat contrastive learning primarily as a representation regularizer rather than a mechanism for *adversarial label set detection*. To the best of our knowledge, CoLAT is the first method to integrate contrastive detection with adversarial training for defense against label poisoning.

5 Experiments

Models and datasets. We report results using the GCN architecture for the classifier and detection models, and evaluate the methods on five benchmark datasets used in prior literature: Cora ML, Citeseer, PubMed (Sen et al., 2008; Namata et al., 2012), Amazon Photo (Shchur et al., 2018), OGBN-ArXiv (Wang et al., 2020), and Actor (Tang et al., 2009). In Sec. 5.2, we provide additional results on Graph Attention Network (GAT) (Veličković et al., 2018) and GraphSAGE (Hamilton et al., 2017). Dataset and architecture details are provided in Sec. A.1 and Sec. A.4 in the appendix.

Adversarial label poisoning attacks. We evaluate our defense against prior adversarial label poisoning attacks, including binary methods (LafAK (Zhang et al., 2020), NTK (Lingam et al., 2024)) and the multi-class attack SMulti (Lingam et al., 2024). They approximate the bilevel problem using linear or kernel surrogates, and do not significantly alter graph homophily (Lingam et al., 2024).

Baselines. We compare our method against a vanilla GNN (referred to as clean training) as well as existing defense for adversarial label poisoning, CP (Zhang et al., 2020). Defenses specifically designed for adversarial label poisoning are relatively scarce. In contrast, several methods have been proposed to address pairwise label noise, where labels are randomly flipped according to a fixed probability, independent of the input features. Adversarial label poisoning instead involves strategically chosen, data-dependent flips designed to maximally degrade model performance. Despite this distinction, and following Lingam et al. (2024), we consider pairwise noise defenses as the closest available baselines for label-flipping attacks. Accordingly, we evaluate against PIGNN (Du et al., 2023), CRGNN (Li et al., 2024b), and RTGNN (Qian et al., 2023), selected based on the review in Wang et al. (2024). We exclude methods that rely on data cleaning prior to training and instead focus on similar robust training strategies, including loss regularization (e.g., CP, PIGNN), contrastive learning (e.g., CRGNN), and iterative label correction (e.g., RTGNN). Our proposed defense also draws on contrastive learning and iterative label flipping, which motivates the selection of these baselines as the most relevant points of comparison. Additional details are provided in Sec. A.2.

Performance metrics. We assess our method using two key metrics: robust and clean accuracy, tracked over a test set with clean labels during training. Robust accuracy reflects model performance tested on clean labels using adversarially poisoned training data. Conversely, clean accuracy measures the performance of models trained and tested on clean-labeled data, offering a benchmark for comparison under both adversarial and non-adversarial conditions.

5.1 Experimental Results

In this section, we report the performance of CoLAT against the baseline methods on Cora ML, Citeseer, and Pubmed datasets against NTK and SMulti attacks. Results on Amazon Photo dataset and the LafAK attack are included in Sec. C.1 and Sec. C.4 in the appendix. As reported in Table 1, CoLAT achieves higher robust accuracy across almost all settings compared to baseline methods. Notably, in scenarios with more severe poisoning levels such as 20 % and 30 % NTK poisoning on Cora ML and Citeseer datasets, CoLAT significantly outperforms all baselines, which experience a marked drop in their accuracy.

When the initial training data is clean, CoLAT provides a proactive defense by introducing adversarial labels during training, thereby effectively reducing the model's sensitivity to potential label attacks. Notably, CoLAT matches performance on par with vanilla GCN on clean data, demonstrating that its robust framework maintains high accuracy without compromising clean accuracy. In all cases, CoLAT even outperforms vanilla GCN on clean data, suggesting that adversarial training by CoLAT can improve generalizability.

Table 1: Test accuracy comparison of CoLAT against baselines and clean training (the undefended GCN baseline) on Cora ML, Citeseer, and PubMed under clean and poisoned labels, all using a GCN architecture. Results are reported for NTK and SMulti across various attack budgets. Red indicates the best test accuracy (stdev) in each setting. CoLAT performs best across most settings.

Dataset	Training	Clean Dataset	NTK				SMulti			
			5%	10%	20%	30%	5%	10%	20%	30%
Cora ML	Clean	0.8270 (0.005)	0.7675 (0.030)	0.6470 (0.033)	0.5378 (0.022)	0.4626 (0.008)	0.7592 (0.021)	0.6673 (0.035)	0.4932 (0.061)	0.3561 (0.032)
	PIGNN	0.8236 (0.0045)	0.7985 (0.011)	0.6727 (0.017)	0.5163 (0.020)	0.4581 (0.087)	0.7887 (0.004)	0.7453 (0.027)	0.6912 (0.024)	0.4891 (0.081)
	RTGNN	0.6297 (0.060)	0.6057 (0.054)	0.5891 (0.040)	0.4784 (0.088)	0.3817 (0.071)	0.6600 (0.035)	0.5739 (0.110)	0.5154 (0.042)	0.4364 (0.040)
	CRGNN	0.8088 (0.014)	0.7459 (0.037)	0.6312 (0.022)	0.5429 (0.017)	0.4636 (0.010)	0.7263 (0.030)	0.6211 (0.040)	0.4927 (0.010)	0.3415 (0.028)
	CP	0.8122 (0.019)	0.7663 (0.025)	0.65587 (0.049)	0.5334 (0.02)	0.4657 (0.01)	0.7281 (0.045)	0.6972 (0.02)	0.5802 (0.056)	0.3760 (0.042)
	CoLAT	0.8377 (0.006)	0.8309 (0.007)	0.7774 (0.02)	0.6689 (0.04)	0.5342 (0.014)	0.8226 (0.017)	0.7879 (0.05)	0.7466 (0.02)	0.6739 (0.05)
Citeseer	Clean	0.7410(0.003)	0.7399 (0.004)	0.6722 (0.01)	0.3682 (0.024)	0.3319 (0.004)	0.6840 (0.031)	0.6742 (0.012)	0.6167 (0.051)	0.4100 (0.067)
	PIGNN	0.6920 (0.028)	0.7266 (0.027)	0.6892 (0.028)	0.3819 (0.005)	0.3565 (0.012)	0.7328 (0.006)	0.7126 (0.005)	0.6981 (0.016)	0.4729 (0.076)
	RTGNN	0.6386 (0.021)	0.6831 (0.035)	0.6765 (0.043)	0.3328 (0.017)	0.3144 (0.045)	0.6268 (0.047)	0.6972 (0.011)	0.5941 (0.028)	0.4300 (0.026)
	CRGNN	0.7082 (0.019)	0.6831 (0.035)	0.595 (0.012)	0.3709 (0.027)	0.3559 (0.019)	0.6579 (0.02)	0.6199 (0.033)	0.4387 (0.073)	0.3087 (0.037)
	CP	0.7070 (0.013)	0.6446 (0.05)	0.5174 (0.067)	0.3693 (0.020)	0.3386 (0.017)	0.6679 (0.034)	0.6209 (0.03)	0.5657 (0.085)	0.3735 (0.09)
	CoLAT	0.7420 (0.011)	0.7565 (0.0067)	0.7287 (0.011)	0.6668 (0.039)	0.4286 (0.057)	0.7095 (0.01)	0.7142 (0.013)	0.6200 (0.05)	0.5937 (0.039)
Pubmed	Clean	0.7940(0.010)	0.7349 (0.013)	0.6935 (0.026)	0.4820 (0.073)	0.3520 (0.04)	0.7484 (0.026)	0.7429 (0.007)	0.6016 (0.12)	0.4989 (0.077)
	PIGNN	0.8095 (0.011)	0.7691 (0.051)	0.7204 (0.042)	0.4957 (0.05)	0.4569 (0.067)	0.7781 (0.018)	0.7983(0.013)	0.6902 (0.052)	0.4657 (0.09)
	RTGNN	0.7173 (0.023)	0.7234(0.018)	0.6916 (0.036)	0.5557 (0.042)	0.3874 (0.063)	0.7325 (0.040)	0.7172(0.021)	0.5394 (0.021)	0.5122 (0.072)
	CRGNN	0.7371 (0.07)	0.7276 (0.036)	0.7396 (0.033)	0.5527 (0.047)	0.3798 (0.015)	0.7156 (0.069)	0.7452 (0.064)	0.6210 (0.042)	0.4382 (0.045)
	CP	0.7875 (0.037)	0.7545 (0.024)	0.7420 (0.027)	0.4975 (0.017)	0.3963 (0.052)	0.7597 (0.008)	0.7343(0.017)	0.6308(0.046)	0.5465(0.042)
	CoLAT	0.8171 (0.009)	0.7697 (0.027)	0.7712 (0.037)	0.7452 (0.055)	0.7636 (0.021)	0.7897 (0.029)	0.7844 (0.017)	0.7488 (0.049)	0.7114 (0.038)

Sanitization effect. In scenarios with already poisoned training data, CoLAT achieves robustness through two key mechanisms: (i) implicitly *sanitizing* corrupted labels and (ii) introducing *additional adversarial labels* to further reduce model sensitivity to attacks.

