Rethinking Label Poisoning for GNNs: Pitfalls and Attacks

Vijay Lingam [1] **Mohammad Sadegh Akhondzadeh** [1] **Aleksandar Bojchevski** [2]

Abstract

Node labels for graphs are usually generated using an automated process, or crowd-sourced from human users. This opens up avenues for malicious users to compromise the training labels, making it unwise to blindly rely on them. While robustness against noisy labels is an active area of research, there are only a handful of papers in the literature that address this for graph-based data. Even more so, the effects of adversarial label perturbations are sparsely studied. A recent work revealed that the entire literature on label poisoning for GNNs is plagued by serious evaluation pitfalls and showed how existing attacks render ineffective post fixing these shortcomings. In this work, we introduce two new simple yet effective attacks that are significantly stronger (up to $\sim 8\%$) than the previous strongest attack. Our work demonstrates the need for more robust defense mechanisms, especially considering the *transferability* of our attacks, where a strategy devised for one model can effectively contaminate numerous other models.

1. Introduction

Graph Neural Networks (GNNs) have emerged as a powerful tool to learn from graph-structured data. From social network analysis and recommendation systems, to bioinformatics and traffic prediction, their wide-ranging applications highlight their importance (Wu et al., 2019b; Akhondzadeh et al., 2023). Consequently, we need to understand their robustness, especially if we aim to integrate them in safety-critical domains. There is a large body of work showing that GNNs are susceptible to feature and structure perturbations, which can significantly degrade their performance (see e.g. surveys by Sun et al. (2018), Chen et al. (2020), and Jin et al. (2021)). These vulnerabilities expose them to potential adversarial attacks, thereby compromising the integrity of their outputs.

Among the various attack vectors, label poisoning poses a distinct threat. In many real-world scenarios, labels for training data are often generated through crowd-sourcing or other non-expert sources. For example, in a federated learning setting where multiple peers collaboratively train a shared model, any peer may submit potentially poisoned labels. Similarly, models are often trained on data scraped from the internet without careful quality control. This opens up an avenue for attackers to insert poisoned labels, thereby manipulating the learning process of GNNs. For illustration purposes, we compute the optimal label flipping attack for the GCN model (Kipf & Welling, 2017) on a tiny version of the Cora-ML dataset by exhaustively enumerating all possible label flips (subsection A.1). We observe that only a single adversarial label flip can drop GCN's test performance from 81.27 (5.64) to 64.36 (6.26), a staggering drop of $\sim$**17%**.

We will show that similar results hold in general – across all datasets and models, a relatively few flips are enough to significantly affect performance. Despite its potential impact, label poisoning for GNNs has been relatively underexplored in the research community, leaving a critical gap in our understanding of their robustness. While label poisoning is better understood for non-graph data (Biggio et al., 2011; Jin et al., 2021), graphs come with unique challenges due to label sparsity and the interdependence between nodes. Recently, Lingam et al. (2023a) have identified severe fallacies in the evaluation setup used by existing label poisoning attacks, and showed how existing attacks are much weaker than claimed after fixing these shortcomings.

In this work, we propose two simple, yet highly effective, families of attacks that stem from different approximations of the (NP-hard) bi-level optimization problem associated with label poisoning. Our two strategies, each with different advantages, rely on *linear surrogates* and *meta learning* respectively. These attacks are significantly stronger than previous ones, affecting both vanilla and noise-aware models. This demonstrates the urgent need for more robust defense mechanisms, especially considering the *transferability* of our attacks, where a strategy devised for one model can effectively contaminate numerous other models.

[1]CISPA Helmholtz Center for Information Security, Saarbrücken, Germany [2]University of Cologne, Germany. Correspondence to: Vijay Lingam <vijaylingam0810@gmail.com>.

2^{nd} AdvML Frontiers workshop at 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Setup: Threat model and existing attacks

We focus on label poisoning attacks for semi-supervised node classification. The adversary can perturb the labels of a small fraction of training nodes, keeping the features and the structure unperturbed.

Problem setting. We are given a graph represented by its $|\mathcal{V}| \times |\mathcal{V}|$ adjacency matrix $\boldsymbol{A}$ and its $|\mathcal{V}| \times D$ feature matrix $\boldsymbol{X}$, where $\mathcal{V}$ is the set of nodes. The D-dimensional features can be discrete or continuous, and the graph can be directed, undirected, weighted or unweighted. We represent the ground-truth labels as one-hot encoded row vectors in a matrix $\boldsymbol{Y} \in \{0,1\}^{|\mathcal{V}| \times C}$ where C is the number of classes. We denote with $\boldsymbol{Y}_l$ and $\boldsymbol{Y}_u$ the labels corresponding to the subset of training (labeled) nodes $\mathcal{V}_l$ and the test (unlabeled) nodes $\mathcal{V}_u$. Given $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}_l$ we learn the parameters $\boldsymbol{\theta}^*$ of a model $f(\boldsymbol{\theta})$, e.g. a GNN, by minimizing some loss $\mathcal{L}$ (usually cross-entropy).

Threat model. The goal of the attacker is to perturb the labels $\boldsymbol{Y}_l$ of a small fraction of training nodes $\mathcal{V}_l$ to reduce the overall performance on the test nodes $\mathcal{V}_u$. We assume that the attacker has complete knowledge of $\boldsymbol{A}$ and $\boldsymbol{X}$ and is not allowed to change them. We consider two settings where the attacker knows: (i) all ground-truth labels $\boldsymbol{Y}$, or (ii) only the ground-truth for the training nodes $\boldsymbol{Y}_l$ and just the predictions $\widetilde{\boldsymbol{Y}}_u$ for the test nodes (deferred to subsection A.6). The first setting corresponds to the worst-cases scenario and is important for understanding the intrinsic robustness of our models, regardless of whether it is feasible to execute such an attack in practice. Extensions where the attacker has partial knowledge of $\boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}$ or $\widetilde{\boldsymbol{Y}}$ are straightforward and left for future work. We can write down the label poisoning attack as the following bi-level optimization problem:

$$\begin{aligned} \widehat{\boldsymbol{Y}}_l^* = \arg\max_{\widehat{\boldsymbol{Y}}_l} \quad & \mathcal{L}(\boldsymbol{\theta}^*; \boldsymbol{A}, \boldsymbol{X}, \boldsymbol{Y}_u) \\ \text{s.t.} \quad & \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X}, \widehat{\boldsymbol{Y}}_l), \\ \|\boldsymbol{Y}_l - \widehat{\boldsymbol{Y}}_l\|_0 \leq 2\epsilon|\mathcal{V}_l| & \end{aligned} \tag{1}$$

where $\widehat{\boldsymbol{Y}}_l$ denotes the poisoned training labels, $\epsilon \in (0,1)$ is the perturbation budget, and $\epsilon|\mathcal{V}_l|$ is the maximum number of labels that can be flipped.

The inner problem corresponds to training a model with potentially poisoned training nodes, while the outer problem maximizes the loss on the test nodes, given the optimal parameters $\boldsymbol{\theta}^*$. Optimally solving Equation 1, or often even just the inner problem itself, is intractable (NP-hard) in general. Next, we briefly describe both heuristic-based and learning-based attacks that currently exist in the literature. In section 3 we describe our approach.

Heuristic-based attacks. The *random label-flipping attack* (**RND**) randomly selects a subset of training nodes within the budget, and then randomly selects incorrect labels for each selected node. The *degree-based label-flipping attack* (**DEG**) selects the highest degree nodes given a budget, and then again randomly selects incorrect labels. As we will show in section 4 even these simple baselines can outperform some prior attacks under a fair evaluation.

Learning-based attacks. All previous attacks approximately solve Equation 1 by using a fixed surrogate model. They all replace the inner problem with a closed-form solution, but differ in how they tackle the outer problem. The poisoned labels are then applied to a target model.

Liu et al. (2019) propose a gradient-based *label propagation attack* (**LP**) for graph-based semi-supervised learning (G-SSL) models. Note that this attack was primarily designed for binary-class datasets that are i.i.d in nature, with G-SSL methods applied on top. Nonetheless, it can be adapted to our graph setting. LP replaces the inner optimization in Equation 1 with the closed-form solution of label propagation. They replace the loss in the outer problem with an expectation over learnable Bernoulli variables that model the probability of flipping the label of a given node. They optimize this new loss using a reparametrization trick and gradient descent.

