PITFALLS IN EVALUATING GNNS UNDER LABEL POISONING ATTACKS

Vijay Lingam, Mohammad Sadegh Akhondzadeh & Aleksandar Bojchevski
CISPA Helmholtz Center for Information Security

ABSTRACT

Graph Neural Networks (GNNs) have shown impressive performance on several graph-based tasks. However, recent research on adversarial attacks shows how sensitive GNNs are to node/edge/label perturbations. Of particular interest is the label poisoning attack, where flipping an unnoticeable fraction of training labels can adversely affect GNNs' performance. While several such attacks were proposed, the latent flaws in the evaluation setup cloud the true effectiveness of the attacks. In this work, we uncover 5 frequent pitfalls in the evaluation setup that plague all existing label-poisoning attacks for GNNs. We observe for some settings that the state-of-the-art attacks are no better than a random label-flipping attack. We propose and advocate for a new evaluation setup that remedies the shortcomings, and can help gauge the potency of label-poisoning attacks fairly. Post remedying the pitfalls, on the Cora-ML dataset, we see a difference in performance of up to 19.37%.[1]

1 INTRODUCTION

Graph Neural Networks (GNNs) are a popular class of methods for graph learning tasks including node classification, graph classification, and link prediction (Wu et al., 2019b). GNNs are efficient at exploiting both the node features and topological structure to learn better representations for the task at hand. Recently, they have made their way into several applications including safety-critical domains (e.g., drug discovery (Xiong et al., 2021)). With the increase in their applicability, it is necessary to understand and ensure the reliability of their predictions.

Recent research on adversarial data poisoning attacks on GNNs (Jin et al., 2021; Sun et al., 2018) exposes the vulnerabilities of GNNs. Adversarial attacks can be broadly classified into evasion and poisoning attacks. Evasion attacks are conducted during the testing phase on a trained model and generally applied in an inductive setting, while poisoning attacks perturb the training data before the training phase. From an attack perspective, evasion is an easier setting because the model is fixed. Poisoning on the other hand is a more difficult setting because the defender has more options to safeguard against the attack. Poisoning attacks apply unnoticeable perturbations to the input graph structure and/or node features and show how this can adversely affect the test performance of GNNs; reaffirming their sensitivity.

One manifestation of poisoning attacks is label poisoning, where a malicious user can introduce corrupted labels into the dataset. It is plausible to imagine such a scenario when training data is scraped from a public source like the internet, where a user can modify the data. It then becomes crucial to analyze how resistant or vulnerable GNNs are to corrupted training labels. Motivated by this, we are interested in evaluating the robustness of GNNs to label poisoning attacks, where a small fraction of training labels are flipped before training.

Several label poisoning attacks (Liu et al., 2019; Zhang et al., 2020; Liu et al., 2022) have been proposed in the literature. To safeguard models against these attacks (Zhang et al., 2020) proposed a defense framework for GNNs against adversarial label poisoning attacks, and (Dai et al., 2021; Yu et al., 2019) proposed defense mechanisms against random label noise. The proposed attacks choose a GNN model, generate poisoned labels, and re-train the GNN model using the default hyper-parameters, which were tuned for clean labels. This evaluation setup, while appearing benign

[1] https://github.com/VijayLingam95/PitfallsOfLabelPoisoningAttacksForGNNs

on the surface, paints a distorted picture of the efficiency and effectiveness of poisoning attacks. In this work, we uncover the pitfalls associated with the common evaluation setup of label poisoning attacks. In a nutshell, our contributions are: 1) We identify 5 pitfalls with the common evaluation setup. 2) We propose a new evaluation setup that is more realistic. 3) We conduct extensive experiments in the new setting and provide insights.

2 Related Work & Background

Over the past few years, the robustness of GNNs against adversarial attacks has received an increasing amount of attention. In several existing works, the adversary is granted the capability to add/remove an unnoticeable amount of edges from the graph (Jin et al., 2021; Geisler et al., 2021), corrupt node features (Ma et al., 2020), and/or manipulate node labels (Liu et al., 2019; Zhang et al., 2020; Liu et al., 2022). In this work, we limit our purview to label poisoning attacks, where the adversary can flip labels for a small fraction of training nodes.

Problem setting: We focus on the semi-supervised node classification task for graphs. We are given access to a graph G = ($\mathcal{V}$, $\mathcal{E}$), where $\mathcal{V}$ is the set of N nodes and $\mathcal{E}$ is the set of edges. G is described by an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N\times N}$. Let $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ denote a d-dimensional feature matrix for all the nodes in the graph, and $\boldsymbol{Y} \in \{1,2,...C\}^N$ denote node labels and represented as one-hot vectors. The goal is to predict labels for unlabeled nodes $\boldsymbol{y}_u$, given $(f(\theta), \mathbf{A}, \mathbf{X}, \boldsymbol{y}_l)$, where $f(\theta)$ is a graph based learning algorithm and $\boldsymbol{y}_l$ denotes the labels of training nodes $\mathcal{V}_l$. In label-poisoning attacks, the objective of the attacker is to perturb a small fraction of training labels $\boldsymbol{y}_l$, such that $\boldsymbol{y}_u$ is largely affected. The label-poisoning attacks for a given model can be viewed as a bi-level optimization problem:

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

in the above equation, N_l = $|\mathcal{V}_l|$, $\widehat{\boldsymbol{y}}_l$ denotes poisoned training labels, and L is the classification loss. ϵN_l is the allowed budget for perturbation and $\epsilon \in (0,1)$; in other words, ϵN_l represents the maximum number of labels that can be flipped.