We analyze these effects in Figure 2. Panels (a)–(c) illustrate the sanitization effect under SMulti 10% attack on Cora ML. During training, CoLAT identifies and restores 36% of the poisoned labels to their clean counterparts while also introducing some additional adversarial labels (purple nodes), resulting in a 12% improvement in test accuracy.

Furthermore, CoLAT operates on a dynamically evolving supervision signal throughout training, whereas some baseline methods rely on fixed poisoned labels throughout. Methods that reweigh nodes, such as PIGNN, or regularize representations, such as CRGNN, do not modify the underlying supervision and therefore remain fully exposed to the corrupted labels. CP attempts label restoration through a one-shot neighborhood aggregation prior to training and cannot adapt once training begins. By contrast, CoLAT updates the supervision signal at each epoch based on both structural inconsistency and gradient contribution, continuously exposing the model to shifting and adversarially grounded corruption patterns. This dynamic correction process allows CoLAT to adapt beyond the observed poison distribution and improve robustness under poisoning.

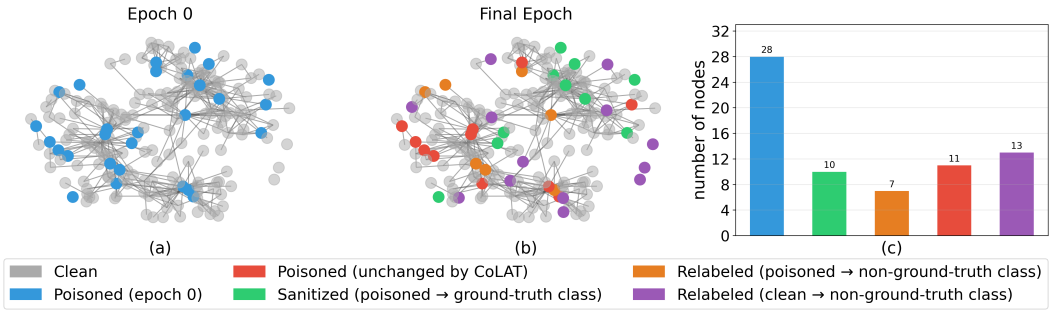


Figure 2: Effectiveness analysis of CoLAT under the SMulti 10% attack on Cora ML. Of the 28 initially poisoned nodes, CoLAT sanitizes 10 to their ground-truth labels and relabels 7 to non-ground-truth classes while also introducing additional adversarial labels by flipping 13 clean node labels.

Scaling the defense. We further evaluate CoLAT on OGBN-ArXiv, a substantially larger dataset [2]. As shown in Table 2, clean training degrades sharply with increasing poisoning, whereas CoLAT maintains strong performance and achieves clear gains under severe attacks. These results demonstrate that CoLAT scales effectively while remaining robust to high levels of adversarial label poisoning.

[2]NTK and SMulti attack implementations do not scale directly to graphs of this size. To address this, we partition the graph into smaller subgraphs, apply the attacks independently, and then combine the perturbed subsets. This procedure provides an approximate realization of the attacks on large-scale graphs.

Table 2: Scaling results of CoLAT on a larger dataset. CoLAT achieves the best performance in almost all poisoned settings on the OBGN-ArXiv dataset.

Training	Clean Dataset	NTK Poisoned				SMulti Poisoned			
		5%	10%	20%	30%	5%	10%	20%	30%
Clean	0.7086(0.035)	0.7001(0.005)	0.6839(0.005)	0.5645(0.023)	0.4741(0.005)	0.6803(0.006)	0.6437(0.008)	0.3923(0.005)	0.207(0.029)
CoLAT	0.7055(0.025)	0.7037(0.014)	0.6806(0.0008)	0.6216(0.008)	0.6184(0.014)	0.6819(0.006)	0.6571(0.003)	0.4831(0.010)	0.2485(0.028)

CoLAT is also more scalable than several baselines. Among the most competitive methods from Table 1, PIGNN relies on a dense $n \times n$ adjacency reconstruction via an inner product decoder, making it impractical for large graphs (e.g., 169k nodes in OGBN-ArXiv) without subgraph sampling, while CRGNN incurs $O(n^2)$ complexity when computing contrastive loss over all node pairs. In contrast, CoLAT operates over edges, yielding $O(\mathcal{E})$ complexity and enabling efficient application to large-scale sparse graphs without architectural modifications. This advantage is reflected not only in memory efficiency but also in reduced training time, as shown in Figure 8 in Appendix E.

Classification under heterophilic datasets. As reported in Table 3, at low poisoning rates on the Actor dataset (Tang et al., 2009), CoLAT performs similarly to clean training under NTK poisoning. However, under the SMulti attack, its effectiveness becomes more evident (e.g., 0.2435 vs. 0.1997 at 10% poisoning, and 0.1444 vs. 0.1079 at 30% poisoning). Although the gains are smaller than those observed on homophilic graphs, these results should be interpreted in light of Actor's heterophilic structure. CoLAT's contrastive noise score treats same-label neighbors as positives, which provides only a sparse signal in heterophilic graphs, where adjacent nodes often have different labels. Moreover, the Actor dataset exhibits weak and noisy structural signal: its edges reflect actor co-occurrence rather than label similarity, which limits neighborhood predictiveness and reduces the usefulness of graph structure as a prior. Adapting CoLAT more naturally to heterophilic graphs would likely require modifying the contrastive objective so that it considers all nodes sharing the same label, rather than only connected same-label nodes. However, this would largely remove the role of graph structure. We investigate this further in Table 15 in Appendix D.

Under the current contrastive formulation, CoLAT is better suited to homophilic graphs. Nevertheless, even with the limited graph signal available in Actor, CoLAT's adversarial label flips still provide robustness gains under strong multiclass attacks such as SMulti.

Table 3: Test accuracy comparison of CoLAT and clean training on the heterophilic Actor dataset under NTK and SMulti adversarial label poisoning attacks.

Training	Clean	NTK Poisoned				SMulti Poisoned			
		5%	10%	20%	30%	5%	10%	20%	30%
Clean	0.2942 (0.008)	0.2849 (0.012)	0.2732 (0.009)	0.2483 (0.007)	0.2179 (0.007)	0.2312 (0.012)	0.1997 (0.016)	0.1429 (0.006)	0.1079 (0.009)
CoLAT	0.2777 (0.014)	0.2756 (0.014)	0.2688 (0.021)	0.2409 (0.006)	0.2381 (0.011)	0.2639 (0.017)	0.2435 (0.015)	0.1690 (0.005)	0.1444 (0.013)

5.2 Ablation Studies and Discussion

Performance on different GNN architectures. We evaluate CoLAT across different GNN architectures on the Cora ML dataset, as shown in Figure 3. Both the detection model and the classification model use either a GAT or GraphSAGE backbone. Overall, CoLAT exhibits similar behavior on GAT and GraphSAGE as observed with GCN, with more noticeable improvements under stronger attacks. For GraphSAGE, the defense remains effective despite its stochastic neighborhood sampling, which may omit contributions from poisoned labels. Notably, under SMulti, GraphSAGE shows the largest relative improvement compared to GAT and GCN, exceeding 72% accuracy across all poisoning rates and achieving up to a 31% gain at 30% poisoning. In contrast, CRGNN, which also relies on contrastive learning, does not adapt as well to architectural changes and in several cases performs worse than clean training. This further suggests that CoLAT is robust to both architectural variations and the stochasticity introduced by neighborhood sampling.

