Symbiotic Adversarial Robustness for Graph Neural Networks: Combining Poisoning and Evasion

Abstract

Deep learning models are known to be vulnerable to small input perturbations, which are known as adversarial examples. Adversarial examples are commonly crafted to deceive a model either at training (poisoning) or testing (evasion). We study the combination of poisoning and evasion attacks. We show that using both threat models can significantly improve the damaging effect of adversarial attacks. Specifically, we study the robustness of Graph Neural Networks (GNNs) under structural perturbations and develop a memory-efficient adaptive end-to-end attack for this novel threat model using first-order optimization.

1 Introduction

Graph neural networks (GNNs) are increasingly used across many different fields, including product recommendations and drug discovery. GNNs are, however, vulnerable to adversarial attacks in many different tasks such as node classification, graph classification, link prediction and node embeddings. Given that such attacks are able to scale to very large graphs, studying the adversarial robustness of GNNs has become increasingly important. GNNs can be attacked at test time (evasion) or during training (poisoning). However, a combined threat model that includes both evasion and poisoning has not been considered in prior literature. Such a model, is, nonetheless, plausible given the public availability of graphs or those extracted from sources such as social media sites.

Our work is based on the concept of a symbiotic attack, which combines both evasion and poisoning attacks. A symbiotic attack aims to minimize classification accuracy on a test set. The attacker is constrained by a global budget and manipulates the entire graph, rather than individual nodes. We provide a comparison of our approach against plain poisoning and evasion attacks. To this end, we adapt the previous PR-BCD attack to the symbiotic threat model, which results in attacks that are memory-efficient and scalable to large graphs. Our main findings are that symbiotic attacks are more effective than poisoning attacks alone, and that evasion attacks are affected by the size of the test set, while symbiotic attacks are less sensitive to test set size. The potential improvement given by the symbiotic threat model indicates that it requires further study.

2 Preliminaries

Notation. We denote a graph by G, with n nodes, an adjacency matrix $A \in \{0,1\}^{n \times n}$, and a feature matrix $X \in \mathbb{R}^{n \times d}$. A GNN applied to the graph is represented by $f_\theta(G)$ with parameters θ. We denote the set of possible adversarial graphs that can be created from G as $\Phi(G)$. Also, L_{atk} and L_{train} denote the adversarial and training objectives.

2.1 Adversarial Robustness of GNNs

An adversarial attack on a GNN can modify the graph's structure, by inserting or removing edges and nodes, or modify the node features. This work focuses on node classification and edge-level structural perturbations.

.

Attacks can be categorized as either evasion or poisoning. In an evasion attack, a fixed GNN (with parameters θ trained on a clean graph) is targeted, and the attacker aims to solve the optimization problem

$$\max_{\hat{G} \in \Phi(G)} L_{atk}(f_\theta(\hat{G})),$$

whereas a poisoning attack is performed before training, aiming to degrade the performance of the GNN after training. This can be described as

$$\max_{\hat{G} \in \Phi(G)} L_{atk}(f_{\theta^*}(\hat{G})), \text{ where } \theta^* = \text{argmin}_\theta L_{train}(f_\theta(\hat{G})).$$

A poisoning attack is generally more challenging. Previous work has investigated using evasion perturbations as poisoning perturbations. Also, the optimization may include unrolling the training procedure to calculate meta-gradients (gradients of L_{atk} with respect to A).

Since we consider only changes to the binary adjacency matrix, we define $\Phi(G)$ to include graphs reachable from G after at most Δ edge perturbations.

2.1.1 PR-BCD

Our work extends on the Projected Randomized Block Coordinate Descent (PR-BCD) attack. Similarly to the Projected Gradient Descent (PGD) attack, the adjacency matrix is relaxed to $P \in [0,1]^{n \times n}$, enabling continuous gradient updates. Each entry indicates the probability of flipping an edge, with the final perturbations sampled from Bernoulli(P). However, as the adjacency matrix grows quadratically with the number of nodes, scaling of the PGD becomes difficult with larger graphs.