We briefly describe both heuristic and more sophisticated label-poisoning attacks that we assess in this paper below. More details can be found in the respective papers.

Heuristics-based attacks. We compare against two baseline attacks. *Random Label Flipping Attack* **(Random)** which randomly perturbs the labels for a budget number of nodes. *Degree-based Label Flipping Attack* **(Degree)** perturbs nodes with the highest degrees given a budget.

Learning-based attacks. These attacks, in general, derive perturbed labels from a fixed surrogate model, for which either an approximate or exact closed-form solution is employed. The poisoned labels are then applied to a target model. *Label Propagation Attack* **(LPAttack)** Liu et al. (2019) proposed the first gradient-based attack 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. Under the lens of Equation 1, LPAttack replaces the inner optimization with the closed-form solution of Label Propagation, thus making the bi-level optimization tractable. A probabilistic solver combined with a gradient descent optimizer is proposed to solve the outer routine.

Label-Flipping Attack **(LafAK)** Zhang et al. (2020) builds on top of LPAttack by replacing the inner optimization in Equation 1 with an approximate closed form solution of a linearized GCN, where the non-linearities between layers are removed. An example of linearizing 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 square loss – i.e. using regression to perform classification: $\boldsymbol{\theta}^* = \frac{1}{N_l}\arg\min_{\boldsymbol{\theta}}\|(\widehat{\boldsymbol{A}}^2\boldsymbol{X}\boldsymbol{\theta})_l - \boldsymbol{y}_l\|_2^2$ – and obtain closed-form solution using the OLS

estimator. Since LAFAK restricts its scope to binary classification, the two most frequent classes are considered and their labels are cast to $\{-1,+1\}^N$. Labels of training nodes are $\boldsymbol{y}_l \in \{-1,+1\}^{N_l}$.

$$\boldsymbol{\theta}^* = ((\widehat{\boldsymbol{A}}^2\boldsymbol{X})_l^T(\widehat{\boldsymbol{A}}^2\boldsymbol{X})_l + \lambda\boldsymbol{I})^{-1}(\widehat{\boldsymbol{A}}^2\boldsymbol{X})_l^T(\boldsymbol{y}_l) \tag{2}$$

In equation 2, the subscript 'l' refers to rows corresponding to training nodes. Similar to LPATTACK, the non-differentiable parts of the outer routine are substituted with continuous surrogates and a gradient descent-based optimizer is employed.

Maximum Gradient Attack **(MG)** Liu et al. (2022) propose a label propagation-based attack for GCN like models. Unlike LPATTACK which uses a similarity matrix that is constructed by applying a gaussian kernel to the feature matrix, MG proposes multiple ways to construct the propagation matrix $\bar{\boldsymbol{A}}$ (e.g., pagerank matrix, higher-order adjacency matrix). The gradient for a given epoch is computed as $g = \nabla_{\boldsymbol{y}_l} - \frac{1}{2}\|\bar{\boldsymbol{A}}\boldsymbol{y}_l - \boldsymbol{y}_u\|$. The top budget number of gradients in g are then selected and traced back to the corresponding nodes. The labels for these nodes are then set to max label class.

For multi-class datasets, LAFAK and LPATTACK consider as a candidate set only the nodes whose labels belong to the two most frequent classes, restricting the attack to flips among these two classes. In the default setting (20 nodes per class), the candidate set is of size at most 20% (the Cora-ML dataset has 7 classes and the two most frequent classes span 20% of the training set). 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. For 20% budget, on default splits, LAFAK and LPATTACK become equivalent and deterministic because all the labels of nodes in the candidate set are flipped to their counterpart.

