2023 IEEE 39th International Conference on Data Engineering (ICDE) | 979-8-3503-2227-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICDE55515.2023.00056

Jointly Attacking Graph Neural Network and its Explanations

Wenqi Fan[1,8*], Han Xu[2*], Wei Jin[2], Xiaorui Liu[3], Xianfeng Tang[4], Suhang Wang[5],
Qing Li[1], Jiliang Tang[2], Jianping Wang[6], and Charu Aggarwal[7]
[1]The Hong Kong Polytechnic University, [2]Michigan State University, [3]North Carolina State University, [4]Amazon,
[5]The Pennsylvania State University, [6]City University of Hong Kong, [7]IBM T.J. Watson, [8]PolyU-Shenzhen Research Institute
wenqifan03@gmail.com, {xuhan1, jinwei2, tangjili}@msu.edu, xiaorui.liu@ncsu.edu, tangxianfeng@outlook.com,
szw494@psu.edu, csqli@comp.polyu.edu.hk, jianwang@cityu.edu.hk, charu@us.ibm.com

***Abstract*—Graph Neural Networks (GNNs) have boosted the performance for many graph-related tasks. Despite the great success, recent studies have shown that GNNs are still vulnerable to adversarial attacks, where adversaries can mislead the GNNs' prediction by modifying graphs. On the other hand, the explanation of GNNs (GNNEXPLAINER for short) provides a better understanding of a trained GNN model by generating a small subgraph and features that are most influential for its prediction. In this paper, we first perform empirical studies to validate that GNNEXPLAINER can act as an inspection tool and have the potential to detect the adversarial perturbations for graphs. This finding motivates us to further investigate a new problem: *Whether a graph neural network and its explanations can be jointly attacked by modifying graphs with malicious desires?* It is challenging to answer this question since the goals of adversarial attack and bypassing the GNNEXPLAINER essentially contradict with each other. In this work, we give a confirmative answer for this question by proposing a novel attack framework (GEAttack) for graphs, which can attack both a GNN model and its explanations by exploiting their vulnerabilities simultaneously. To the best of our knowledge, this is the very first effort to attack both GNNs and explanations on graph-structured data for the trustworthiness of GNNs. Comprehensive experiments on various real-world datasets demonstrate the effectiveness of the proposed method.**

***Index Terms*—Graph Neural Networks, Adversarial Attacks, Explanations, GNNEXPLAINER, Trustworthy GNNs.**

I. INTRODUCTION

Graph neural networks (GNNs) have achieved significant success for graphs in various real-world data mining applications [1]–[4], such as node classification [5], recommender systems [6]–[12], and natural language processing [13]. Despite the great success, recent studies show that GNNs are vulnerable to adversarial attacks [14], [15], which has raised great concerns for employing GNNs in security-critical applications [13], [15], [16]. More specifically, attackers can insert adversarial perturbations into graphs, which can lead a well-designed model to produce incorrect outputs or have bad overall performance [13], [17], [18]. For example, adversaries can build well-designed user profiles to promote/demote items in bipartite graphs at many e-commerce platforms such as Alibaba and Amazon [19], [20]; or some hackers intend to damage an electoral opponent's reputation by propagating fake news in social media [16].

