Graph Neural Network Explanations are Fragile

Jiate Li [1 2] Meng Pang [1] Yun Dong [3] Jinyuan Jia [4] Binghui Wang [2 5]

Abstract

Explainable Graph Neural Network (GNN) has emerged recently to foster the trust of using GNNs. Existing GNN explainers are developed from various perspectives to enhance the explanation performance. We take the first step to study GNN explainers under adversarial attack—We found that an adversary slightly perturbing graph structure can ensure GNN model makes correct predictions, but the GNN explainer yields a drastically different explanation on the perturbed graph. Specifically, we first formulate the attack problem under a practical threat model (i.e., the adversary has limited knowledge about the GNN explainer and a restricted perturbation budget). We then design two methods (i.e., one is loss-based and the other is deduction-based) to realize the attack. We evaluate our attacks on various GNN explainers and the results show these explainers are fragile.[1]

1. Introduction

Graph Neural Network (GNN) (Scarselli et al., 2008; Kipf & Welling, 2017; Hamilton et al., 2017; Velickovic et al., 2018; Xu et al., 2019) is the mainstream paradigm for learning graph data: it takes a graph as input and learns node or graph representations to capture the relation among nodes or graph structural information. GNNs have achieved state-of-the-art performance in graph-related tasks such as node/graph classification and link prediction (Wu et al., 2020).

Explainable GNN has emerged in recent years and has wide applications including molecular property prediction (Wu et al., 2023), disease diagnosis (Pfeifer et al., 2022), drug analysis (Yang et al., 2022), and fake news spreader prediction (Rath et al., 2021). Concretely, given a graph and a predicted (node/graph) label by a GNN model, explainable GNNs aim to determine the subgraph (include edges and the connected nodes) from the graph that ensures the best predictability about the label (see Figure 1 an example). This subgraph is also called *explanatory subgraph*. To achieve the goal, various GNN explainers from different perspectives (Ying et al., 2019; Luo et al., 2020; Yuan et al., 2021; Zhang et al., 2022; Wang & Shen, 2023) have been proposed. Among them, perturbation-based explainers (Ying et al., 2019; Luo et al., 2020; Schlichtkrull et al., 2021; Funke et al., 2022; Wang et al., 2021b; Yuan et al., 2021; Zhang et al., 2022) are widely studied and show promising explanation performance (more details are in Section 2).

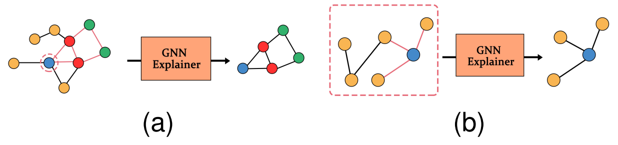


Figure 1. GNN explanation for (a) node classification and (b) graph classification—It identifies the subgraph that ensures the best prediction for the target node and target graph, respectively.

In this paper, instead of enhancing the explanation performance, we focus on understanding the robustness of GNN explainers when facing adversaries. Particularly, we study the research problem: *Given a graph, a well-trained GNN model, and a GNN explainer, can an adversary manipulate the graph structure such that the GNN predictions are still accurate, but the explanation result of the GNN explainer is drastically changed?*[2] We emphasize this problem has serious security implications in real applications. Lets take GNN-based malicious user detection in social networks as an instance. Assume a Facebook user is predicted as malicious by a GNN. When applying a GNN explainer, a reasonable explanation would be that the user has connected with other users that also exhibit malicious activities. Now assume an adversary has carefully added connections between the user and certain normal users. If the user is still predicted as malicious, the explainer may wrongly interpret this is because the user has connections with those normal users, and suggest flagging them as malicious users.

[1]Nanchang University, China [2]Illinois Institute of Technology, USA [3]Milwaukee School of Engineering, USA [4]The Pennsylvania State University, USA [5]Jiate did this research as an intern in Wang's lab. Correspondence to: Meng Pang <mengpang@ncu.edu.cn>, Binghui Wang <bwang70@iit.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1]Code is at: https://github.com/JetRichardLee/Attack-XGNN

[2]Many existing studies propose to attack GNNs for the classification purpose. Their goal is different from ours and more detailed discussions about the differences are shown in Appendix E.

We take the first step on attacking the most widely-studied perturbation-based GNN explainers with graph structure perturbations (i.e., add new edges to or/and remove existing edges from the graph). We first characterize the threat model of the attack under three aspects: *1) Attack goal*—ensure the difference between the explanation result with our attack and that without attack be as large as possible; *2) Attack knowledge*—know the explanation loss (since we want to attack the specific explainer), while unknown to the internal model and training details of the explainer; *3) Attack constraint*—have limited perturbation budget, maintain the graph structure information, and ensure correct predictions (like no attacks). We then formalize our attack based on the threat model. However, the attack problem is NP-hard and thus challenging to be solved. To address it, we propose to relax the problem and find approximate solutions, leading to our two attack design—one is loss-based and the other is deduction-based. Specifically, our two attacks are inspired by an observation: a feasible attack should be able to efficiently score edges based on their importance and then identify the most important (added and deleted) edges to perturb the graph structure.

Loss-based attack: Perturbation-based GNN explainers are aiming to minimize an explanation loss (see Equation 1). This attack then uses the loss change induced by modifying an edge to indicate the edge importance. Particularly, an existing edge largely increasing the loss when it is deleted, or a new edge largely increasing the loss when it is added, are deemed important. This attack then designs a new loss to capture this property. When optimized, it can uncover important edges to be perturbed.

Deduction-based attack: A drawback of the loss-based attack is it lacks a direct connection to the formulated attack objective. Our deduction-based attack aims to mitigate this drawback. Particularly, the key idea is to simulate the dynamic learning process of the perturbation-based GNN explainers. Then the attack objective can be rewritten in the form of a loss that has a close relationship with the loss used by the GNN explainer. It then optimizes this new loss and identifies important edges to be perturbed.

We systematically evaluate our attacks on multiple graph datasets, GNN tasks, and perturbation-based GNN explainers. Our experimental results show that existing GNN explainers are fragile. For instance, when perturbing only 2 edges, the explanatory edges can be 70% different from those without the attack. Our results also show the generated perturbations transfer well to attacking other types of GNN explainers, thus demonstrating the generality. For instance, only 2 perturbed edges generated by our attacks ensure 50% explanatory edges in a state-of-the-art surrogate-based GNN explainer be changed. Our key contributions are as below:

- To our best knowledge, this is the first work to comprehensively understand the robustness of perturbation-based GNN explainers against graph perturbations.
- We design two attacks on perturbation-based GNN explainers. Our attacks are practical (i.e., limited knowledge on the GNN explainer), stealthy (i.e., small perturbation and maintain graph structure) , and faithful (i.e., correct GNN predictions).
- We evaluate our attacks on multiple graph datasets, GNN tasks, and diverse types of GNN explainers.

2. Related Work

Explainable Graph Neural Networks: Generally, GNN explainers can be classified as *instance-level based* and *model-level based*. Instance-level based methods provide input instance-dependent explanation and identify the important part of a graph (e.g., edges, nodes, and features) for predicting the input instance. In contrast, model-level based methods (e.g., XGNN (Yuan et al., 2020) and GNNInterpreter (Wang & Shen, 2023)) do not consider specific input instances, but generate graph patterns to explain a class of instances. In the paper, we focus on instance-level explainers as they are more widely studied.

Instance-level GNN explainers can be further classified into five categories: *decomposition-based*, *gradient-based*, *surrogate-based*, *generation-based*, and *perturbation-based*. Details of these categories are seen in Appendix B. We primarily focus on perturbation-based methods, as their explanation results are more accurate.

Attacking Neural Network Explanations: These studies (Ghorbani et al., 2019; Dombrowski et al., 2019; Slack et al., 2021; Heo et al., 2019) have explored the robustness of explanations, primarily on image models, against adversarial attacks. Particularly, their goal is introducing subtle perturbations to input images that do not alter image predictions, but can drastically change the explanations. As we will see, attacking explainable GNN methods is much more challenging in that the attack has more constraints and the attack problem is essentially NP-hard.

Attacking Graph Neural Networks: Existing attacks on GNNs all focus on misleading classification models and they can be categorized as *test-time evasion attacks* (Dai et al., 2018; Zügner et al., 2018; Xu et al., 2019; Wu et al., 2019; Ma et al., 2020; Mu et al., 2021; Wang et al., 2022a; 2023a; 2024) and *training-time poisoning attacks* (Xu et al., 2019; Zügner & Günnemann, 2019; Wang & Gong, 2019; Wang et al., 2023a). Take GNNs for node classification as an instance. In evasion attacks, given a trained GNN model and a (clean) graph, an attacker carefully perturbs the graph structure such that the GNN model misclassifies as many testing nodes as possible in the perturbed graph. In poisoning attacks, given a GNN algorithm and a graph, an attacker carefully perturbs the graph structure in the training phase, such that the learnt GNN model misclassifies as many testing nodes as possible in the testing phase.

A closely relevant work to ours is the GEAttack (Fan et al., 2023). However, the threat model and attacker goal are different from ours. First, GEAttack has white-box access to the GNN explainer; Second, GEAttack also aims to alter GNN *predictions*—it adds new edges to a graph such that the GNN classifier produces *wrong* predictions for nodes in the perturbed graph and the added edges are *within* the explanatory subgraph outputted by the GNN explainer.

3. Background

3.1. Perturbation-based GNN Explainers

Given a graph $G = (V, E)$ with node set V, edge set E, a label y (on a node $v \in V$ or the entire graph G); and a well-trained GNN model f with an accurate prediction, i.e., $f(G) = y$. The objective of GNN explainers is identifying a subgraph[3] $G_S = (V_S, E_S) \subset G$ that ensure f's best predictability for the label y, i.e., $\max_{G_S} Pr(f(G_S) = y)$. Note that when the important edges E_S are determined, the connected nodes V_S are determined accordingly. For simplicity, many GNN explainers hence focus on identifying E_S which is called the *explanatory edges*.

In this paper, we consider the widely-studied perturbation-based GNN explainers and briefly review it as below:

1. An edge mask $M \in [0, 1]^{|E|}$ is defined and initialized deterministically or stochastically. This induces a masked graph G_M with the masked edges defined as $E \otimes M$, where $\otimes$ means the element-wise product. A mask value M_e indicates the important score of an edge $e \in E$. For instance, $M_e = 1$ means e is extremely important, while $M_e = 0$ means e is extremely unimportant for prediction.

2. A GNN explainer-dependent objective function $\mathcal{L}$:[4]

$$\mathcal{L}(M) = l(f(G_M), y) + \mathcal{C}(M), \tag{1}$$

where l denotes the prediction loss with respect to the masked graph G_M; and $\mathcal{C}$ is a constraint function on the mask. The prediction loss and constraint functions vary among different explainers. Table 7 shows these functions in representative perturbation-based GNN explainers.