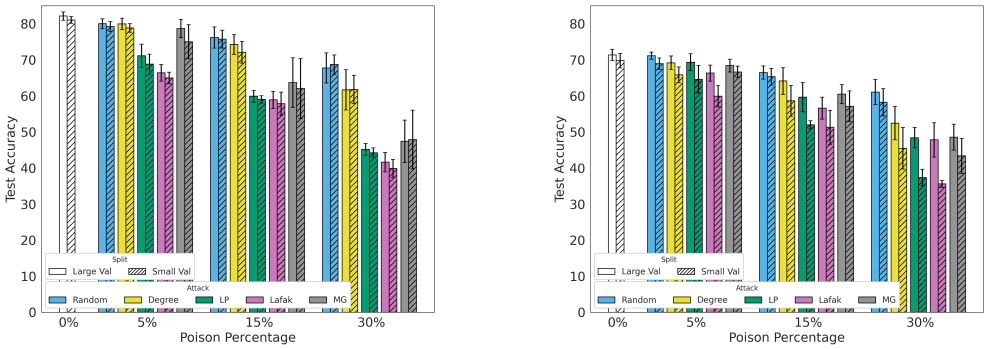
3 PITFALLS

On closer inspection, we infer that the current attacks do not simulate the full potential of a defender. In short, the dataset splits and the training/tuning routine simulated by the attacker when evaluating the strength of their proposed attacks in all previous works is not realistic. In the rest of this section, we carve out the specifics of the pitfalls we identified and empirically validate them with our experiments and provide remedies. We conduct our experiments on three datasets namely Cora-ML (McCallum et al., 2000), Citeseer (Sen et al., 2008), and Pubmed (Namata et al., 2012)[2]. We report our experimental results using three different GNNs: GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and APPNP (Klicpera et al., 2019). We sweep the poisoning budget over the range [5%, 10%, 15%, 20%, 30%]. In practice, smaller budgets are of more relevance. Additional experiments and the setup details can be in found in the Appendix (A.3 and A.1).

1) Large Validation Set. All existing label poisoning attacks (LPATTACK, LAFAK, MG) evaluate on data splits with validation (val) set size much larger than that of training (train) set. To substantiate, in the default setting as tabulated in Table 1 in the Appendix, the val set contains 500 nodes and the train set contains 20 nodes per class (e.g. 140 nodes in the Cora-ML dataset). Moreover, the labels of the validation set are assumed to be clean – not poisoned. We hypothesize that having a larger val set with clean labels might aid the model in recovering accuracy by avoiding overfitting on the poisoned train labels. (Shchur et al., 2018) expose the shortcomings of using a large val set for graph data. Besides, in this unrealistic scenario, the defender could simply ignore the given (potentially poisoned) training data and train the model on val data, thus circumventing the attacker.

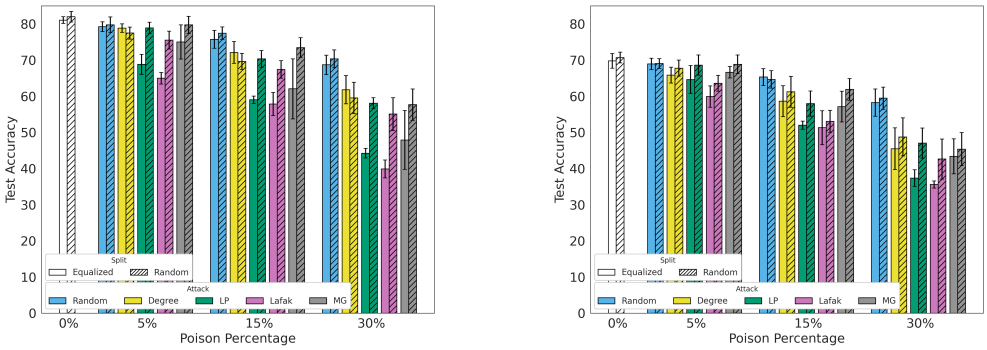
To alleviate this issue, we intervene on the val set size with labels intact. We postpone our intervention on the fact that val set labels are clean to a later experiment to better isolate the effect of the two. We create 10 new data splits by sampling 20 nodes per class for both train and val sets. We use the default hyper-parameters (we explore the effect of tuning in Pitfall 3) suggested by the authors of

[2]In line with previous work we use the largest connected component, since attacks that use LP as a surrogate are likely to suffer if there are multiple components. This is another potential pitfall that is out of our scope.



(a) Cora-ML (b) Citeseer

Figure 1: Effect of the validation set size on attack performance across datasets. Larger val set helps recover accuracy in general.



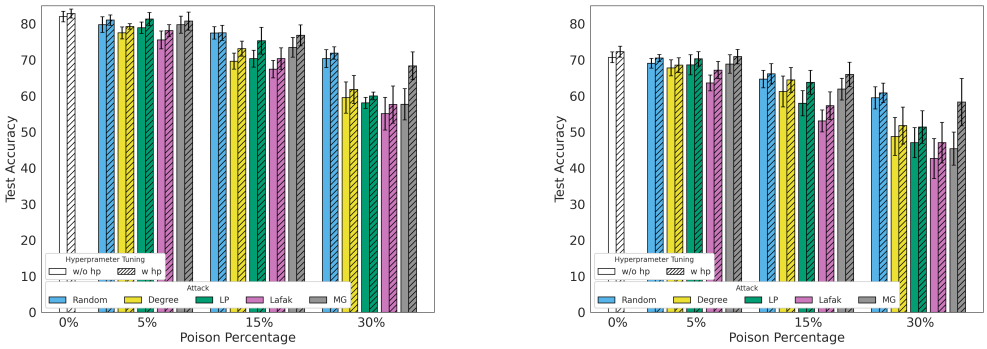
(a) Cora-ML (b) Citeseer

Figure 2: Attack performance using class equalized v/s random splits. Learning-based attack performance significantly degrades by switching to a more realistic setting (random).

respective models. Figure 1 validates our hypothesis that having a large val set can help recover test performance in general. The drop in accuracy is steeper for learning-based attacks. Shifting from large to small val setting, the test performance can drop by as much as 5%.