PR-BCD uses Randomized Block Coordinate Descent (R-BCD), updating a block of P at each iteration. The projection step ensures the budget is enforced in expectation, i.e. $E[\text{Bernoulli}(P)] = \sum P < \Delta$ and $P \in [0,1]^{n \times n}$. After each iteration, rather than sampling the block again, the promising entries of the block are kept, and only the remaining entries are resampled.

PGD can also be applied for a poisoning attack (Meta-PGD). In our attacks, we employ the same principle with PR-BCD for better scalability. While we only consider a single global budget Δ, it is possible to include more complex constraints when needed for a given application.

3 Symbiotic Attacks

The Symbiotic Objective. A symbiotic attack has a similar form to the bi-level optimization problem but has an added dependence on the evasion graph G^* in addition to the parameters θ^*:

$$\max_{\hat{G} \in \Phi(G)} L_{pois}(f_{\theta^*}(G^*)) \text{ where } \theta^* = \text{argmin}_\theta L_{train}(f_\theta(\hat{G})), \text{ and } G^* = \text{argmax}_{\hat{G} \in \Phi(\hat{G})} L_{ev}(f_{\theta^*}(\hat{G}))$$

Here, L_{pois} and L_{ev} are separated for clarity even though they could be the same loss.

Threat Model. We model an attacker who aims to reduce a model's performance on node classification tasks. Our attacker has full access to the graph, has knowledge of the model's architecture, can create surrogate models, and can only access the trained model as a black-box. Finally, our attacker has a limited global budget of edge insertions/removals.

The Sequential Attack. A simple way to launch a symbiotic attack is to divide the budget and launch a poisoning attack with the first half, followed by an evasion attack with the second half. In this attack, the poisoning step is not aware of a future evasion, but can improve performance by reducing the classification margin of certain nodes.

The Joint Attack. The poisoning attack can be designed to "fit" the future evasion graph by including the evasion attack in the poisoning loss. The poisoning loss is computed using the poisoned model over the evasion graph. This results in a poisoning attack which not only reduces the model's accuracy, but also makes it more vulnerable to evasion.

Both the sequential and joint attacks can be instantiated using different evasion/poisoning attacks. We build upon PR-BCD because it scales well to larger graphs. Note that the sequential attack is actually a special case of the joint attack, with zero iterations per inner evasion attack.

4 Evaluation

4.1 Setup

We compare the symbiotic threat model with evasion and poisoning attacks, using PR-BCD to implement the evasion and poisoning attacks. These are evaluated on Cora, CiteSeer, and PubMed datasets. We study the robustness of GCN, GAT, APPNP, and GPRGNN models. We also consider R-GCN and Jaccard purification as potential defense mechanisms. For each dataset, we allocate 20 nodes of each class for the labeled training set and 10

Table 1: Numbers of nodes, edges, and classes in the datasets we include in our evaluations.

Dataset	Nodes	Edges	Classes
Cora	2,708	10,556	7
CiteSeer	3,327	9,104	6
PubMed	19,717	88,648	3

4.2 Results

Table 2 displays the perturbed accuracy values on the test set (10 percent of nodes) for our benchmark datasets and models, averaged over 10 runs, with the standard error of the mean also shown. The attacker is given a 5 percent budget of the number of edges, and this budget is split equally between poisoning and evasion for the symbiotic attacks. We report the best performing of the two symbiotic attacks, and also note that the symbiotic attacks are consistently stronger than the poisoning attacks, and stronger than plain evasion. The symbiotic threat model is especially evident on the larger PubMed graph, where the accuracy drops to almost zero, for example, using a GCN.