Sensitivity to contrastive detection model. We replace the detection model with a simpler heuristic that flips the labels of nodes based on their contribution to the classification loss to assess its importance. The simpler heuristic uses logits as a proxy to select nodes, rather than relying on the contrastive objective. Note that this heuristic coarsely approximates the inner maximization in a purely node-wise and local manner, without accounting for structural dependencies. The logit-based

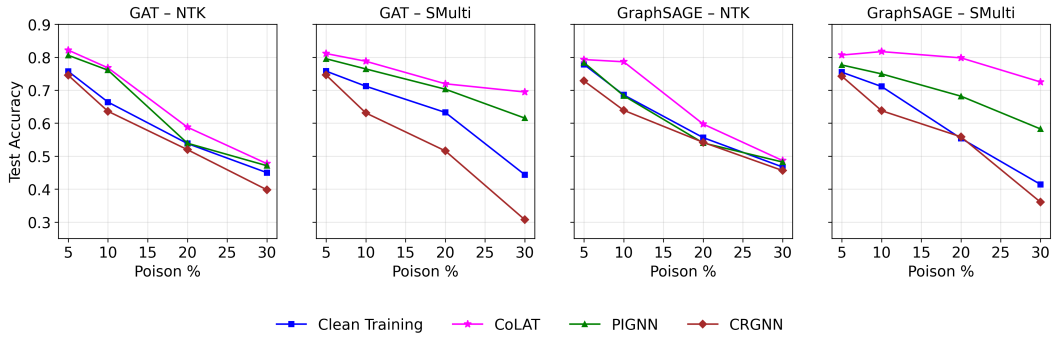


Figure 3: Test accuracy as a function of poisoning rate under NTK and SMulti attacks for GAT and GraphSAGE, to assess CoLAT's performance across different architectures.

heuristic yields only marginal improvements over clean training (see Figure 4).For example, on Cora ML at 30% poisoning, the logit-based heuristic yields only a 0.4% gain under NTK, compared with 13% using the detection model. Under SMulti, it achieves about 5%, whereas the detection model reaches up to 31%. This comparison demonstrates that the contrastive loss is a key component of our structurally guided scoring mechanism, whereas logit-based selection using classification loss is insufficient to capture influential adversarial nodes.

Moreover, we highlight that the detection model and the classification model do not need to share the same architecture, as shown in Table 15 in Sec. D. This makes CoLAT a flexible framework.

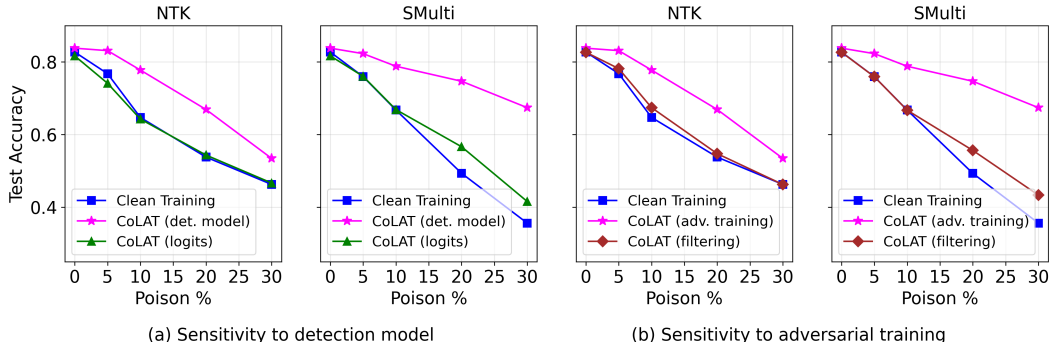


Figure 4: Sensitivity analysis of CoLAT under NTK and SMulti attacks. We assess two ablations as the poisoning rate increases: replacing the contrastive detection model with a logit-based heuristic, and replacing adversarial training with one-step label filtering.

Sensitivity to adversarial training. We evaluate the contribution of adversarial training by ablating this component and instead using the detection model as a one-step filtering mechanism. We first train the detection model on poisoned labels for 100 epochs on the Cora ML dataset and use it to identify and flip the labels once before training the classification model with early stopping on the filtered labels. As shown in Figure 4, this filtering-based approach yields only marginal improvements over clean training. In contrast, adversarial training yields much larger gains. Under NTK with 20% poisoning, it improves performance by about 13%, compared with roughly 1% for filtering. Under SMulti at 20%, adversarial training achieves around 26%, whereas filtering reaches only about 8%. These results highlight the importance of the iterative optimization in the bilevel min-max attacker-defender formulation, where the defender is repeatedly updated in response to adversarial perturbations, rather than relying on a single adversarially labeled configuration of the dataset.

Sensitivity to update interval of detection model. For large datasets such as OGBN-ArXiv, we update the detection model at a lower rate than the classification model to reduce training time, using update intervals of $1, 5, 10$, and 20 as presented in Figure 5. Here, the update interval δ denotes that we perform one gradient descent step on the detection model for every δ gradient descent steps on

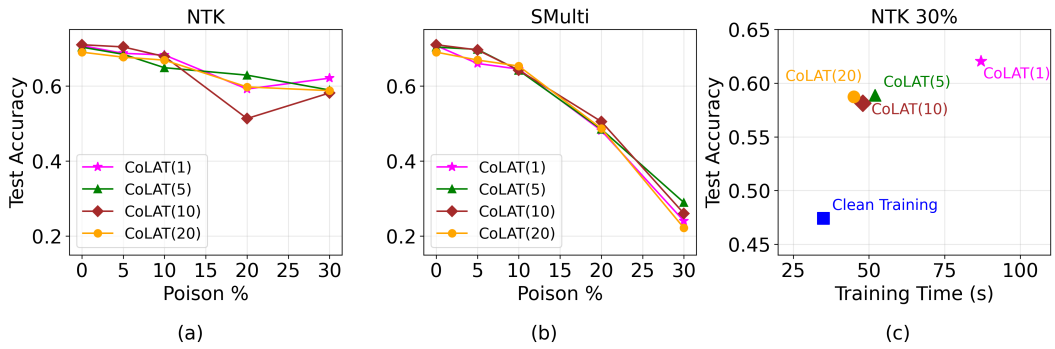


Figure 5: Sensitivity of CoLAT to the detection-model update interval on OGBN ArXiv. We consider update intervals $\delta \in \{1, 5, 10, 20\}$, where the detection model is updated once every δ classification-model updates. Panels (a) and (b) show test accuracy under NTK and SMulti attacks across poisoning rates, and panel (c) presents the training time-accuracy trade-off at 30% NTK poisoning.

the classification model. We observe that this approach still achieves improved test accuracy on label poisoning attacks, with significant improvement over clean training alongside reduced training time.

Ablation on Node Selection. We ablate our choice of selecting the most inconsistent nodes by instead selecting the most consistent nodes. As shown in Table 4, selecting inconsistent nodes consistently yields substantially better performance across both NTK and SMulti attacks and all poisoning rates on Cora-ML. This supports our design choice of prioritizing inconsistent nodes.

Table 4: Ablation on node selection on Cora-ML. Selecting the most inconsistent nodes consistently outperforms selecting the most consistent nodes across attacks and poisoning rates.

Training	Clean Dataset	NTK Poisoned				SMulti Poisoned			
		5%	10%	20%	30%	5%	10%	20%	30%
Clean	0.8270 (0.005)	0.7675 (0.030)	0.6470 (0.033)	0.5378 (0.022)	0.4626 (0.008)	0.7592 (0.021)	0.6673 (0.035)	0.4932 (0.061)	0.3561 (0.032)
CoLAT (select consistent nodes)	0.6871 (0.095)	0.7159 (0.025)	0.5736 (0.046)	0.4978 (0.027)	0.3806 (0.096)	0.7253 (0.014)	0.6054 (0.033)	0.4918 (0.050)	0.3757 (0.041)
CoLAT (select inconsistent nodes)	0.8377 (0.006)	0.8309 (0.007)	0.7774 (0.02)	0.6689 (0.04)	0.5342 (0.014)	0.8226 (0.017)	0.7879 (0.05)	0.7466 (0.02)	0.6739 (0.05)

6 Conclusion

We propose CoLAT, a contrastive-adversarial framework for robust node classification under label poisoning. By coupling contrastive, structure-aware detection with adversarial label flipping in a bilevel formulation, CoLAT identifies nodes that are most inconsistent with their local graph context and concentrates training on their worst-case perturbations. This yields an efficient alternating optimization scheme that improves robustness without degrading clean accuracy. While no defense is universally robust across all attack settings (Papernot et al., 2016), we evaluate CoLAT against a diverse range of label poisoning strategies and benchmarks from the literature, where it consistently achieves improved robustness. Overall, our results suggest that incorporating structural consistency into adversarial training is a promising direction for improving the resilience of graph learning models to label corruption.