2. Equalized vs Random split. We previously advocated using a small validation set to reflect a more realistic setting. Both the train and val sets contain equal number of samples per class. We refer to this setting as class equalized (equalized). However, this setting does not preserve global class distribution and treats each class equally. Another potential way to split the dataset is to randomly sample nodes for train and val sets (referred to as random setting). We argue this to be a more realistic setting, as the class distribution of train and val sets would closely align with the true class distribution. To create random splits, we randomly sample $|\text{training set}|$ nodes for train and val sets separately. For equalized splits, we retain the previous splits where we sampled 20 nodes per class for both train and val sets. We create 10 different splits for both these settings and run our experiments using default hyper-parameters and plot our results in Figure 2.

From Figure 2, we deduce that switching to a more realistic setting (random) significantly degrades the performance of label-poisoning attacks. RANDOM and DEGREE attacks are less susceptible to change in data splits. This phenomenon is accentuated for learning-based attacks, revealing their sensitivity to the data splits.



(a) Cora-ML

(b) Citeseer

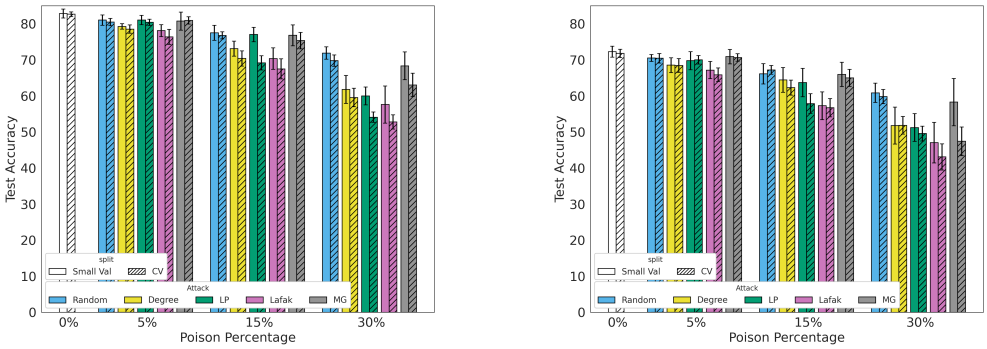
Figure 3: Attack accuracy with and without hyper-parameters (h-params) tuning across datasets. 'w/o hyp' and 'w hp' refers to without and with h-params tuning settings. H-params tuning enables the model to recover accuracy.

3. Hyper-parameters Tuning. While using random splits with small validation sets fixes the aforementioned shortcomings, a crucial component is missing in the evaluation setup. It has been well studied that GNNs are usually very sensitive to hyper-parameters (h-params) (Li & King, 2020). Despite establishing this, all the label-poisoning attacks for GNNs still use default h-params to evaluate the effectiveness of the proposed attack. The default h-params were tuned to work with clean labels (unpoisoned). To study the effect of h-params tuning on the attack efficacy, we select the random splits described in Pitfall 2, and run experiments with and without h-params tuning across models and datasets. We plot our results in Figure 3.

We can infer from Figure 3 that hyper-parameters tuning can significantly degrade the attack performance – seen as increase in test accuracy. The recovery in performance is more pronounced in learning-based attacks and higher poison percentages. At 5% budget and sometimes higher budgets, the claimed state-of-the-art attack, MG, is as ineffective as the RANDOM attack. We observe this anomaly across models and datasets as documented in the Appendix A.3. LAFAK attack seems to be the strongest attack among the suite of existing attacks followed by LPATTACK.

4. Using Clean Validation Set is Not Realistic. It is an unrealistic assumption for an attacker to treat train and val sets as separate entities. In other words, the current evaluation setup in literature for label-poisoning attacks training nodes. The val set is untouched and remains sanitized. This does not simulate the poisoning attack setting truthfully, as the defender can simply ignore the training set during training. In reality, the defender gets a partially-labeled and potentially poisoned graph. Then, they split the labeled nodes (sometimes called development set) into a train and val set, but crucially poisoned nodes have an equal chance to land in either of these sets since the defender does not know which labels are corrupted. To simulate this setting, we fuse the train and val set, and apply the poisoning attack. Next, we create 10 different train and val sets by bifurcating the fused set using stratified sampling to preserve the class distribution maintained before fusing. Finally, the average performance across splits is reported.

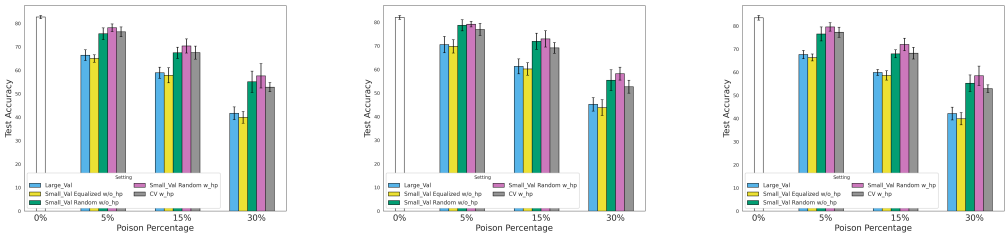
We perform experiments on the proposed evaluation setting (referred to as the CV setting), and plot the results in Figure 4. The CV setting boosts the efficacy of the attacks. This is expected, as the random setting with the small val set (small val) has clean labels compared to poisoned labels in the CV setting.
In Figure 10, we sequentially apply a series of remedies we have so far described and plot the attack performance. Moving from class equalized splits to random splits has the largest impact on average, followed by hyper-parameters tuning. Comparing the first and last bar for a given budget shows that the attack performance on average worsens by $\sim$9% for LAFAK attack across models for the Cora-ML dataset. Additional comparisons are provided in the appendix section A.3. We advocate for using this setting as it reflects a real scenarios better.



