Trustworthy anomaly detection for smart manufacturing

Li, Z.

Chapter 5

Explainable Graph Neural Networks Under Fire

Authors: **Zhong Li**, Simon Geisler, Yuhang Wang, Stephan Günnemann, Matthijs van Leeuwen

Submitted to IEEE Transactions on Knowledge and Data Engineering.

Abstract

Predictions made by graph neural networks (GNNs) usually lack interpretability due to their complex computational behavior and the abstract nature of graphs. In an attempt to tackle this, many GNN explanation methods have emerged. Their goal is to explain a model's predictions and thereby obtain trust when GNN models are deployed in decision critical applications. Most GNN explanation methods work in a post-hoc manner and provide explanations in the form of a small subset of important edges and/or nodes. In this paper we demonstrate that these explanations can unfortunately not be trusted, as common GNN explanation methods turn out to be highly susceptible to adversarial perturbations. That is, even small perturbations of the original graph structure that preserve the model's predictions may yield drastically different explanations. This calls into question the trustworthiness and practical utility of post-hoc explanation methods for GNNs. To be able to attack GNN explanation models, we devise a novel attack method dubbed *GXAttack*, the first *optimization-based* adversarial white-box attack method for post-hoc GNN explanations under such settings. Due to the devastating effectiveness of our attack, we call for an adversarial evaluation of future GNN explainers to demonstrate their robustness. For reproducibility, our code is available via GitHub.

5.1 Introduction

Graph Neural Networks (GNNs) [187] have achieved promising results in various learning tasks, including representation learning [188, 189], node classification [190, 191], link prediction [192, 193], and anomaly detection [55, 59]. However, intricate data representations and non-linear transformations render their interpretation and thus understanding their predictions non-trivial [194].

This has led to the introduction of GNN explanation methods, which can improve a model's transparency and thus increase trust in a GNN model, especially when it is deployed in a decision-critical application (e.g., where fairness, privacy, or safety are important) [195]. Moreover, they can help identify the scenarios where a GNN model may fail [196]. Explainability of a GNN model can be obtained by designing a self-explainable GNN model, which usually employs a simple model architecture and few parameters, leading to explainability at the cost of suboptimal prediction accuracy; or by extracting post-hoc explanations that do not influence the internal workings of a GNN model and thus maintains its high prediction accuracy. In this paper we only consider post-hoc GNN explanation methods, as many such methods have merged over the past decade, including but not limited to [195, 197, 198, 199, 196, 200, 201, 202, 203].

It has been shown that traditional deep neural networks (DNNs) are vulnerable to adversarial attacks [204, 205], where an imperceptible well-designed perturbation to input data can lead to substantially different prediction results. Recently, it was demonstrated that the explanations of predictions made by these DNNs are also fragile [206]. That is, an unnoticeable, well-designed perturbation to input data can result in completely different explanations while the model's predictions remain unchanged. However, the vulnerability of explanation methods has been primarily studied for DNN models designed for image and text data. In contrast, the vulnerability of GNN explanation methods to adversarial attacks has received very limited attention [207, 208], although a plethora of GNN explanation methods have recently been proposed [101, 209, 210].

Importantly, existing studies have only investigated the (in)stability of GNN explanations under relatively "mild" settings: they only consider *random* perturbations and/or *prediction-altering* perturbations. Specifically, [211] theoretically analyzed the stability of three GNN explainers under random perturbations. Further, [208] considered two scenarios: 1) graphs whose predictions are not changed after randomly flipping 20% edges; and 2) severe perturbations using an off-the-shelf attack algorithm

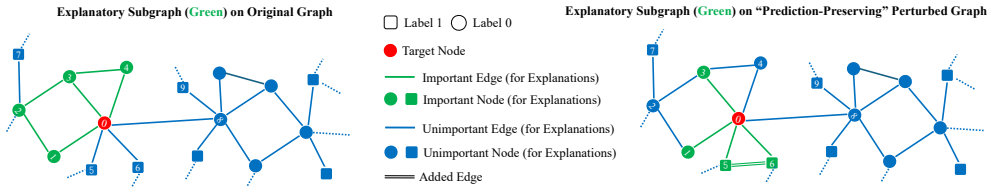


Figure 5.1: When using post-hoc GNN explainers, the explanatory subgraph on a graph with "prediction-preserving" perturbations (right) can strongly differ from that on the original graph (left).

to perturb at most 10% edges to change the GNN predictions (as well as the explanations). Besides, [212] proposed an adversarial attack to simultaneously change the GNN predictions and their explanations. By altering predictions and explanations, however, the graph can become inherently different. [207] explored evaluation metrics for GNN explanations from the perspective of adversarial robustness and resistance to out-of-distribution input, where they defined "adversarial robustness" as the difficulty of reversing a GNN prediction by perturbing the complementary of the explanatory subgraph.

Different from previous studies, we aim to perform structural perturbations that are both carefully crafted (i.e., *optimized* rather than *random*) and imperceptible (i.e., such that the GNN predictions remain unchanged but their explanations differ substantially). As shown in Figure 5.1, we empirically found that small *prediction-preserving* perturbations can result in largely different explanations generated by post-hoc GNN explainers. To our knowledge, we are the first to formally formulate and systematically investigate this problem under a white-box setting. Specifically, we devise *GXAttack*, the first *optimization-based* adversarial white-box attack on post-hoc GNN explanations. We employ a widely used GNN explainer, PGExplainer [199], as example target when designing our attack algorithm. Results on various datasets demonstrate the effectiveness of our approach. Moreover, our experiments show that other widely used GNN explanation methods, such as GradCAM [197], GNNExplainer [195], and SubgraphX [201], are also fragile under the attacks optimized for PGExplainer. Due to the devastating effectiveness of our attack, we call for an adversarial evaluation of future GNN explainers to demonstrate their robustness.

We first summarize related work in Chapter 5.2. Following this, we provide necessary background in Chapter 5.3. Then, we formalize the GNN explanation attack problem and introduce our novel attack algorithm GXAttack in Chapter 5.4. Next, we describe the experimental setups in Chapter 5.5. Experiment results and correspond-

ing analyses are given in Chapter 5.6. Finally, we conclude the paper in Chapter 5.7.

5.2 Related Work

5.2.1 GNN Explanations

[101, 209, 210] perform comprehensive surveys and propose excellent taxonomies of GNN explanation methods. Specifically, GNN explanations methods mainly include: 1) **Gradients-based methods**: Grad [213], GradCAM [197], GuidedBP [104], Integrated Gradients (IG) [214]; 2) **Decomposition-based methods**: GraphLRP [103], GNN-LRP [215]; 3) **Surrogate-based methods**: GraphLIME [203], PGM-Explainer [196]; 4) **Generation-based methods**: XGNN [198]; 5) **Perturbation-based methods**: GNNExplainer [195], PGExplainer [199], GraphMask [216], SubgraphX [201]; and 6) **CF-based methods**: CF-GNNExplainer [202], RCExplainer [200].

5.2.2 Attacks and Defenses of Traditional XAI Methods

It has been shown that traditional XAI methods (namely those designed to explain deep neural networks for image and text data) are susceptible to malicious perturbations [217, 218, 219]. Please refer to [206] for a comprehensive survey.

5.2.3 Robustness/Stability of GNN Explanations

[101, 211] point out that a reliable GNN explainer should exhibit stability, namely minor perturbations that do not impact the model predictions should not largely change the explanations. Particularly, [211] theoretically analyze the stability of three GNN explainers, including vanilla Grad, GraphMask and GraphLIME, for which they derive upper bounds on the instability of explanations. Importantly, [200, 207, 208, 212] are closely related but are different from our work and they are detailed as follows.

OAR. [207] present a novel metric for evaluating GNN explanations, termed Out-Of-Distribution-resistant Adversarial Robustness (OAR). OAR aims to solve the limitations of current removal- and generative-based evaluations. More concretely, they evaluate post-hoc explanation subgraphs by computing their robustness under attack, which consists of three steps: 1) they formulate the adversarial robustness tailored for GNN explanations problem, which is the minimum adversarial perturbation on

the structure of complementary subgraph; 2) they introduce a tractable and easy-to-implement objective of adversarial robustness for GNN explanations; after relaxation, the objective becomes a constrained graph generation problem (namely random perturbations) rather than an adversarial attack problem; 3) they present an Out-Of-Distribution (OOD) reweighting block that aims to confine the evaluation on original data distribution. This work is an initial attempt to explore evaluation metrics for GNN explanations from the perspectives of adversarial robustness and resistance to OOD. In contrast, we focus on optimization-based adversarial attack on post-hoc GNN explanations rather than providing an evaluation metric.