3. The edge mask is optimized to decide the explanatory edges. First, the GNN explainer learns the mask M by minimizing the loss $\mathcal{L}(M)$ via gradient descent: $M = M - r \cdot \partial\mathcal{L}(M)/\partial M$, where r is the learning rate. Then, the edges with the k highest values in the learnt mask M are selected as the explanatory edges E_S:

$$E_S = E.\text{top}_k(M). \tag{2}$$

[3]Some GNN explainers (e.g., Ying et al. (2019)) also interpret node features, which are not as effective as graph structure.

[4]We omit E in the loss $\mathcal{L}$ and its meaning is clear from context.

3.2. Power-Law Likelihood Ratio Test

This test is used to measure the similarity between two graphs. Suppose we have a graph G and a generated graph G_A. To ensure G_A possesses similar structural properties as G, we employ a two-sample test for power-law distributions, which mainly evaluates the likelihood between the degree distributions of both G and G_A.

Given a power-law distribution $p(x) \propto x^{-\alpha}$, the initial step involves estimating the scaling parameter α. Although there exists no precise solution for discrete data, such as the degree distribution (Bessi, 2015; Zügner et al., 2018) proposed an approximate expression for G (or G_A) as:

$$\alpha_G = 1 + |\mathcal{D}_G| \cdot [\Sigma_{d_i \in \mathcal{D}_G} \log(d_i/(d_{\text{min}} - 1/2))]^{-1} \tag{3}$$

Here, d_{min} is the minimal degree of nodes in a test set. $\mathcal{D}_G = \{d_v^G \mid v \in V, d_v^G \geq d_{\text{min}}\}$ is a multi-set containing node degrees. Using the obtained scaling parameter α_G for sample $\mathcal{D}_G$, its log-likelihood can be assessed by:

$$l(\mathcal{D}_G) = |\mathcal{D}_G| \cdot \log\alpha_G + |\mathcal{D}_G| \cdot \alpha_G \cdot \log\alpha_G - (\alpha_G + 1) \sum_{d_i \in \mathcal{D}_G} \log d_i \tag{4}$$

Similarly, we can compute the log-likelihood for $\mathcal{D}_{G_A}$ and for $\mathcal{D}_G \cup \mathcal{D}_{G_A}$. The ratio test statistic is then given by:

$$\Lambda(G, G_A) = -2 \cdot l(\mathcal{D}_G \cup \mathcal{D}_{G_A}) + 2 \cdot l(\mathcal{D}_G) + 2 \cdot l(\mathcal{D}_{G_A}) \tag{5}$$

This statistic adheres to a χ^2 distribution with a single degree of freedom. For instance, in this paper, we require that the generated graph G_A exhibits more than 99% structural similarity with G. Then, we only accept G_A when: $\Lambda(G, G_A) < \tau \approx 0.000157$.

4. Our Attack Design

In this section, we present our attack method in detail. We first define the threat model that characterizes the attacker's goal, knowledge, and constraints (e.g., stealthiness, faithfulness, practicability). We then formalize our attack by integrating the threat model, and propose two attack methods to solve the attack problem.

4.1. Attack Formulation

Threat model: We first characterize our threat model[5].

[5]Our threat model is suitable for practical scenarios where GNN explainers are as-a-service (e.g., they are deployed as an API to provide visualized explanations with users' input graphs). For instance, Drug Explorer (Wang et al., 2022b) is a recent explainable GNN tool for drug repurposing (reuse existing drugs for new diseases), where users input a drug graph and the tool outputs the visualized explanation result.

Attack Goal: Given an explanation instance $\{G, y, f, E_S\}$, an attack introduces edge perturbations (via injecting new edges or deleting the existing edges) to the graph G, which produces a modified edge set $\tilde{E}$ (and a modified graph $\tilde{G}$) with a new mask $\tilde{M}$ after the attack. As the edge status of any pair of nodes in the graph can be perturbed and hence the attack mask $\tilde{M} \in [0,1]^{|V|^2}$. GNN explainers can then yield a new explanation $\tilde{E}_S$ based on $\tilde{M}$:

$$\tilde{M} = \arg\min_{\tilde{M}} \mathcal{L}(\tilde{M}) = l(f(\tilde{G}_{\tilde{M}}), y) + \mathcal{C}(\tilde{M}), \quad (6)$$

$$\tilde{E}_S = \tilde{E}.\text{top}_k(\tilde{M}), \quad (7)$$

The attack goal is to ensure the difference between $\tilde{E}_S$ after the attack and E_S without attack be as large as possible.

Attack Knowledge: In real-world applications, the attacker often has limited knowledge about the GNN explainer. We consider this scenario in this paper. Specifically, we assume the attacker only knows the explanation loss $\mathcal{L}$ (as s/he wants to attack a specific explainer) and the explanatory edges E_S (not the mask values M_{E_S}) outputted by the explainer. The attacker does not know the internal model parameters or architecture, nor the training details of the explainer.

Attack Constraint: To ensure our attack be realistic, we consider the following constraints on the attack:

1 The explanatory edges E_S on the clean graph $G = (V, E)$ should be kept in the perturbed graph $\tilde{G} = (V, \tilde{E})$. Otherwise, there is a trivial solution where the attacker can simply remove E_S to fool the GNN explainer.

2 The number of perturbed edges should not exceed a perturbation budget ξ. This is to enforce that the attacker has constrained resources and make the attack practical.

3 $\tilde{G}$ and G should be structurally similar. This ensures the edge perturbations are stealthy, i.e., hard to be detected via checking the graph structure. Here, we follow the power-law likelihood ratio test in Section 3.2 to measure graph structure similarity.

4 The GNN model f still makes accurate predictions on the perturbed graph. This ensures the model faithfulness.

Problem formulation: Based on the above threat model, we formalize our attack as below:

$$\arg\max_{\tilde{E}} |E_S - \tilde{E}_S \cap E_S| \quad (8)$$

$$s.t. \quad E_S \subseteq \tilde{E} \quad (9)$$

$$|E \cup \tilde{E} - E \cap \tilde{E}| \leq \xi \quad (10)$$

$$\Lambda(G, \tilde{G}) < \tau \quad (11)$$

$$f(\tilde{G}) = y \quad (12)$$

However, the objective function of Equation (8) involves the set operation, which makes it hard to be optimized.

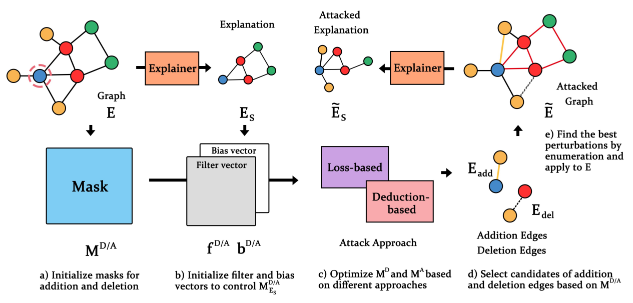

Explanation
Attacked Explanation
Explainer
Explainer
Graph
E
E_S
$\tilde{E}_S$
Attacked Graph
$\tilde{E}$
e) Find the best perturbations by enumeration and apply to E
Mask
Bias vector
Filter vector
Loss-based
Deduction-based
E_{add}
E_{del}
$M^{D/A}$
$f^{D/A}$ $b^{D/A}$
Attack Approach
Addition Edges
Deletion Edges
a) Initialize masks for addition and deletion
b) Initialize filter and bias vectors to control $M^{D/A}_{E_S}$
c) Optimize M^D and M^A based on different approaches
d) Select candidates of addition and deletion edges based on $M^{D/A}$

Figure 2. Overview of our attacks. Take node classification for instance: we find perturbations that maximize the difference between E_S and $\tilde{E}_S$ while satisfying the attack constraints.

Next, we propose to relax the objective function defined on the set to be that defined on numeral value. Let's first consider the ideal case, where the mask values are binary. We observe that the sum of the attack mask $\tilde{M}$'s values on the explanatory edges E_S (without attack), denoted as an ℓ_1-norm $\|\tilde{M}_{E_S}\|_1$, equals to the set size of $|\tilde{E}_S \cap E_S|$. Then we could rewrite Equation (8) as:

$$|E_S - \tilde{E}_S \cap E_S| = |E_S| - |\tilde{E}_S \cap E_S| = k - \|\tilde{M}_{E_S}\|_1.$$

As k is a constant, Equation (8) could be reformulated as:

$$\underset{\tilde{E}}{\text{argmin}} \quad \|\tilde{M}_{E_S}\|_1 \quad (13)$$

This means the attack objective is now to learn an attack mask $\tilde{M}$ on the perturbed graph $\tilde{G}$ that maximally decreases the mask values on E_S. However, due to the attack constrains on the sets, this attack optimization problem is combinatorial, which is NP-hard (a detailed proof is provided in Appendix A). This implies determining the optimal edge perturbations within polynomial time is impossible. To address it, we propose our attack that finds an approximate solution to solve this NP-hard problem.

4.2. Attack Methodology

Our attack is inspired by the observation: A majority of the edges are not important for prediction/explanation, hence a feasible attack should be able to efficiently *score edges based on their importance and then identify the most important edges for addition and deletion.* Next we propose our two attack methods based on this motivation.

4.2.1. Loss-based Attack

This attack method uses the loss change induced by modifying an edge to indicate the edge importance. Note that perturbation-based GNN explainers aim to minimize an explanation loss defined in Equation 1. Hence, we deem an edge as important if it is an existing edge and largely decreases the loss when deleted, or it is a new edge and largely increases the loss when it is added. Deleting these existing edges or/and adding these new edges can significantly affect the explanatory edges E_S. Hence, this attack aims to uncover the edges that largely change the loss.

Loss on $\tilde{M}^D$ and $\tilde{M}^A$: Obviously, added edges are restricted to the complementary edges in G, denoted as $E^C = \{(i,j) : \forall i,j \in V, (i,j) \notin E, i \neq j\}$, while deleted edges are from the existing edges E in G. Due to the **Attack Constraint 1)**, we do not allow deleting the explanatory edges E_S and denote the edge deletion set as $E^D = E - E_S$. We also denote E^A as $E^A = E^C \cup E_S$ for description simplicity. With it, we have $|E^A|+|E^D| = |\tilde{M}|$.