(a) Cora-ML

(b) Citeseer

Figure 4: Comparing the effect of proper cross-validation on attack performance across datasets.



(a) GCN

(b) GAT

(c) APPNP

Figure 5: Analyzing the effect of evaluation setting on LAFAK's attack performance. All experiments were conducted on the Cora-ML dataset across three models. The effect of the evaluation setting is consistent across all models.

5. Missing variances. An important statistic that is missing in previous works is variance. We visually depict the variance in all our plots to check for statistical significance across attacks. This measure would help assess the effectiveness of a given attack model. We see that the variance is relatively large, often of the same order as the supposed improvement by an attack.

6: Missing Comparison against Defense Models. (Bonus) While the effect of label poisoning attacks was studied for vanilla or unvaccinated GNN models, a thorough comparison against GNN defense models is missing. We leave this study to future work.

4 RECOMMENDATIONS & DISCUSSION

In this work, we uncover the flaws in the evaluation setup widely used by label-poisoning attacks for GNNs. To ensure a fairer and more realistic evaluation, we make the following recommendations. 1) Random splits with a validation set size similar to the train set should be preferred over class equalized splits. 2) Thorough hyper-parameters tuning must be ensured to simulate a fair defender and gauge the potency of the proposed attack. 3) The proposed CV setting, where training data is bifurcated into train/val sets post-poisoning should be used for comparing attacks. Overall, LAFAK attack is the most effective followed by LPATTACK. The major difference between these attacks is the surrogate model; switching from an LP to an SGC (Wu et al., 2019a) surrogate model significantly improves the attack performance as validated by our experiments. Note that the current attacks perform a transfer attack instead of a direct attack – poisoned labels are generated by attacking an approximate surrogate model and used on a target model. Extrapolating from our observations and the recommendations from (Mujkanovic et al., 2022), we believe adaptive attacks can further boost the attack performance.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *ArXiv*, abs/1907.10902, 2019.

Enyan Dai, Charu Aggarwal, and Suhang Wang. Nrgnn: Learning a label noise-resistant graph neural network on sparsely and noisily labeled graphs. *SIGKDD*, 2021.

Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. Robustness of graph neural networks at scale. In *Neural Information Processing Systems, NeurIPS*, 2021.

Wei Jin, Yaxing Li, Han Xu, Yiqi Wang, Shuiwang Ji, Charu Aggarwal, and Jiliang Tang. Adversarial attacks and defenses on graphs. *SIGKDD Explor. Newsl.*, pp. 19–34, 2021.

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

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

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Combining neural networks with personalized pagerank for classification on graphs. In *International Conference on Learning Representations (ICLR)*, 2019.

Yaoman Li and Irwin King. AutoGraph: Automated graph neural network. In *Neural Information Processing*, pp. 189–201. Springer International Publishing, 2020. doi: 10.1007/978-3-030-63833-7_16. URL https://doi.org/10.1007%2F978-3-030-63833-7_16.

Ganlin Liu, Xiaowei Huang, and Xinping Yi. 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.

Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. *A Unified Framework for Data Poisoning Attack to Graph-Based Semi-Supervised Learning*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. Towards more practical adversarial attacks on graph neural networks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 2000.

Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. Are defenses for graph neural networks robust? In *Neural Information Processing Systems, NeurIPS*, 2022.

Galileo Mark Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven active surveying for collective classification. In *Workshop on Mining and Learning with Graphs*, 2012.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *ArXiv*, abs/1811.05868, 2018.

Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, Yixin Liu, Philip S. Yu, Lifang He, and Bo Li. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. *ArXiv*, abs/1902.07153, 2019a.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019b.

Jiacheng Xiong, Zhaoping Xiong, Kaixian Chen, Hualiang Jiang, and Mingyue Zheng. Graph neural networks for automated de novo drug design. *Drug Discovery Today*, 26(6):1382–1393, 2021. ISSN 1359-6446. doi: https://doi.org/10.1016/j.drudis.2021.02.011. URL https://www.sciencedirect.com/science/article/pii/S1359644621000787.

Xingrui Yu, Bo Han, Jiangchao Yao, Gang Niu, Ivor Tsang, and Masashi Sugiyama. How does disagreement help generalization against label corruption? In *International Conference on Machine Learning*, pp. 7164–7173, 2019.

Mengmei Zhang, Linmei Hu, Chuan Shi, and Xiao Wang. 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

The appendix is structured as follows. In Section A.1, we provide comprehensive details on the hyper-parameters ranges and other experiments details. In Section A.2, we provide dataset statistics of different settings. In Section A.3, we describe details of additional experiments to bolster our observations made in the main paper.