V-InFoR. [208] point out that "existing GNN explainers are not robust to the structurally corrupted graphs, namely graphs with noisy or adversarial edges." This study investigates the negative effect of minor structural corruptions (which do not change the predictions) and severe structural corruptions (which change the predictions) on GNN explainers. Particularly, they provide quantitative evidence that existing GNN explainers are fragile to structurally corrupted graphs. They evaluate 6 GNN explainers under two settings: 1) minor corruptions where they select the perturbed graphs whose predictions are not changed after randomly flipping 20% edges; 2) severe corruptions where they employ adversarial attack algorithm GRABNEL [220] to perturb at most 10% edges to change the predictions. In contrast, our method focus on optimization-based (rather than random) and prediction-preserving (rather than prediction-altering) adversarial attacks.

RCExplainer. [200] aim to find a small set of edges of the input graph such that the prediction result will substantially change if we remove those edges. Specifically, they propose RCExplainer to generate robust counter-factual explanations for GNNs in two steps: 1) they employ a set of decision regions to model the decision logic of a GNN, where each decision region determines the predictions on multiple graphs that are predicted to be the same class; 2) they leverage a DNN model to explore the decision logic, and thereby extract robust counter-factual explanations as a small set of edges of the input graph. This method is only applicable to GNNs that belong to Piecewise Linear Neural Networks. In this paper, we focus on attacking differentiable post-hoc GNN explanation methods that can be applied to any GNNs.

GEAttack. [212] empirically demonstrate that GNN explanation methods can be employed as tools to detect the adversarial perturbations on graphs. On this basis, they propose an attack framework dubbed *GEAttack* that can jointly attack both GNN and its explanations. Specifically, they formulate *GEAttack* as a bi-level optimization problem: 1) they mimic the GNN explanation method optimization process to obtain

graph explanations in the inner loop; 2) they compute the gradient of the attack objective w.r.t. the explanations in the outer loop. Different from GEAttack, we consider prediction-preserving attacks, which are inherently more challenging.

Concurrent Work. While the concurrent work [221] also addresses a similar topic, our study provides an alternative perspective by performing white-box attacks (namely we assume knowing the full knowledge about the GNN classifier and the GNN explainer), while they perform gray-box attacks by assuming knowing only the explanation loss and the generated explanatory edges of the GNN explainer.

5.2.4 Adversarial Robustness of GNNs

Starting with the seminal works [222, 223], that both proposed adversarial attacks on GNNs, a rich literature formed in the realm of GNN-specific adversarial attacks, defenses, and certifications [224, 225]. Most attacks primarily focus on altering predictions via perturbations of the discrete graph structure (edge insertions or deletions), either between existing nodes [222] or via the insertion of new (adversarial) nodes [226]. Optimization-based attacks either operate globally (number of edge insertions or deletions) [227, 228], alter a local neighborhood [222], or can handle a combination of both constraints [229]. While adversarial attacks on GNN explanation methods may benefit from optimization methods over the discrete graph structure, a careful discussion and derivation of appropriate attack objectives and metrics was missing.

5.3 Preliminaries

We utilize lowercase letters, bold lowercase letters, uppercase letters and calligraphic fonts to represent scalars (x), vectors ($\mathbf{x}$), matrices ($\mathbf{X}$), and sets ($\mathcal{X}$), respectively.

Definition 5.1 (Attributed Graph)**.** We denote an attributed graph as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ where $\mathcal{V} = \{v_1, ..., v_n\}$ is the set of nodes. Let $\mathcal{E} = \{e_{ij}\}_{i,j \in \{1,...,n\}}$ be the set of edges, where $e_{ij} = 1$ if there exists an edge between v_i and v_j and $e_{ij} = 0$ otherwise. We use $\mathbf{A} \in \{0, 1\}^{n \times n}$ to denote the corresponding adjacency matrix. $\mathbf{X} \in \mathbb{R}^{n \times d}$ represents the node attribute matrix, where the i-th row vector $\mathbf{x}_i$ denotes the node attribute of v_i. Thus, a graph can also be represented as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$.

5.3.1 Node Classification with GNNs

We consider the task of node classification. Given an input graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{X}\}$ (or $\mathcal{G} = (\mathbf{A}, \mathbf{X})$), a prediction model $f_\theta(\cdot)$ assigns a label c from a pre-defined set $\mathcal{C}$ to

each node $v \in \mathcal{V}$. In other words, we have $f_\theta(\mathcal{G}(v)) = c$, where $\mathcal{G}(v)$ denotes the computation graph of node v. Typically, $f_\theta(\cdot)$ consists of two components: 1) a GNN model used to learn node representations, and 2) a multi-layer perceptron (MLP) classifier that assigns labels. More specifically, given a set of training nodes and their labels $\{(v_1, y_1), ..., (v_n, y_n)\}$ from graph $\mathcal{G}$, the objective function of a GNN classifier can be defined as

$$\min_\theta \mathcal{L}_{\text{GNN}}(f_\theta) := \sum_{i=1}^{n} l(f_\theta(\mathcal{G}(v_i), y_i)) \tag{5.1}$$

$$= \sum_{i=1}^{n} \sum_{c}^{C} \mathbb{I}(y_i = c) \ln(f_\theta(\mathcal{G}(v_i), y_i)^{(c)}), \tag{5.2}$$

where θ is the set of learnable parameters for $f_\theta(\cdot)$, $l(\cdot)$ is the cross-entropy loss function, and $\mathbb{I}(\cdot)$ denotes the Kronecker function. Further, $f_\theta(\mathcal{G}(v_i), y_i)^{(c)}$ indicates the c-th softmax output for the prediction of node v_i, and C is the cardinality of label set $\mathcal{C}$.

Now we define the design detail of the GNN classifier $f_\theta(\cdot)$. For the first component, for simplicity, we employ a 2-layer GCN model with parameters $(\mathbf{W}_1, \mathbf{W}_2)$, i.e.,

$$GCN(\mathcal{G}) = GCN(\mathbf{A}, \mathbf{X}) := \tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2, \tag{5.3}$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}(\mathbf{A} + \mathbf{I})\tilde{\mathbf{D}}^{-1/2}$ (with $\mathbf{I}$ the identity matrix and $\tilde{\mathbf{D}}$ the diagonal matrix of $\mathbf{A} + \mathbf{I}$). Further, $\sigma(\cdot)$ is an activation function such as ReLU. For the second component, we simply use a softmax$(\cdot)$ function rather than a MLP. Therefore, the full GNN classifier f_θ is defined as

$$f_\theta(\mathbf{A}, \mathbf{X}) := \text{softmax}(\tilde{\mathbf{A}}\sigma(\tilde{\mathbf{A}}\mathbf{X}\mathbf{W}_1)\mathbf{W}_2). \tag{5.4}$$

5.3.2 GNN Explainer

We consider post-hoc GNN explanation methods that provide instance-level explanations: given a graph $\mathcal{G}$, a target node v, and prediction $f_\theta(\mathcal{G}(v))$, the explainer $g_\lambda(v, f(\mathcal{G}(v)))$ aims to 'explain' this particular prediction. Following common explanation methods, we assume that explanations are given in the form of an *explanatory subgraph*, namely $\mathcal{G}_s(v) = g(v, f(\mathcal{G}(v)))$, consisting of a small set of important edges and their associated nodes, where the edge importance is measured by a so-called *edge importance score*.

In this paper, we take PGExplainer [199] as an example to attack for three reasons. First, [230] pointed out that GNN explanations providing edge importance are preferable to explanations providing node importance, due to the fact that edges have more fine-grained information than nodes. Second, [231] empirically showed that PGExplainer generates the least unstable explanations, exhibiting a 35.35% reduction in explanation instability relative to the average instability of other GNN explainers. Third, the objective function of PGExplainer is differentiable. For completeness, we briefly revisit the goal of PGExplainer. For brevity, we denote $\mathcal{G}(v)$ as $\mathcal{G}$ (and $\mathcal{G}_s(v)$ as $\mathcal{G}_s$ analogously) throughout the remainder of the paper.

PGExplainer. Given a graph $\mathcal{G}$ and a target node v, PGExplainer provides an *explanatory subgraph* $\mathcal{G}_s(v) = (\mathbf{A}_s, \mathbf{X}_s)$ to explain why $f_\theta(\mathcal{G}(v)) = c$, where $\mathbf{A}_s$ is the subgraph structure and $\mathbf{X}_s$ is the node features associated with nodes residing in the subgraph. Specifically, PGExplainer maximizes the mutual information between the GNN's prediction $\mathbf{y}$ and the explanatory subgraph $\mathcal{G}_s$, i.e.,

$$\max_{\mathcal{G}_s} \text{MI}(\mathbf{y}, \mathcal{G}_s) := H(\mathbf{y}) - H(\mathbf{y}|\mathcal{G}_s), \tag{5.5}$$

where $H(\mathbf{y})$ is the entropy of $f_\theta(\mathcal{G})$, and $H(\mathbf{y}|\mathcal{G}_s)$ is the conditional entropy of the GNN's prediction on the explanatory graph, namely $f_\theta(\mathcal{G}_s)$.