Recently, interpretation methods for GNNs (GNNEXPLAINER for short) [21]–[24] have been proposed to explain the inner working mechanisms of GNNs. In particular, given a trained GNN model and its prediction on a test node, GNNEXPLAINER [21], [22] will return a small subgraph together with a small subset of node features that are most influential for its prediction, so human can interpret the model's decision via the output subgraphs and features. In our work, we hypothesize that GNNEXPLAINER can also provide great opportunities for human (inspectors or system designers) to inspect the "confused" predictions from adversarial perturbations. For example, in the e-commerce platform systems, once there exist abnormal predictions for certain products given by the GNN model, the GNNEXPLAINER can help us locate the most influential users which cause the model to make such a prediction. In this way, we can figure out the abnormal users, or adversaries who deliberately propagate such adversarial information to our system. As an illustrative example, in Figure 1, an attacker (Attacker 1) changes node v_1's prediction of the GNN model by adding an adversarial edge (v_1, v_7) (cf. Figure 1 (b)). At this time, if people find v_i's prediction is problematic, they can apply graph interpretation methods (cf. Figure 1 (c)) to "inspect" this anomaly. Note that an GNNEXPLAINER can figure out the most influential components for v_i's prediction. In such case, these adversarial edges made by attackers are highly likely to be chosen by the GNNEXPLAINER and then detected by the inspector or system designers (cf. Figure 1 (c)). Therefore, adversarial users / edges can be excluded from the GNN model to improve the system's safety. In fact, this hypothesis is verified via empirical studies, where the details can be found in Section III-C. Please note that similar observations about model explanation techniques can be found on non graph-structured data in other domains [25]–[28].

Motivated by the fact that GNNEXPLAINER can act as an inspection tool for graph adversarial perturbations, we further investigate a new problem: *Whether a graph neural network and its* GNNEXPLAINER *can be jointly attacked by modifying graphs with malicious desires?* For example, as shown in Figure 1 (e-f), when an attacker (Attacker 2) inserts an adversarial edge (v_1, v_{11}) to mislead the model's prediction, he can also successfully evade the detection by

*Equal contribution

2375-026X/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDE55515.2023.00056

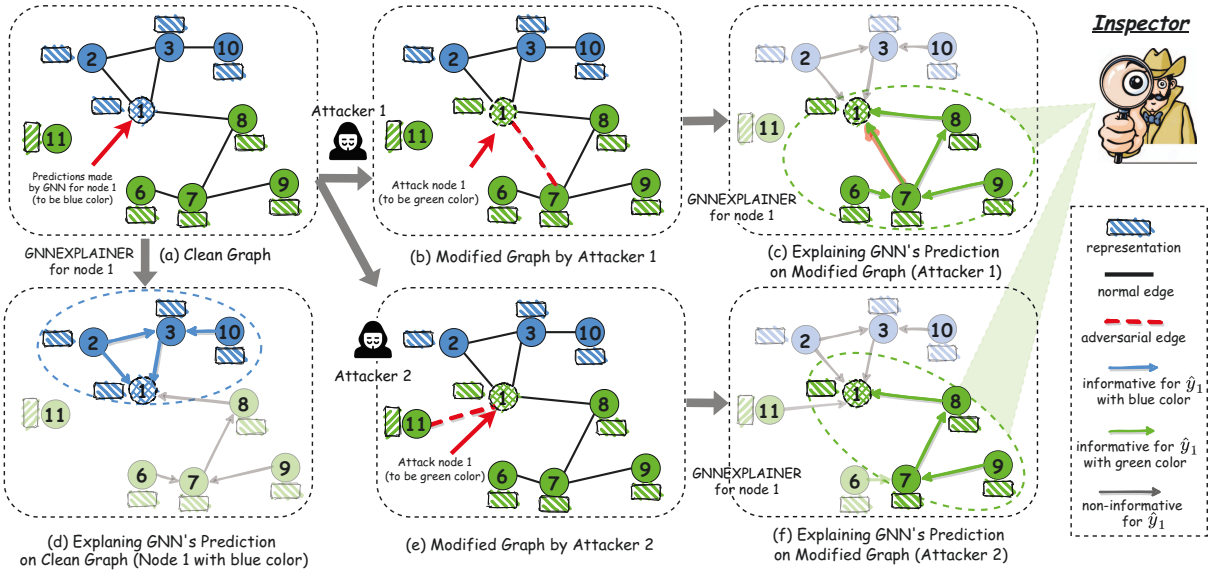


Figure 1: Adversarial attacks and the explanations (e.g., GNNEXPLAINER [21] and PGExplainer [22]) for prediction made by a GNN model. Some edges form important message-passing pathways (in dotted circle with blue/green color) while others do not (in translucent). The attacker 1 can successfully change the GNN's prediction to green color on target node v_1, while the added adversarial edge (v_1, v_7) is included into a subgraph generated by GNNEXPLAINER. And the attacker 2 can attack the GNN model, as well as fool the GNNEXPLAINER, where the added adversarial edge (v_1, v_{11}) is not included into a subgraph and successfully evade the detection by a inspector.