Now we split the attack mask $\tilde{M}$ into an edge deletion mask $\tilde{M}^D \in [0,1]^{|E|}$ on E and an edge addition mask $\tilde{M}^A \in [0,1]^{|E^A|}$ on E^A. We then require $\tilde{M}^D$ and $\tilde{M}^A$ trainable while not learning the mask value on E_S (e.g., set a constraint $\tilde{M}^D_{E_S} = \gamma$ and $\tilde{M}^A_{E_S} = \gamma$) during training. Motivated by the perturbation-based GNN explanation, we design below two losses on $\tilde{M}^D$ and $\tilde{M}^A$ respectively:

$$\min_{\tilde{M}^D} \mathcal{L}(\tilde{M}^D)|_{\tilde{M}^D_{E_S}=\gamma}, \quad \max_{\tilde{M}^A} \mathcal{L}(\tilde{M}^A)|_{\tilde{M}^A_{E_S}=\gamma} \tag{14}$$

where $\mathcal{L}(\tilde{M}^D)|_{\tilde{M}^D_{E_S}=\gamma}$ means $\mathcal{L}(\tilde{M}^D)$ defined on $\tilde{M}^D$ with values on E_S set to be γ, and similar to $\mathcal{L}(\tilde{M}^A)|_{\tilde{M}^A_{E_S}=\gamma}$.

To make the above losses trainable, we introduce four vector variables: $\mathbf{f}^D, \mathbf{b}^D \in \{0,1\}^{|E|}$, and $\mathbf{f}^A, \mathbf{b}^A \in \{0,1\}^{|E^A|}$, where $\mathbf{f}^D$ and $\mathbf{f}^A$ are called filter vectors, and $\mathbf{b}^D$ and $\mathbf{b}^A$ are called bias vectors for $\tilde{M}^D$ and $\tilde{M}^A$, respectively. And we set the values as below:

$$\mathbf{f}^D_e = 0,\ \mathbf{b}^D_e = 1,\ \text{if } e \in E_S;\ \mathbf{f}^D_e = 1,\ \mathbf{b}^D_e = 0,\ \text{if } e \in E^D.$$
$$\mathbf{f}^A_e = 0,\ \mathbf{b}^A_e = 1,\ \text{if } e \in E_S; \mathbf{f}^A_e = 1,\ \mathbf{b}^A_e = 0,\ \text{if } e \in E^C. \tag{15}$$

Then we have:

$$\min_{\tilde{M}^D} \mathcal{L}(\tilde{M}^D)|_{\tilde{M}^D_{E_S}=\gamma} = \mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D + \gamma \cdot \mathbf{b}^D) \tag{16}$$

$$\max_{\tilde{M}^A} \mathcal{L}(\tilde{M}^A)|_{\tilde{M}^A_{E_S}=\gamma} = \mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A + \gamma \cdot \mathbf{b}^A) \tag{17}$$

We observe that, for any $\gamma \in [0,1]$, the results of $\tilde{M}^D$ and $\tilde{M}^A$ on E_S are always γ. Besides, since $\mathbf{f}^D_e = 0$ and $\mathbf{f}^A_e = 0$ for all $e \in E_S$, the mask values $\tilde{M}^D_{E_S}$ and $\tilde{M}^A_{E_S}$ are never learnt during training.

Candidate edges in $\tilde{M}^D$ ($\tilde{M}^A$) for deletion (addition): After learning $\tilde{M}^D$ (or $\tilde{M}^A$), for an edge $e \in E^D$ (or $e \in E^A$), if it has a higher mask value $\tilde{M}^D_e$ (or $\tilde{M}^A_e$), it has more positive effect on *decreasing* (or *increasing*) the loss w.r.t. E_S, and conversely, deleting (or adding) it could increase the loss more. We hence select the edges in E^D (or E^A) with the highest mask values as edge deletion (or addition) candidates E_{del} (or E_{add}).

Deciding the edges for perturbation: To decide the best combination of deleted edges and added edges, we enumerate over different ξ_D and ξ_A such that $\xi_D + \xi_A = \xi$ (*satisfy Attack Constraint 2*), where we select the top ξ_D and top ξ_A candidates in E_{del} and E_{add}, respectively. A generated perturbed graph $\tilde{G}$ under a pair (ξ_D, ξ_A) is kept, if yields a correct GNN prediction $f(\tilde{G}) = y$ (*satisfy Attack Constraint 3*), and passes the likelihood test $\Lambda(G, \tilde{G})$ (*satisfy Attack Constraint 4*). The perturbed graph, when evaluated by a GNN explainer, leading to the largest $|E_S - E_S \cap \tilde{E}_S|$ will be chosen as our final attack result. Algorithm 1 in Appendix D shows the details of the loss-based attack.

Drawbacks: The loss-based attack is effective to some extent (see our results in Section 5). However, a key drawback is that it lacks a direct connection to the attack objective in either Equation (8) or Equation (13). This highlights the necessity on designing an attack method that directly considers the attack objective. Next, we propose another deduction-based attack to mitigate the drawback.

4.2.2. Deduction-based Attack

Our key idea in this attack is to simulate the dynamic mask learning process by the perturbation-based GNN explainer. Then we evaluate the potential of deleted edges from E^D and added edges from E^A to minimize $||\tilde{M}^*_{E_S}||_1$.

Deductive reasoning: Assume the GNN explainer learns the final mask M^* from the initial mask M^0 on the clean graph G with a loss $\mathcal{L}(M)$ defined in Equation 1 and a constant learning rate r. Then, in each iteration $i \in \{1, \cdots, \infty\}$, the mask M^i can be updated as

$$M^i = M^{i-1} - r\frac{\partial \mathcal{L}(M)}{\partial M}|_{M=M^{i-1}}. \tag{18}$$

When applied to the explanatory edges E_S:

$$M^i_{E_S} = M^{i-1}_{E_S} - r\frac{\partial \mathcal{L}(M)}{\partial M_{E_S}}|_{M_{E_S}=M^{i-1}_{E_S}}$$

Applying this equation iteratively, the final mask on E_S is:

$$M^*_{E_S} = M^0_{E_S} - r\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial \mathcal{L}(M)}{\partial M_{E_S}}|_{M_{E_S}=M^i_{E_S}}$$

As a result, the ℓ_1-norm of $M^*_{E_S}$ can be expressed as:

$$||M^*_{E_S}||_1 = ||M^0_{E_S} - r\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial \mathcal{L}(M)}{\partial M_{E_S}}|_{M_{E_S}=M^i_{E_S}}||_1$$

Given that the elements in $M^*_{E_S}$ are all positive (otherwise E_S cannot be the important edges), the ℓ_1-norm of $M^*_{E_S}$ equals to the sum of all its elements $M^*_e, \forall e \in E_S$:

$$\begin{aligned} ||M^*_{E_S}||_1 &= \sum_{e\in E_S}\left[M^0_e - r\lim_{n\to\infty}\sum_{i=0}^{n-1}\frac{\partial \mathcal{L}(M)}{\partial M_e}|_{M_e=M^i_e}\right] \\ &= \sum_{e\in E_S} M^0_e - r\lim_{n\to\infty}\sum_{i=0}^{n-1}\sum_{e\in E_S}\frac{\partial \mathcal{L}(M)}{\partial M_e}|_{M_e=M^i_e} \end{aligned} \tag{19}$$

As the real mask learning process is black-box to the attacker, we have no access to the true values of $M^i_{E_S}, i \in \{1, \cdots, \infty\}$. To address it, we introduce a new variable $\alpha \in [0, 1]$ and its value sequence $\alpha_i, \forall i \in \{1, \cdots, \infty\}$ to approximate the mask sequence $M^i_{E_S}, \forall i \in \{1, \cdots, \infty\}$. At each iteration i, we assume the mask values M^i of important edges E_S equal to (relatively large) α_i:

$$M_e = \alpha|_{M_e = M^i_e, \alpha=\alpha_i}, \forall e \in E_S, i \in \{1, \cdots, \infty\} \tag{20}$$

With this assumption, the following properties are satisfied:

$$\frac{\partial M_e}{\partial \alpha} = 1|_{M_e = M^i_e, \alpha=\alpha_i}, \forall e \in E_S, i \in \{1, \cdots, \infty\} \tag{21}$$

$$\sum_{e \in E_S} M^0_e = \sum_{e \in E_S} \alpha_0 = |E_S|\alpha_0 \tag{22}$$

Incorporating Equations 20, 21 and 22 into Equation 19, we rewrite Equation 19 via a differential equation on α:

$$\begin{aligned} ||M^*_{E_S}||_1 &= |E_S|\alpha_0 - r \lim_{n \to \infty} \sum_{i=0}^{n-1} \sum_{e \in E_S} \frac{\partial \mathcal{L}(M)}{\partial M_e} \frac{\partial M_e}{\partial \alpha}|_{M_{E_S}=\alpha, \alpha=\alpha_i} \\ &= |E_S|\alpha_0 - r \lim_{n \to \infty} \sum_{i=0}^{n-1} \frac{\partial \mathcal{L}(M)|_{M_{E_S}=\alpha}}{\partial \alpha}|_{\alpha=\alpha_i} \\ &\approx |E_S|\alpha_0 - r \int_{\alpha_0}^{\alpha_\infty} \frac{\partial \mathcal{L}(M)|_{M_{E_S}=\alpha}}{\partial \alpha} \end{aligned} \tag{23}$$

Here, α_0 represents an initial mask value and α_∞ approaches the final mask value for edges $e \in E_S$. Here, we set $\alpha_0 = 0$ and approximate the integral in Equation 23 via sampling. Specifically, we use N samples and define constants $\{\beta_i\}^N_{i=1}$ based on a lower bound constant $\beta \in [0, 1]$:

$$\beta_i = \frac{i-1}{N-1} \times (1-\beta) + \beta, \quad \forall i \in \{1, 2, \cdots, N\}$$

Consequently, we express $\|M^*_{E_S}\|_1$ as below

$$||M^*_{E_S}||_1 \approx -\frac{r}{N} \sum_{i=1}^{N} \left(\mathcal{L}(M)|_{M_{E_S}=\beta_i} - \mathcal{L}(M)|_{M_{E_S}=0}\right),$$

where $\mathcal{L}(M)|_{M_{E_S}=\beta_i (\text{or }=0)}$ means $\mathcal{L}(M)$ on the mask M with values on the explanatory edges E_S set to be β_i (or 0).

As an attack, instead, we reverse the mask learning:

$$||\tilde{M}^*_{E_S}||_1 \approx \frac{r}{N} \sum_{i=1}^{N} \left(\mathcal{L}(\tilde{M})|_{\tilde{M}_{E_S}=\beta_i} - \mathcal{L}(\tilde{M})|_{\tilde{M}_{E_S}=0}\right).$$

Then, we can follow the loss-based attack to define the loss on the edge deletion mask $\tilde{M}^D$ and edge addition mask $\tilde{M}^A$ and decide the deleted and added edges for perturbation.