Zhang et al. (2020) build on top of the LP attack and propose LAFAK (**LFK**) that replaces the inner optimization in Equation 1 with a regression-based closed-form solution of a linearized GCN. They remove the non-linearities in a 2-layer GCN (Kipf & Welling, 2017): $\text{SOFTMAX}(\widehat{\boldsymbol{A}}\,\text{RELU}(\widehat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{\theta}^1)\,\boldsymbol{\theta}^2) \rightarrow \text{SOFTMAX}(\widehat{\boldsymbol{A}}^2\boldsymbol{X}\boldsymbol{\theta})$, where $\widehat{\boldsymbol{A}}$ is the symmetric normalized adjacency matrix, and $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$ are reparameterized into a single matrix $\boldsymbol{\theta} \in \mathbb{R}^D$. Here, the (implicit) surrogate model is SGC (Wu et al., 2019a). Next, they replace the cross-entropy loss with a least squares loss, using regression to perform classification. That is, $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\widehat{\boldsymbol{X}}_l\boldsymbol{\theta} - \hat{\boldsymbol{y}}_l\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2$, where $\hat{\boldsymbol{y}}_l \in \{-1,+1\}^{|\mathcal{V}_l|}$ is the vector of (potentially poisoned) *binary* training labels, $\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{A}}^2\boldsymbol{X}$ are the diffused features, $(\cdot)_l$ select the subset of rows corresponding to training nodes, and λ is the regularization strength. Now, $\boldsymbol{\theta}^* = (\widehat{\boldsymbol{X}}_l^T\widehat{\boldsymbol{X}}_l + \lambda\boldsymbol{I})^{-1}\widehat{\boldsymbol{X}}_l^T\hat{\boldsymbol{y}}_l$. Similar to LP, they substitute the non-differentiable parts of the outer problem with continuous surrogates and use a gradient descent-based optimizer.

Liu et al. (2022) propose the *maximum gradient attack* (**MG**) based on label propagation. Unlike LP which uses a similarity matrix that is constructed by applying a Gaussian kernel to the feature matrix, MG proposes multiple ways to construct a propagation matrix $\bar{\boldsymbol{A}}$ (e.g., PageRank matrix, higher-order adjacency matrix). Then, they greedily select the top nodes within the budget with the highest gradient w.r.t. the outer loss, and set their label to the max false class.

Binary vs. multi-class. A big limitation of LFK and LP is

that they only support binary labels. Thus, for multi-class datasets they consider only the subset of the graph whose nodes belong to the two most frequent classes, restricting their attack to flips among these two classes. Note, this restriction is only for the attack phase, during evaluation the entire (multi-class) graph is used. This mismatch may be sub-optimal. Moreover, this binary approach may limit the maximum perturbation budget. In subsection A.5 we discuss the issue in detail and propose a minor baseline extension to alleviate it.

Discussion. All these attacks have shortcomings. The LP attack was primarily built for G-SSL datasets that do not contain a graph, limiting its efficacy on message-passing based GNNs. LFK relies on several continuous surrogates to enable gradient flow. These approximations, along with only being able to handle binary classes, hurt its performance. MG's greedy approach can be myopic. While these existing label attacks claim severe degradation in GNNs performance with minimal label perturbations, Lingam et al. (2023a) identified serious flaws in their evaluation setup, and proposed a new evaluation scheme. Existing label poisoning attacks for GNNs are not as effective as they claim under the corrected evaluation setup.

3. Label poisoning attacks

Motivated by the results in (Lingam et al., 2023a) where previous attacks are not as effective as claimed, we design new attacks. We propose two different approximation strategies to solve the bi-level poisoning problem in Equation 1 that rely on: (i) linear surrogate models and (ii) meta learning.

3.1. Linear surrogate attack

We start by replacing the inner optimization problem with a linear surrogate for which we can obtain the optimal weights $\boldsymbol{\theta}^*$ in closed-form. We explore two different variants. First, similar to Zhang et al. (2020) we use an SGC (Wu et al., 2019a) surrogate which is equivalent to linearizing a vanilla 2-layer GCN (Kipf & Welling, 2017). We use SGC, since despite its simplicity it shows similar performance to its non-linear counterpart across datasets. Second, we use the Neural Tangent Kernel (NTK) for an infinitely-wide GCN (Sabanayagam et al., 2022). We use the NTK since it is theoretically well-founded and it has been show to capture the behavior of its finite-width counterpart.

In both cases, unlike Zhang et al. (2020) which only support binary labels, we replace the inner problem with $\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|\widehat{\boldsymbol{X}}_l\boldsymbol{\theta} - \widehat{\boldsymbol{Y}}_l\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2$. In this *multi-output* ridge regression problem, $\boldsymbol{\theta} \in \mathbb{R}^{D\times C}$, we regress against the one-hot labels $\widehat{\boldsymbol{Y}}_l$, thereby supporting mutli-class problems. For the SGC variant $\widehat{\boldsymbol{X}} = \widehat{\boldsymbol{A}}^2\boldsymbol{X}$, and for the NTK variant $\widehat{\boldsymbol{X}}$ is the kernel matrix derived by Sabanayagam et al. (2022)(see subsection A.3 for more detail). The closed-form solution is readily obtained as $\boldsymbol{\theta}^* = \widetilde{\boldsymbol{X}}_l\widehat{\boldsymbol{Y}}_l$ where $\widetilde{\boldsymbol{X}} = (\widehat{\boldsymbol{X}}^T\widehat{\boldsymbol{X}} + \lambda\boldsymbol{I})^{-1}\widehat{\boldsymbol{X}}^T$ is a constant that can be pre-computed once[1] for each dataset. Now, we rewrite the surrogate variant of Equation 1 as the following mixed-integer linear program (MILP):

$$\min_{\boldsymbol{H}\in\{0,1\}^{n\times C},\, \boldsymbol{b}\in\{0,1\}^n} \mathcal{L}(\boldsymbol{Y}_u, \widetilde{\boldsymbol{Y}}_u) \tag{2a}$$

$$\widehat{\boldsymbol{Y}}_l = \boldsymbol{b}\odot\boldsymbol{H} + (\boldsymbol{1}_N - \boldsymbol{b})\odot\boldsymbol{Y}_l \tag{2b}$$

$$\boldsymbol{H} \neq \boldsymbol{Y}_l \tag{2c}$$

$$\widetilde{\boldsymbol{Y}}_u = \widehat{\boldsymbol{X}}_u\widetilde{\boldsymbol{X}}_l\widehat{\boldsymbol{Y}}_l \tag{2d}$$

$$\boldsymbol{H}\boldsymbol{1}_C = \boldsymbol{1}_N \tag{2e}$$

$$\boldsymbol{b}^T\boldsymbol{1}_N \leq \epsilon N \tag{2f}$$

where $\boldsymbol{H}$ and $\boldsymbol{b}$ are binary variables, $\odot$ is the Hadamard (elementwise) product, $N = |\mathcal{V}_l|$ is the number of training nodes, and $\boldsymbol{1}_N$ is the all-ones vector of size N. Here, Equation 2b and Equation 2c construct the final (poisoned) labels, Equation 2d computes the predictions $\widetilde{\boldsymbol{Y}}_u$ for the test nodes, Equation 2e enforces a one-hot encoding, and Equation 2f enforces the budget constraint. We explore several variants for the loss in Equation 2a. One choice is $\mathcal{L}(\boldsymbol{Y}_u, \widetilde{\boldsymbol{Y}}_u) = \mathrm{sum}(\boldsymbol{Y}_u \odot \widetilde{\boldsymbol{Y}}_u)$ which is equivalent to minimizing the predicted "probability" for the true class. The benefit of this loss is that it is linear in $\boldsymbol{Y}_u$ and $\widetilde{\boldsymbol{Y}}_u$. Alternatively, we can compute the most likely label for each node, by computing the $\arg\max$ over the rows of $\widetilde{\boldsymbol{Y}}_u$. This requires additional auxiliary binary variables to encode the $\arg\max$, making it inherently less scalable (see subsection A.6 for a more detailed discussion).

The MILP problem in Equation 2 finds the optimal subset of nodes to perturb (via $\boldsymbol{b}$) and which labels to perturb them to (via $\boldsymbol{H}$). We also explore another variant where $\boldsymbol{H}$ is now fixed, e.g. we always choose to perturb to the false label with the highest probability as predicted by a clean model (see subsection A.4 for details). We call the first variant SGC and the second SGC-BIN. For SGC-BIN we only have to find the optimal subset of nodes which should be significantly easier. In fact, we have the following result.