A.1 REPRODUCIBILITY

In all our experiments without hyper-parameter tuning, we use the hyper-parameters reported by the authors of the models. For optimization, we use the Adam optimizer (Kingma & Ba, 2015). We set the maximum epochs to 2000, and employ early stopping with patience of 200. For remaining experiments with hyper-parameters tuning, we use the Optuna framework (Akiba et al., 2019) and set the number of trials to 30 to optimize the hyper-parameters search. The learning rate is swept over [0.01, 0.05, 0.08, 0.1], weight decay over [0.0, 0.005, 0.0005, 0.00005], dropout over [0.3, 0.5, 0.7]. For APPNP model, we additionally tune the alpha over the range [0.1, 0.3, 0.5, 0.8]. The hidden dimensions in all experiments is set to 64. For all the datasets, we use undirected graphs with self-loops and perform symmetric normalization. We report test performance along with standard deviation corresponding to the best validation accuracy in our experiments.

Our experiments were performed on a machine with an AMD EPYC 7F32 3.7Ghz processor, 1TB ram, NVIDIA A100 GPU with 40GB of memory, Python 3.8.12, and PyTorch 1.11.0 (Paszke et al., 2019).

A.2 DATASET STATISTICS FOR DEFAULT SETTING

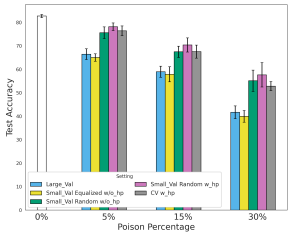
In Table 1, we tabulate dataset statistics. We additionally include the train/val/test split statistics for the default and the proposed CV setting. Note that in the CV setting, the test accuracy is measured over all the remaining unlabeled nodes, and the train and val set have the same size.

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

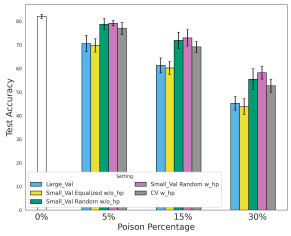
Table 1: Dataset statistics

A.3 ADDITIONAL EXPERIMENTS

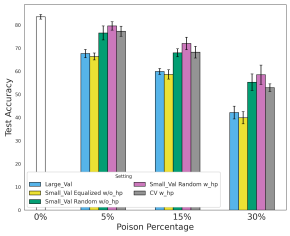
In this section, we extend the experiments we performed in the main paper to more datasets and models. Most of our observations are inline with those reported in the main paper. For the Citeseer and Pubmed dataset, for some budgets, we observe our proposed CV splits to be favorable to the attacker compared to the default setting for the LAFAK attack. An opposite behavior was seen for the Cora-ML dataset.



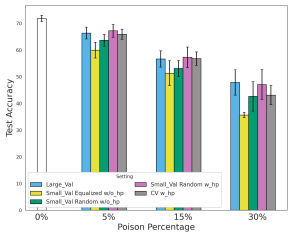
(a) GCN - Cora-ML



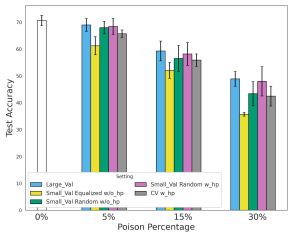
(b) GAT - Cora-ML



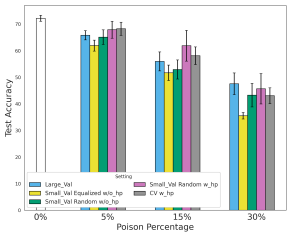
(c) APPNP - Cora-ML



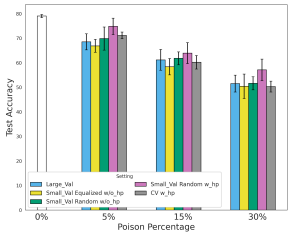
(d) GCN - CiteSeer



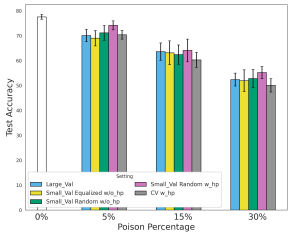
(e) GAT - CiteSeer



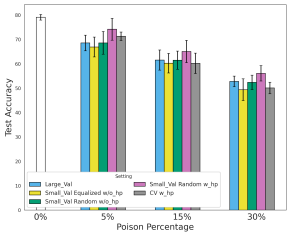
(f) APPNP - CiteSeer



(g) GCN - Pubmed

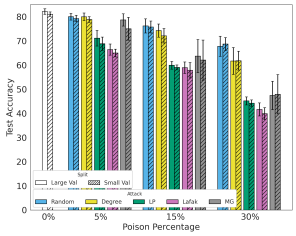


(h) GAT - Pubmed

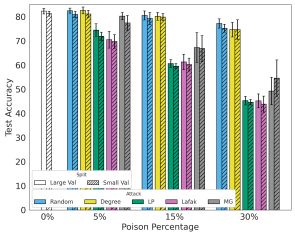


(i) APPNP - Pubmed

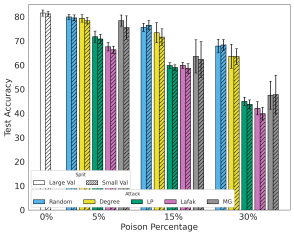
Figure 6: Analyzing the effect of each evaluation setting on the performance of the LafAK attack across three models and three datasets.