Loss on $\tilde{M}^D$ and $\tilde{M}^A$: We first define a loss $\bar{\mathcal{L}}$ on $\tilde{M}^D \in [0, 1]^{|E|}$ for the edge set E to be minimized[6]:

$$\begin{aligned} \bar{\mathcal{L}}(\tilde{M}^D) &= \sum_{i=1}^{N} \left[\mathcal{L}(\tilde{M}^D)|_{\tilde{M}^D_{E_S}=\beta_i} - \mathcal{L}(\tilde{M}^D)|_{\tilde{M}^D_{E_S}=0}\right] \\ &= \sum_{i=1}^{N} \left[\mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D + \beta_i \cdot \mathbf{b}^D) - \mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D)\right]. \end{aligned} \tag{24}$$

Similarly, we define another loss $\bar{\mathcal{L}}$ on $\tilde{M}^A \in [0, 1]^{|E^A|}$ for the edge set E^A to be maximized:

$$\begin{aligned} \bar{\mathcal{L}}(\tilde{M}^A) &= \sum_{i=1}^{N} \left[\mathcal{L}(\tilde{M}^A)|_{\tilde{M}^A_{E_S}=\beta_i} - \mathcal{L}(\tilde{M}^A)|_{\tilde{M}^A_{E_S}=0}\right] \\ &= \sum_{i=1}^{N} \left[\mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A + \beta_i \cdot \mathbf{b}^A) - \mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A)\right], \end{aligned} \tag{25}$$

where $\mathbf{f}^D$, $\mathbf{f}^A$, $\mathbf{b}^D$, and $\mathbf{b}^A$ are defined in Equation 15.

Deciding the edges for perturbation: After minimizing $\bar{\mathcal{L}}(\tilde{M}^D)$, the edges in E^D with the highest masked values in $\tilde{M}^D$ are the ones that reduce $\|\tilde{M}_{E_S}\|_1$ the most when they are deleted from G. After maximizing the loss $\bar{\mathcal{L}}(\tilde{M}^A)$, the edges in E^A with the highest masked values in $\tilde{M}^A$ are the ones that reduce $\|\tilde{M}_{E_S}\|_1$ the most when they are added to G. We then follow the enumeration strategy proposed before and identify the best deleted and added edges to generate the perturbed graph as our attack. Algorithm 2 in Appendix D details the deduction-based attack.

5. Experiment

5.1. Experiment Setup

Datasets: We evaluate GNN explanations on both node classification and graph classification tasks. For node classification, following existing works (Ying et al., 2019; Luo et al., 2020) we choose three synthetic datasets, i.e., BA House, BA Community, and Tree Cycle. We also add one large real-world dataset OGBN-Products (Bhatia et al., 2016). For graph classification, we use two real-world datasets, MUTAG (Kriege & Mutzel, 2012) and Reddit-Binary (Yanardag & Vishwanathan, 2015). Example graphs of the datasets and detailed descriptions are shown in Appendix C.

Base GNN model and GNN explainers: We use the graph convolutional network (GCN) (Kipf & Welling, 2017) as the base GNN model for node and graph classification, following (Ying et al., 2019). The testing accuracy (all are $> 80\%$) of the trained GCN on the datasets are shown in Table 1. *To ensure the explanation quality, we only select the testing nodes/graphs that are correctly predicted by the GNN model for evaluation.* We choose three well-known perturbation-based GNN explainers: GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), and GSAT (Miao et al., 2022) (More details see Table 7).

Evaluation metric: As different datasets use different number of explanatory edges, we evaluate the attack performance using the overlap ratio between the explanatory edges

[6] W.l.o.g, we omit the constant r/N for description simplicity

Table 1. Testing accuracy on the trained GCN model.

Model	House	Community	Cycle	OGBN-P	MUTAG	REDDIT
GCN	92.32%	83.17%	92.53%	81.12%	86.28%	80.88%

Table 2. Overall attack performance.

Explainer	Method	House	Community	Cycle	OGBN-P	MUTAG	REDDIT
GNNExp.	Random	5.04%	16.75%	22.99%	21.09%	34.55%	33.26%
	Kill-hot	11.11%	18.25%	14.77%	17.62%	42.40%	34.40%
	Loss-based	19.05%	28.35%	57.92%	40.62%	63.40%	47.96%
	Dedu.-based	**20.02%**	**29.79%**	**64.42%**	**42.15%**	**64.80%**	**48.07%**
PGExp.	Random	16.55%	10.50%	5.30%	7.72%	44.66%	29.72%
	Kill-hot	14.50%	13.76%	5.46%	7.29%	55.60%	29.55%
	Loss-based	35.83%	24.46%	37.59%	35.65%	**64.80%**	42.63%
	Dedu.-based	**41.50%**	**24.59%**	**47.93%**	**36.12%**	**64.80%**	**43.05%**
GSAT	Random	34.04%	13.72%	9.30%	11.94 %	26.08%	26.78%
	Kill-hot	36.92%	13.36%	8.33%	8.12 %	26.50%	28.05%
	Loss-based	**51.25%**	20.20%	32.33%	30.82%	68.50%	50.25%
	Dedu.-based	**51.25%**	**45.54%**	**48.45%**	**31.88 %**	**69.90%**	**57.40%**

after and before the attack obtained by the explainers. That is, $|E_S - E_S \cap \tilde{E}_S|/|E_S|$. This overlap ratio is 0 if $\tilde{E}_S$ and E_S are exactly the same, and 1 if they are completely different. Hence a larger ratio indicates a better attack performance. In our results, we report the average overlap ratio on a set of testing nodes and graphs.

Parameter setting: Table 8 in Appendix C summarizes the default values of the key parameters in the explainers and our attack, e.g., perturbation budget ξ, top-k selection parameter k. When studying the impact of each parameter, we fix the others to be the default value.

Compared attacks: There exist no attack methods on GNN explainers. Here, we propose two baselines for comparison.

Random attack: This attack iteratively and randomly selects a pair of nodes. If there exists no edge between them, we add one; otherwise, if the existing edge is not from the original explanation, we delete it. Finally, we ensure the total added and deleted edges to be ξ and pass the attack constraints, e.g., likelihood ratio test and maintain the accurate prediction.

Kill-hot attack: This attack assumes the adversary knows the mask generated by the explainer. It first decides the non-explanatory edges with the highest mask values and then deletes the top-ξ edges, presuming these edges could heavily influence the mask.

5.2. Experimental Results

Overall results: Table 2 shows the attack performance of the compared attacks on the three GNN explainers and six datasets. We have several key observations: 1) Random attack performs the worst, indicating randomly selecting edges for perturbation is not effective enough. 2) Kill-hot attack performs (slightly) better than random attack in most cases, but much worse than our two attacks. This implies the “most important" non-explanatory edges contain some information to influence the final mask, but is not sufficient;

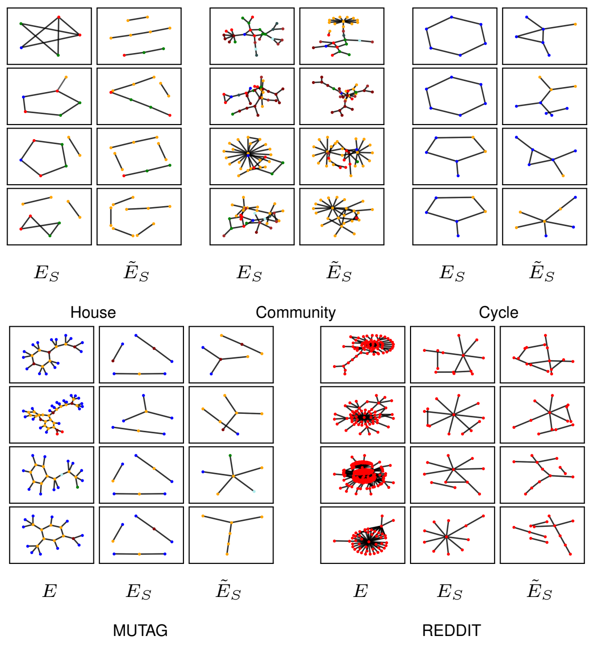


Figure 3. Visualization of the explanation results before and after our deduction-based attack against the PGExplainer. We do not show OGBN-P as it is too big/dense to be visualized.

3) Deduction-based attack performs the best and loss-based attack performs the second best. This shows loss change is a good indicator to identify certain important edges for perturbation, while simulating the learning procedure of (black-box) GNN explainers can be more beneficial to identify the important edges; 4) We do not see a strong connection between the attack performance on explainers and GNN's testing accuracy, by comparing Table 2 and Table 1.

Note that when running our attacks, we filter the intermediate perturbed graphs whose predictions are not maintained. Here we also report the fraction of GNN mis-predictions during the attack in Table 3. We can see the fraction is low, indicating the perturbed graph does not change the prediction often. The main reason is that we focus more on attacking the GNN explanations, instead of attacking the GNN predictions like, e.g., Fan et al. (2023).

Visualizing explanation results: To better understand the vulnerability caused by our attacks, we visualize the explanation results without and with our deduction-based attack on PGExplainer in Figure 3 We can categorize these results into two types: 1) Both explanatory edges obtained by the explainer without attack and those obtained with our attack are in a connected subgraph, but they look significantly different; 2) Explanatory edges obtained by the explainer without attack are in a connected subgraph, but they are disconnected after our attack.

Ablation study: In this experiment, we show the impact of key parameters in our attack on the attack performance.

Table 3. Fraction of failed attacks during training.

Explainer	Method	House	Community	Cycle	OGBN-P	MUTAG	REDDIT
GNNExp.	Loss-based	4.2%	14.1%	11.4%	4.6%	1.7%	0.0%
	Dedu.-based	7.5%	16.5%	5.0%	7.7%	0.0%	0.0%
PGExp.	Loss-based	2.9%	12.6%	5.0%	0.0%	4.2%	0.0%
	Dedu.-based	2.1%	9.2%	10.4%	0.0%	0.0%	0.0%
GSAT	Loss-based	3.8%	12.6%	6.7%	0.0%	0.0%	0.0%
	Dedu.-based	2.1%	9.6%	8.3%	0.0%	1.7%	0.0%

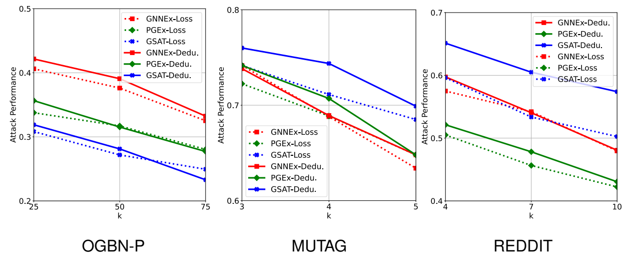
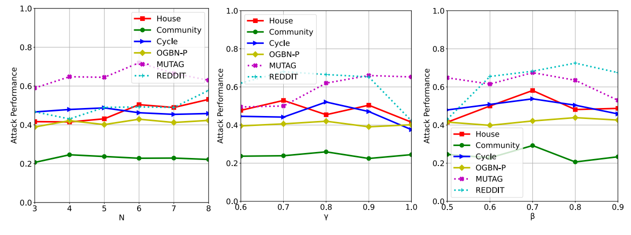


Figure 4. Impact of k on our attack in three real-world datasets.