Proposition 3.1. *For a fixed $\boldsymbol{H}$ the optimal $\boldsymbol{b}$ in problem Equation 2 can be obtained by returning the subset of nodes corresponding to the smallest ϵN elements of an N-dimensional vector $\mathbf{c}$, where the n-th element of $\mathbf{c}$ is computed as $c_n = \sum_{ij} Q_{in}P_{nj}R_{ij}$ where $\boldsymbol{Q} = \widehat{\boldsymbol{X}}_u\widetilde{\boldsymbol{X}}_l, \boldsymbol{P} = \boldsymbol{Y}_l - \boldsymbol{H}$ and $\boldsymbol{R} = \widetilde{\boldsymbol{Y}}_u$.*

See subsection A.4 for a proof. Intuitively, when $\boldsymbol{H}$ is fixed we can rewrite the MILP in a canonical form as

[1] We use numerically stable algorithms to compute $\widetilde{\boldsymbol{X}}$ without explicitly computing the matrix inverse. For very large datasets we can also compute approximations using iterative solvers.

$\min_{\boldsymbol{b}\in\{0,1\}^N} \boldsymbol{c}^T\boldsymbol{b}$ subject to $\boldsymbol{b}^T\mathbf{1}_N \leq \epsilon N$ where $\boldsymbol{c}$ is a constant, which can be efficiently solved by returning the ϵN smallest (negative) elements in $\boldsymbol{c}$.

3.2. Meta attacks

For the second family of attacks we take an orthogonal approach where we directly optimize the poisoned labels $\widehat{\boldsymbol{Y}}_l$ via gradient descent. Since Equation 1 is a bi-level problem we need to differentiate through the inner optimization. Here, unlike the previous attack the inner problem is w.r.t. the actual target model (e.g. the non-linear GCN) which is itself trained with gradient descent. Therefore, we resort to computing *meta gradients* by unrolling the inner optimization for a fixed number of epochs and differentiating through it. Unrolled optimization is common in the meta learning literature, where the goal is e.g. to learn the optimal hyper-parameters (or the structure) of a model. Intuitively, one can think of $\widehat{\boldsymbol{Y}}_l$ as "hyper-parameters" to be optimized for the inner problem. We explore three different ways (standard, fixed, expanded) to parametrize the poisoned labels.

Standard meta attack. We first initialize the free parameter matrix $\widetilde{\boldsymbol{H}} \in \mathbb{R}^{N\times C}$ to learn the log-probability of flipping to a given target label for each training node. We obtain $\boldsymbol{H}$ by sampling proportional to the log-probabilities in $\widetilde{\boldsymbol{H}}$ using the Gumbel-softmax trick (Jang et al., 2016) on each row. In the forward pass $\widetilde{\boldsymbol{H}}$ is "hard" (one-hot encoding), while using the softmax probabilities in the backward pass. We also initialize a parameter vector $\tilde{\boldsymbol{b}} \in \mathbb{R}^N$ with the goal of learning which subset of nodes are a good candidate for poisoning. To enforce the budget we apply *soft-top-k* followed by *k-subset-selection* (Paulus et al., 2020) to obtain $\boldsymbol{b} = \text{top}_k(\tilde{\boldsymbol{b}})$. We construct the final poisoned labels:

$$\widehat{\boldsymbol{Y}}_l = \boldsymbol{b} \odot \boldsymbol{H} + (1 - \boldsymbol{b}) \odot \boldsymbol{Y}_l \tag{3}$$

We feed $\widehat{\boldsymbol{Y}}_l$ into the inner optimization problem which we train for a fixed number of epochs, and use meta-gradients of $\boldsymbol{b}$ w.r.t. the outer loss for learning. During the forward pass we used hard top-k for $\boldsymbol{b}$, sampled proportional to the respective soft-top-k scores which are used during the backward pass.

Gumbel-softmax loss. The choice of loss function in the outer problem can impact the attack performance. The loss of interest is the 0-1 loss (i.e. test accuracy), which is non-differentiable. There are several options to circumvent this issue by constructing different approximations. The simplest idea is to just use the cross-entropy loss (which is also used in the inner problem) on the "soft" predictions $\widetilde{\boldsymbol{Y}}_l$. Another approach which was shown to perform better for other attacks (see e.g. (Mujkanovic et al., 2023)) is the margin loss, where we minimize the margin between the true and the most probable false class. We propose a third alternative that exploits the Gumbel-Softmax trick (Jang et al., 2016). Specifically, in the forward pass we sample hard predictions from $\widetilde{\boldsymbol{Y}}_u$ using the Gumbel-Softmax reparametrization trick, while using the soft predictions for the backward pass. That is for node i we sample a prediction via $\text{onehot}(\arg\max_c\{g_c + p_{ic}\})$ where $g_c \sim \text{Gumbel}(0, 1)$ and p_{ic} is the predicted probability for class c obtained from the (i, c)-th entry of $\widetilde{\boldsymbol{Y}}_u$. Given the one-hot encoded hard predictions we can compute the accuracy via the Hadamard product with the ground-truth $\boldsymbol{Y}_u$. We study the choice of loss functions and their effects in subsection A.6, since our newly proposed loss performs the best on average we use it in the main experiments.

Fixed and expanded meta attacks. We construct a fixed one-hot $\boldsymbol{H} \in \{0, 1\}^{N\times C}$ similar to our SGC attack by determining a fixed target label for each training node (the most likely false class for a clean model).

Similar to the standard meta attack, we let $\boldsymbol{b}$ learn to select the subset of poisoned labels, which now always flip to a fixed target label. We discuss the expanded variant in subsection A.2

Adaptive attacks. Our meta attacks are an instance of, so called, adaptive attacks which were shown to be significantly better than surrogate-based attacks. This bitter lesson was learned for both computer vision (Carlini & Wagner, 2017; Athalye et al., 2018; Tramèr et al., 2020) and GNNs (Mujkanovic et al., 2023). As we will see in section 4, adaptive attacks are not always superior for label poisoning.

3.3. Binary subset variants

We explore additional variants of all of our attacks, where instead of allowing the attacker to select the poisoned labels from the entire training set, we restrict their access only to a subset of labels corresponding to two classes. We refer to these as -BIN variants. This will allows us to compare the the prior baselines on their own terms since they do not natively support multi-class attacks. Similar to LAFAK we choose 2 candidate classes and always flip between them, e.g. the two most frequent classes. As we will see, in practice the binary variants perform surprisingly well, often even better than their multi-class variants. This is interesting, since in theory any feasible solution for the binary variants is also feasible for the multi-class variants, i.e. multi-class is strictly more powerful. Even the RND and DEG baselines benefit. We investigate this phenomenon in more depth in section 4.

4. Experimental evaluation

We thoroughly compare our family of attacks against all baselines. Additionally, for the first time, we evaluate against two GNNs designed to be robust to perturbed labels: CPGCN (Zhang et al., 2020) that introduces a clustering-based loss, and RTGNN (Qian et al., 2023) that

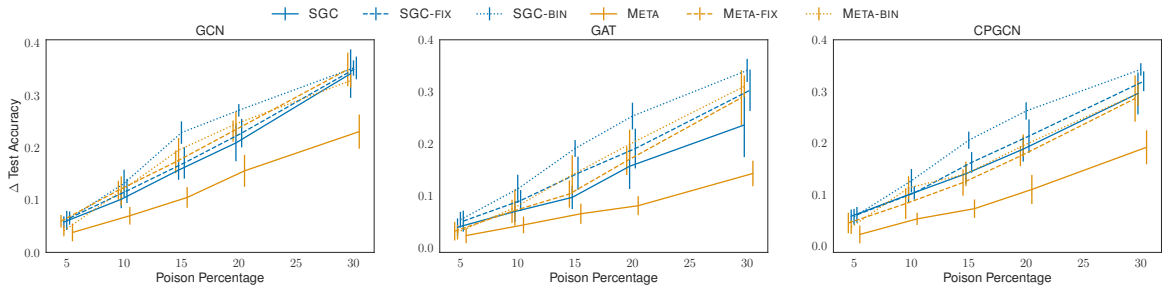

Figure 1. Different variants of the attacks that we propose. Binary variants are best on average.

uses self-reinforcement and consistency regularization. See Appendix A for implementation, code, and tuning details. In subsection A.6 we provide additional results, including larger scale graphs and ablations. In all experiments, we use the evaluation setting from (Lingam et al., 2023a).