Acknowledgments and Disclosure of Funding

We thank the German Research Foundation for the support. This research was also supported by the Max Planck & Amazon Science Hub.

References

Alom, Z., Ngo, T. G. B., Kantarcioglu, M., and Akcora, C. G. (2025). GOttack: Universal adversarial attacks on graph neural networks via graph orbits learning. In *International Conference on Learning Representations*.

Bal, M. I., Cevher, V., and Muehlebach, M. (2025). Adversarial training for defense against label poisoning attacks. In *International Conference on Learning Representations*.

Biggio, B., Nelson, B., and Laskov, P. (2012). Poisoning attacks against support vector machines. In *International Conference on Machine Learning*.

Dai, E., Aggarwal, C., and Wang, S. (2021). NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial attack on graph structured data. In *International Conference on Machine Learning*, volume 80, pages 1115–1124.

Du, X., Bian, T., Rong, Y., Han, B., Liu, T., Xu, T., Huang, W., Li, Y., and Huang, J. (2023). Noise-robust graph learning by estimating and leveraging pairwise interactions. *Transactions on Machine Learning Research*.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. (2020). All you need is low (rank): Defending against adversarial attacks on graphs. In *International Conference on Web Search and Data Mining*.

Fang, A., Ilharco, G., Wortsman, M., Wan, Y., Shankar, V., Dave, A., and Schmidt, L. (2022). Data determines distributional robustness in contrastive language image pre-training (CLIP). In *International Conference on Machine Learning*, pages 6216–6234.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2015). Explaining and harnessing adversarial examples. *ArXiv*, 1412.6572.

Gosch, L., Geisler, S., Sturm, D., Charpentier, B., Zügner, D., and Günnemann, S. (2023). Adversarial training for graph neural networks: Pitfalls, solutions, and new directions. *Advances in Neural Information Processing Systems*, 36:58088–58112.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., and Tang, J. (2020). Graph structure learning for robust graph neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 66–74.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *ArXiv*, 1609.02907.

Li, K., Chen, Y., Liu, Y., Wang, J., He, Q., Cheng, M., and Ao, X. (2024a). Boosting the adversarial robustness of graph neural networks: An OOD perspective. In *International Conference on Learning Representations*.

Li, X., Li, Q., Li, D., Qian, H., and Wang, J. (2024b). Contrastive learning of graphs under label noise. *Neural Networks*, 172:106113.

Lingam, V., Akhondzadeh, M. S., and Bojchevski, A. (2024). Rethinking label poisoning for GNNs: Pitfalls and attacks. In *International Conference on Learning Representations*.

Liu, G., Huang, X., and Yi, X. (2022). Adversarial label poisoning attack on graph neural networks via label propagation. In *European Conference on Computer Vision*.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *ArXiv*, 1706.06083.

McAuley, J., Targett, C., Shi, Q., and van den Hengel, A. (2015). Image-based recommendations on styles and substitutes. In *ACM SIGIR Conference on Research and Development in Information Retrieval*.

McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual Review of Sociology*, 27(1):415–444.

Mujkanovic, F., Geisler, S., Günnemann, S., and Bojchevski, A. (2022). Are defenses for graph neural networks robust? *Advances in Neural Information Processing Systems*, 35:8954–8968.

Namata, G., London, B., Getoor, L., Huang, B., and Edu, U. (2012). Query-driven active surveying for collective classification. In *International Workshop on Mining and Learning with Graphs*, volume 8, page 1.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *ArXiv*, 1807.03748.

Papernot, N., McDaniel, P., and Goodfellow, I. (2016). Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *ArXiv*, 1605.07277.

Platonov, O., Kuznedelev, D., Diskin, M., Babenko, A., and Prokhorenkova, L. (2024). A critical look at the evaluation of gnns under heterophily: Are we really making progress? *ArXiv*.

Qian, S., Ying, H., Hu, R., Zhou, J., Chen, J., Chen, D. Z., and Wu, J. (2023). Robust training of graph neural networks via noise governance. In *ACM International Conference on Web Search and Data Mining*.

Robey, A., Latorre, F., Pappas, G., Hassani, H., and Cevher, V. (2024). Adversarial training should be cast as a non-zero-sum game. In *International Conference on Learning Representations*.

Sabanayagam, M., Gosch, L., Günnemann, S., and Ghoshdastidar, D. (2025). Exact certification of (graph) neural networks against label poisoning. In *International Conference on Learning Representations*.

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

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *ArXiv*, abs/1811.05868.

Sun, L., Dou, Y., Yang, C., Zhang, K., Wang, J., Yu, P. S., He, L., and Li, B. (2022). Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):7693–7711.

Tang, J., Sun, J., Wang, C., and Yang, Z. (2009). Social influence analysis in large-scale networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, page 807–816.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive multiview coding. In *European Conference on Computer Vision*, pages 776–794.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al. (2018). Graph attention networks. In *International Conference on Learning Representations*.

Wang, K., Shen, Z., Huang, C., Wu, C.-H., Dong, Y., and Kanakia, A. (2020). Microsoft academic graph: When experts are not enough. *Quantitative Science Studies*, 1:396–413.

Wang, Z., Sun, D., Zhou, S., Wang, H., Fan, J., Huang, L., and Bu, J. (2024). NoisyGL: A comprehensive benchmark for graph neural networks under label noise. In *Advances in Neural Information Processing Systems*.

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

Xue, Y., Whitecross, K., and Mirzasoleiman, B. (2022). Investigating why contrastive learning benefits robustness against label noise. In *International Conference on Machine Learning*, pages 24851–24871.

Yin, N., Shen, L., Wang, M., Luo, X., Luo, Z., and Tao, D. (2023). OMG: Towards effective graph classification against label noise. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12873–12886.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., and Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823.

Yuan, J., Luo, X., Qin, Y., Zhao, Y., Ju, W., and Zhang, M. (2023). Learning on graphs under label noise. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

Yuan, X., Zhang, C., Tian, Y., Ye, Y., and Zhang, C. (2024). Mitigating emergent robustness degradation while scaling graph learning. In *International Conference on Learning Representations*.

Zang, X., Xie, Y., Chen, J., and Yuan, B. (2020). Graph universal adversarial attacks: A few bad actors ruin graph learning models. *ArXiv*, 2002.04784.

Zhang, M., Hu, L., Shi, C., and Wang, X. (2020). Adversarial label-flipping attack and defense for graph neural networks. In *IEEE International Conference on Data Mining*, pages 791–800.

Zhang, X. and Zitnik, M. (2020). GNNGuard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems*, 33:9263–9275.

Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Zügner, D. and Günnemann, S. (2019a). Adversarial attacks on graph neural networks via meta learning. In *International Conference on Learning Representations*.

Zügner, D. and Günnemann, S. (2019b). Certifiable robustness and robust training for graph convolutional networks. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 246–256.

A Experimental Details

A.1 Datasets

We follow the exact data splits of Lingam et al. (2024) for Cora ML, Citeseer and Pubmed datasets and adopt their evaluation protocol. Their work highlights key pitfalls in existing adversarial attack evaluations, particularly the common assumption of a clean validation set, which is unrealistic in label poisoning attacks. We extract the largest connected component in each dataset and report the details in Table 5.

Table 5: Dataset statistics for the benchmark graphs used in our experiments.

Dataset	**Nodes**	**Edges**	**Features**	**Classes**	**Homophily**	**Train/Val/Test**		
						Train	Val	Test
Cora ML	2810	9386	2879	7	0.81	140	140	2530
Citeseer	2110	4723	3703	6	0.74	120	120	1870
Pubmed	19717	44324	500	3	0.80	60	60	19597
Amazon Photos	7640	238162	745	8	0.80	160	240	7250
OGBN-ArXiv	169343	1166243	128	40	0.63	90941	29799	48603
Coauthor-Physics	34,493	495,924	8,415	5	0.93	100	150	34,243
Coauthor-CS	18,333	163,788	6,805	15	0.81	300	450	17,583
Actor	7600	34259	932	5	0.22	4028	2432	1140

Coauthor datasets split are stratified by classes in 20 per class/30 per class/ rest for train/val/test as in theoriginal paper (Shchur et al., 2018).