(a) Impact of N (b) Impact of γ (left) and β (right)

Figure 5. (a) Impact of N on our deduction-based attack; (b) Impact of γ and β on our attack performance.

Impact of k: As synthetic graphs have a fixed size of the groundtruth explanation (e.g., $k = 6$ in BA House), we only study the impact of k (the size of the explanation results) on the real-world MUTAG and REDDIT graphs. The results are shown in Figure 4. We can see our attacks perform better on a smaller k. This is understandable, as it would be naturally more challenging to affect larger number of edges, with a limited perturbation budget $\xi = 2$.

Impact of N: Our deduction-based attack uses a finite number of N samples to approximate the integral. We analyze the effect of N and show the results in Figure 5(a). Overall, the attack results are stable with different N's, e.g., the performance difference is within 5%. A larger N shows slightly better performance. One reason could be that larger N yields more accurate approximation of the integral.

Impact of γ and β: Both our attacks force the mask values on the explanatory edges by the explainer without attack to be a constant (γ or dependent on β) during attack optimization. Here we study the impact of this parameter and show the results in Figure 5(b). The results show our attacks obtain better performance when γ and β are not too large (e.g., 1.0) or not too smaller (e.g., < 0.6). One reason is too small values underestimate the effect of explanatory edges, while too large values overestimate the effect.

Table 4. Attack performance of transferring the generated perturbed graphs on the perturbation-based GNN explainer by baseline and our attacks to other types of GNN explainers.

Explainer	Method	House	Community	Cycle	OGBN-P	MUTAG	REDDIT
CAM	Random	11.25%	22.83%	18.04%	2.71%	18.45%	16.85%
	Kill-hot	13.13%	27.41%	25.42%	2.82%	21.50%	19.25%
	Loss-based	25.33%	46.65%	42.00%	30.31%	33.50%	32.50%
	Dedu.-based	**25.42%**	**51.96%**	**43.50%**	**35.23%**	**37.50%**	**35.50%**
SA	Random	12.71%	17.77%	17.63%	18.83%	17.70%	13.25%
	Kill-hot	15.83%	32.37%	18.33%	14.31%	26.50%	19.50%
	Loss-based	27.38%	46.96%	28.33%	32.00%	47.50%	44.50%
	Dedu.-based	**41.25%**	**50.18%**	**34.17%**	**41.54%**	**53.50%**	**55.25%**
GraphSVX	Random	11.45%	18.48%	11.67%	21.32%	14.85%	18.75%
	Kill-hot	22.17%	26.29%	21.25%	24.15%	27.50%	26.75%
	Loss-based	31.46%	45.67%	23.52%	28.77%	31.50%	33.06%
	Dedu.-based	**35.42%**	**49.69%**	**26.46%**	**30.62** %	**34.50%**	**36.67%**
GEM	random	25.21%	24.93%	14.21%	5.88%	11.40%	15.45%
	Kill-hot	29.58%	29.64%	19.17%	6.38%	19.50%	20.25%
	Loss-based	**52.92%**	40.89%	**38.50%**	28.50%	36.50%	38.25%
	Dedu.-based	47.71%	**44.91%**	36.50%	**29.38** %	**39.50%**	**40.75%**

Table 5. Average runtime of our attacks.

Method	House	Community	Cycle	OGBN-P	MUTAG	REDDIT
Loss-based	6.3s	39.4s	2.5s	124.1s	3.6s	1.8s
Dedu.-based	7.7s	44.8s	2.7s	147.4s	4.0s	2.5s

Impact of ξ: Figure 6 shows the results with different perturbation budgets ξ. We can see our attacks become more effective with larger ξ. This is because a larger ξ ensures the adversary has more capability to perturb the input graph.

Transferability results: Our attacks are designed mainly for perturbation-based GNN explainers. In this experiment, we study the effectiveness of the generated perturbed graphs by our attacks (default setting) on other four types of GNN explainers (See Section B in Appendix). We choose decomposition-based CAM (Pope et al., 2019), gradient-based SA (Baldassarre & Azizpour, 2019), surrogate-based GraphSVX (Duval & Malliaros, 2021), and generation-based GEM (Lin et al., 2021) for evaluation. Table 4 shows the results. As a comparison, we also show results on the two baselines. We can see the transferability results are not good with the random and kill-hot attacks. Instead, all explainers are more vulnerable to the perturbed graphs generated by our (deduction-based) attack. Such results validate the promising transferability of our deduction-based attack is due to its produced attack edges are effective in general.

Runtime: Theoretically, the worst-case complexity of our attacks against GNN explainers per graph is $O(T|V|^2)$, where T is the number of iterations to score the addition/deletion masks and V is the number of nodes in the graph. This is because, in the worst case, our attacks need to find all addition/deletion candidates using a complete graph. Note that this complexity is the same as the GNN explainers themselves[7]. Empirically, Table 5 shows the average

[7]To handle large graphs, the current GNN explainers only use

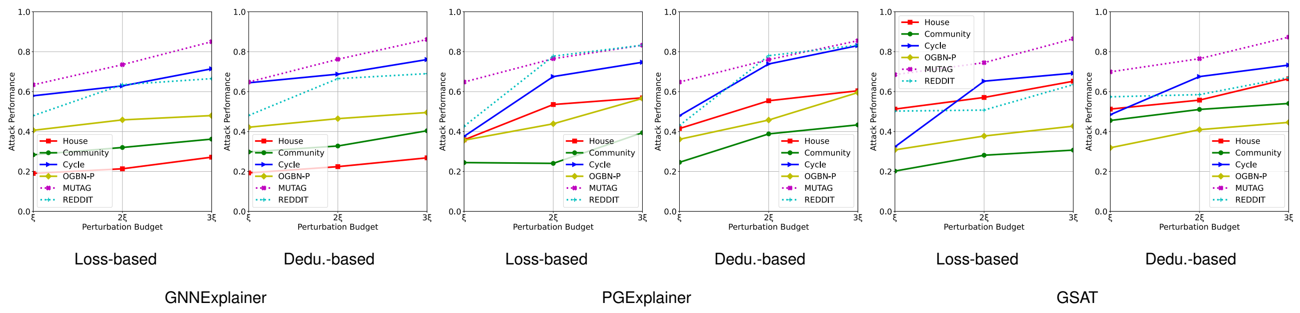


Figure 6. Impact of ξ on our attack performance.

Table 6. Black-box (deduction-based) attack with a surrogate loss.

	House	Community	Cycle	OGB-P	MUTAG	REDDIT
GNNExp.	10.83%	28.75%	58.33%	25.42%	42.50%	45.53%
PGExp.	33.75%	16.88%	34.48%	23.89%	49.00%	42.63%
GSAT	40.21%	30.49%	37.36%	25.43%	47.50%	38.68%

runtime on the 6 datasets in the default setting without any GPU (Macbook with a 2.30GHz CPU and an 8GB RAM).

6. Discussion

Black-box attack on GNN explainers: We also test the black-box setting where the attacker is unknown to the explanation loss. To simulate this, the attacker uses a surrogate explanation loss, e.g., Cross Entropy loss + mask loss $\sum_{i \in E} |m_i|$). Table 6 shows the results with our deduction-based attack using the surrogate loss. We can observe this attack still performs (much) better than the random and kill-hot baselines (see Table 2), but is inferior to our loss-based and deduction-based attacks with known explanation loss.

Defending against the proposed attack: We test two empirical defenses, but found they are not effective enough.

Structure difference detection: This defense aims to detect whether adversarially perturbed graphs generated by our attack are structurally different from clean graphs. Specifically, we utilize the structure similarity metrics NetSim and DeltaCon proposed in (Wills & Meyer, 2020). We found all the perturbed graphs generated by our deduction-based attack and the respective clean graphs have a similarity larger than 0.95. This shows it is hard to differentiate perturbed graphs from clean ones, since the generated perturbed graphs are stealthy under the constraint in Equation (11).

Only API access: This defense limits the attack knowledge by only providing the attacker the API access to target GNN explainers. However, it is still possible for the attacker to break this defense. For instance, the attacker can use a *surrogate explanation loss* and perform our attack. Table 6 shows this surrogate loss based attack is somewhat effective.

the node's local computation graph (i.e., a subgraph formed by the node's within 2 or 3-hop neighboring nodes and their connections) for explanation. This can significantly scale down the graph size to be processed. Our attacks also perform on this subgraph.

We emphasize the designed attack uses limited knowledge about GNN explainers. Hence, a robust explainer that can defend against this *weak* attack does not imply it is effective against stronger attacks. To verify whether a GNN explainer is intrinsically robust, it should test on the worst-case scenario, i.e., the white-box attack whether the attacker has full knowledge about the GNN explainer. A potential direction is hence to design provably robust GNN explainers.

Provably robust GNN explainers: All the existing provable defenses for GNNs (Bojchevski et al., 2020; Wang et al., 2021a; Zhang et al., 2021b; Jin et al., 2020; Xia et al., 2024) focus on classification models that map an input graph to a *scalar* label/confidence score. However, GNN explainers essentially map an input graph to a *matrix* mask, making all existing defenses inapplicable. We will leave designing provably robust GNN explainers as future work.

Attacking other types of GNN explainers. Different types of GNN explainers use different explanation mechanisms. To specifically attack these explainers, we need to redesign the attack objective and solution to fit each of them. Note that the transferability results in Table 4 have somewhat demonstrated the effectiveness of the adversarial edges generated by our attack on attacking all the other types of GNN explainers. It is valuable future work to study the vulnerability of other types of GNN explainers by designing the customized attack objectives and solutions.

7. Conclusion

We perform the first study on understanding the robustness of GNN explainers under graph structure perturbations, with an emphasis on perturbation-based GNN explainers. We formulate the attack under a realistic threat model, i.e., the attacker has limited knowledge about the explainer, allows small perturbation and maintains graph structure, and ensures correct GNN predictions. We propose two attack methods (loss-based and deduction-based) to realize the attack. Extensive evaluations validate existing GNN explainers are vulnerable to our proposed attacks. Future works include designing attacks against non-perturbation-based GNN explainers; and designing (provably) robust GNN explainers.