5.4 GXAttack: GNN Explanation Adversarial Attacks

Given an input graph $\mathcal{G}$, a GNN classifier $f_\theta(\cdot)$, and a GNN explainer $g_\lambda(\cdot)$, the original prediction for the target node v is $f_\theta(\mathcal{G}(v))$ and its corresponding original *explanatory subgraph* is given by $\mathcal{G}_s(v) = g_\lambda(f_\theta(v, \mathcal{G}(v)))$. From an attacker's perspective, a desirable manipulated graph $\mathcal{G}_{adv} := \hat{\mathcal{G}} = \phi(\mathcal{G})$ for attacking node v should have the following properties, where $\phi(\cdot)$ is an attack function that perturbs its input, i.e., it removes and/or adds a small set of edges:

1. The norm of the perturbation is so small that the change is considered imperceptible, i.e., $\mathcal{G} \approx \hat{\mathcal{G}}$. This can be formalized as $\|\mathbf{A} - \hat{\mathbf{A}}\| < B$, where B is the perturbation budget;

2. As GNNs can be highly sensitive to input perturbations, we require that the output of the GNN classifier remains approximately the same, i.e., perturbations must be *prediction-preserving*. Formally, we have $f(\mathcal{G}(v)) = f(\hat{\mathcal{G}}(v))$. We hypothesize

that perturbations that are both small and prediction-preserving are very likely to preserve the essence of the learned predictive models;

3. The explanatory subgraph for node v on original graph $\mathcal{G}_s(v) = g_\lambda(f_\theta(v, \mathcal{G}(v)))$ and that on perturbed graph $\hat{\mathcal{G}}_s(v) = g_\lambda(f_\theta(\hat{\mathcal{G}}(v)))$ are substantially different: $\mathcal{G}_s(v) \neq \hat{\mathcal{G}}_s(v)$. In particular, we aim to make $\hat{\mathbf{A}}_s$ (i.e., the adjacency matrix of explanatory subgraph on perturbed graph) strongly different from $\mathbf{A}_s$ (i.e., the adjacency matrix of explanatory subgraph on original graph).

5.4.1 Attack Objectives

Generic Objective. This paper focuses on adversarial structure attacks on post-hoc GNN explanation for node classification:

$$\max_{\hat{\mathbf{A}} \text{ s.t. } \|\mathbf{A}-\hat{\mathbf{A}}\|_0 < B} \mathcal{L}(g_\lambda(f_\theta(\hat{\mathbf{A}}, \mathbf{X}))), \tag{5.6}$$

with loss function $\mathcal{L}$, perturbation budget B, and *fixed* parameters θ and λ. The GNN predictor $f_\theta(\cdot)$ is applied to a graph $\mathcal{G} = \{\mathbf{A}, \mathbf{X}\}$ to obtain node label predictions $f_\theta(\mathbf{A}, \mathbf{X})$. Next, GNN explainer $g_\lambda(\cdot)$ is applied to GNN predictions $f_\theta(\mathbf{A}, \mathbf{X})$ to generate post-hoc explanations $g_\lambda(f_\theta(\mathbf{A}, \mathbf{X}))$. We focus on *evasion* (test time) attacks and *local* attacks on a single node with a budget B. Moreover, we study white box attacks as they represent the "worst-case noise" of a model. In other words, we assume perfect knowledge about the graph, labels, prediction model, and post-hoc explanation model.

Loss Function. We should instantiate the loss function $\mathcal{L}$ such that it pertains to the desirable properties of manipulations discussed before. More specifically, it is defined as

$$\mathcal{L} := \mathrm{MI}(f_\theta(\mathbf{A}, \mathbf{X}), f_\theta(\hat{\mathbf{A}}, \mathbf{X})) - \beta \cdot \mathrm{MI}(g_\lambda(f_\theta(\mathbf{A}, \mathbf{X})), g_\lambda(f_\theta(\hat{\mathbf{A}}, \mathbf{X}))). \tag{5.7}$$

The first term ensures that the GNN predictions before and after perturbations are approximately the same. Further, the second term enforces that the manipulated explanation is different from the original explanatory subgraph. Their relative importance is weighted by hyperparameter $\beta \in \mathbb{R}_+$.

Final Objective. We directly model the edge perturbation matrix $\mathbf{P} \in \{0, 1\}^{n \times n}$, where we flip $\mathbf{A}_{ij}$ (namely $0 \to 1$ or $1 \to 0$) if $\mathbf{P}_{ij} = 1$, and keep $\mathbf{A}_{ij}$ intact otherwise. On this basis, Eq 5.6 can be rewritten as

$$\max_{\substack{\mathbf{P}\in\{0,1\}^{n\times n}\\ \text{s.t. } \sum \mathbf{P}_{ij}<B}} \mathrm{MI}(f_\theta(\mathbf{A},\mathbf{X}), f_\theta(\mathbf{A}\oplus\mathbf{P},\mathbf{X})) - \beta\cdot \mathrm{MI}(g_\lambda(f_\theta(\mathbf{A},\mathbf{X})), g_\lambda(f_\theta(\mathbf{A}\oplus\mathbf{P},\mathbf{X}))), \tag{5.8}$$

where $\oplus$ means *element-wise exclusive or* operation, and B is the edge flipping budget.

5.4.2 Relaxations and Loss Definitions

Unfortunately, directly minimizing $\mathcal{L}$ w.r.t. $\mathbf{P}$ is intractable since there are $\mathcal{O}(2^{n\times n})$ candidates for $\mathbf{P}$. We therefore approximate this discrete optimization problem, i.e., we make some relaxations to make $\mathcal{L}$ (approximately) differentiable w.r.t. $\mathbf{P}$ so that we can perform gradient-based attacks.

Relaxations. We assume that edge flipping graph $\mathcal{M}$ corresponding to $\mathbf{P}\in\{0,1\}^{n\times n}$ (namely using $\mathbf{P}$ as the adjacency matrix) is a Gilbert random graph. In other words, edge occurrences are conditionally independent from each other. As a result, with slight abuse of notations, the probability of graph $\mathcal{M}$ can be factorized as $P(\mathcal{M}) = \prod_{1\le i,j\le n} P(m_{ij})$, dubbed $\tilde{\mathbf{P}}$. Next, we instantiate $P(m_{ij})$ using a Bernoulli distribution, namely $P(m_{ij}) \sim \text{Bernoulli}(\tilde{\mathbf{P}}_{ij})$ with $P(m_{ij}=1) = \tilde{\mathbf{P}}_{ij}$. As a result, Eq 5.8 is equivalent to

$$\max_{\mathbf{P}} \mathcal{L}(\mathbf{P}) = \max_{\mathcal{M}} \mathbb{E}_{\mathcal{M}}[\mathcal{L}(\mathcal{M})] \approx \max_{\tilde{\mathbf{P}}} \mathbb{E}_{\mathcal{M}\sim\tilde{\mathbf{P}}}[\mathcal{L}(\mathcal{M})]. \tag{5.9}$$

In this way, we relax the $\mathbf{P}\in\{0,1\}^{n\times n}$ from binary variables to continuous variables $\tilde{\mathbf{P}}\in[0,1]^{n\times n}$. Hence, we can utilize gradient-based methods to optimize the objective function. At the last step of the attack, we then discretize the result (see Chapter 5.4.3).

Loss Term 1. We approximate the first term in $\mathcal{L}$ as follows:

$$\max_{\tilde{\mathbf{P}}} \mathrm{MI}(f_\theta(\mathbf{A},\mathbf{X}), f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.10}$$

$$:= \max_{\tilde{\mathbf{P}}} H(f_\theta(\mathbf{A},\mathbf{X})) - H(f_\theta(\mathbf{A},\mathbf{X})|f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.11}$$

$$\propto \max_{\tilde{\mathbf{P}}} \; - H(f_\theta(\mathbf{A},\mathbf{X})|f_\theta(\mathbf{A}\oplus\tilde{\mathbf{P}},\mathbf{X})) \tag{5.12}$$

$$\approx \max_{\tilde{\mathbf{P}}} \sum_{c=1}^{C} P_\theta(y=c|\mathcal{G}) \ln(P_\theta(y=c|\hat{\mathcal{G}})). \tag{5.13}$$

Eq 5.11 holds by definition of mutual information; Eq 5.12 holds as $\theta, \mathbf{A}, \mathbf{X}$ are fixed; Eq 5.13 holds when we replace conditional entropy $H(f_\theta(\mathbf{A}, \mathbf{X})|f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))$ with cross entropy $H(f_\theta(\mathbf{A}, \mathbf{X}), f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))$ for practical optimization reason. Moreover, we define $\mathbf{A}_{ij} \oplus \tilde{\mathbf{P}}_{ij} = \mathbf{A}_{ij} + \tilde{\mathbf{P}}_{ij}$ if $\mathbf{A}_{ij} = 0$, and $\mathbf{A}_{ij} \oplus \tilde{\mathbf{P}}_{ij} = \mathbf{A}_{ij} - \tilde{\mathbf{P}}_{ij}$ otherwise.