A.2 Baselines

CP (Zhang et al., 2020) investigates the impact of adversarial label-flipping attacks on the generalization performance of GCNs. To mitigate such attacks, the authors propose a defense framework that leverages community-level labels as auxiliary supervision for node classification. Specifically, the method introduces a community-based regularization term to discourage overfitting to corrupted labels.

PIGNN (Du et al., 2023) improves robustness to label noise by incorporating pairwise label information into GNN training. The key idea is that pairwise relationships are more reliable than individual node labels. Specifically, a pairwise interaction (PI) label is defined as 1 if two nodes share the same class and 0 otherwise. The method estimates these PI labels based on the similarity of node embeddings. During training, the estimated PI labels are used as confidence signals to guide the node classifier, thereby regularizing learning and reducing the impact of noisy labels.

RTGNN (Qian et al., 2023) combines graph refinement with label correction during training. It constructs a refined graph based on feature similarity to enhance supervision and employs a small-loss criterion to identify trustworthy nodes, leveraging the observation that clean samples tend to incur lower training loss. The method then iteratively updates the training process by emphasizing reliable nodes and correcting the labels of likely corrupted ones. This joint refinement of graph structure and labels helps mitigate the propagation of noise and improves robustness.

CRGNN (Li et al., 2024b) leverages contrastive learning to improve the robustness of GNNs to noisy and sparse labels. It generates augmented graph views through feature masking and edge perturbations while preserving node semantics. A contrastive objective is used to capture local structural information and reduce the impact of noisy labels, while a dynamic cross-entropy loss mitigates overfitting. Additionally, a cross-space consistency constraint enforces alignment between different embedding spaces.

A.3 Hyperparameters

We tune both CoLAT and all baselines using the hyperparameter search ranges specified in Table 6. For the baselines, we follow the implementations and hyperparameter search spaces using the official

codebase[3], provided by Wang et al. (2024), which offers a unified benchmark and standardized experimental settings for fair comparison. For each method, we perform 50 hyperparameter trials and report results averaged over 5 random seeds. Hyperparameters are selected based on validation performance and fixed across all experiments, with early stopping applied to CoLAT during training.

Table 6: Hyperparameter search space for all methods.

Algorithm	**Hyper-parameter**	**Search Space**
General Settings	learning rate	1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 0.1
	weight decay	1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2, 5e-2, 0.1
	replications	5
CP	λ	0.1, 0.2, 0.3
RTGNN	τ	0, 0.05, 0.1, 0.2
	λ	0.05, 0.1, 0.2
	α	0.03, 0.1, 0.3, 1
PIGNN	N/A	N/A
CRGNN	α	0.1, 0.2, 0.3, 0.5, 1
	β	0.1, 0.2, 0.3, 0.5, 1
CoLAT	k (flip budget)	3%, 5%, 10%
	γ (margin)	0.3, 0.5, 0.7

A.4 Model Architecture and Training Details

For datasets Cora ML, Citeseer, PubMed, and Actor, we use standard GNN architectures (GCN, GAT, and GraphSAGE) with two layers, hidden dimension 16, and dropout rate 0.5, trained for 200 epochs for all baselines. For the larger OGBN-Arxiv dataset, we adopt a deeper and wider architecture to account for its scale and complexity. Specifically, we use three layers with a hidden dimension of 256 and train for 500 epochs. For all settings, we performed five replications.

All models use the same architecture across methods to ensure fair comparison. All experiments are conducted on A100 or H100 GPUs when GPU acceleration is required. Exact details are specified where applicable. Unless otherwise noted, training times are reported on an A100-SXM4-80GB GPU.

B Effectiveness Analysis

As shown earlier in Figure 2 and in Figure 6, CoLAT achieves robustness through two key mechanisms: (i) implicit sanitization of corrupted labels and (ii) the introduction of adversarial labels that reduce the model's sensitivity to attacks.

Here, we highlight an edge case shown in Figure 7, where no explicit sanitization occurs. Despite the absence of label correction, the model still benefits from proactively introduced adversarial labels under the NTK 10% poisoning setting, yielding a 10% improvement in test accuracy.

This result demonstrates that CoLAT does not rely solely on label sanitization, but improves robustness through the adversarial training effect, leading to a more resilient model even when corrupted labels remain uncorrected.

C Additional Experimental Results

To demonstrate the generalizability of CoLAT across datasets, we present additional results on the Amazon Photos dataset (Shchur et al., 2018) in Sec. C.1 and on the Co-author datasets (Shchur et al., 2018) in Sec. C.2, where we compare CoLAT with baseline methods under the NTK and SMulti attacks.

[3] `https://github.com/eaglelab-zju/NoisyGL`

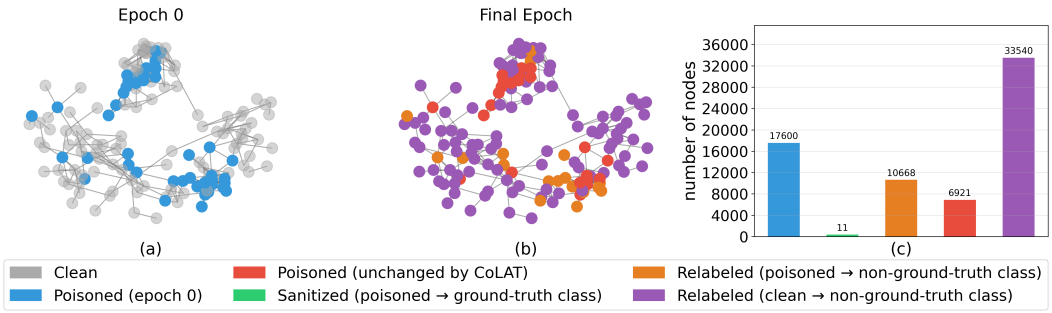


Figure 6: Effectiveness analysis of CoLAT under the NTK attack with a 20% poisoning rate on OBGN-ArXiv.

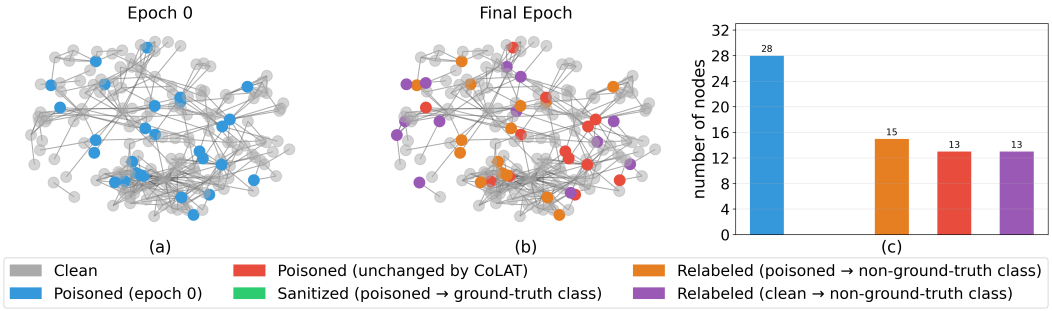


Figure 7: Effectiveness analysis of CoLAT under the NTK attack with a 10% poisoning rate onb Cora-ML, showing that CoLAT remains robust through adversarial training even in the absence of explicit label sanitization.

In Sec. C.3, we focus on demonstrating the effectiveness of CoLAT under very high poisoning levels on the Cora-ML dataset and in Sec. C.4, we instead examine generalization across attack types by evaluating CoLAT under the LafAK attack on the Cora-ML, Citeseer, and PubMed datasets.

C.1 Amazon-Photo Dataset

We evaluate CoLAT on the Amazon Photo dataset to demonstrate its effectiveness beyond the citation-style benchmarks considered in Table 1.

The Amazon Photo dataset is a segment of the Amazon co-purchase graph (McAuley et al., 2015), where nodes represent products, edges indicate that two products are frequently bought together, node features are bag-of-words representations of product reviews, and class labels correspond to product categories.

Table 7 shows that CoLAT consistently outperforms baseline methods at higher poisoning levels on the Amazon Photo dataset. For example, under the NTK 10% poisoning setting, none of the baselines outperform clean training, whereas CoLAT achieves a 9% improvement. Under the SMulti attack, CoLAT achieves a 31% gain over clean training at 30% poisoning and a 22% improvement at 20% poisoning.

We do not report results for the NTK 30% poisoning setting, as the optimization problem described by Lingam et al. (2024) failed to converge when generating poisoned labels for the Amazon Photo dataset using the NTK attack.