Acknowledgments

We thank the anonymous reviewers for their insightful reviews and discussions. We thank Youqi Li for polishing the proof. Pang is supported in part by the Natural Science Foundation of Jiangxi Province of China (20232BAB212025) and High-level and Urgently Needed Overseas Talent Programs of Jiangxi Province (20232BCJ25024). Wang is supported in part by the Cisco Research Award and the National Science Foundation under grant Nos. ECCS-2216926, CCF-2331302, CNS-2241713 and CNS-2339686. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Impact Statement

Graph Neural Networks (GNNs) become the mainstream methodology for learning graph data. Explainable GNNs aim to foster the trust of using GNNs. However, this paper shows that state-of-the-art GNN explainers can be broken by practical, stealthy, and faithful adversarial attacks. Such vulnerabilities raise the concerns of using GNNs for safety/security-critical applications such as disease diagnosis and malware detection. Our findings urge the community to design more (provably) robust GNN explainers.

References

Baldassarre, F. and Azizpour, H. Explainability techniques for graph convolutional networks. *ICML Workshop*, 2019.

Bessi, A. Two samples test for discrete power-law distributions, 2015.

Bhatia, K., Dahiya, K., Jain, H., Kar, P., Mittal, A., Prabhu, Y., and Varma, M. The extreme classification repository: Multi-label datasets and code, 2016.

Bojchevski, A., Gasteiger, J., and Günnemann, S. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *ICML*, 2020.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. In *ICML*, pp. 1115–1124. PMLR, 2018.

Dombrowski, A.-K., Alber, M., Anders, C., Ackermann, M., Müller, K.-R., and Kessel, P. Explanations can be manipulated and geometry is to blame. In *NeurIPS*, 2019.

Duval, A. and Malliaros, F. D. Graphsvx: Shapley value explanations for graph neural networks. In *PKDD*, 2021.

Fan, W., Jin, W., Liu, X., Xu, H., Wang, S., Li, Q., Tang, J., Wang, J., and Aggarwal, C. Jointly attacking graph neural network and its explanations. In *ICDE*, 2023.

Feng, Q., Liu, N., Yang, F., Tang, R., Du, M., and Hu, X. Degree: Decomposition based explanation for graph neural networks. In *ICLR*, 2022.

Funke, T., Khosla, M., Rathee, M., and Anand, A. Zorro: Valid, sparse, and stable explanations in graph neural networks. *IEEE TKDE*, 2022.

Gao, G., Xiao, M., Wu, J., Han, K., Huang, L., and Zhao, Z. Opportunistic mobile data offloading with deadline constraints. *IEEE Transactions on Parallel and Distributed Systems*, 28(12):3584–3599, 2017.

Ghorbani, A., Abid, A., and Zou, J. Interpretation of neural networks is fragile. In *AAAI*, 2019.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.

Heo, J., Joo, S., and Moon, T. Fooling neural network interpretations via adversarial model manipulation. In *NeurIPS*, 2019.

Huang, Q., Yamada, M., Tian, Y., Singh, D., and Chang, Y. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE TKDE*, 2022.

Jin, H., Shi, Z., Peruri, V. J. S. A., and Zhang, X. Certified robustness of graph convolution networks for graph classification under topological attacks. In *NeurIPS*, 2020.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kriege, N. and Mutzel, P. Subgraph matching kernels for attributed graphs. In *ICML*, 2012.

Lin, W., Lan, H., and Li, B. Generative causal explanations for graph neural networks. In *ICML*, 2021.

Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., and Zhang, X. Parameterized explainer for graph neural network. In *NeurIPS*, 2020.

Ma, J., Ding, S., and Mei, Q. Towards more practical adversarial attacks on graph neural networks. In *NeurIPS*, 2020.

Miao, S., Liu, M., and Li, P. Interpretable and generalizable graph learning via stochastic attention mechanism. In *ICML*, 2022.

Mu, J., Wang, B., Li, Q., Sun, K., Xu, M., and Liu, Z. A hard label black-box adversarial attack against graph neural networks. In *CCS*, 2021.

Pereira, T., Nascimento, E., Resck, L. E., Mesquita, D., and Souza, A. Distill n'explain: explaining graph neural networks using simple surrogates. In *AISTATS*, 2023.

Pfeifer, B., Saranti, A., and Holzinger, A. Gnn-subnet: disease subnetwork detection with explainable graph neural networks. *Bioinformatics*, 38, 2022.

Pope, P. E., Kolouri, S., Rostami, M., Martin, C. E., and Hoffmann, H. Explainability methods for graph convolutional neural networks. In *CVPR*, 2019.

Rath, B., Morales, X., and Srivastava, J. Scarlet: explainable attention based graph neural network for fake news spreader prediction. In *PAKDD*, 2021.

Scarselli, F., Gori, M., Tsoi, C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE TNN*, 2008.

Schlichtkrull, M. S., Cao, N. D., and Titov, I. Interpreting graph neural networks for NLP with differentiable edge masking. In *ICLR*, 2021.

Schnake, T., Eberle, O., Lederer, J., Nakajima, S., Schütt, K. T., Müller, K.-R., and Montavon, G. Higher-order explanations of graph neural networks via relevant walks. *IEEE TPAMI*, 2021.

Shan, C., Shen, Y., Zhang, Y., Li, X., and Li, D. Reinforcement learning enhanced explainer for graph neural networks. In *NeurIPS 2021*, December 2021.

Slack, D. Z., Hilgard, S., Lakkaraju, H., and Singh, S. Counterfactual explanations can be manipulated. In *NeurIPS*, 2021.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., et al. Graph attention networks. In *ICLR*, 2018.

Vu, M. and Thai, M. T. Pgm-explainer: Probabilistic graphical model explanations for graph neural networks. *NeurIPS*, 2020.

Wang, B. and Gong, N. Z. Attacking graph-based classification via manipulating the graph structure. In *CCS*, 2019.

Wang, B., Jia, J., Cao, X., and Gong, N. Z. Certified robustness of graph neural networks against adversarial structural perturbation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 1645–1653, 2021a.

Wang, B., Li, Y., and Zhou, P. Bandits for structure perturbation-based black-box attacks to graph neural networks with theoretical guarantees. In *CVPR*, 2022a.

Wang, B., Pang, M., and Dong, Y. Turning strengths into weaknesses: A certified robustness inspired attack framework against graph neural networks. In *CVPR*, 2023a.

Wang, B., Lin, M., Zhou, T., Zhou, P., Li, A., Pang, M., Li, H., and Chen, Y. Efficient, direct, and restricted black-box graph evasion attacks to any-layer graph neural networks via influence function. In *WSDM*, 2024.

Wang, Q., Huang, K., Chandak, P., Zitnik, M., and Gehlenborg, N. Extending the nested model for user-centric xai: A design study on gnn-based drug repurposing. *IEEE TVCG*, 2022b.

Wang, X. and Shen, H. W. GNNInterpreter: A probabilistic generative model-level explanation for graph neural networks. In *ICLR*, 2023.

Wang, X., Wu, Y., Zhang, A., He, X., and Chua, T.-S. Towards multi-grained explainability for graph neural networks. *NeurIPS*, 34:18446–18458, 2021b.

Wang, X., Wu, Y., Zhang, A., Feng, F., He, X., and Chua, T.-S. Reinforced causal explainer for graph neural networks. *IEEE TPAMI*, pp. 2297–2309, 2023b.

Wills, P. and Meyer, F. G. Metrics for graph comparison: a practitioner's guide. *Plos one*, 2020.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*, 2019.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Philip, S. Y. A comprehensive survey on graph neural networks. *IEEE TNNLS*, 2020.

Wu, Z., Wang, J., Du, H., Jiang, D., Kang, Y., Li, D., Pan, P., Deng, Y., Cao, D., Hsieh, C.-Y., et al. Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking. *Nature Communications*, 14(1):2585, 2023.

Xia, Z., Yang, H., Wang, B., Jia, J., et al. Gnncert: Deterministic certification of graph neural networks against adversarial perturbations. In *ICLR*, 2024.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *KDD*, 2015.

Yang, Z., Zhong, W., Zhao, L., and Chen, C. Y.-C. Mgraphdta: deep multiscale graph neural network for explainable drug–target binding affinity prediction. *Chemical science*, 13(3):816–833, 2022.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. GNNExplainer: Generating explanations for graph neural networks. In *NeurIPS*. 2019.

Yuan, H., Tang, J., Hu, X., and Ji, S. Xgnn: Towards model-level explanations of graph neural networks. In *KDD*, 2020.

Yuan, H., Yu, H., Wang, J., Li, K., and Ji, S. On explainability of graph neural networks via subgraph explorations. In *ICML*, 2021.

Zhang, S., Liu, Y., Shah, N., and Sun, Y. Gstarx: Explaining graph neural networks with structure-aware cooperative games. *NeurIPS*, 35:19810–19823, 2022.

Zhang, Y., Defazio, D., and Ramesh, A. Relex: A model-agnostic relational model explainer. In *AIES*, 2021a.

Zhang, Z., Jia, J., Wang, B., and Gong, N. Z. Backdoor attacks to graph neural networks. In *SACMAT*, 2021b.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. In *ICLR*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *SIGKDD*, pp. 2847–2856, 2018.

A. Proof of the NP-Hardness of our Problem in Equations (8)-(12)

Generally speaking, there are two approaches to prove the NP-hardness of a problem: 1) following the definition of NP-hardness, 2) reduction from a special case which is a well-known NP-hard problem (Gao et al., 2017). Here, we follow the latter one to prove the NP-hardness of our problem via the 0-1 knapsack problem, which is NP-hard.

For ease of description, we reformulate our attack problem via a matrix representation. We use the binary matrix $\mathbf{M}$ and $\tilde{\mathbf{M}}$ to denote the explanation result outputted by a parameterized GNN explainer (e.g., g_θ) before and after the attack (i.e., $\mathbf{M}_{(i,j)} = 1$ if the edge $(i, j) \in E_S$, otherwise $\mathbf{M}_{(i,j)} = 0$; similarly, $\tilde{\mathbf{M}}_{(i,j)} = 1$ if the edge $(i, j) \in \tilde{E}_S$, otherwise $\tilde{\mathbf{M}}_{(i,j)} = 0$). We use $\mathbf{A}$ and $\tilde{\mathbf{A}}$ to represent the adjacency matrix of a graph $G = (V, E)$ and the perturbed counterpart $\tilde{G} = (V, \tilde{E})$, and a binary matrix $\mathbf{S} \in \{0, 1\}^{|V|\times|V|}$ to indicate whether edges in $\mathbf{A}$ are perturbed or not (i.e. $\tilde{\mathbf{A}} = \mathbf{A} \oplus \mathbf{S}$, $\oplus$ is the `XOR` operator). Based on these relation, we have $\mathbf{M} = g(\mathbf{A})$ and $\tilde{\mathbf{M}} = g(\tilde{\mathbf{A}}) = g(\mathbf{A} \oplus \mathbf{S})$.