4.3 Effect of the Number of Test Nodes

To highlight the differences between poisoning and evasion objectives, Figure 2 shows the perturbed accuracies for evasion, poisoning, and symbiotic attacks with varying fractions of test nodes with a GCN and a 5

As the number of test nodes increases, evasion becomes much more challenging across all datasets. Although poisoning and symbiotic attacks also become more difficult with more test nodes, especially on PubMed, they are more robust than the evasion attack. Therefore, the reduction in performance cannot be explained by the attacks having to target a larger number of nodes with the same budget. The poisoning attack is less affected since it can manipulate the flow of information during training. The symbiotic attacks also benefit from this since they can reduce the base accuracy, making nodes easier to misclassify during the evasion phase. The symbiotic attacks are also stronger than poisoning alone.

4.4 Hyperparameters

Block size. Figure 3 shows the results of the four attacks with varying block sizes, using a fixed 5 percent budget and 125 iterations against a GCN. For small block sizes, the attacks are less effective since the PR-BCD optimization can only cover a small part of the adjacency matrix. However, larger blocks have decreasing marginal benefit when a large part of the adjacency matrix can be covered.

Budget. Figure 4 shows how all four attacks follow a similar trend when increasing budget size. On PubMed, changing 5 percent of edges is enough to achieve near-zero accuracy under the symbiotic model. This highlights the devastating effect of joint attacks, especially in larger graphs with a small number of labeled train nodes.

5 Conclusion and Future Work

In this work, we have introduced the symbiotic threat model for GNNs, which combines evasion and poisoning attacks. We proposed two methods to generate adversarial perturbations for this model and

Table 2: Average ($\pm$ standard error) perturbed accuracies for the evasion, poisoning, and symbiotic attacks with a 5 percent budget. The -J suffix indicates the graph has been pre-processed with Jaccard purification. (ind.) stands for inductive learning. The strongest (lowest accuracy) results for each setup are written in bold.