Loss Term 2. Next, we approximate the second term in $\mathcal{L}$ as follows:

$$\min_{\tilde{\mathbf{P}}} \text{MI}(g_\lambda(f_\theta(\mathbf{A}, \mathbf{X})), g_\lambda(f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}, \mathbf{X}))) \tag{5.14}$$

$$\propto \max_{\tilde{\mathbf{P}}} \text{Dist}(\mathbf{M}_s, \hat{\mathbf{M}}_s) \tag{5.15}$$

$$=: \max_{\tilde{\mathbf{P}}} \left(1 - \frac{\text{vec}(\mathbf{M}_{pr}) \cdot \text{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}\right), \tag{5.16}$$

where $\mathbf{M}_s$ and $\hat{\mathbf{M}}_s$ are the explanatory subgraphs on original graph and perturbed graph, respectively; $\mathbf{M}_{pr}$ and $\hat{\mathbf{M}}_{pr}$ are the predicted explanations (edge importance matrices) on original graph and perturbed graph, respectively. In other words, the explanatory subgraph is represented as an edge importance matrix in this study. Eq 5.15 holds by definition; Eq 5.16 holds when we define the *Dist* function as the *cosine distance metric*, where vec$(\cdot)$ flattens a matrix to a vector, and $\|\cdot\|_F$ represents the Frobenius Norm.

Final Relaxed Objective. As a result, optimizing Eq 5.8 amounts to optimizing

$$\max_{\substack{\tilde{\mathbf{P}} \in [0,1]^{n \times n} \\ \text{s.t. } \sum \tilde{\mathbf{P}}_{ij} < B}} \sum_{c=1}^{C} P_\theta(y = c|\mathcal{G}) \ln(P_\theta(y = c|\hat{\mathcal{G}}) + \beta \cdot \left(1 - \frac{\text{vec}(\mathbf{M}_{pr}) \cdot \text{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}\right). \tag{5.17}$$

As a result, $\mathcal{L}(\tilde{\mathbf{P}})$ (i.e., Eq 5.17) is differentiable w.r.t. $\tilde{\mathbf{P}}$.

5.4.3 Optimization

To guarantee that $\tilde{\mathbf{P}}$ parametrizes a valid distribution and that we obey the budget in expectation, we employ Projected Gradient Ascent Algorithm (based on [227]) to optimize objective function (5.17), and the details are given in Algorithm 4. To go back from the continuous $\tilde{\mathbf{P}}_{ij}$ to discrete $\mathbf{P}_{ij}$, we perform Bernoulli sampling with $\mathbf{P}_{ij} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_{ij})$ after optimization (Line 10). We note that the Projected Randomized Block Coordinate Descent (PR-BCD) [228] can perform this optimization more efficiently, but do not compare both methods due to the low scalability of the

evaluated explainers.

Algorithm 4 Pseudo-code for GXAttack

Input: graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, predictor $f_\theta(\cdot)$, explainer $g_\lambda(\cdot)$, loss $\mathcal{L}$, budget B, #epochs T, learning rate α_t
Output: perturbed adjacency matrix $\hat{\mathbf{A}}$

$\mathbf{y} \leftarrow f_\theta(\mathbf{A}, \mathbf{X})$ ▷ Get predictions on original graph
$\mathcal{G}_s \leftarrow g_\lambda(f_\theta(\mathbf{A}, \mathbf{X}))$ ▷ Get explanations for the predictions on original graph
$\tilde{\mathbf{P}} \leftarrow \mathbf{0} \in \mathbb{R}^{n \times n}$ ▷ Initialize zeros for continuous perturbation variables
for $t \in \{1, 2, ..., T\}$ **do**
  $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}_{t-1}, \mathbf{X})$ ▷ Get predictions on perturbed graph
  $\hat{\mathcal{G}}_s \leftarrow g_\lambda(f_\theta(\mathbf{A} \oplus \tilde{\mathbf{P}}_{t-1}, \mathbf{X}))$ ▷ Get explanations for the predictions on perturbed graph
  $\tilde{\mathbf{P}}_t \leftarrow \tilde{\mathbf{P}}_{t-1} + \alpha_t \cdot \nabla_{\tilde{\mathbf{P}}_{t-1}} \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y}, \hat{\mathcal{G}}_s, \mathcal{G}_s)$ ▷ Update gradients
  $\tilde{\mathbf{P}}_t \leftarrow \prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) \leq B]}(\tilde{\mathbf{P}}_t)$ ▷ Project gradients, see Appendix 5.7 for more detail
end for
$\mathbf{P} \sim \text{Bernoulli}(\tilde{\mathbf{P}})$ s.t. $\sum \mathbf{P}_{ij} \leq B$ ▷ Sample binary perturbation variables
Return $\mathbf{A} \oplus \mathbf{P}$ ▷ Which is defined as $\mathbf{A}_{ij} + \mathbf{P}_{ij}(1 - 2\mathbf{A}_{ij})$

5.4.4 Complexity Analysis

We first discuss the dense case here and then make a discussion of GNNs/explainers implemented with sparse matrices. The time complexity is as follows:

1. **Initial Operations**: $O(f_\theta + g_\lambda)$, typically $O(n^2)$ for a dense GNN and explainer (sometimes even $O(n^3)$).

2. **Each Epoch**:

   - Prediction and explanation on perturbed graph: $O(2T \cdot (f_\theta + g_\lambda))$, where T is the number of epochs.
   - Gradient computation and update: Dependent on the efficiency of backpropagation through the GNN, but generally $O(T \cdot n^2)$ for gradient computation and a similar order for updates.

3. **Sampling Step**: $O(n^2)$.

Meanwhile, the space complexity is as follows:

1. **Space for Storing Matrices**: Storing $\mathbf{A}$, $\mathbf{X}$, $\tilde{\mathbf{P}}$ each requires $O(n^2)$ space.

2. **Additional Space for Operations**: Space required for intermediate gradients and outputs during predictions, typically $O(n^2)$.

Since the explainer uses dense matrices, we also choose an attack operating on all n^2 edges. Thus, the total complexity $O((T+1) \cdot (f_\theta + g_\lambda) + n^2)$ since we have T attack steps plus the initial predictions as well as explanations and, last, we sample the final perturbations. Assuming that $O(f_\theta) = O(g_\lambda) = O(n^2 d + nd^2)$ and $d \ll n$, this yields a total time complexity of $O(n^2)$ with number of attack iterations $T \ll n$, and number of nodes n. This is the same asymptotic complexity as the dense GNN f_θ and explainer g_λ.

Assuming GNN and explainer are in $O(f_\theta) = O(g_\lambda) = O(Ed+nd^2)$ with number of edges E, a dense attack would dominate the overall complexity ($O(n^2)$). An immediate remedy would yield PRBCD [228] since its complexity is $O(B+E)$ with edge flipping budget B and all additional components (e.g., loss function) could be implemented in $O(E)$ with sparse matrices. This yields an overall time and space complexity of $O(E)$, assuming $B \ll E$ and hidden dimensions $d \ll n$, which is the same asymptotic complexity as the sparse GNN f_θ and explainer g_λ.

5.5 Experimental Setup

We aim to answer the following research questions: (**RQ1**) How effective are the attacks generated by GXAttack on PGExplainer? (**RQ2**) How effective are the attacks generated by GXAttack when compared to trivial attackers that use random perturbations? and (**RQ3**) Can the attacks generated by GXAttack (optimized for PGExplainer) transfer to other GNN explanation methods?

5.5.1 Datasets

To answer these questions, we need to evaluate the GNN explanation quality before and after the attacks. However, as pointed out by [231], evaluating the quality of post-hoc GNN explanations is challenging. This is because existing benchmark graph datasets lack ground-truth explanations or their explanations are not reliable. To mitigate this, they proposed a synthetic graph data generator *ShapeGGen*. It can generate a variety of graph datasets with ground-truth explanations, where graph size, node degree distributions, the ground-truth explanation sizes, and more can be varied. As a result, *ShapeGGen* allows us to mimic graph data in various real-world applications. More importantly, *ShapeGGen* can ensure that the generated data and its ground-truth explanations do not suffer from GNN explanation evaluation pitfalls [230], namely trivial explanations, redundant explanations, and weak GNN predictors.

Table 5.1: Summary of synthetic datasets with ground-truth explanations. #Nodes and #Edges represent the number of expected nodes and edges, respectively. We do not consider larger datasets for two reasons: 1) explanation accuracy will be very low (< 0.3) for most explainers, and it is not meaningful to attack low accuracy explainers; 2) computation will be slow for the attacker (and the explainer).