Attack variants. We first study the different variants of our attacks. We denote with -FIX and -BIN the fixed and binary subset variants respectively. On Figure 1 we see that on average linear surrogate attacks are better than meta-based attacks. One reason is the fact that we are able to solve the MILP in Equation 2 exactly (and efficiently for -FIX and -BIN). That is, we find the optimal attack (given the surrogate), while the meta-based attacks suffer from optimization issues. We also observe that binary variants are significantly stronger than multi-class variants. Similar trends hold for the baselines (see subsection A.6). This is surprising since the solution space spanned by the multi-class variants encompasses the binary space. To better understand this strange phenomenon we design additional experiments.

Binary vs. multi-class. Our linear surrogate attacks use regression to simulate classification which can be one cause for the discrepancy between the binary and multi-class performance. To study this we compute the optimal attacks and compare models trained with least squares vs. cross-entropy. Then we compute the difference in accuracy when we switch from the regression setting (i.e. the attack's surrogate) to the classification setting (i.e. the defender's actual evaluation) denoted with $\Delta_{\text{acc}}^{\text{bin}}$ for the binary attack (and $\Delta_{\text{acc}}^{\text{mul}}$ for the multi-class attack). In Table 1 we see that difference in accuracy is negligible when using the binary attack, and quite large for the multi-class attack, even growing with increased budget ϵ. The loss shows no significant difference. Thus, the binary variants perform better since they incur a smaller "approximation gap" from regression to classification.

We also investigate the effect of hyper-parameter tuning. In Table 2 we see that when we run the SGC attack on an *untuned* model, the multi-class variant is better, however, this is not true for the tuned model (see Figure 1). Both findings indicate that the multi-class attacks are more likely to overfit – to either the linear regression surrogate or the untuned model – and thus are unable to find poisons that generalize to the final tuned model used by the defender.

Attacks. For ease of comparison, we pick the best performing (on average) attack from each family that we introduced in section 3, namely: SGC-BIN, NTK-BIN and META-BIN. We compute the difference in test performance between the clean-accuracy and poisoned accuracy for each budget across attacks, and plot them on Figure 2. Higher difference shows a stronger attack. We observe that all our attacks (in solid), particularly SGC-BIN attack, outperform baseline attacks with maximum gains of up to $\sim 13\%$. We observe similar trends across different models and datasets as we show in subsection A.6.

CPGCN and RTGNN are two recent defenses that have been introduced to specifically tackle pairwise noise in labels – where the flips between classes is fixed. Attacks like LFK, LP, and the binary variants of our proposed attacks can be viewed as (*adversarial*) pairwise noise. While RTGNN has claimed impressive recovery in performance with pairwise noise rate of up to 40%, we find that our simple attacks can cause a debilitating effect on its performance with as low as 10% noise in labels.

Surprisingly the SGC-based attacks transfer well to a wide range of vanilla GNNs as well as robust GNNs. Moreover, our simple attack formulation significantly outperforms prior attacks that use complicated routines. In contrast to the success of meta-gradient attacks for graph/feature perturbations (Mujkanovic et al., 2023), we observe meta attacks to be inferior on average.

On Figure 3 we show how many times a given attack wins, i.e. leads to lowest accuracy, across all splits and all models (for Cora-ML and Citeseer). Different variants of our linear attacks win most often.

5. Related work

Adversarial attacks for machine learning (ML) have been studied extensively in the literature (Jin et al., 2021; Sun et al., 2018; Liang et al., 2022; Cinà et al., 2023). Attacks can be broadly classified into *evasion* and *poisoning*. For evasion attacks the model is fixed and the attacker perturbs the input at test time. Poisoning attacks are conducted *before* the model is trained, where the attacker can perturb

Table 1. Difference in performance when training the model with cross-entropy vs. least squares loss for Cora-ML. The multi-class attack has significantly higher Δ in accuracy, overfitting to the linear regression surrogate.

ϵ	$\Delta^{\text{bin}}_{\text{acc}}$	$\Delta^{\text{mul}}_{\text{acc}}$	$\Delta^{\text{bin}}_{\text{loss}}$	$\Delta^{\text{mul}}_{\text{loss}}$
5%	-0.02	+1.79	+0.76	+0.78
10%	+1.55	+3.76	+0.86	+0.88
15%	+0.69	+6.01	+0.95	+0.96
20%	-0.82	+8.36	+1.09	+1.06
30%	-2.18	+9.88	+1.33	+1.27

Table 2. Accuracy comparison between multi-class and binary SGC attacks *without* hyper-parameter tuning, on Cora-ML and GCN. The multi-class attack is stronger here, but not after tuning (see Figure 1) indicating overfitting.

ϵ	**multi**	**binary**
5%	74.49 (2.22)	75.01 (1.13)
10%	67.02 (2.43)	70.35 (1.80)
15%	59.10 (1.81)	65.50 (1.86)
20%	54.76 (1.46)	60.08 (3.04)
30%	47.76 (1.45)	45.78 (4.20)

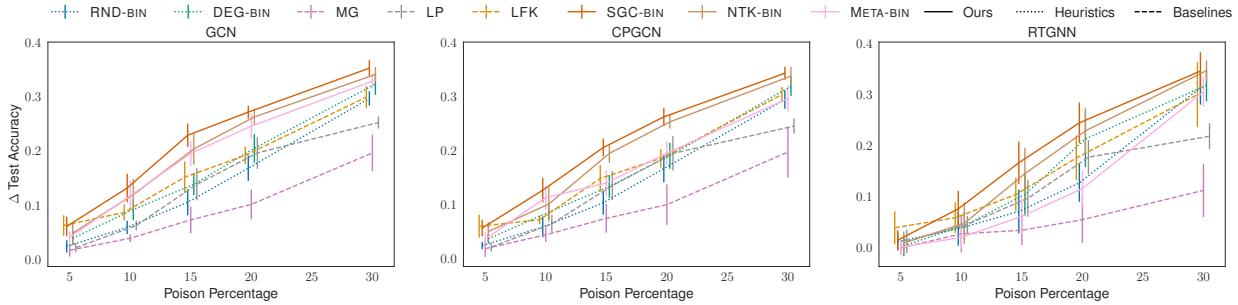

Figure 2. Our strongest attacks significantly outperform the baselines across various models.

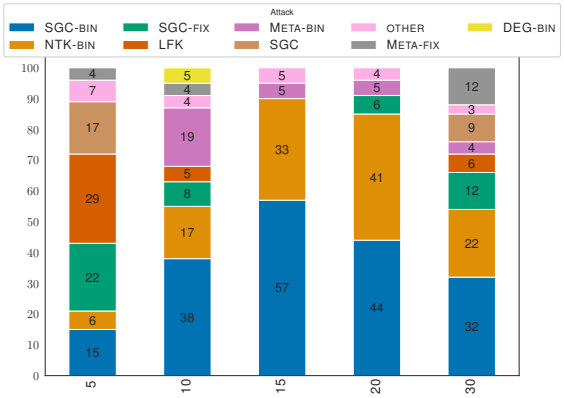

Figure 3. Strongest per split.

the training data with an intention to degrade model performance or control model behavior. Poisoning is significantly more challenging than evasion since it results in a difficult bi-level optimization problem.

Robustness of GNNs. Currently, we have an extensive body of work on the (adversarial) robustness of GNNs. The overwhelming majority of studies focus on adversarial attacks that add or remove a small fraction of edges and/or perturb the node features. Consequently, various heuristic defenses have been developed, as well as certificates to provide provable robustness guarantees. For a detailed overview of both attacks and defenses we recommend the surveys by Sun et al. (2018), (Chen et al., 2020), and Jin et al. (2021)). Label poisoning attacks, despite their importance and potential impact, have been surprisingly sparse with only a few exceptions (Liu et al., 2019; Zhang et al., 2020; Liu et al., 2022). Interestingly, Mujkanovic et al. (2023) recently showed that adaptive attacks – designed specifically to circumvent a given target model or defense – are still able to manipulate almost all models. In other words, we have made significantly less progress than it initially appears towards designing robust models. One conclusion from their work is that we have to carefully think about the evaluation setup to avoid overly optimistic results. This is echoed in our pitfalls findings.

Poisoning attacks. Most poisoning attack assume that the attacker has access to the training data, and position the proposed attack as a worst-case robustness analysis. Such attacks have been extensively studied for classical ML models like SVMs (Biggio et al., 2011), as well as recent (deep learning) models. Lots of different threat models have been considered, including allowing arbitrary changes (both features and labels), clean label poisoning (only features), and poisoning only the labels, which is our focus. See Jagielski et al. (2018) and Tian et al. (2022) for a comprehensive overview.