Model	Dataset	Clean	Evasion	Poisoning	Symbiotic
GCN	CiteSeer	0.68 ± 0.01	0.41 ± 0.01	0.4 ± 0.01	$\mathbf{0.38 \pm 0.01}$
	CiteSeer (ind.)	0.67 ± 0.01	0.41 ± 0.01	0.62 ± 0.01	$\mathbf{0.33 \pm 0.01}$
	CiteSeer-J	0.68 ± 0.01	0.41 ± 0.01	0.41 ± 0.02	$\mathbf{0.38 \pm 0.01}$
	Cora	0.78 ± 0.01	0.41 ± 0.01	0.46 ± 0.02	$\mathbf{0.35 \pm 0.01}$
	Cora (ind.)	0.75 ± 0.02	0.42 ± 0.01	0.68 ± 0.03	$\mathbf{0.3 \pm 0.01}$
	Cora-J	0.74 ± 0.01	0.39 ± 0.01	0.43 ± 0.02	$\mathbf{0.36 \pm 0.01}$
	PubMed	0.78 ± 0.01	0.41 ± 0.01	0.12 ± 0.02	$\mathbf{0.03 \pm 0.01}$
	PubMed-J	0.77 ± 0.01	0.41 ± 0.01	0.11 ± 0.01	$\mathbf{0.02 \pm 0.0}$
GAT	CiteSeer	0.62 ± 0.02	0.27 ± 0.02	0.41 ± 0.02	$\mathbf{0.3 \pm 0.03}$
	CiteSeer (ind.)	0.68 ± 0.01	0.37 ± 0.01	0.64 ± 0.02	$\mathbf{0.56 \pm 0.02}$
	CiteSeer-J	0.64 ± 0.01	0.32 ± 0.03	0.41 ± 0.03	$\mathbf{0.3 \pm 0.03}$
	Cora	0.69 ± 0.02	0.22 ± 0.02	0.48 ± 0.03	$\mathbf{0.29 \pm 0.02}$
	Cora (ind.)	0.77 ± 0.01	0.21 ± 0.01	0.61 ± 0.04	$\mathbf{0.35 \pm 0.03}$
	Cora-J	0.67 ± 0.01	0.23 ± 0.02	0.45 ± 0.02	$\mathbf{0.28 \pm 0.02}$
	PubMed	0.73 ± 0.01	0.38 ± 0.04	0.41 ± 0.01	$\mathbf{0.2 \pm 0.03}$
	PubMed-J	0.74 ± 0.01	0.34 ± 0.04	0.38 ± 0.04	$\mathbf{0.19 \pm 0.02}$
APPNP	CiteSeer	0.69 ± 0.01	0.45 ± 0.01	0.56 ± 0.01	$\mathbf{0.47 \pm 0.01}$
	CiteSeer (ind.)	0.71 ± 0.01	0.47 ± 0.01	0.66 ± 0.02	$\mathbf{0.4 \pm 0.01}$
	CiteSeer-J	0.68 ± 0.01	0.43 ± 0.01	0.52 ± 0.02	$\mathbf{0.45 \pm 0.02}$
	Cora	0.82 ± 0.02	0.48 ± 0.03	0.64 ± 0.02	$\mathbf{0.51 \pm 0.04}$
	Cora (ind.)	0.82 ± 0.02	0.53 ± 0.02	0.78 ± 0.01	$\mathbf{0.37 \pm 0.01}$
	Cora-J	0.82 ± 0.01	0.5 ± 0.01	0.67 ± 0.01	$\mathbf{0.54 \pm 0.01}$
	PubMed	0.79 ± 0.0	0.46 ± 0.01	0.21 ± 0.02	$\mathbf{0.09 \pm 0.01}$
	PubMed-J	0.77 ± 0.01	0.45 ± 0.01	0.19 ± 0.03	$\mathbf{0.1 \pm 0.02}$
GPRGNN	CiteSeer	0.66 ± 0.01	0.34 ± 0.01	0.44 ± 0.02	$\mathbf{0.33 \pm 0.01}$
	CiteSeer (ind.)	0.67 ± 0.01	0.37 ± 0.01	0.56 ± 0.01	$\mathbf{0.34 \pm 0.01}$
	CiteSeer-J	0.65 ± 0.01	0.35 ± 0.01	0.44 ± 0.01	$\mathbf{0.35 \pm 0.01}$
	Cora	0.82 ± 0.01	0.46 ± 0.01	0.53 ± 0.01	$\mathbf{0.4 \pm 0.01}$
	Cora (ind.)	0.8 ± 0.02	0.44 ± 0.01	0.74 ± 0.01	$\mathbf{0.35 \pm 0.01}$
	Cora-J	0.79 ± 0.01	0.44 ± 0.01	0.54 ± 0.01	$\mathbf{0.4 \pm 0.01}$
	PubMed	0.78 ± 0.01	0.42 ± 0.01	0.28 ± 0.03	$\mathbf{0.08 \pm 0.02}$
	PubMed-J	0.78 ± 0.01	0.42 ± 0.01	0.38 ± 0.04	$\mathbf{0.15 \pm 0.04}$
RGCN	CiteSeer	0.63 ± 0.01	0.39 ± 0.01	0.59 ± 0.02	$\mathbf{0.47 \pm 0.01}$
	Cora	0.74 ± 0.02	0.44 ± 0.01	0.74 ± 0.01	$\mathbf{0.52 \pm 0.02}$
	PubMed	0.77 ± 0.01	0.43 ± 0.01	0.42 ± 0.04	$\mathbf{0.15 \pm 0.03}$

showed that symbiotic attacks can be more effective than the evasion or poisoning approaches on their own. We will outline several avenues for future work.

The joint attack can be implemented using other evasion attacks, or attacks designed for the symbiotic threat model. In addition, our work considered global budgets, but it is easy to consider per-node local budgets and targeted attacks as well. Moreover, we did not consider the use of different loss functions for the poisoning and evasion parts, which may also further improve attack performance. We plan to include further evaluations on these settings as our next step. Finally, novel poisoning attacks can be developed which utilize knowledge of a future evasion attack.