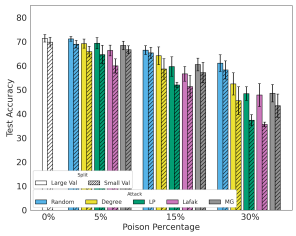
(a) GCN - Cora-ML



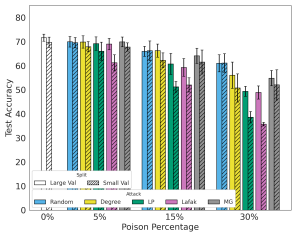
(b) GAT - Cora-ML



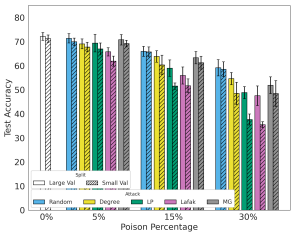
(c) APPNP - Cora-ML



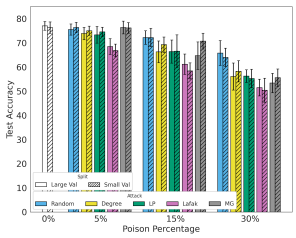
(d) GCN - CiteSeer



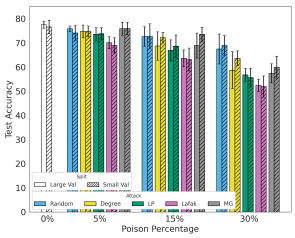
(e) GAT - CiteSeer



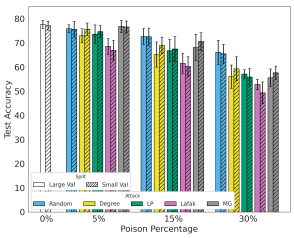
(f) APPNP - CiteSeer



(g) GCN - Pubmed

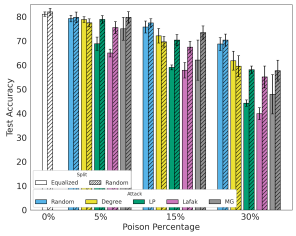


(h) GAT - Pubmed

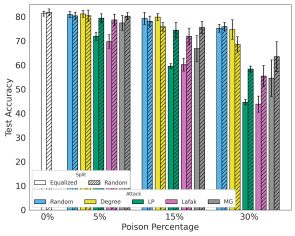


(i) APPNP - Pubmed

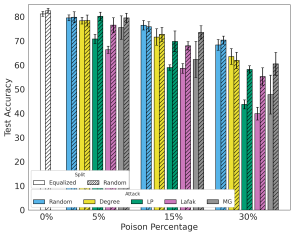
Figure 7: Effect of validation set size on the performance of poisoning attacks across three models and three datasets.



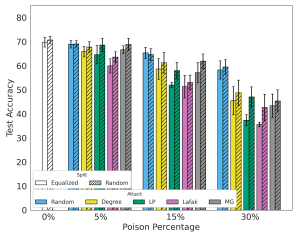
(a) GCN - Cora-ML



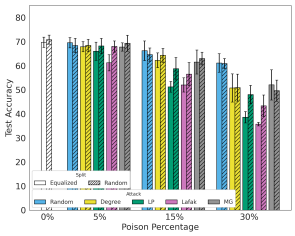
(b) GAT - Cora-ML



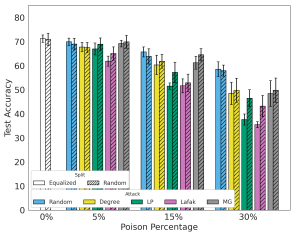
(c) APPNP - Cora-ML



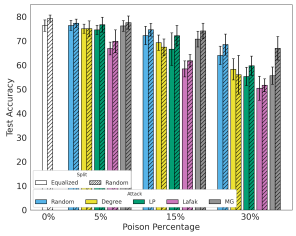
(d) GCN - CiteSeer



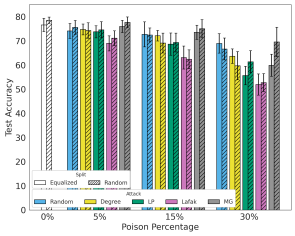
(e) GAT - CiteSeer



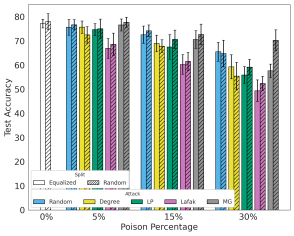
(f) APPNP - CiteSeer



(g) GCN - Pubmed

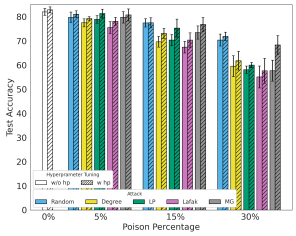


(h) GAT - Pubmed

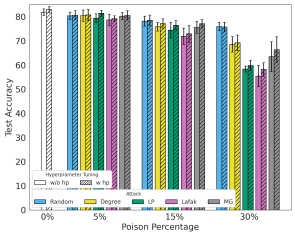


(i) APPNP - Pubmed

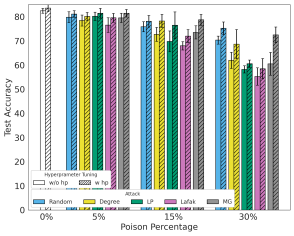
Figure 8: Effect of data split on the performance of poisoning attacks across three models and three datasets.