6. Conclusion and discussion

Simple label poisoning attacks are surprisingly powerful – all examined models, including those designed to be robust, are highly vulnerable. Interestingly, in our setting linear surrogate attacks outperform adaptive meta attacks.

Moreover, we observe that the binary variants where the attacker's threat model is limited outperform multi-class variants, likely due to overfitting. Our findings highlight the urgent need to further study poisoning attacks, as well as develop robust defenses and certificates against them.

References

Akhondzadeh, M. S., Lingam, V., and Bojchevski, A. Probing graph representations. In Ruiz, F., Dy, J., and van de Meent, J.-W. (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 11630–11649. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/akhondzadeh23a.html.

Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. *ArXiv*, abs/1907.10902, 2019.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, 2018.

Biggio, B., Nelson, B., and Laskov, P. Support vector machines under adversarial label noise. In Hsu, C.-N. and Lee, W. S. (eds.), *Proceedings of the Asian Conference on Machine Learning*, volume 20 of *Proceedings of Machine Learning Research*, pp. 97–112, South Garden Hotels and Resorts, Taoyuan, Taiwain, 14–15 Nov 2011. PMLR.

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

Carlini, N. and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 2017.

Chen, L., Li, J., Peng, J., Xie, T., Cao, Z., Xu, K., He, X., and Zheng, Z. A survey of adversarial learning on graphs. *ArXiv*, abs/2003.05730, 2020.

Cinà, A. E., Grosse, K., Demontis, A., Vascon, S., Zellinger, W., Moser, B. A., Oprea, A., Biggio, B., Pelillo, M., and Roli, F. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *ACM Comput. Surv.*, 55(13s), jul 2023. ISSN 0360-0300. doi: 10.1145/3585385. URL https://doi.org/10.1145/3585385.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 19–35, 2018.

Jang, E., Gu, S. S., and Poole, B. Categorical reparameterization with gumbel-softmax. *ArXiv*, abs/1611.01144, 2016.

Jin, W., Li, Y., Xu, H., Wang, Y., Ji, S., Aggarwal, C., and Tang, J. Adversarial attacks and defenses on graphs. *SIGKDD Explor. Newsl.*, pp. 19–34, 2021.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Liang, H., He, E., Zhao, Y., Jia, Z., and Li, H. Adversarial attack and defense: A survey. *Electronics*, 11(8), 2022. ISSN 2079-9292. doi: 10.3390/electronics11081283. URL https://www.mdpi.com/2079-9292/11/8/1283.

Lingam, V., Akhondzadeh, M. S., and Bojchevski, A. Pitfalls in evaluating GNNs under label poisoning attacks. In *ICLR 2023 Workshop on Pitfalls of limited data and computation for Trustworthy ML*, 2023a. URL https://openreview.net/forum?id=qGKj3AHlSXv.

Lingam, V., Sharma, M., Ekbote, C., Ragesh, R., Iyer, A., and Sellamanickam, S. A piece-wise polynomial filtering approach for graph neural networks. In Amini, M.-R., Canu, S., Fischer, A., Guns, T., Kralj Novak, P., and Tsoumakas, G. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 412–452, Cham, 2023b. Springer International Publishing. ISBN 978-3-031-26390-3.

Liu, G., Huang, X., and Yi, X. Adversarial label poisoning attack on graph neural networks via label propagation. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part V*, pp. 227–243, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20064-9. doi: 10.1007/978-3-031-20065-6_14. URL https://doi.org/10.1007/978-3-031-20065-6_14.

Liu, X., Si, S., Zhu, X., Li, Y., and Hsieh, C.-J. *A Unified Framework for Data Poisoning Attack to Graph-Based Semi-Supervised Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Mujkanovic, F., Geisler, S., Gunnemann, S., and Bojchevski, A. Are defenses for graph neural networks robust? *ArXiv*, abs/2301.13694, 2023.

Paulus, M. B., Choi, D., Tarlow, D., Krause, A., and Maddison, C. J. Gradient estimation with stochastic softmax tricks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Qian, S., Ying, H., Hu, R., Zhou, J., Chen, J., Chen, D. Z., and Wu, J. Robust training of graph neural networks via noise governance. WSDM '23, pp. 607–615, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394079. doi: 10.1145/3539597.3570369. URL https://doi.org/10.1145/3539597.3570369.

Sabanayagam, M., Esser, P. M., and Ghoshdastidar, D. Representation power of graph convolutions : Neural tangent kernel analysis. *ArXiv*, abs/2210.09809, 2022.

Sun, L., Dou, Y., Yang, C., Zhang, K., Wang, J., Liu, Y., Yu, P. S., He, L., and Li, B. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

Tian, Z., Cui, L., Liang, J., and Yu, S. A comprehensive survey on poisoning attacks and countermeasures in machine learning. *ACM Computing Surveys*, 55(8):1–35, 2022.

Tramèr, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *ArXiv*, abs/2002.08347, 2020.

Wu, F., Zhang, T., de Souza, A. H., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks. *ArXiv*, abs/1902.07153, 2019a.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019b.

Zhang, M., Hu, L., Shi, C., and Wang, X. Adversarial label-flipping attack and defense for graph neural networks. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 791–800, 2020. doi: 10.1109/ICDM50108.2020.00088.

A. Appendix

In subsection A.1, we provide additional details on the motivation experiment. In subsection A.2, we provide details on the expanded meta attack. subsection A.3 contains details regarding our NTK-based attacks. In subsection A.4, we provide the proof for the proposition introduced in the main paper and additional theoretical results on the MILP. In subsection A.6, we provide additional experiments along with ablative studies to offer more insights. In subsection A.5, we provide details on extending the baselines to multi-class. Finally, in subsection A.7, we provide details on dataset statistics. First, we discuss how we tune the hyper-parameters in both the attack and the defense phase of our evaluation.

Attack phase tuning. For our LSA family of attacks, we tune the regularisation term λ in range [1e-3, 1e3] in logarithmic steps. For the meta attacks, we set the learning rate for the inner and the outer routine optimization to 1e-2 and 1e-1. The weight decay for both the optimizers is set to 5e-4. The number of inner optimization iterations (epochs) is set to 15. Additional gains in the attack strength can be achieved by further fine-tuning these hyper-parameters. We refrain from doing this due to lack of computational resources, since our evaluation setup is already quite expensive. For the learning-based baseline attacks (LP, MG, LFK), we sweep through the hyper-parameters ranges suggested by the authors in their papers.

Defence phase tuning. Post obtaining the poisoned labels, we perform thorough hyper-parameter tuning of the target model. Specifically, for the shared general hyper-parameters we sweep the following ranges: [0.1, 0.01, 0.05, 0.08] for the learning rate, [0.0, 0.005, 0.0005, 0.00005] for weight decay, and [0.3, 0.5, 0.7] for dropout. We fix the hidden dimensions to 64, again due to limited computation resources. Model specific hyper-parameters are tuned in the ranges suggested by the respective papers. We empirically observe that higher learning rates allow models to recover better from label poisoning attacks. We use Optuna (Akiba et al., 2019) to optimize the hyper-parameters search, and set the number of trials to 20. For learning we use the Adam optimizer (Kingma & Ba, 2015), with 1000 maximum number of epochs and an early stopping patience of 100. The majority of our experiments are run on an Nvidia k80 GPU with 24GB memory, and the remaining experiments were run using an A100 GPU with 40GB of memory.

A.1. Motivation experiment

For the motivation experiment described in the main paper, we consider the Cora-ML dataset and sub-sample the largest-connected-component with roughly 100 nodes. We refer to this dataset as Cora-ML-tiny. The final sampled dataset has 93 nodes and 3 classes. We create 10 different splits with training, validation and test set sizes equal to 19, 19 and 55 correspondingly. After exhaustively enumerating over all possible label flips with budget equal to 1, we find that a single adversarial label flip can cause a staggering reduction of $\sim$17% in test performance.