A Proof of Theorem 2.1

Proof. Let $x \in A_i$. Then, $\sigma_i(x) = 0$, and for all $b \in O$ where $b_i = 0$, $w_b(x) = 0$. Thus,

$$F(x) = \sum_{b \in O, b_i = 1} w_b(x) G_b(x)$$

If $b_i = 1$, then $G_b(x) \in B_i$, and therefore $F(x)$ is also in B_i due to the convexity of B_i.

B Sub-Gaussian Covering Numbers for ReLU Networks

Figure 2 depicts an example of applying our safe predictor to a notional regression problem. This example uses inputs and outputs in 1-D with one input-output constraint. The unconstrained network consists of a single hidden layer with a dimension of 10, ReLU activations, and a fully connected layer. The safe predictor shares this structure with constrained predictors, G_0 and G_1, but each predictor has its own fully connected layer. The training uses a sampled subset of points from the input space. Figure 3 shows an example of applying the safe predictor to a notional regression problem with a 2-D input and 1-D output and two overlapping constraints. The unconstrained network has two hidden layers of dimension 20 with ReLU activations, followed by a fully connected layer. The constrained predictors, G_{00}, G_{10}, G_{01}, and G_{11}, share the hidden layers and have an additional hidden layer of size 20 with ReLU followed by a fully connected layer. Again, training uses a sampled subset of points from the input space and the learned predictors are shown for the continuous input space.

C Details of VerticalCAS Experiment

C.1 Safeability Constraints

The "safeability" property from prior work can be encoded into a set of input-output constraints. The "safeable region" for a given advisory is the set of input space locations where that advisory can be selected such that future advisories exist that will prevent an NMAC. If no future advisories exist, the advisory is "unsafeable" and the corresponding input region is the "unsafeable region". Examples of these regions, and their proximity functions are shown in Figure 5 for the CL1500 advisory.

The constraints we enforce in our safe predictor are: $x \in A_{\text{unsafeable},i} \Rightarrow F_i(x) < \max_j F_j(x)$, $\forall i$. To make the output regions convex, we approximate by enforcing $F_i(x) = \min_j F_j(x) - \epsilon$, for all $x \in A_{\text{unsafeable},i}$.

C.2 Proximity Functions

We start by generating the unsafeable region bounds. Then, a distance function is computed between points in the input space ($v_O - v_I$, h, τ), and the unsafeable region for each advisory. These are not true distances but are 0 if and only if the data point is within the unsafeable set. These are then used to produce proximity functions. Figure 5 shows examples of the unsafeable region, distance function, and proximity function for the CL1500 advisory.

C.3 Structure of Predictors

The compressed policy tables for ACAS Xu and VerticalCAS use neural networks with six hidden layers with a dimension of 45, and ReLU activation functions. We used the same architecture for the unconstrained network. For constrained predictors, we use a similar architecture, but share the first four layers for all predictors. This provides a common learned representation of the input space, while allowing each predictor to adapt to its constraints. Each constrained predictor has two additional hidden layers and their outputs are projected onto our convex approximation of the safe output region, using $G_b(x) = \min_j G_j(x) - \epsilon$. In our experiments, we used $\epsilon = 0.0001$.

With this construction, we needed 30 separate predictors to enforce the VerticalCAS safeability constraints. The number of nodes for the unconstrained and safe implementations were 270 and 2880, respectively. Our safe predictor is smaller than the original look-up tables by several orders of magnitude.

C.4 Parameter Optimization

We use PyTorch for defining our networks and performing parameter optimization. We optimize both the unconstrained network and our safe predictor using the asymmetric loss function, guiding the network to select optimal advisories while accurately predicting scores from the look-up tables. Each dataset is split using an 80/20 train/test split with a random seed of 0. The optimizer is ADAM, with a learning rate of 0.0003, a batch size of 216, and training for 500 epochs.