Finally, we use a matrix $\mathbf{W}$ to represent the cost for perturbing each edge: for $(i, j) \notin E_S$, $\mathbf{W}_{(i,j)} = 1$; for $(i, j) \in E_S$, $\mathbf{W}_{(i,j)} \approx +\infty$, which is to satisfy the constraint in Eqn 9 of protecting explanatory edges from being modified. Then, our attack problem can be rewritten as solving $\mathbf{S}$ as below:

$$\arg\min_{\mathbf{S}\in\{0,1\}^{|V|\times|V|}} \sum_{(i,j)\in|V|\times|V|} \mathbf{M}_{(i,j)} * \tilde{\mathbf{M}}_{(i,j)} \tag{26}$$

$$s.t. \quad \sum_{(i,j)\in|V|\times|V|} \mathbf{W}_{(i,j)}\mathbf{S}_{(i,j)} \leq \xi \tag{27}$$

$$\Lambda(\mathbf{A}, \mathbf{A} \oplus \mathbf{S}) < \tau \tag{28}$$

$$f(\mathbf{A} \oplus \mathbf{S}) = y \tag{29}$$

This problem is reduced from the 0-1 knapsack problem (https://en.wikipedia.org/wiki/Knapsack_problem), which is NP-hard. Next, we provide a more complete and detailed proof.

Our key idea to prove our problem being NP-hard is to prove one of its subproblems to be NP-hard under certain requirements. Here, we generate a subproblem that is a 0-1 knapsack problem, where the variables $\mathbf{S}$ are considered as "items" and the (bounded) "cost" is generated with our assumptions made on the explainer g, the GNN model, and the value of τ and ξ.

First, we make an assumption on the explainer g. For every non-existent edge $e \in E_A$ $(= V \times V \setminus E)$, we define a corresponding edgeset $\hat{E}_e \subseteq E \setminus E_S$ (and $\hat{E}_e$ can be a null set) and require:

$$\sum_{e\in E_A} |\hat{E}_e| \leq k$$

$$\forall e, e' \in E_A, e \neq e' \rightarrow \hat{E}_e \cap \hat{E}_{e'} = \emptyset$$

When explaining any perturbed graph $\tilde{G} = (V, \tilde{E})$, g is assumed to follow the rules: (1) For any $e \in E_A$ added into $\tilde{E}$, g selects all edges in $\hat{E}_e$ into explanation; (2) if the explanation edges are not full (i.e., less than k), g chooses the remaining edges from E_S. With this, we define an matrix $\mathbf{V}$ as below:

$$\mathbf{V}_{(i,j)} = \begin{cases} |\hat{E}_{(i,j)}|, & (i,j) \in E_A \\ 0, & \text{otherwise.} \end{cases}$$

where $\mathbf{V}_{(i,j)}$ represents the number of changed explanation edges by perturbing (i, j) and its value ranges from 0 to k.

Then, Eq (26) could be rewritten as:

$$\arg\max_{\mathbf{S}\in\{0,1\}^{|V|\times|V|}} \sum_{(i,j)\in|V|\times|V|} \mathbf{V}_{(i,j)} * \mathbf{S}_{(i,j)} \tag{30}$$

Next, we assume f is a binary GNN classifier (parameterized by $\mathbf{W}$) and the prediction is based on:

$$f(\mathbf{A} \oplus \mathbf{S}) = \begin{cases} 1, & \sum_{(i,j)\in|V|\times|V|} \mathbf{W}_{(i,j)}(\mathbf{A}_{(i,j)} \oplus \mathbf{S}) \geq 0 \\ 0, & \text{else} \end{cases}$$

where $\mathbf{W}$ assumes to have the same size as $\mathbf{A}$.

We now define a constant matrix $\mathbf{C}$ by:

$$\mathbf{C}_{(i,j)} = \begin{cases} \mathbf{W}_{(i,j)}, & \text{if } \mathbf{A}_{(i,j)} = 1 \\ -\mathbf{W}_{(i,j)}, & \text{if } \mathbf{A}_{(i,j)} = 0 \end{cases}$$

Assume the prediction of the graph G as $y = 1$. Then, the constraint $f(\mathbf{A} \oplus \mathbf{S}) = y$ could be rewritten as:

$$\sum_{(i,j)\in|V|\times|V|} \mathbf{C}_{(i,j)}\mathbf{S}_{(i,j)} \leq \sum_{(i,j)\in|V|\times|V|} \mathbf{W}_{(i,j)}\mathbf{A}_{(i,j)}$$

By making further assumptions that $\tau = +\infty$ and $\xi = +\infty$, the problem is expressed as:

$$\underset{\mathbf{S}\in\{0,1\}^{|V|\times|V|\setminus E_S}}{\arg\max} \sum_{(i,j)\in|V|\times|V|} \mathbf{V}_{(i,j)}\mathbf{S}_{(i,j)} \tag{31}$$

$$s.t. \quad \sum_{(i,j)\in|V|\times|V|} \mathbf{C}_{(i,j)}\mathbf{S}_{(i,j)} \leq \sum_{(i,j)\in|V|\times|V|} \mathbf{W}_{(i,j)}\mathbf{A}_{(i,j)} \tag{32}$$

In this formulation, $\mathbf{V}_{(i,j)} \in \{0, \cdots, k\}$, $\mathbf{C}_{(i,j)} \in (-\infty, +\infty)$ and $\mathbf{S}_{i,j} \in \{0, 1\}$ correspond to the value, weight, and item in the 0-1 knapsack problem, which cannot be solved in the polynomial time. Note that the above reductions can be completed in finite steps, and the special case belongs to the NP-hard class. The complicated cases should be harder than the special case since more complex constraints are considered in our problems. Therefore, our problem is ensured to be NP-hard.

B. More Related Work

(i) *Decomposition-based methods* consider the prediction of the model as a score and decompose it backward layer-by-layer until it reaches the input. The score of different parts of the input can be used to explain its importance to the prediction. Such methods include Excitation-BP (Pope et al., 2019), DEGREE (Feng et al., 2022), CAM (Pope et al., 2019) and GNN-LRP (Schnake et al., 2021).

(ii) *Gradient-based methods*, including SA (Baldassarre & Azizpour, 2019), Guided-BP (Baldassarre & Azizpour, 2019) and Grad-CAM (Pope et al., 2019), take the gradient of the prediction against the input to show the sensitivity of a prediction to the input. The sensitivity can be used to explain the input for that prediction.

(iii) *Surrogate-based methods* replace the complex GNN model as a simple and interpretable surrogate model. Representative methods are PGMExplainer (Vu & Thai, 2020), GraphLime (Huang et al., 2022), GraphSVX (Duval & Malliaros, 2021), RelEx (Zhang et al., 2021a), and DistilnExplain (Pereira et al., 2023).

(iv) *Generation-based methods* use generative models or graph generators to generate explanations from the instance or model level. These methods include SubgraphX (Yuan et al., 2021), GEM (Lin et al., 2021), RGExplainer (Shan et al., 2021) and RCExplainer (Wang et al., 2023b). For instance, RCExplainer applies reinforcement learning to generate subgraphs as explanations. In every step the agent takes an action to drop or add an edge from the current subgraph, and receives reward based on the new one. When it terminates, the final subgraph is used as the explanation.

(v) *Perturbation-based methods* aim to find the important subgraph or/and features as explanations by perturbing the input graph. Well-known methods include GNNExplainer (Ying et al., 2019), PGExplainer (Luo et al., 2020), Graph-Mask (Schlichtkrull et al., 2021), Zorro (Funke et al., 2022), Refine (Wang et al., 2021b) and GStarX (Zhang et al., 2022). In the paper, we mainly focus on perturbation-based methods: their explanation results are more accurate as they use both the original graph and target GNN model.

C. More Experimental Settings

Dataset description: We provide more details of the datasets used in Section 5. Some example graphs are in Figure 7.

BA House: This graph stems from a base Barabási-Albert (BA) graph attached with “house"-structured motifs as the groundtruth explanation. Nodes within the base graph have a label 0, while nodes positioned at the top, middle, or bottom of the “house"-motif have labels 1, 2, and 3, respectively.

Table 7. Loss, constraint, and mask initialization of representative perturbation-based GNN explainers.

Explainer	Prediction Loss	Constraint Function	Learning Process
GNNExplainer (Ying et al., 2019)	Cross entropy	$\sum_{i\in E} \|m_i\| + m_i \log m_i + (1-m_i)\log(1-m_i)$	Deterministic
PGExplainer (Luo et al., 2020)	Cross entropy	$\sum_{i\in E} \|m_i\| + m_i \log m_i + (1-m_i)\log(1-m_i)$	Stochastic
GSAT (Miao et al., 2022)	Cross entropy	$\sum_{i\in E} m_i \log \frac{m_i}{r} + (1-m_i)\log\frac{1-m_i}{1-r} + c(n,r)$	Stochastic

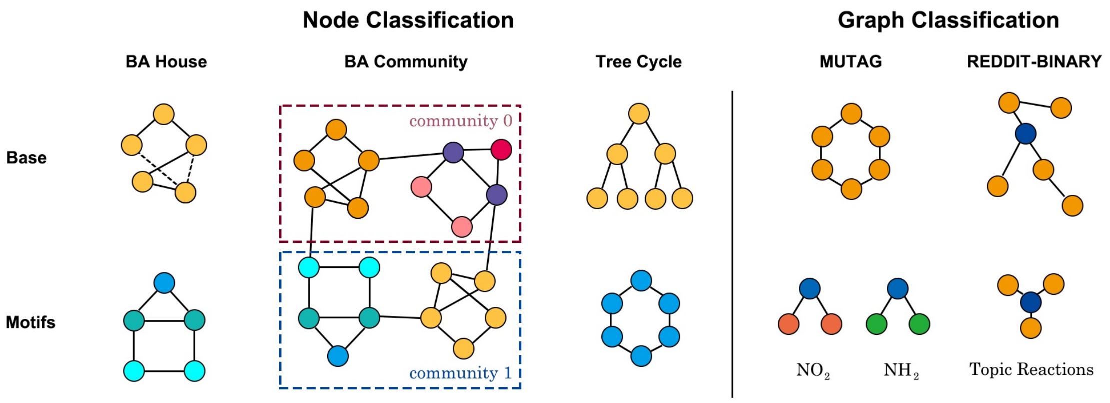


Figure 7. Example synthetic graphs and real-world graphs and their groudtruth explanations.

Table 8. Parameter Setting.