C.2 Coauthor Datasets

We further evaluate CoLAT on two medium-scale datasets, Coauthor-CS and Coauthor-Physics. Both are co-authorship graphs derived from the Microsoft Academic Graph. In these datasets, nodes represent authors, with an edge connecting two authors if they have co-authored a paper. Node features are derived from the keywords of each author's publications, while class labels

Table 7: Test accuracy comparison of CoLAT and baseline methods on the Amazon Photo dataset under clean training and adversarial label poisoning attacks. CoLAT consistently achieves the strongest performance at higher poisoning levels.

Training	Clean	NTK Poisoned			SMulti Poisoned			
		5%	10%	20%	5%	10%	20%	30%
Clean Training	0.8853 (0.009)	0.8745 (0.01)	0.7565 (0.08)	0.4859 (0.007)	0.8643 (0.014)	0.7456 (0.017)	0.5656 (0.033)	0.4771 (0.013)
PIGNN	0.8700 (0.020)	0.8242 (0.030)	0.7372 (0.055)	0.4909 (0.007)	0.8450 (0.032)	0.7448 (0.037)	0.5904 (0.053)	0.4791 (0.005)
RTGNN	0.6713 (0.061)	0.5914 (0.034)	0.6049 (0.026)	0.4376 (0.046)	0.6087 (0.023)	0.5595 (0.095)	0.6087 (0.056)	0.4601 (0.018)
CRGNN	0.8553 (0.027)	0.7145 (0.13)	0.6337 (0.10)	0.4695 (0.075)	0.6889 (0.12)	0.5891 (0.11)	0.5728 (0.065)	0.4726 (0.053)
CP	0.8902 (0.010)	0.8798 (0.020)	0.7388 (0.047)	0.4866 (0.004)	0.8754 (0.017)	0.7926 (0.023)	0.5623 (0.046)	0.4664 (0.010)
CoLAT	0.8800 (0.035)	0.8657 (0.054)	0.8422 (0.094)	0.6038 (0.14)	0.8453 (0.020)	0.8361 (0.053)	0.7900 (0.034)	0.7935 (0.082)

correspond to the authors' primary fields of study. We perform 15 hyperparameter-tuning runs for each configuration. As shown in Table 8, CoLAT outperforms all baselines across both datasets in all but one configuration.

Table 8: Test accuracy comparison of CoLAT against baselines and clean training (the undefended GCN baseline) Coauthor-CS, and Coauthor-Physics under clean and poisoned labels, all using a GCN architecture. Results are reported for NTK and SMulti across various attack budgets. Red indicates the best test accuracy (stdev) in each setting. CoLAT performs best across most settings.

Dataset	Training	Clean Dataset	NTK				SMulti			
			5%	10%	20%	30%	5%	10%	20%	30%
Coauthor-Physics	Clean	0.9163(0.015)	0.9161 (0.014)	0.6579 (0.072)	0.4998 (0.018)	0.4397 (0.006)	0.7663 (0.031)	0.5910 (0.061)	0.4410 (0.011)	0.4030 (0.024)
	PIGNN	0.9005 (0.007)	0.9124 (0.027)	0.7614 (0.031)	0.4762 (0.039)	0.4476 (0.008)	0.8587 (0.026)	0.7594 (0.035)	0.4412 (0.009)	0.4403 (0.021)
	RTGNN	0.9206 (0.012)	0.9212 (0.009)	0.9247 (0.019)	0.4591 (0.024)	0.4250 (0.021)	0.8175 (0.051)	0.5137 (0.027)	0.4797 (0.038)	0.4208 (0.020)
	CRGNN	0.9289 (0.004)	0.9247 (0.012)	0.7130 (0.033)	0.5145 (0.026)	0.4271(0.032)	0.6579 (0.02)	0.6199 (0.033)	0.4387 (0.073)	0.3087 (0.037)
	CP	0.8962 (0.029)	0.9156 (0.014)	0.6832 (0.091)	0.5281 (0.043)	0.4533 (0.023)	0.8587 (0.029)	0.7012 (0.092)	0.4344 (0.015)	0.4021 (0.022)
	CoLAT	0.9362 (0.005)	0.9248 (0.002)	0.9063 (0.002)	0.5280 (0.076)	0.5028 (0.057)	0.8907 (0.039)	0.8559 (0.044)	0.5388 (0.210)	0.5026 (0.024)
Coauthor-CS	Clean	0.9121(0.007)	0.8283 (0.011)	0.7414 (0.073)	0.6136 (0.045)	0.3518(0.030)	0.7717 (0.041)	0.6585 (0.014)	0.5785 (0.042)	0.3826 (0.031)
	PIGNN	0.9072 (0.005)	0.8610 (0.013)	0.916 (0.030)	0.6416 (0.016)	0.3882 (0.031)	0.8638 (0.011)	0.6631 (0.014)	0.5336 (0.035)	0.4157 (0.019)
	RTGNN	0.8966 (0.027)	0.8296 (0.042)	0.6766 (0.010)	0.6371 (0.063)	0.3856 (0.041)	0.8061 (0.055)	0.6843 (0.010)	0.5622 (0.026)	0.4746 (0.078)
	CRGNN	0.9061 (0.006)	0.8129 (0.088)	0.6921 (0.013)	0.5742 (0.024)	0.3111 (0.040)	0.7146 (0.063)	0.6525 (0.017)	0.5283 (0.033)	0.3436 (0.034)
	CP	0.9033 (0.005)	0.8180 (0.032)	0.7618 (0.065)	0.5891 (0.050)	0.3355 (0.067)	0.7359 (0.017)	0.6630 (0.019)	0.5333 (0.050)	0.3714 (0.028)
	CoLAT	0.9124 (0.004)	0.9031 (0.008)	0.8932 (0.015)	0.84754 (0.043)	0.7486 (0.054)	0.8432 (0.067)	0.9085 (0.007)	0.8085 (0.059)	0.6516 (0.059)

C.3 Effectiveness at High Poison Levels

Label poisoning attacks are typically budget-constrained to remain inconspicuous, with prior defenses commonly evaluated at poisoning rates up to 30%. To assess robustness under more extreme conditions, we evaluate CoLAT at 40% and 50% poisoning rates on Cora-ML, Citeseer, and PubMed. As shown in Sec. C.3, CoLAT consistently outperforms clean training across all datasets, attacks, and poisoning rates, demonstrating robustness even under severe label poisoning.

Table 9: Performance under high poisoning rates. CoLAT consistently outperforms clean training across datasets and attacks at 40% and 50% poisoning rates.

Dataset	Training	NTK		SMulti	
		40%	50%	40%	50%
Cora-ML	Clean	0.2295 (0.018)	0.1405 (0.023)	0.3305 (0.027)	0.2764(0.004)
	CoLAT	0.3471 (0.039)	0.2256 (0.049)	0.6296 (0.021)	0.6114 (0.049)
Citeseer	Clean	0.1621 (0.015)	0.1441 (0.011)	0.2195 (0.032)	0.2204(0.044)
	CoLAT	0.1854(0.034)	0.1816 (0.034)	0.5639 (0.086)	0.5086 (0.125)
Pubmed	Clean	0.4037 (0.028)	0.3192 (0.025)	0.4145(0.062)	0.2455(0.098)
	CoLAT	0.4359 (0.060)	0.3873 (0.031)	0.5649 (0.127)	0.5130 (0.072)

C.4 LafAK Attack

The LafAK attack is a binary adversarial attack that reformulates the original bilevel optimization problem as a single-level problem by leveraging a closed-form solution of the linearized inner model (Zhang et al., 2020).

We report results under the LafAK attack across the Cora ML, Citeseer, and PubMed datasets, including comparisons with baseline methods (Sec. C.4.1) and ablation studies on Cora ML (Sec. C.4.2).

These ablations demonstrate (i) the advantage of adversarial training over one-step label correction under the LafAK attack, and (ii) the effectiveness of the proposed detection model.

C.4.1 Comparison with Baselines

Table 10 shows that CoLAT achieves the highest robustness on the clean dataset and is the only method that improves generalization compared to standard clean training.

On poisoned datasets, CoLAT consistently attains the best performance, particularly at higher poisoning levels under the LafAK attack across all three datasets.

Table 10: Test accuracy comparison of CoLAT and baseline methods under the LafAK attack across Cora ML, Citeseer, and PubMed. All methods use a GCN architecture. CoLAT consistently achieves the best performance across high poisoning rates.