Dataset	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6	Syn7
Shape	House	House	House	Circle	House	House	House
#Subgraphs	10	50	50	50	50	50	100
Subgraph Size	6	6	10	10	10	10	10
P(Connection)	0.3	0.06	0.06	0.06	0.12	0.20	0.06
#Classes	2	2	2	2	2	2	2
#Nodes	59	299	521	511	489	492	1018
#Edges	164	970	1432	1314	1664	1898	3374
Average Degree	2.8	3.2	2.7	2.6	3.4	3.9	3.3
Max Budget	10	15	15	15	15	15	15

This makes *ShapeGGen* very suitable for studying the limitations of GNN explainers. We consider the synthetic datasets in Table 5.1.

Why not real-world datasets? We do not consider real-world datasets for two key reasons: 1) synthetic datasets with ground truth are *de-facto* standards for evaluating post-hoc GNN explainers (for node classification), including GNNExplainer, PGExplainer, PGM-Explainer, SubgraphX, CF-GNNExplainer, RCExplainer, etc. 2) without ground-truth information we can only compute some metrics, such as cosine similarity of explanation before and after attack (which will be defined later) to quantify attack performance. However, as shown in Table 5.2, the cosine similarity and the real attack performance (in terms of ΔGEA, which will be defined later) are not necessarily related. Therefore, considering such metrics may not be meaningful.

5.5.2 Metrics and Baselines

Graph Explanation Accuracy (GEA). We employ the *graph explanation accuracy* proposed in [231] to quantify the quality of generated explanations when the ground-truth explanations are available. For simplicity, we assume that each edge is binary coded as '0' (non-contributory) or '1' (contributory) to model predictions in both ground-truth (provided by *ShapeGGen*) and predicted explanation masks (provided by the GNN explainer). Specifically, *graph explanation accuracy* measures the correctness of generated explanations by computing the Jaccard Index between ground-truth

explanation $\mathbf{M}_{gt}$ and predicted explanation $\mathbf{M}_{pr}$ as

$$\mathrm{JAC}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) = \frac{\mathrm{TP}(\mathbf{M}_{gt}, \mathbf{M}_{pr})}{\mathrm{TP}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) + \mathrm{FP}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) + \mathrm{FN}(\mathbf{M}_{gt}, \mathbf{M}_{pr})}, \tag{5.18}$$

where TP, FP, FN denote True Positives, False Positives, and False Negatives, respectively. Higher values indicate larger consistency between the generated explanation and the ground-truth explanation. As a result, the difference in this metric before and after perturbation can be used to measure the performance of attacks:

$$\Delta\mathrm{GEA} =: \mathrm{JAC}(\mathbf{M}_{gt}, \mathbf{M}_{pr}) - \mathrm{JAC}(\mathbf{M}_{gt}, \hat{\mathbf{M}}_{pr}), \tag{5.19}$$

where higher values indicates more successful attacks.

Cosine Similarity. To quantify the performance of attacks when ground-truth explanations are **not** available, we use cosine similarity to measure the stability of graph explanations:

$$\mathrm{Sim}_{\cos}(\mathbf{M}_{pr}, \hat{\mathbf{M}}_{pr}) = \frac{\mathrm{vec}(\mathbf{M}_{pr}) \cdot \mathrm{vec}(\hat{\mathbf{M}}_{pr})}{\|\mathbf{M}_{pr}\|_F \|\hat{\mathbf{M}}_{pr}\|_F}, \tag{5.20}$$

where $\mathrm{vec}(\cdot)$ flattens its input as a vector, and $\|\cdot\|_F$ represents the Frobenius Norm. Higher values of $\mathrm{Sim}_{\cos}$ (within $[-1, 1]$) indicate higher graph explanation stability, and thus less successful attacks.

Prediction Change. To measure the impact of edge perturbations on GNN predictions, we consider two metrics as follows: 1) **ΔLabel**: the ratio of nodes of which the predicted label has changed after attacks; and 2) **ΔProb**: the average absolute change of predicted probability (of the original predicted class) of all nodes, which is formally defines as

$$\Delta\mathrm{Prob} = \frac{\sum_{i=1}^{N} |P(y_i = y_i^*) - \hat{P}(y_i = y_i^*)|}{N},$$

where $P(y_i = y_i^*)$ is the prediction probability of node v_i being classified as y_i^* (the original predicted class with the highest probability) on the original graph. Moreover, $\hat{P}(y_i = y_i^*)$ is the prediction probability of node v_i being classified as y_i^* (the original predicted class with the highest probability on the original graph) on the perturbed graph.

Attacker Baselines. Given the novelty of our problem setting, we only consider two trivial baselines for edge perturbations: 1) **random flipping**: we randomly flip

B_1 edges; and 2) **random rewiring**: we randomly rewire B_2 edges within k-hop neighbors of the target node.

5.5.3 Attack Transferability Settings

We consider attack transferability on the following five types of GNN explainers: 1) **Gradient**-based: GradCAM [197], Integrated Gradients (IG) [214]; 2) **Perturbation**-based: GNNExplainer [195], SubgraphX [201]; 3) **Surrogate**-based: PGMExplainer [196]; 4) **Decomposition-based methods**: GNN-LRP [215] and 5) **Random** Explainer that randomly generates random important edges.

5.5.4 Training Protocols and Compute Resources

GNN Predictor Training Protocols. To avoid pitfalls caused by weak GNN predictor [230], we should train the GNN predictor close to its maximum possible performance. When training the GNN predictor, we employ an Adam optimizer where the learning rate is 1e-2, weight decay is 1e-5, the number of epochs is 300, and the hidden dimension is 32.

GNN Explainer Training Protocols. We followed the authors' recommendations for setting the hyperparameters of GNN explanation methods. We select top-k ($k = 25\%$) important edges to generate explanations for all GNN explainers. Particularly, when training the PGExplainer, the number of training epochs is 10.

GNN Explanation Attacker Training Protocols. We set the trade-off hyperparameter in the final loss as 0.1 such that the magnitudes of the two loss terms are approximately the same. Besides, the learning rate in gradient ascent is set to 0.1. Importantly, we set the maximal training epochs as 100 due to computation time reason, while further increasing this number can largely improve the attack performance (in terms of ΔGEA). Moreover, we set the perturbation budget to 15 at maximum for synthetic data (note that we attempt to employ a hard minimum budget of 5 with 2000 maximum trials of Bernoulli sampling). We perform sensitivity analysis regarding these hyperparameters, and the results and analysis will be discussed in Chapter 5.6.4.

Explanation Accuracy Computation. The $get\_acc(\cdot)$ function of GraphXAI considers all non-zero edges as predicted positives (namely with binary code '1') by default when computing JAC. Following [231], we set the top 25% (of edges with the highest edge importance scores) as 1 and the rest as zeros in our evaluations when computing both GEA and cosine similarity.

Software and Hardware. The evaluated GNN explainers are implemented mainly based on GraphXAI [1] [231]. Algorithms are implemented in Python 3.8 (using PyTorch [109] and PyTorch Geometric [110] libraries when applicable) and ran on internal clusters with AMD 128 EPYC7702 CPUs (with 512GB RAM). All experiments can be finished within 7 days if using 10 such machine instances. All code and datasets are available on GitHub[2].

5.6 Experimental Results and Analysis

Before answering the research questions, we state three presumptions and check them empirically in Chapter 5.6.1. Next, we answer RQ1 and RQ2 in Chapter 5.6.2, and RQ3 in Chapter 5.6.3. Following this, we perform sensitivity analysis in Chapter 5.6.4 and property analysis in Chapter 5.6.5.

5.6.1 Experiments to Check Presumptions

Presumption 1. *The prediction correctness of a GNN predictor and explanation accuracy of a GNN explainer are* ***not*** *related.* For each dataset, we first get the set of correctly predicted nodes and the set of wrongly predicted nodes; then we compare the distribution of their explanation accuracy. As shown in Figure 5.6 in Appendix, there is no notable difference. In other words, we do **not** observe that wrongly labelled nodes tend to have lower explanation accuracy despite [230] claim that it is unfair to use wrongly labelled nodes to evaluate GNN explanations (as they may not use the ground-truth explanations to make predictions and thus tend to have lower explanation accuracy).

Presumption 2. *The prediction confidence of a GNN predictor and explanation accuracy of a GNN explainer are related,* where prediction confidence is defined as the probability assigned to the predicted class. For each dataset, we first divide the nodes into five groups based on their prediction confidence; then we compare the distribution of their explanation accuracy. Table 5.4 in Appendix show that group 'G9' (i.e., the set of nodes with prediction confidence in $[0.9, 1.0]$) indeed tends to have lower explanation accuracy on the original graph than other groups.