A.2. Expanded meta attack

We first construct $\boldsymbol{H} \in \{0,1\}^{N \times C-1 \times C}$ by enumerating over all possible false (one-hot) labels for every train node. Next, we initialize a parameter matrix $\boldsymbol{B} \in \mathbb{R}^{N \times C-1 \times C}$ from a uniform distribution. The idea is to learn $\boldsymbol{B}$ to select a good subset of poisoned labels. To enforce the budget we apply *soft-top-k* followed by *k-subset-selection* (Paulus et al., 2020) on top of $\mathrm{vec}(\boldsymbol{B})$ where $\mathrm{vec}(\cdot)$ vectorizes the matrix.[2] We construct the final poisoned labels as:

$$\widehat{\boldsymbol{Y}}_l = \sum_{c=0}^{C-1} (\boldsymbol{B} \odot \boldsymbol{H})_{:,c,:} + ((1 - \boldsymbol{B}) \odot \boldsymbol{Y}_l)_{:,c,:} \tag{4}$$

We feed $\widehat{\boldsymbol{Y}}_l$ into the inner optimization problem which we train for a fixed number of epochs, and use meta-gradients of $\boldsymbol{B}$ w.r.t. the outer loss for learning. During the forward pass we used hard top-k for $\boldsymbol{B}$, sampled proportional to the respective soft-top-k scores which are used during the backward pass.

A.3. NTK attack

Our variants of the NTK-based attacks are exactly the same as our SGC variants, except in how we construct $\widehat{\boldsymbol{X}}$. For the NTK variants, $\widehat{\boldsymbol{X}}$ corresponds to the NTK kernel matrix for a 2 layer ReLU GCN as derived by Sabanayagam et al. (2022) (see Eq. 5 in their paper). Given $\widehat{\boldsymbol{X}}$ we generate poisoned labels following the procedure in subsection 3.1.

[2] We place no additional constraints on $\boldsymbol{B}$ to ensure multiple flips are not selected for the same label. In practice this is not an issue, since the optimizer learns to use the entire budget and select k-unique label flips.

A.4. Additional detailed on the MILP formulation of our linear surrogate attacks

In one variation of our LSA attacks instead of learning $\boldsymbol{H}$, we fix it to some preset target false labels. As we showed in Theorem 3.1 the benefit of fixing $\boldsymbol{H}$ is that we can solve the MILP extremely efficiently. We provide the proof below. We explore the following variants:

- $\boldsymbol{H}$-MARGIN: we perturb labels to the false label with the highest probability as predicted by a clean model (trained using ground truth training labels).
- $\boldsymbol{H}$-RANDOM: we perturb labels to a randomly chosen false class.
- $\boldsymbol{H}$-BINARY+: we sort all classes w.r.t their frequency in a descending order and pair consecutive classes to flip them.

We compare these variants for $\boldsymbol{H}$ in subsection A.6. In the main text we use $\boldsymbol{H}$-MARGIN for the -FIX attacks.

Proof (Theorem 3.1). When $\boldsymbol{H}$ is fixed and $\mathcal{L}(\boldsymbol{Y}_u, \widetilde{\boldsymbol{Y}}_u) = \text{sum}(\boldsymbol{Y}_u \odot \widetilde{\boldsymbol{Y}}_u)$ we can rewrite the MILP in Equation 2 as

$$\underset{\boldsymbol{b}\in\{0,1\}^N}{\arg\min}\ \text{vec}(\boldsymbol{Y}_u)^T \text{vec}\left(\widehat{\boldsymbol{X}}_u \widetilde{\boldsymbol{X}}_l \left(\boldsymbol{b} \odot \boldsymbol{H} + (\boldsymbol{1}_N - \boldsymbol{b}) \odot \boldsymbol{Y}_l\right)\right) \qquad \boldsymbol{b}^T \boldsymbol{1}_N \leq \epsilon N \tag{5}$$

We can simplify Equation 5 without changing the minimum by omitting the terms that do not depend on $\boldsymbol{b}$:

$$\underset{\boldsymbol{b}\in\{0,1\}^N}{\arg\min}\ \text{vec}(\boldsymbol{Y}_u)^T \text{vec}\left(\widehat{\boldsymbol{X}}_u \widetilde{\boldsymbol{X}}_l \left(\boldsymbol{b} \odot (\boldsymbol{H} - \boldsymbol{Y}_l)\right)\right) \qquad \boldsymbol{b}^T \boldsymbol{1}_N \leq \epsilon N \tag{6}$$

Now if we define $\boldsymbol{Q} = \widehat{\boldsymbol{X}}_u \widetilde{\boldsymbol{X}}_l$, $\boldsymbol{P} = \boldsymbol{Y}_l - \boldsymbol{H}$ and $\boldsymbol{R} = \boldsymbol{Y}_u$ we can again rewrite as:

$$\underset{\boldsymbol{b}\in\{0,1\}^N}{\arg\min}\ \boldsymbol{c}^T \boldsymbol{b} \qquad \boldsymbol{b}^T \boldsymbol{1}_N \leq \epsilon N$$

where the n-th element of $\boldsymbol{c}$ is computed as $c_n = \sum_{ij} Q_{in} P_{nj} R_{ij}$. Since the objective in Equation 6 equals $\sum_{ij} \sum_{n=1}^{N} (Q_{in} \cdot b_n \cdot P_{nj} \cdot R_{ij})$, exchanging the order of the sums and taking b_n outside we obtain $\sum_n b_n c_n = \boldsymbol{c}^T \boldsymbol{b}$. This problem is equivalent to selecting up to ϵN elements from $\boldsymbol{c}$ such that their sum is minimized. The optimal solution is to select the ϵN smallest negative elements. □

Efficiently solving the original MILP. When $\boldsymbol{H}$ is fixed we can efficiently obtain the optimal solution from Theorem 3.1. However, as well show next, even when we optimize over $\boldsymbol{H}$ (i.e. it is not fixed) we can still solve the MILP efficiently. First we rewrite Equation 2 in the following equivalent form where we now only optimize over $\boldsymbol{H}$, omitting $\boldsymbol{b}$.

$$\begin{aligned} \boldsymbol{H}^* = \underset{\boldsymbol{H}\in\{0,1\}^{N\times C},}{\arg\min}\ & \text{sum}(\boldsymbol{Y}_u \odot \widehat{\boldsymbol{X}}_u \widetilde{\boldsymbol{X}}_l \boldsymbol{H}) \\ & \boldsymbol{H}\boldsymbol{1}_C = \boldsymbol{1}_N \\ & \|\boldsymbol{H} - \boldsymbol{Y}_l\|_1 \leq \epsilon 2N \end{aligned} \tag{7}$$

We multiple by 2 in the budget constraint since $\boldsymbol{H}$ and $\boldsymbol{Y}_l$ differ in either 0 or 2 entries in each row.

We will show that relaxing $\boldsymbol{H}$ from $\{0,1\}^{N\times C}$ to $[0,1]^{N\times C}$ in Equation 7 still results in an integral (i.e. binary) solution. Therefore, we can solve the MILP by solving the corresponding relaxed LP.

Proposition A.1. *The optimal solution of Equation 7 is integral, i.e. $\boldsymbol{H}^* \in \{0,1\}^{N\times C}$, when relaxing the MILP to an LP such that $\boldsymbol{H} \in [0,1]^{N\times C}$.*

Proof. Let $M = N \cdot C$. We can rewrite Equation 7 in the following canonical form

$$\begin{aligned} \min_{\boldsymbol{x}\in\{0,1\}^{2\cdot M}}\ & \boldsymbol{c}^T x \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h} \end{aligned}$$

where $\boldsymbol{A} = [\underbrace{\boldsymbol{I}_N, \ldots, \boldsymbol{I}_N}_{C \text{ times}}, \boldsymbol{O}]$, $\boldsymbol{I}_N$ is the identify matrix, $\boldsymbol{O}$ is an $N \times M$ [zero matrix, $\boldsymbol{b} = \mathbf{1}_N$ is the ones vector, $\boldsymbol{G} = \begin{bmatrix} -\boldsymbol{I}_M & -\boldsymbol{I}_M \\ \boldsymbol{I}_M & -\boldsymbol{I}_M \\ \mathbf{0}_M & \mathbf{1}_M \end{bmatrix}$, $\mathbf{0}_M$ is the zeros vector, and $\boldsymbol{h} = [-\text{vec}(\boldsymbol{Y}_l), \text{vec}(\boldsymbol{Y}_l), \epsilon 2N]$. Here, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ implements the one-hot constraint, while $\boldsymbol{G}\boldsymbol{x} \leq \boldsymbol{h}$ implements the L_1-norm budget constraint, where we have twice the number of variables to encode the absolute value. Since $\boldsymbol{b}$ and $\boldsymbol{h}$ are integral (because $\boldsymbol{Y}_l$ is one-hot encoded), to show that the optimal solution is integral we only need to show that $\boldsymbol{A}$ and $\boldsymbol{G}$ are totally unimodular. Since the identity matrix $\boldsymbol{I}_N$ is totally unimodular, the result directly follows given the fact that if $\boldsymbol{T}$ is totally unimodular then so are $-\boldsymbol{T}, \boldsymbol{T}^T, [\boldsymbol{T}, \boldsymbol{I}]$, and $[\boldsymbol{T}, -\boldsymbol{T}]$. □