Dataset	Method	Clean	LafAK			
			5%	10%	20%	30%
Cora ML	Clean Training	0.8270 (0.005)	0.8006 (0.010)	0.7368 (0.022)	0.6145 (0.030)	0.5203 (0.025)
	PIGNN	0.8236 (0.004)	0.8182 (0.015)	0.7776 (0.015)	0.5936 (0.041)	0.6099 (0.011)
	RTGNN	0.6297 (0.060)	0.5538 (0.085)	0.6141 (0.084)	0.5591 (0.039)	0.5148 (0.017)
	CRGNN	0.8088 (0.014)	0.7660 (0.015)	0.7024 (0.040)	0.6066 (0.024)	0.4909 (0.019)
	CP	0.8122 (0.019)	0.7863 (0.031)	0.7340 (0.038)	0.6100 (0.047)	0.4942 (0.022)
	CoLAT	0.8377 (0.006)	0.8213 (0.016)	0.7898 (0.02)	0.7008 (0.06)	0.6221 (0.012)
Citeseer	Clean Training	0.7410 (0.003)	0.6791 (0.013)	0.6475 (0.032)	0.5780 (0.004)	0.3620 (0.005)
	PIGNN	0.6920 (0.028)	0.7252 (0.010)	0.7012 (0.009)	0.5376 (0.066)	0.3735 (0.038)
	RTGNN	0.6386 (0.021)	0.5990 (0.038)	0.5997 (0.032)	0.5088 (0.028)	0.3377 (0.018)
	CRGNN	0.7082 (0.019)	0.6403 (0.059)	0.5845 (0.040)	0.5290 (0.025)	0.3390 (0.014)
	CP	0.7070 (0.013)	0.6204 (0.050)	0.5932 (0.059)	0.5280 (0.016)	0.3537 (0.029)
	CoLAT	0.7420 (0.011)	0.7090 (0.021)	0.7316 (0.006)	0.5780 (0.015)	0.5745 (0.029)
PubMed	Clean Training	0.7940 (0.010)	0.7210 (0.009)	0.6786 (0.018)	0.5636 (0.092)	0.4831 (0.021)
	PIGNN	0.8095 (0.011)	0.7443 (0.011)	0.7565 (0.002)	0.5227 (0.020)	0.4340 (0.051)
	RTGNN	0.7173 (0.023)	0.6889 (0.062)	0.5662 (0.038)	0.6362 (0.051)	0.5084 (0.030)
	CRGNN	0.7321 (0.070)	0.7287 (0.035)	0.7076 (0.014)	0.5220 (0.200)	0.4906 (0.026)
	CP	0.7875 (0.037)	0.7314 (0.050)	0.7228 (0.033)	0.5802 (0.031)	0.4705 (0.070)
	CoLAT	0.8171 (0.009)	0.7211 (0.037)	0.8091 (0.004)	0.7233 (0.048)	0.7138(0.025)

C.4.2 Ablation Studies on LafAK attack

Performance on different architectures We evaluate CoLAT on GAT and GraphSAGE architectures using the Cora ML dataset under the LafAK attack, comparing its performance with two baseline methods, PIGNN and CRGNN, to assess robustness to architectural variation.

As shown in Table 11, CoLAT consistently achieves the best performance across both architectures under the LafAK attack, providing significant improvements over both clean training and baseline methods. In particular, CoLAT attains approximately a 20% performance gain over both baselines and clean training at 30% poisoning.

Table 11: Test accuracy comparison of CoLAT and baseline methods under the Lafak attack across GAT and GraphSAGE architectures on Cora ML. CoLAT consistently achieves the best performance across high poisoning rates.

Architecture	Method	Clean	LafAK			
			5%	10%	20%	30%
GAT	Clean Training	0.8277 (0.010)	0.7913 (0.014)	0.7583 (0.011)	0.6291 (0.028)	0.5298 (0.029)
	PIGNN	0.8325 (0.006)	0.8182 (0.010)	0.7801 (0.009)	0.6224 (0.028)	0.5287 (0.026)
	CRGNN	0.8145 (0.015)	0.7792 (0.009)	0.7253 (0.019)	0.6161 (0.014)	0.5044 (0.042)
	CoLAT	0.8435 (0.008)	0.8353 (0.011)	0.7853 (0.026)	0.7890 (0.022)	0.7605 (0.027)
GraphSAGE	Clean Training	0.8277 (0.010)	0.8044 (0.018)	0.7458 (0.011)	0.5958 (0.016)	0.5797 (0.016)
	PIGNN	0.8049(0.019)	0.8073 (0.017)	0.7372 (0.012)	0.56 (0.052)	0.5259 (0.041)
	CRGNN	0.8024(0.014)	0.7752 (0.025)	0.6995 (0.032)	0.5884 (0.018)	0.5179 (0.054)
	CoLAT	0.8103 (0.01)	0.8021 (0.019)	0.8019 (0.020)	0.7257 (0.04)	0.7523 (0.03)

Sensitivity to adversarial training As described in Sec. 5.2, the adversarial training in CoLAT is replaced with a one-step filtering procedure based on the trained detection model. This variant is evaluated on the Cora ML dataset under the LafAK attack. The results in Table 12 show that adversarial training, which continuously exposes the model to diverse adversarial configurations, is more effective than a one-step filtering approach based on a fixed configuration.

Table 12: Sensitivity to adversarial training under the LafAK attack on the Cora ML dataset. We compare CoLAT with full adversarial training against a variant that replaces it with one-step filtering based on the detection model. Results are reported for clean data and poisoning rates of 5%, 10%, 20%, and 30%. Adversarial training consistently yields stronger robustness.

Training	Clean Dataset	LafAK			
		5%	10%	20%	30%
Clean Training	0.8270 (0.005)	0.8006 (0.010)	0.7368 (0.022)	0.6145 (0.030)	0.5203 (0.025)
CoLAT (filtering)	0.8264 (0.008)	0.7966 (0.006)	0.7669 (0.009)	0.6638 (0.008)	0.5413 (0.022)
CoLAT (adversarial training)	0.8377 (0.006)	0.8213 (0.016)	0.7898 (0.02)	0.7008 (0.06)	0.6221 (0.012)

Sensitivity to contrastive detection model As shown in Table 13, we empirically demonstrate under the LafAK attack that selecting nodes for the attack set based solely on their contribution to the classification loss is insufficient to capture truly influential nodes. This comparison highlights that the contrastive loss is a key component of our structurally guided scoring mechanism.

Table 13: Sensitivity to the contrastive detection model under the LafAK attack on the Cora ML dataset. We compare CoLAT using a logit-based heuristic for node selection with the full contrastive detection model. Results show that the contrastive detection model provides superior robustness across poisoning rates.

Training	Clean Dataset	LafAK			
		5%	10%	20%	30%
Clean Training	0.8270 (0.005)	0.8006 (0.010)	0.7368 (0.022)	0.6145 (0.030)	0.5203 (0.025)
CoLAT (using logits)	0.8164 (0.012)	0.8002 (0.014)	0.7258 (0.015)	0.5934 (0.027)	0.5213 (0.023)
CoLAT (using det. model)	0.8377 (0.006)	0.8213 (0.016)	0.7898 (0.02)	0.7008 (0.06)	0.6221 (0.012)

D Extension to Heterophilic Datasets

As reported in Table 3 and in Table 14, CoLAT outperforms clean training at high poison levels but otherwise performs similarly to clean training on the Actor and Amazon Ratings dataset (Platonov et al., 2024). These results should be interpreted in light of the heterophilic structure which reduces the usefulness of graph structure as a prior.

Adapting CoLAT more naturally to heterophilic graphs would likely require modifying the contrastive objective so that it considers all nodes sharing the same label, rather than only connected same-label nodes. However, this would largely remove the role of graph structure, making a simple MLP more appropriate. We investigate this in Table 15 by comparing two variants of CoLAT: one that removes the detector and uses the logits of the GCN classification loss for scoring, and another that replaces the detector with an MLP trained with cross-entropy loss and uses its logits for scoring label flips of the GCN-based classification model. The MLP-based variant performs better, suggesting that leveraging graph structure as a prior is not effective in this setting. This is consistent with the fact that GCNs are better suited for homophilic graph data, so even clean GCN training is already suboptimal for the Actor dataset.

Moreover, the detection model and the classification model do not need to share the same architecture, which means that the detection model can leverage a simple MLP to make the GCN classification model more robust, as shown in Table 15. This makes CoLAT a flexible framework.