Presumption 3. *The prediction confidence of a GNN predictor and attack performance of a GNN explanation attacker are related.* To check this presumption, we

[1] https://github.com/mims-harvard/GraphXAI
[2] `https://github.com/ZhongLIFR/GXAttack`

Table 5.2: Results on all synthetic datasets (average ± standard deviations across 5 trials). **O. GEA** means explanation accuracy on original graph and **P. GEA** indicates explanation accuracy on perturbed graph. Moreover, **ΔGEA** is the explanation accuracy change after perturbation, $\mathbf{Sim}_{\cos}$ is the cosine similarity between explanations (before and after perturbations), and **#Pert.** represents the number of flipped edges. **ΔLabel** is the average change of predicted labels, while **ΔProb** denotes the average absolute change of predicted probability (of the original predicted class). "GXAttack", "Rnd. Flipping", and "Rnd. Rewiring" correspond to our attack method, random flipping, and random rewiring, respectively. ' ↑' indicates larger values are preferred while ' ↓' means the opposite.

	Dataset	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6	Syn7
	O. GEA	$0.678_{\pm 0.046}$	$0.641_{\pm 0.022}$	$0.494_{\pm 0.012}$	$0.469_{\pm 0.006}$	$0.439_{\pm 0.014}$	$0.306_{\pm 0.025}$	$0.429_{\pm 0.013}$
GXAttack	**P. GEA**(↓)	$0.497_{\pm 0.041}$	$0.375_{\pm 0.011}$	$0.308_{\pm 0.010}$	$0.312_{\pm 0.003}$	$0.290_{\pm 0.004}$	$0.217_{\pm 0.013}$	$0.287_{\pm 0.014}$
GXAttack	**ΔGEA** (↑)	$0.181_{\pm 0.020}$	$0.267_{\pm 0.014}$	$0.186_{\pm 0.013}$	$0.157_{\pm 0.010}$	$0.149_{\pm 0.015}$	$0.088_{\pm 0.014}$	$0.142_{\pm 0.009}$
GXAttack	$\mathbf{Sim}_{\cos}$(↓)	$0.921_{\pm 0.007}$	$0.975_{\pm 0.001}$	$0.980_{\pm 0.004}$	$0.976_{\pm 0.003}$	$0.978_{\pm 0.002}$	$0.988_{\pm 0.001}$	$0.971_{\pm 0.040}$
GXAttack	**#Pert.** (↓)	$3.300_{\pm 0.255}$	$5.620_{\pm 0.045}$	$6.680_{\pm 0.148}$	$5.900_{\pm 0.122}$	$6.560_{\pm 0.055}$	$5.940_{\pm 0.167}$	$7.020_{\pm 0.130}$
GXAttack	**ΔLabel**(↓)	$0.000_{\pm 0.000}$	$0.002_{\pm 0.003}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	$0.001_{\pm 0.001}$	$0.000_{\pm 0.001}$	$0.001_{\pm 0.001}$
GXAttack	**ΔProb**(↓)	$0.106_{\pm 0.008}$	$0.140_{\pm 0.004}$	$0.163_{\pm 0.003}$	$0.150_{\pm 0.002}$	$0.143_{\pm 0.004}$	$0.089_{\pm 0.003}$	$0.147_{\pm 0.001}$
Rnd. Flipping	**P. GEA**(↓)	$0.640_{\pm 0.056}$	$0.638_{\pm 0.023}$	$0.493_{\pm 0.012}$	$0.467_{\pm 0.009}$	$0.438_{\pm 0.013}$	$0.306_{\pm 0.024}$	$0.429_{\pm 0.013}$
Rnd. Flipping	**ΔGEA**(↑)	$0.038_{\pm 0.029}$	$0.001_{\pm 0.005}$	$0.001_{\pm 0.002}$	$0.002_{\pm 0.003}$	$0.001_{\pm 0.001}$	$0.000_{\pm 0.005}$	$0.000_{\pm 0.001}$
Rnd. Flipping	$\mathbf{Sim}_{\cos}$(↓)	$0.800_{\pm 0.006}$	$0.945_{\pm 0.004}$	$0.964_{\pm 0.006}$	$0.943_{\pm 0.005}$	$0.966_{\pm 0.001}$	$0.978_{\pm 0.001}$	$0.981_{\pm 0.001}$
Rnd. Flipping	**#Pert.**(↓)	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$	$15_{\pm 0.000}$
Rnd. Flipping	**ΔLabel**(↓)	$0.051_{\pm 0.012}$	$0.011_{\pm 0.007}$	$0.012_{\pm 0.004}$	$0.009_{\pm 0.006}$	$0.004_{\pm 0.002}$	$0.002_{\pm 0.002}$	$0.004_{\pm 0.002}$
Rnd. Flipping	**ΔProb**(↓)	$0.048_{\pm 0.003}$	$0.009_{\pm 0.005}$	$0.006_{\pm 0.001}$	$0.008_{\pm 0.001}$	$0.006_{\pm 0.001}$	$0.003_{\pm 0.001}$	$0.003_{\pm 0.001}$
Rnd. Rewiring	**P. GEA**(↓)	$0.149_{\pm 0.012}$	$0.045_{\pm 0.005}$	$0.069_{\pm 0.004}$	$0.067_{\pm 0.004}$	$0.020_{\pm 0.002}$	$0.013_{\pm 0.001}$	$0.020_{\pm 0.002}$
Rnd. Rewiring	**ΔGEA**(↑)	$0.530_{\pm 0.043}$	$0.596_{\pm 0.022}$	$0.425_{\pm 0.010}$	$0.402_{\pm 0.003}$	$0.419_{\pm 0.014}$	$0.293_{\pm 0.025}$	$0.409_{\pm 0.012}$
Rnd. Rewiring	$\mathbf{Sim}_{\cos}$(↓)	$0.723_{\pm 0.031}$	$0.941_{\pm 0.006}$	$0.975_{\pm 0.006}$	$0.959_{\pm 0.008}$	$0.951_{\pm 0.004}$	$0.970_{\pm 0.002}$	$0.967_{\pm 0.009}$
Rnd. Rewiring	**#Pert.**(↓)	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$	$16_{\pm 0.000}$
Rnd. Rewiring	**ΔLabel**(↓)	$0.274_{\pm 0.030}$	$0.199_{\pm 0.022}$	$0.202_{\pm 0.004}$	$0.184_{\pm 0.011}$	$0.214_{\pm 0.007}$	$0.159_{\pm 0.013}$	$0.274_{\pm 0.014}$
Rnd. Rewiring	**ΔProb**(↓)	$0.234_{\pm 0.016}$	$0.210_{\pm 0.013}$	$0.198_{\pm 0.007}$	$0.200_{\pm 0.003}$	$0.221_{\pm 0.005}$	$0.174_{\pm 0.013}$	$0.259_{\pm 0.005}$

first divide the nodes into five groups based on their prediction confidence; then we compare the distribution of their attack performance. Table 5.4 in Appendix show that group 'G9' also tends to have less successful attacks (i.e., smaller ΔGEA) when compared to other groups. Possible reasons are that: 1) the nodes in this group have lower explanation accuracy on original graph; and 2) the prediction confidence is close to 1, leaving no much room for perturbations. These two facts together make the attack more challenging.

5.6.2 Experiments to Answer RQ1 and RQ2: Attack Effectiveness

From Table 5.2, we have the following main observations:

Observation 1. *The larger the subgraph size, the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). For this, compare the results on Syn2 (Subgraph Size = 6) to those on Syn3 (Subgraph Size = 10).

Observation 2. *The type of motifs (namely ground-truth explanations) appears to have minimal impact on both explanation accuracy and attack performance.* This can be obtained by comparing the results on Syn3 (Motif ="House") with those on Syn4 (Motif ="Circle").

Observation 3. *The larger the connection probability (namely node degree), the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). This can be obtained by comparing the results on Syn3 (P(Connection) = 0.06), Syn5 (P(Connection) = 0.12), and Syn6 (P(Connection) = 0.20).

Observation 4. *The more subgraphs, the lower the original explanation accuracy of PGExplainer and thus the poorer the attack performance* (i.e., the smaller the value of ΔGEA). This can be obtained by comparing the results on Syn3 (#Subgraphs=50) with those on Syn7 (#Subgraphs=100).

Answer to RQ1. From Table 5.2, we can see that GXAttack can achieve a ΔGEA ranging from 0.088 to 0.267 while using a very small perturbation budget (less than 7.1 on all datasets) and keeping the predicted labels nearly unchanged (i.e., ΔLabel is nearly zero) after perturbation.

Answer to RQ2. In contrast, the random flipping attack employs a budget of 15 but achieves ΔGEA of almost zero on all datasets, indicating it is ineffective. Meanwhile, the random rewiring attack uses a budget of 16 and achieves a very large ΔGEA (ranging from 0.293 to 0.596). However, this comes at the cost of large ΔLabel (ranging from 0.159 to 0.274). In other words, the predicted labels of many nodes have been changed after the attacks, and this violates our objective.