A.5. Extending the baselines to multi-class

As a consequence of the binary-class approach, the baselines might have a limit on the maximum perturbation budget. For example, in the default (previously-used, non-CV) evaluation setting for the Cora-ML dataset the candidate set of potential flips is of size at most 20% since Cora-ML has 7 classes and the two most frequent classes span up to 20% of the training set depending on the split. Therefore, LAFAK and LPATTACK by default cannot accommodate higher budgets. To enable this, while not deviating from the original design of the attack, we propose a minor extension. We first exhaust the 20% budget by perturbing labels of the two most frequent classes and then we fix these perturbed labels. For the remaining budget, we perturb the clean labels, but by restricting the attack scope to a candidate set consisting of the next two most frequent classes. This process is repeated until the budget is completely exhausted. The multi-class setting can be handled better, however, we restrain from such adaptations to remain true to the original design of the attack. This also means that for the 20% budget, in the previous evaluation setting, LAFAK and LPATTACK become equivalent and deterministic because all the labels of the nodes in the candidate set are flipped to their counterpart.

A.6. Additional experiments

A.6.1. ADDITIONAL EXPERIMENTS ON STRONGEST ATTACKS

To extend our main paper analysis on strongest attacks, we provide additional experimental results for our strongest attacks (SGC-BIN and META-BIN) on several datasets and models. We plot the results in Figure 6. Our findings from the main paper that our attacks significantly outperform previous attacks still hold.

A.6.2. SCALING OUR ATTACKS TO LARGER GRAPHS AND GRAPHS WITH LARGER NUMBER OF CLASSES

To further study the efficacy of our proposed attacks, we extend our analysis to two additional datasets. Specifically, we evaluate on the CoraFull dataset (Bojchevski & Günnemann, 2017) that contains 70 output classes, and the OGBN-ArXiv dataset (Hu et al., 2020) – a medium sized graph with $\sim$ 169K nodes. Following Lingam et al. (2023b), we apply a dimensionality reduction on the node features of CoraFull using PCA (top 500 dimensions). The statistics for all 5 datasets are tabulated in Table 7.

Larger number of classes. Since the number of classes for CoraFull is relatively higher compared to the other datasets, the choice of how we fix $\boldsymbol{H}$ can be potentially more impactful. Therefore, we compare the three variants outlined in subsection A.4 against the baselines. We can infer from Table 3 that our attacks outperforms existing attacks, and that the -BINARY+ variant is the best on average. These results signify that our attacks also scale well to datasets with larger number of classes.

Table 3. Comparing different attacks on the CoraFull dataset. The strategy for how to initialize $\boldsymbol{H}$ can impact the attack performance. The -MARGIN and -BINARY+ variants outperform -RANDOM.

ϵ	H-MARGIN	H-BINARY+	H-RANDOM	DEG	RND	LP	MG
5%	$56.14_{\pm 0.53}$	**55.96**$_{\pm \mathbf{0.61}}$	$57.41_{\pm 0.52}$	**55.70**$_{\pm \mathbf{0.42}}$	$57.01_{\pm 0.45}$	$57.38_{\pm 0.26}$	$56.19_{\pm 0.53}$
10%	$51.27_{\pm 0.54}$	**50.22**$_{\pm \mathbf{0.84}}$	$54.49_{\pm 0.50}$	$54.02_{\pm 0.62}$	$55.21_{\pm 0.59}$	$53.52_{\pm 0.27}$	$52.15_{\pm 0.48}$
15%	$46.31_{\pm 0.57}$	**43.99**$_{\pm \mathbf{0.27}}$	$51.67_{\pm 0.69}$	$52.86_{\pm 0.66}$	$54.74_{\pm 0.73}$	$49.66_{\pm 0.12}$	$47.58_{\pm 0.53}$
20%	$41.47_{\pm 0.28}$	**38.10**$_{\pm \mathbf{0.54}}$	$48.30_{\pm 0.93}$	$51.53_{\pm 0.71}$	$53.56_{\pm 0.52}$	$49.08_{\pm 0.08}$	$43.02_{\pm 0.56}$
30%	$33.29_{\pm 0.68}$	**27.92**$_{\pm \mathbf{0.30}}$	$40.93_{\pm 1.60}$	$48.56_{\pm 0.86}$	$51.29_{\pm 0.75}$	$49.10_{\pm 0.10}$	$31.71_{\pm 0.08}$

Table 4. Comparing different attacks on OGBN-ArXiv. Our SGC-BINattack also scales to large graphs, and still outperforms existing attacks by a significant margin. The LFKattack, with its current implementation, could not be executed because of TLE (Time Limit Exceeded) error.

ϵ	SGC-BIN	RND	DEG	LP	MG	LFK
1%	**64.31**	67.14	66.56	66.15	67.21	TLE
3%	**62.17**	67.09	66.36	60.56	66.69	TLE
5%	**48.20**	66.90	66.12	55.62	66.72	TLE

Table 5. Evaluation of the extended variant of our meta attack on a GCN model for different datasets.

ϵ	CoraML	Citeseer	Pubmed
5%	$80.68_{\pm 1.65}$	$70.62_{\pm 1.31}$	$76.15_{\pm 1.34}$
10%	$79.57_{\pm 1.86}$	$68.85_{\pm 0.73}$	$73.20_{\pm 1.97}$
15%	$77.76_{\pm 1.30}$	$66.96_{\pm 1.80}$	$69.02_{\pm 3.60}$
20%	$75.34_{\pm 1.74}$	$64.24_{\pm 3.03}$	$61.28_{\pm 3.86}$
30%	$71.73_{\pm 2.13}$	$60.08_{\pm 3.41}$	$50.51_{\pm 6.71}$

Table 6. The effect of loss function on META-BIN attack's performance on the Cora-ML dataset. The proposed Gumbel-softmax loss function outperforms the regular cross-entropy loss and previously proposed margin loss.

ϵ	Cross Entropy Loss	Margin Loss	Gumbel-softmax Loss
5%	**78.51 (2.18)**	79.25 (1.68)	79.80 (2.12)
10%	75.96 (1.39)	75.27 (1.93)	**73.35 (2.34)**
15%	67.78 (1.81)	68.75 (3.20)	**65.81 (3.30)**
20%	63.36 (3.56)	63.88 (2.90)	**61.60 (3.37)**
30%	56.13 (3.05)	56.17 (2.74)	**54.53 (2.28)**

Larger graphs. Next, we study the medium-sized OGBN-ArXiv dataset to test scalability to larger graphs. Since the OGBN-ArXiv dataset contains large number of training nodes, we evaluate the attacks on budget ranges $\{1\%, 3\%, 5\%\}$. We also use the default splits since the data is chronologically split. In Table 4 we see that our proposed attack, SGC-BIN, outperforms existing attacks.

A.6.3. CONFUSION MATRIX ANALYSIS

To analyse how the test node predictions are affected by the strongest baseline attack (LFK) and our strongest proposed attack (SGC-BIN), we plot the confusion matrices for the 15% budget in Figure 4. Even though both LFK and SGC-BIN poison the training nodes belonging to the same two classes (class 2 and class 4), after the SGC-BIN attack the model more consistently confuses the test predictions for these two classes.