Under the current contrastive formulation, CoLAT is better suited to homophilic graphs. Nevertheless, even with the limited graph signal available in Actor, CoLAT's adversarial label flips still provide robustness gains under strong multiclass attacks such as SMulti.

Table 14: Test accuracy comparison of CoLAT and clean training on the heterophilic Amazon-ratings dataset under NTK and SMulti adversarial label poisoning attacks.

Training	Clean	NTK Poisoned						SMulti Poisoned					
		5%	10%	20%	30%	40%	50%	5%	10%	20%	30%	40%	50%
Clean	0.4411 (0.005)	0.4336 (0.005)	0.4215 (0.007)	0.3683 (0.003)	0.3213 (0.006)	0.2927(0.055)	0.2806(0.014)	0.4113(0.002)	0.3736(0.006)	0.2391 (0.018)	0.1660(0.008)	0.1110(0.011)	0.071(0.008)
CoLAT	0.3891 (0.015)	0.3760 (0.008)	0.3832 (0.018)	0.3621 (0.018)	0.3371 (0.014)	0.3090(0.032)	0.2903(0.012)	0.3854 (0.006)	0.3527 (0.026)	0.2586(0.026)	0.1413 (0.050)	0.069(0.016)	0.050(0.009)

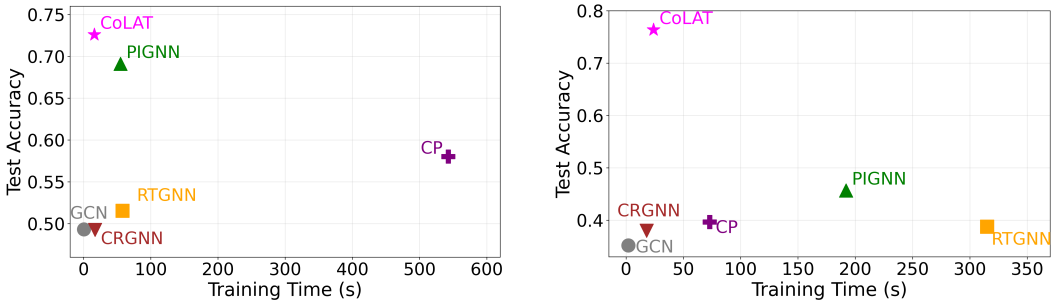
Table 15: Ablation study on the Actor dataset highlighting the impact of graph structure in heterophilic settings. We compare CoLAT variants using a GCN-based detection model with an MLP-based detection model under NTK and SMulti attacks. Results show that when graph structure provides weak or noisy signals, leveraging non-graph-based detection (MLP) can be more effective.

Training	Clean	NTK Poisoned				SMulti Poisoned			
		5%	10%	20%	30%	5%	10%	20%	30%
MLP(logits)	0.2853 (0.017)	0.2770 (0.019)	0.2856 (0.007)	0.2477 (0.015)	0.2246 (0.011)	0.2533 (0.020)	0.2511 (0.017)	0.2274 (0.019)	0.1618 (0.041)
CoLAT(logits)	0.2783 (0.017)	0.2579 (0.006)	0.2719 (0.003)	0.2460 (0.007)	0.2244 (0.020)	0.2472 (0.015)	0.2384 (0.017)	0.2079 (0.011)	0.1724 (0.047)

E Training Time and Memory Comparison

We compare the computational efficiency of CoLAT with clean training and the baselines in terms of training time and memory usage. All classification models are trained for 200 epochs. Figure 8 reports training time and test accuracy on Cora-ML and PubMed under CPU-only and NVIDIA A100-SXM4-80GB GPU settings, respectively. Across both settings, CoLAT achieves the highest test accuracy while requiring among the lowest training times.

We further evaluate training time and memory consumption on the larger Coauthor-Physics dataset in Table 16. CoLAT requires only 7.67 seconds and 2.77 GB of memory, close to clean GCN training (4.23 seconds and 2.76 GB), while incurring substantially lower computational overhead than the defense baselines. In particular, several baselines require considerably longer training times or higher memory, with PiGNN exceeding the available GPU memory. These results demonstrate that CoLAT introduces only modest computational and memory overhead, while remaining scalable to larger graphs.



(a) Training on Cora ML under the SMulti attack (20% poisoning) with CPU-only training (16 CPUs).

(b) Training on PubMed under the NTK attack (30% poisoning) using a A100-SXM4-80GB GPU.

Figure 8: Training time versus test accuracy for CoLAT and baseline methods under different hardware settings and attack scenarios. CoLAT achieves the best trade-off, requiring among the least training time while attaining the highest test accuracy across both CPU and GPU configurations.

F Limitations

We evaluate CoLAT against a diverse set of label poisoning attacks from the literature. However, as with most defenses, robustness cannot be guaranteed against all possible or adaptive attack strategies (Papernot et al., 2016). In particular, an attacker that explicitly targets the detection mechanism or exploits its structural assumptions may degrade performance. We do not consider fully adaptive label poisoning attacks that are tailored to our defense. Constructing such attacks against adversarial training defenses is shown to be computationally expensive and non-trivial (Mujkanovic

Table 16: Training time and memory consumption of different baselines on the Coauthor-physics dataset Numbers reported on NVIDIA A100-SXM4-40GB. Lower is better.

	Training Time (s)	Memory Used (GB)
Clean GCN	4.23	2.76
PiGNN	N/A (638.16)	OOM (69.22)
RTGNN	858.62	9.10
CRGNN	24.8	18.2
CP	173.6	3.82
CoLAT	7.67	2.77

et al., 2022): For our setting, the adversary would need to jointly account for contrastive node selection and the evolving adversarial training dynamics within a bilevel optimization framework, which is computationally demanding and difficult to scale. As a result, designing strong adaptive attacks for this setting remains an open problem. Nevertheless, the consistent improvements observed across a wide range of attack types and budgets suggest that structure-aware adversarial training provides a useful inductive bias for improving robustness in practice.

G Impact statement

In this work, we study robust node classification in graph learning under label poisoning, a realistic threat in applications where labels are adversarially manipulated or crowd-sourced, and propose a contrastive-adversarial training-based defense called CoLAT. By improving robustness to such label perturbations, our method can enhance the reliability of graph-based systems in domains such as social network analysis, recommendation systems, and biological networks. In particular, increased resilience to corrupted labels may help prevent the propagation of malicious information through graph-structured pipelines.

At the same time, techniques for adversarial training and attack modeling may have dual-use implications. While our goal is defensive, the formulation of structured label perturbations could potentially inform the design of stronger poisoning attacks or adaptive attacks. However, our method does not introduce new attack strategies; rather, it builds on existing threat models to improve robustness. As with other adversarial defenses, performance under adaptive, defense-aware attacks remains an open challenge. This work encourages further research into principled and scalable defenses for graph learning under adversarial conditions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both the abstract and introduction (Sec. 1) clearly state the addressed problem, the proposed approach, and contributions made in the paper along with the takeaways from computational results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We explicitly discuss the limitations of our approach in Sec. F in the appendix. Furthermore, in Sec. 5, we clearly state the experimental setup, including datasets and model sizes, on which we test our method. Lastly, we discuss the hardware and computational cost of our method in Sec. A.4 and Sec. E in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: Our paper does not include theoretical results. We only provide theoretically motivated formulations to frame our CoLAT approach in Sec. 3.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We state Algorithm 1 for reproducibility our proposed CoLAT approach. Further, we provide the experimental details in Sec. 5, and in Appendix A along with the specific hyperparameter values (see Table 6) and model architectures (see Sec. A.4) used in the experiments. We also submitted our code as supplementary material for reproducibility.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the references for the graph datasets we used in our experiments in Sec. 5. In Appendix A.1, we further report the details on these datasets. We report the open-source codebases on which our implementation is based in Appendix 6. We submitted our open-access code as supplementary material in our submission.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The details on the experimental setup are provided in Sec. 5 and Appendix A, with a table reporting hyperparameter choices (see Table 6).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We evaluated our CoLAT approach across replications (as stated in Table 6); hence, we report the standard deviation values in all of the tables in our experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide details on compute resources in Sec. A.4 and runtime comparison in Sec. E in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is conducted with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: We motivate the impact of our proposed approach in Sec. 1. We explicitly discuss the societal impact of our research in Appendix G.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks as we use open-source graph datasets found in the literature.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We provide citations for the benchmark datasets and baseline models employed in the study.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.