5.6.3 Experiments to Answer RQ3: Attack Transferability

Answer to RQ3. From Table 5.3, we can see that some post-hoc explainers, including PGMExplainer, and IG, suffer from poor explanation accuracy on the original graphs, even giving lower accuracy than the Random Explainer. As a result, investigating the attack transferability on these explainers may not be meaningful. On the contrary, like PGExplainer, explainers such as GradCAM, GNNExplainer, SubgraphX, and GNN-LRP (highlighted in gray) can achieve good performance on most original graphs. For those, the attacks optimized for PGExplainer are also harmful, leading to a dramatic degradation of explanation accuracy. For instance, the explanation accuracy of GNNexplainer decreases from 0.711 to 0.397 on Syn2. This demonstrates the transferability of the attacks generated by GXAttack.

5.6.4 Sensitivity Analysis

Specifically, we perform analysis in terms of ΔGEA, ΔProb and used budget when varying the following hyperparameters, respectively: 1) the loss trade-off hyperparameter; 2) the maximally allowed perturbation budget (Max Budget) in Figure 5.2; and 3) the maximal training epochs of attacker in Figure 5.3.

Table 5.3: Transferability of attacks on synthetic datasets Syn1-6 with a typical run (Results on Syn7 are omitted due to excessive computation time). We report the explanation accuracy (i.e., GEA) on the original graph → explanation accuracy on the perturbed graph.

Explainer	Syn1	Syn2	Syn3	Syn4	Syn5	Syn6
PGExplainer	0.728 → 0.510	0.648 → 0.347	0.485 → 0.304	0.471 → 0.311	0.450 → 0.285	0.332 → 0.225
GradCAM	0.745 → 0.551	0.708 → 0.397	0.510 → 0.300	0.511 → 0.310	0.455 → 0.278	0.414 → 0.241
IG	0.118 → 0.104	0.134 → 0.103	0.136 → 0.101	0.109 → 0.092	0.116 → 0.085	0.078 → 0.065
GNNExplainer	0.738 → 0.550	0.711 → 0.397	0.539 → 0.341	0.514 → 0.327	0.483 → 0.283	0.431 → 0.205
SubgraphX	0.113 → 0.015	0.532 → 0.375	0.296 → 0.141	0.417 → 0.282	0.394 → 0.312	0.408 → 0.305
PGMExplainer	0.015 → 0.064	0.046 → 0.039	0.031 → 0.034	0.036 → 0.041	0.038 → 0.043	0.037 → 0.040
GNN-LRP	0.752 → 0.573	0.713 → 0.394	0.509 → 0.300	0.515 → 0.311	0.457 → 0.284	0.410 → 0.241
RandomExplainer	0.184 → 0.146	0.170 → 0.096	0.135 → 0.102	0.124 → 0.090	0.128 → 0.100	0.133 → 0.086

Figure 5.2: Sensitivity analysis w.r.t. maximally allowed perturbation budget (Max Budget) on Syn2 using GXAttack.

Trade-off Hyperparameter vs Attack Performance. Our sensitivity analysis shows that the attack performance (in term of ΔGEA), ΔProb, and the used budget are nearly constant when we vary the trade-off hyper-parameter β's value from 0.000001 to 10. We omit the figures here.

Attack Budget vs Attack Performance. From Figure 5.2, we can see that as the Max Budget increases from 1 to 10, ΔGEA and ΔProb rise sharply, showing large gains from higher allowed budgets. Beyond a (maximally allowed) budget of 10, ΔGEA and ΔProb both stabilize, indicating no further gains. The underlying reason is that the used budget stop increasing after 10 (maximally allowed) when we set the training epochs as 100. In Tables 5.2 and 5.3, we report the results with a maximum budget equal to 15.

Attack Training Epochs vs Attack Performance. From Figure 5.3, we observe a steady increase in the used budget as epochs increase. However ΔProb first increases and then decreases (due to overfitting) when increasing the training epochs. Similarly, ΔGEA first increases and then decreases (due to overfitting) with the increase of training epochs. For computation time reason, we only report the results with 100 epochs in Tables 5.2 and 5.3. However, it is important to note that the attack performance (in terms of ΔGEA) can be largely improved if we increase the number of training epochs. For instance, ΔGEA can be improved from 0.269 to 0.364 if we increase the training epochs from 100 to 300.

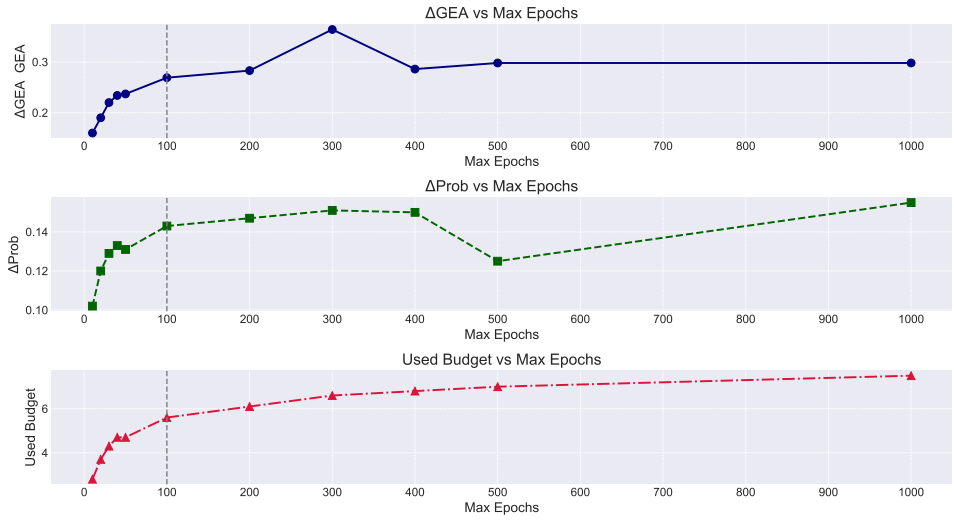
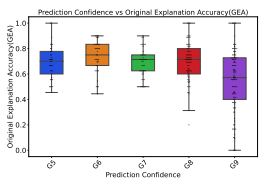
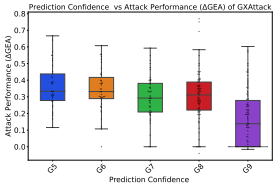


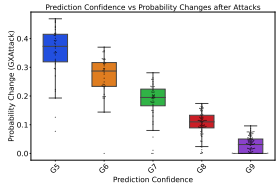
Figure 5.3: Sensitivity analysis w.r.t. maximal training epochs on Syn2 using GXAttack.



(a) Prediction confidence vs. original explanation accuracy on Syn2.



(b) Prediction confidence vs. attack performance (ΔGEA) on Syn2.



(c) Prediction confidence vs. probability change after attacks on Syn2.

Figure 5.4: Property analysis regarding prediction confidence on Syn2. Other datasets show consistent results and are omitted.

5.6.5 Property Analysis

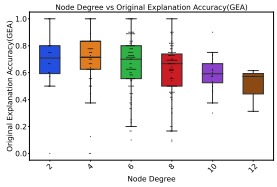
We also conducted some property analysis of GXAttack in terms of the prediction confidence, and node degree.

Property Analysis on Prediction Confidence. From Figure 5.4(a), we can clearly see that nodes in 'G9' tend to have lower original explanation accuracy. As a result, as shown in Figure 5.4(b), the attack performance of GXAttack on these nodes (in 'G9') tend to be less successful (i.e., smaller ΔGEA). This is because the nodes in 'G9' leave no much room for perturbations under the constraint that the predictions remain unchanged, namely much smaller probability changes as shown in Figure 5.4(c).

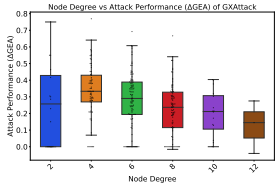
Property Analysis on Node Degree. From Figure 5.5(a), we can see that nodes with higher degrees tend to have lower original explanation accuracy. Figure 5.5(b) shows that the attack performance of GXAttack on these nodes with higher degrees tends to be less successful (i.e., smaller ΔGEA). Possible reasons are that 1) the original explanation accuracy tends to be lower; and 2) small perturbations on nodes with high degrees have limited impact on the predictions, resulting in smaller probability changes as shown in Figure 5.5(c).

5.7 Conclusions

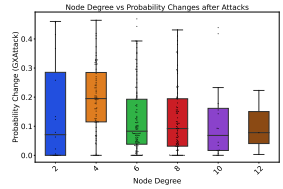
GNN explanations for enhancing transparency and trust of graph neural networks are becoming increasingly important in decision-critical domains. Our research reveals that existing GNN explanation methods are vulnerable to adversarial perturbations that strongly change the explanations without affecting the predictions. This vulner-



(a) Node degree vs. original explanation accuracy on Syn2.



(b) Node degree vs. attack performance (ΔGEA) on Syn2.



(c) Node degree vs. probability change after attacks on Syn2.