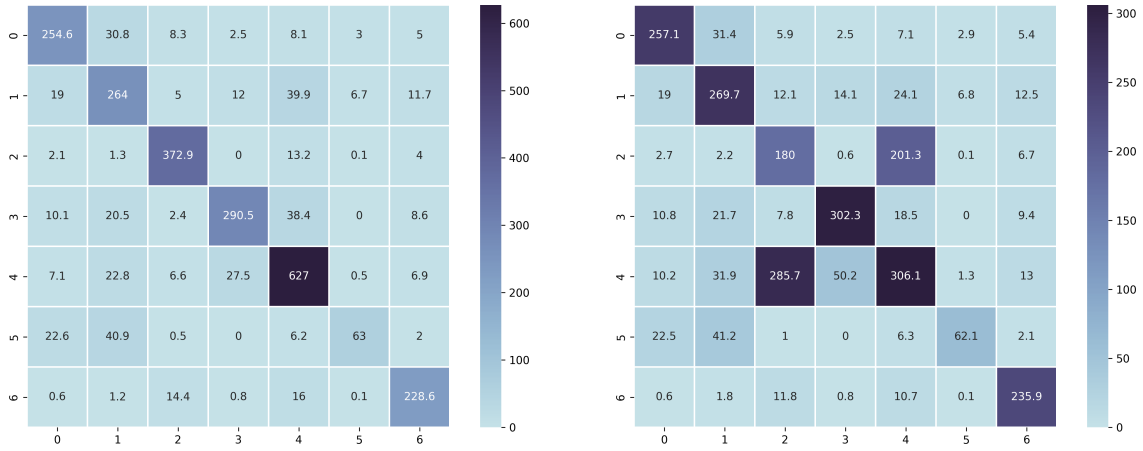

Figure 4. Confusion matrices for test set for a GCN model poisoned with the LFK (left) and SGC-BIN (right) attacks using a 15% poisoning budget on the Cora-ML dataset.

A.6.4. EFFECT OF LOSS FUNCTION

We proposed a new Gumbel-Softmax based loss function in the main paper. To study the effect of loss function on attack performance, we perform an ablative study by varying the loss function for the META-BINattack and tabulate the results in Table 6. We observe that the proposed loss function performs better in general.

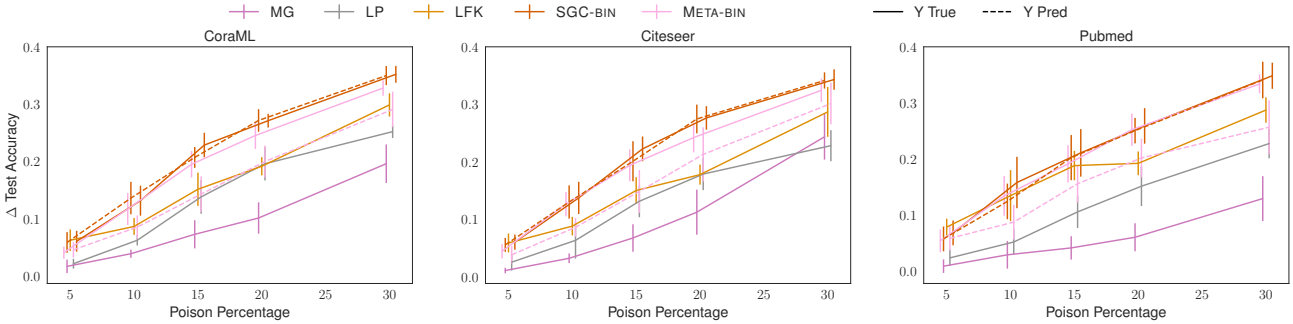

Figure 5. Evaluating our attacks under the two different threat models given access to: ground-truth test labels vs. only test predictions. Our attacks outperform the baselines even without access to ground-truth test labels.

A.6.5. Extended meta attack

In Table 5 we evaluate the extended variant of our meta attack. Overall, we can conclude that this variant is weaker compared to the other meta variants (and our surrogate attacks) while still outperforming some of the baselines. In any case, we include these results for completeness.

A.6.6. Attacking without access to ground-truth test labels

In our main paper, we described a second threat model, where the ground truth labels are only available for training nodes and only the predictions are available for test nodes (no ground-truth labels). We evaluate our strongest attacks SGC-BIN and META-BIN in this setting and plot the results on Figure 5. We observe that the performance of our attacks in this setting is close to the worst-case setting where the attacker knows the test ground-truth labels. Especially for SGC-BIN there is almost no difference between using the ground-truth labels vs. the predictions. Additionally, our attacks that use predictions also outperform the baselines that use ground-truth labels.

A.7. Dataset statistics

In Table 7, we tabulate dataset statistics. We additionally include the train/val/test split statistics for the default and the proposed CV setting. Note that in the CV setting, the test accuracy is measured over all the remaining unlabeled nodes, and the train and val set have the same size.

Table 7. Dataset statistics.

Dataset	Nodes	Features	Classes	Default Train/Val/Test	CV Train/Val/Test
Cora-ML	2,810	2,879	7	140 / 500 / 1000	140 / 140 / 2530
Citeseer	2,110	3,703	6	120 / 500 / 1000	120 / 120 / 1870
Pubmed	19717	500	3	60 / 500 / 1000	60 / 60 / 19597
CoraFull	18800	500	70	-	1400 / 1400 / 16000
OGBN-ArXiv	169343	128	40	90941 / 29799 / 48603	90941 / 29799 / 48603

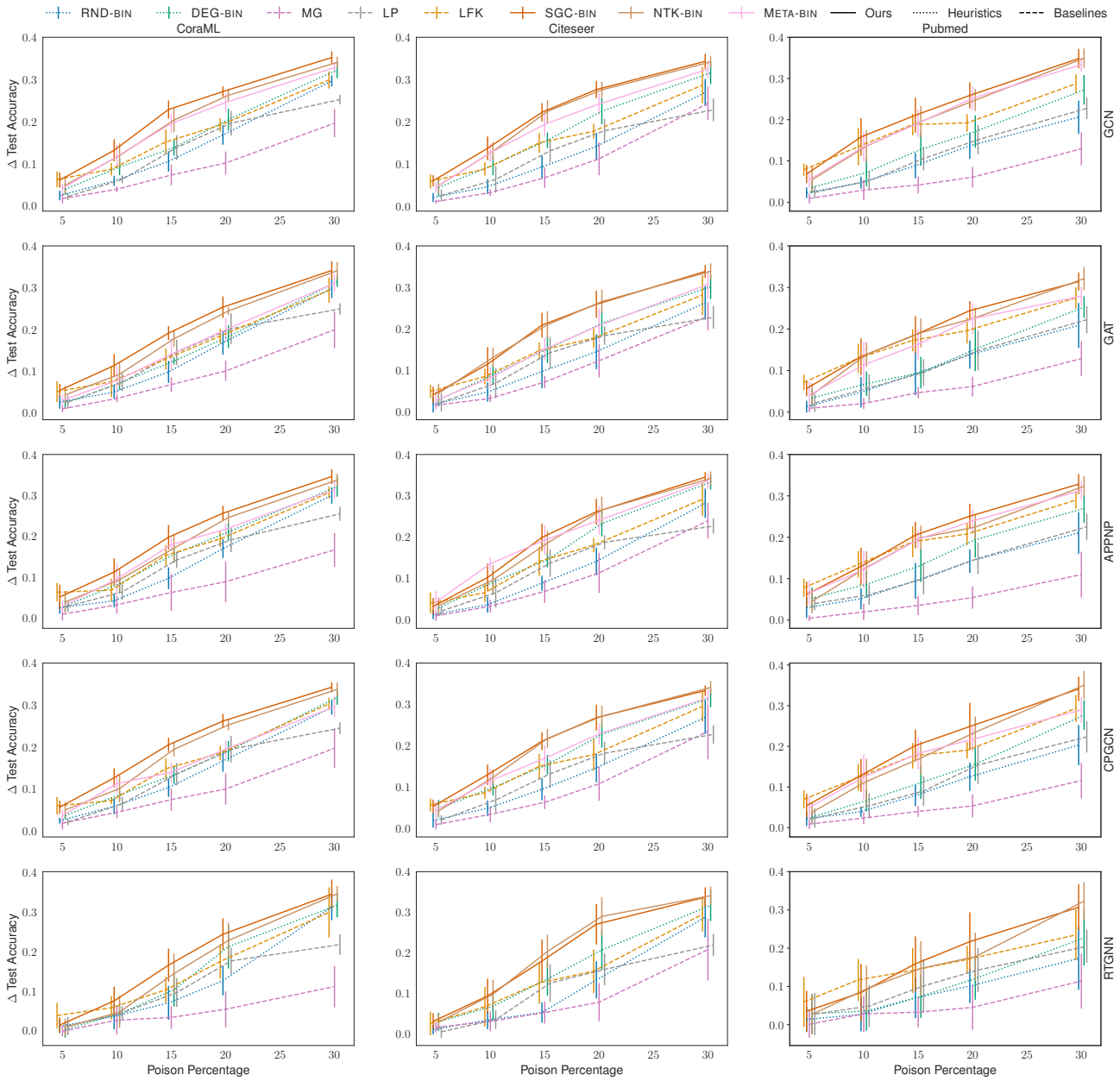

Figure 6. Our strongest attacks significantly outperform the baselines across various models and datasets.