Task	Dataset	#Cases	$\|E\|_{\text{avg}}$	k	ξ	N	β	γ
Node Explanation	BA House	80	4110	6	5	4	0.7	0.7
	BA Community	160	8920	28	10	4	0.7	0.7
	Tree Cycle	160	1950	6	2	4	0.7	0.7
	OGBN-Prod.	40	126036	25	10	4	0.7	0.7
Graph Explanation	MUTAG	40	97	5	2	4	0.7	0.7
	Reddit-Bin.	40	152	10	2	4	0.7	0.7

BA Community: This dataset has two BA-Shapes graphs with two “house"-motifs. Nodes are labelled based on their structural roles and community memberships, with eight classes in total.

Tree Cycle: At its core lies a balanced binary tree. Six-node cycle motifs are appended to randomly chosen nodes from the base graph. The cycle motifs is the groundtruth explanation.

OGBN-Products: This dataset is an undirected unweighted graph, representing an Amazon product co-purchasing network. Nodes represent products sold in Amazon, and edges between two products indicate that the products are purchased together.

MUTAG: It consists of 4,337 molecular graphs, labeled in alignment with their mutagenic effect on the Gram-negative bacterium, Styphimurium. NO_2 and NH_2 motifs are considered as the groundtruth.

Reddit-Binary: It includes 2,000 graphs, each corresponding an online discussion thread on Reddit. Within a graph, nodes signify users, and edges denote interactions between them. The label of a graph is indicative of the interaction type within its thread. A star shape motif is considered as the groundtruth.

Parameter setting: Table 8 shows the default parameter settings in our experiment.

D. Algorithm Details

Algorithm 1 and Algorithm 2 describe the detailed implementation of our two proposed attack algorithms.

Algorithm 1 Loss-based Attack

Input: A graph $G = (V, E)$ with a label y, explanatory edges E_S outputted by a GNN explainer, perturbation budget ξ, ratio test parameter $\tau = 0.000157$, the explainer loss $\mathcal{L}$, and constant γ.

```
 1: Initialize the maximum set difference $md = 0$
 2: Initialize a mask $\tilde{M}^D$ for all edges in $E$
 3: Initialize a filter matrix $\mathbf{f}^D$ and a bias matrix $\mathbf{b}^D$ for all edges in $E$: for $e \in E_S$, $\mathbf{f}_e^D = 0$, $\mathbf{b}_e^D = 1$; otherwise, $\mathbf{f}_e^D = 1$, $\mathbf{b}_e^D = 0$
 4: Set a loss function for $\tilde{M}^D \in [0,1]^{|E|}$ as $\mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D + \gamma \cdot \mathbf{b}^D)$. Learn $\tilde{M}^D$ by minimizing $\bar{\mathcal{L}}(\tilde{M}^D)$
 5: Take $\xi$ edges from $E^D$ with the highest $\xi$ masked values in $\tilde{M}^D$ as deletion candidates $E_{del} = E^D.\text{top}_\xi(\tilde{M}^D)$
 6: Initialize a mask $\tilde{M}^A$ for all edges in $E^A = E^C \cup E_S$
 7: Initialize a filter matrix $\mathbf{f}^A$ and a bias matrix $\mathbf{b}^A$ for all edges in $E^A$: for $e \in E_S$, $\mathbf{f}_e^A = 0$, $\mathbf{b}_e^A = 1$; otherwise, $\mathbf{f}_e^A = 1$, $\mathbf{b}_e^A = 0$
 8: Set a loss function for $\tilde{M}^A$ as $\bar{\mathcal{L}}(\tilde{M}^A) = \mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A + \gamma \cdot \mathbf{b}^A)$. Learn $\tilde{M}^A$ by maximizing $\bar{\mathcal{L}}(\tilde{M}^A)$
 9: Take $\xi$ edges from $E^A$ with the highest $\xi$ masked values in $\tilde{M}^A$ as addition candidates $E_{add} = E^A.\text{top}_\xi(\tilde{M}^A)$
10: for $0 \le \xi_A \le \xi$, $\xi_D = \xi - \xi_A$ do
11:     Add top $\xi_A$ edges in $E_{add}$ to $E$, resulting in $\tilde{E}_{add}$
12:     Delete top $\xi_D$ edges in $E_{del}$ from $\tilde{E}_{add}$, resulting in $\tilde{E}$
13:     if $f(\tilde{G})! = y$ or $\Lambda(G, \tilde{G}(V, \tilde{E})) \ge \tau$ then
14:         Continue
15:     end if
16:     Obtain new explanation $\tilde{E}_S$ on $\tilde{E}$ via Equations 6 and 7
17:     if $|E_S - \tilde{E}_S \cap E_S| > md$ then
18:         $md = |E_S - \tilde{E}_S \cap E_S|$
19:     end if
20: end for
21: return $md$ and $\tilde{E}_S$
```

Algorithm 2 Deduction-based Attack

Input: A graph $G = (V, E)$ with a label y, explanatory edges E_S outputted by a GNN explainer, perturbation budget ξ, ratio test parameter τ, the explainer loss $\mathcal{L}$, number of samples N, and β.

```
 1: Initialize the maximum set difference $md = 0$
 2: Initialize $\beta_i = \frac{i-1}{N-1} \times (1-\beta) + \beta$ for all $i \in \{1, 2, \cdots, N\}$
 3: Initialize a mask $\tilde{M}^D$ for all edges in $E$
 4: Initialize a filter matrix $\mathbf{f}^D$ and a bias matrix $\mathbf{b}^D$ for all edges in $E$: for $e \in E_S$, $\mathbf{f}_e^D = 0$, $\mathbf{b}_e^D = 1$; otherwise, $\mathbf{f}_e^D = 1$, $\mathbf{b}_e^D = 0$
 5: Set a loss function for $\tilde{M}^D \in [0,1]^{|E|}$ as $\bar{\mathcal{L}}(\tilde{M}^D) = \sum_{i=1}^{N}[\mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D + \beta_i \cdot \mathbf{b}^D) - \mathcal{L}(\tilde{M}^D \otimes \mathbf{f}^D)]$. Learn $\tilde{M}^D$ by minimizing
    $\bar{\mathcal{L}}(\tilde{M}^D)$
 6: Take $\xi$ edges from $E^D$ with the highest $\xi$ masked values in $\tilde{M}^D$ as deletion candidates $E_{del} = E^D.\text{top}_\xi(\tilde{M}^D)$
 7: Initialize a mask $\tilde{M}^A$ for all edges in $E^A = E^C \cup E_S$
 8: Initialize a filter matrix $\mathbf{f}^A$ and a bias matrix $\mathbf{b}^A$ for all edges in $E^A$: for $e \in E_S$, $\mathbf{f}_e^A = 0$, $\mathbf{b}_e^A = 1$; otherwise, $\mathbf{f}_e^A = 1$, $\mathbf{b}_e^A = 0$
 9: Set a loss function for $\tilde{M}^A$ as $\bar{\mathcal{L}}(\tilde{M}^A) = \sum_{i=1}^{N}[\mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A + \beta_i \cdot \mathbf{b}^A) - \mathcal{L}(\tilde{M}^A \otimes \mathbf{f}^A)]$. Learn $\tilde{M}^A$ by maximizing $\bar{\mathcal{L}}(\tilde{M}^A)$
10: Take $\xi$ edges from $E^A$ with the highest $\xi$ masked values in $\tilde{M}^A$ as addition candidates $E_{add} = E^A.\text{top}_\xi(\tilde{M}^A)$
11: for $0 \le \xi_A \le \xi$, $\xi_D = \xi - \xi_A$ do
12:     Add top $\xi_A$ edges in $E_{add}$ to $E$, resulting in $\tilde{E}_{add}$
13:     Delete top $\xi_D$ edges in $E_{del}$ from $\tilde{E}_{add}$, resulting in $\tilde{E}$
14:     if $f(\tilde{G})! = y$ or $\Lambda(G, \tilde{G}(V, \tilde{E})) \ge \tau$ then
15:         Continue
16:     end if
17:     Obtain new explanation $\tilde{E}_S$ on $\tilde{E}$ via Equations 6 and 7
18:     if $|E_S - \tilde{E}_S \cap E_S| > md$ then
19:         $md = |E_S - \tilde{E}_S \cap E_S|$
20:     end if
21: end for
22: return $md$ and $\tilde{E}_S$
```

Table 9. Average confidence difference of the most-likely class before and after our attack.

	House	Community	Cycle	OGB-P	MUTAG	REDDIT
GNNExp.	0.0015	0.0575	-0.0495	0.0135	0.0005	-0.0052
PGExp.	0.0165	0.0148	-0.0426	0.0156	0.0036	0.0039
GSAT	0.0235	0.0160	-0.0007	0.0145	-0.0098	0.0051

Table 10. Average number of different explanations edge outputted by GNN explainers in five runs.

	House	Community	Cycle	OGB-P	MUTAG	REDDIT
GNNExp.	0.17/6	3.38/28	0.75/6	1.77/25	0.12/5	1.82/10
PGExp.	0/6	0/28	0/6	0/25	0/5	0/10
GSAT	0/6	0/28	0/6	0/25	0/5	0/10

E. More Discussions

Differences between attacking GNN classifiers and attacking GNN explainers: Both attacks on GNN classifiers and GNN explainers perturb the graph structure, but their attack goals (and hence attack objectives) are different. The former attack focuses on altering *GNN predictions*, and the latter attack altering *GNN explanations* while retaining *GNN predictions*. Specifically, the former attack perturbs the graph structure such that a target GNN classifier produces a wrong (node/graph) *label prediction* on the perturbed graph. In contrast, the latter attack perturbs the graph structure such that a target GNN explainer generates a wrong *explanatory subgraph*, while maintaining the correct GNN prediction on the perturbed graph.

From the formulation of attack objective, the former attack optimizes the *differentiable* loss used by a GNN classifiers, while the later optimizes the *non-differentiable* edge set outputted by a GNN explainer, under certain constraints (this problem is proved to be NP-hard). A possible solution is to adapt the idea of existing attacks by relaxing our attack objective with an approximate differentiable loss (i.e., our loss-based attack), but our results show the attack performance is not promising. We then design a more accurate deduction-based attack that directly mimics the learning dynamics of the GNN explainer.

Do the proposed attacks affect the GNN prediction confidence? We record the average confidence score difference of the most likely class before and after our deduction-based attack. Table 9 shows the results, where positive/negative value means a decrease/increase. We see our attack nearly has no influence on the score of the most confident class.

How consistent are the GNN explanation results, or whether the proposed attack is statistically meaningful? We tested the explanation consistency on the studied perturbation-based GNN explainers. For every (node/graph) instance, we first explain it once with a target GNN explainer to obtain E_s, and then rerun five times of the explainer. We calculate the average number of different edges (denoted as A) in these explanations compared with edges in E_s. The results of $A/|E_s|$ are shown in Table 10. We observe GNNExp. has marginal inconsistency, while PGExp. and GSAT do not have the inconsistency issue. Hence, our attack results are statistically meaningful.