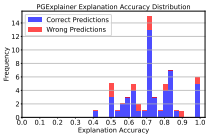
Figure 5.5: Property analysis regarding node degree on Syn2. Other datasets show consistent results and thus are omitted.

ability undermines the reliability of these methods in crucial settings. Specifically, we introduced *GXAttack*, an optimization-based adversarial method to assess the robustness of post-hoc GNN explanations. GXAttack's effectiveness underscores the need for new evaluation standards that incorporate adversarial robustness as a fundamental aspect. Future research should focus on developing explanation methods that are both interpretable and robust against adversarial attacks, as enhancing the stability of GNN explanations is crucial for ensuring their consistency and reliability in practical applications.

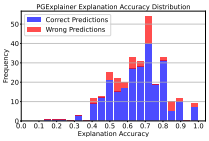
Appendix A: Projected Gradient Descent

After each gradient update, $\tilde{\mathbf{P}}_t \in \mathbb{R}^{n \times n}$, although it must lie in $[0,1]^{n \times n}$ for its interpretation as a Bernoulli random variable. Moreover, we must ensure that sampling $\mathbf{P}_t \in \{0,1\}^{n \times n} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_t)$ will yield a discrete perturbation $\hat{\mathbf{A}} = \mathbf{A} \oplus \mathbf{P}_t$ obeying the admissible perturbations $\|\mathbf{A} - \hat{\mathbf{A}}\|_0 < B$ with sufficient likelihood. Following [227, 228], we ensure that $\mathbb{E}_{\mathbf{P} \sim \text{Bernoulli}(\tilde{\mathbf{P}}_t)}[\|\mathbf{P}\|_0] \leq B$ and then obtain the final perturbation form a small set of samples. To guarantee that $\tilde{\mathbf{P}}_t$ parametrizes a valid distribution and that we obey the budget in expectation, we employ the projection $\prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) \leq B]}(\tilde{\mathbf{P}}_t)$.

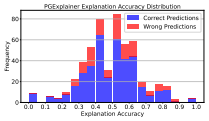
The space spanned by $\tilde{\mathbf{P}}_t \in [0,1]^{n \times n}$ describes a $n \times n$ dimensional hypercube intersected with the region below the B-simplex defining $\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}}_t) = B]$. The region below the B-simplex can be expressed as $\vec{1}^\top \tilde{\mathbf{P}} \vec{1} = \sum_{i,j} \tilde{\mathbf{P}}_{i,j} \leq B$. Thus, after



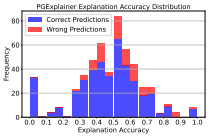
(a) Syn1



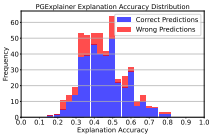
(b) Syn2



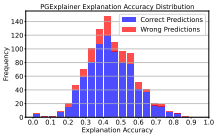
(c) Syn3



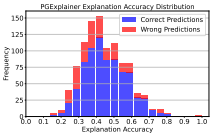
(d) Syn4



(e) Syn5



(f) Syn6



(g) Syn7

Figure 5.6: Comparative analysis of explanation accuracy with respect to prediction correctness on all synthetic datasets. The results show that explanation accuracy is not necessarily correlated with the prediction correctness. Particularly, we did **not** observe that wrongly predicted nodes tend to have lower explanation accuracy.

each gradient update, we solve

$$\prod_{\mathbb{E}[\text{Bernoulli}(\tilde{\mathbf{P}})\leq B]}(\tilde{\mathbf{P}}) = \arg\min_{\tilde{\mathbf{S}}} \|\tilde{\mathbf{P}} - \tilde{\mathbf{S}}\|_F^2 \tag{5.21}$$
$$\text{subject to } \sum_{i,j} \tilde{S}_{i,j} \leq B \text{ and } \tilde{\mathbf{S}} \in [0,1]^{n\times n}$$

for clipping to the hyperplane and projecting back towards the simplex using a bisection search. See [227] for details and the derivation.

Appendix B: Presumptions Checking Results

Table 5.4: Complete results on Syn1-Syn7. **O. GEA** means explanation accuracy on original graph and **P. GEA** indicates explanation accuracy on perturbed graph. Moreover, **ΔGEA** is the explanation accuracy change after perturbation, $\mathbf{Sim}_{\cos}$ is the cosine similarity between explanations (before and after perturbations), and **#Pert.** represents the number of flipped edges. Besides, **ΔLabel** is the average change of predicted labels, while **ΔProb** denotes the average absolute change of predicted probability (of the original predicted class). "G5" is the set of nodes of which the original prediction confidence located in [0.5,0.6). Similarly, "G6" for [0.6,0.7), "G7" for [0.7,0.8), "G8" for [0.8,0.9), "G9" for [0.9,1.0], and "All" for all nodes.

Dataset	Confidence Group	G5	G6	G7	G8	G9	All
Syn1	**O. GEA**	0.696	0.763	0.726	0.775	0.683	0.728
	P. GEA (↓)	0.403	0.463	0.485	0.584	0.679	0.560
	ΔGEA (↑)	0.293	0.299	0.241	0.190	0.004	0.168
	$\mathbf{Sim}_{\cos}$ (↓)	0.817	0.868	0.898	0.937	0.987	0.923
	#Pert. (↓)	6.5	5.0	4.5	2.7	0.2	3.0
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.327	0.204	0.147	0.065	0.002	0.108
Syn2	**O. GEA**	0.694	0.739	0.690	0.694	0.546	0.648
	P. GEA (↓)	0.329	0.396	0.390	0.387	0.381	0.347
	ΔGEA (↑)	0.365	0.293	0.300	0.307	0.165	0.269
	$\mathbf{Sim}_{\cos}$ (↓)	0.958	0.960	0.971	0.979	0.991	0.977
	#Pert. (↓)	7.1	6.6	6.2	6.3	3.8	5.6
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.353	0.268	0.181	0.103	0.003	0.143
Syn3	**O. GEA**	0.484	0.496	0.479	0.510	0.440	0.480
	P. GEA (↓)	0.305	0.303	0.279	0.335	0.310	0.306
	ΔGEA (↑)	0.179	0.193	0.200	0.175	0.130	0.179
	$\mathbf{Sim}_{\cos}$ (↓)	0.966	0.973	0.962	0.981	0.989	0.978
	#Pert. (↓)	8.0	7.4	6.6	5.8	4.7	6.7
	ΔLabel (↓)	0.008	0	0	0	0	0.002
	ΔProb (↓)	0.242	0.206	0.166	0.099	0.039	0.165
Syn4	**O. GEA**	0.500	0.500	0.493	0.495	0.416	0.471
	P. GEA (↓)	0.309	0.314	0.319	0.326	0.296	0.311
	ΔGEA (↑)	0.192	0.187	0.179	0.168	0.120	0.159
	$\mathbf{Sim}_{\cos}$ (↓)	0.973	0.972	0.962	0.979	0.989	0.980
	#Pert. (↓)	6.9	6.8	6.4	6.2	3.9	5.7
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.333	0.267	0.177	0.111	0.030	0.149
Syn5	**O. GEA**	0.456	0.470	0.443	0.461	0.430	0.450
	P. GEA (↓)	0.252	0.264	0.282	0.305	0.336	0.293
	ΔGEA (↑)	0.204	0.207	0.161	0.157	0.094	0.157
	$\mathbf{Sim}_{\cos}$ (↓)	0.967	0.971	0.975	0.979	0.984	0.976
	#Pert. (↓)	7.5	7.6	7.0	6.6	5.2	6.6
	ΔLabel (↓)	0.013	0	0	0	0	0.002
	ΔProb (↓)	0.287	0.226	0.147	0.085	0.031	0.138
Syn6	**O. GEA**	0.415	0.412	0.407	0.382	0.291	0.332
	P. GEA (↓)	0.260	0.249	0.239	0.259	0.217	0.231
	ΔGEA (↑)	0.155	0.162	0.168	0.123	0.074	0.101
	$\mathbf{Sim}_{\cos}$ (↓)	0.975	0.975	0.979	0.986	0.984	0.989
	#Pert. (↓)	7.3	7.8	7.5	7.0	5.0	5.9
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.348	0.270	0.189	0.106	0.029	0.092
Syn7	**O. GEA**	0.428	0.415	0.406	0.405	0.401	0.409
	P. GEA (↓)	0.266	0.250	0.294	0.263	0.278	0.263
	ΔGEA (↑)	0.161	0.165	0.156	0.142	0.123	0.146
	$\mathbf{Sim}_{\cos}$ (↓)	0.986	0.983	0.989	0.991	0.992	0.990
	#Pert. (↓)	8.2	7.9	7.5	7.1	6.1	7.2
	ΔLabel (↓)	0	0	0	0	0	0
	ΔProb (↓)	0.319	0.253	0.169	0.097	0.035	0.147