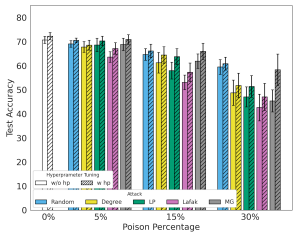
(a) GCN - Cora-ML



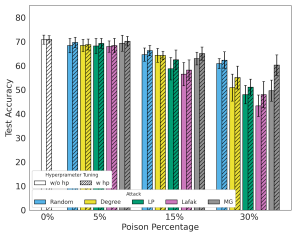
(b) GAT - Cora-ML



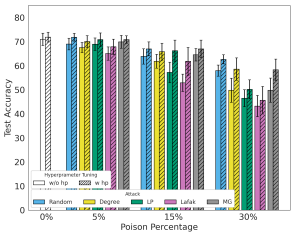
(c) APPNP - Cora-ML



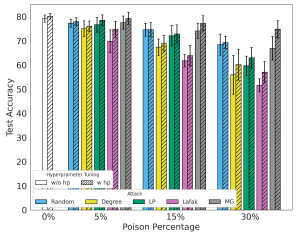
(d) GCN - CiteSeer



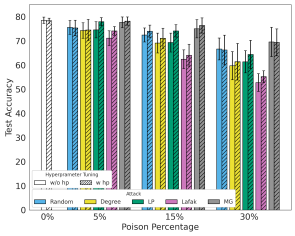
(e) GAT - CiteSeer



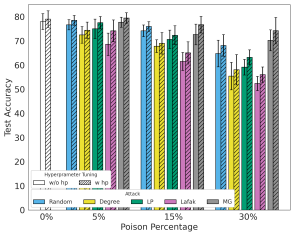
(f) APPNP - CiteSeer



(g) GCN - Pubmed

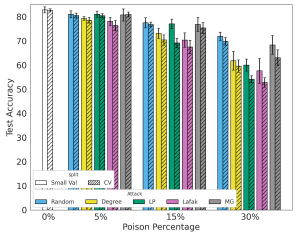


(h) GAT - Pubmed

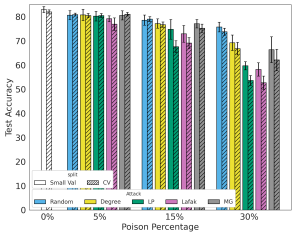


(i) APPNP - Pubmed

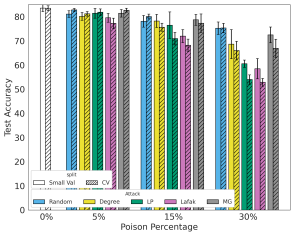
Figure 9: Analyzing the effect of proper hyper-parameters tuning on the performance of poisoning attacks across three models and three datasets.



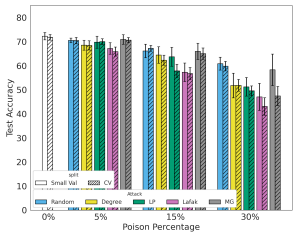
(a) GCN - Cora-ML



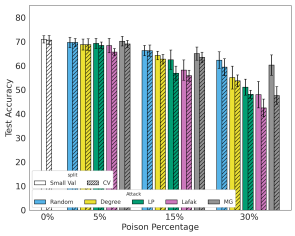
(b) GAT - Cora-ML



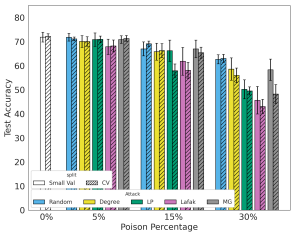
(c) APPNP - Cora-ML



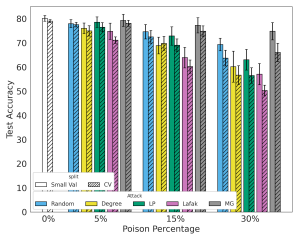
(d) GCN - CiteSeer



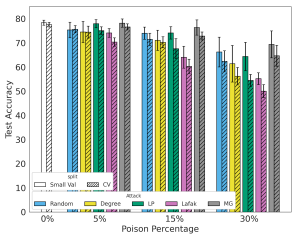
(e) GAT - CiteSeer



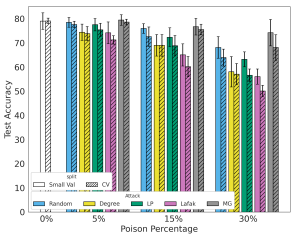
(f) APPNP - CiteSeer



(g) GCN - Pubmed



(h) GAT - Pubmed



(i) APPNP - Pubmed

Figure 10: Analyzing the effect of proper cross-validation on the evaluated performance of poisoning attacks across three models and three datasets.