a GNNEXPLAINER. Therefore, the attacker becomes more dangerous because he can even misguide these inspection approaches, leading to more severe safety issues for GNNs. However, jointly fooling graph neural networks and the GNNEXPLAINER faces tremendous challenges. The biggest obstacle is that the goals of adversarial attack and bypassing the GNNEXPLAINER essentially contradict with each other. After all, the adversarial perturbations on graph are highly correlated with the target label because it is the perturbations that cause such malicious prediction. Therefore, the GNNEXPLAINER has a high chance to detect these perturbations based on the mutual information [21]. Moreover, although there exist extensive works on adversarial attack for GNNs model [14], [15], [17], [29], the attack on GNNEXPLAINER is a new problem and it calls for a new solution.

To address aforementioned challenges, we propose a novel attack framework (**GEAttack**) for graphs, where the attacker can successfully fool the GNN model and misguide the inspection from GNNEXPLAINER simultaneously. Our major contributions can be summarized as follows:

- We discover that GNNEXPLAINER tools can be utilized to understand and inspect the problematic outputs from adversarially perturbed graph data, which paves a way to improve the safety of GNNs.
- We propose a new attacking problem where we seek to jointly attack a graph neural network method and its explanations. Our proposed algorithm **GEAttack** successfully resolves the dilemma between attacking GNN and its explanations by exploiting their vulnerabilities simultaneously. To the best of our knowledge, we are the very first to study this problem that reveals more severe safety concerns, and investigate interactions between adversarial attacks and explainability for the trustworthiness of GNNs model.
- We conduct comprehensive experiments on three real-world graph datasets to show the effectiveness of the proposed model.

II. Related Work

Our work is related to two general lines of works adversarial attacks on GNNs models and their interpretation methods.

A. Adversarial Attacks on Graphs

GNNs generalize deep neural networks to graph structured data and become powerful tools for graph representations learning [1], [30], [31]. However, recent studies have demonstrated that GNNs suffer from the same issue as other deep neural networks: they are highly vulnerable to adversarial attacks [13], [32], [33]. Specifically, attackers can generate graph adversarial perturbations by manipulating the graph structure or node features to deceive the GNNs model to make incorrect predictions [13]–[15]. *Nettack* [14] is one of the first methods that perturbs the graph structure data by preserving degree distribution and feature co-occurrence to perform attack on GNN model [1]. *RL-S2V* [15] is the first work to employ reinforcement learning to generate adversarial perturbations on graph data. *NIPA* [34] also proposes a deep reinforcement learning based method to perform fake node injection attack on

graph by simulating the attack process and sequentially adds the adversarial edges and designs labels for the injected fake nodes. *IG-Attack* [17] introduces an integrated gradients based attack method to accurately reflect the effect of perturbing certain features or edges on graph data. *Metattack* [35] is proposed to globally perturb the graph based on meta-learning. In our work, we first claim that GNNEXPLAINER tools can serve as an alternative way to improve the GNN model's safety, by people (such as system inspectors or designers) doing inspections on the problematic prediction outcomes of GNN models, and then locating the potential adversarial perturbations in the graph.

B. Interpretation for Graph Neural Networks

The explanation techniques of deep models aim to study the underlying relationships behind the predictions of deep models, and provide human-intelligible explanations, which can make the deep models more trustable [36]. Some recent efforts have been made to explain the deep models for image and text data [36]–[39]. However, the explainability of graph neural networks models on graph structured data are less explored, which is critical for understanding deep graph neural networks [21]–[24], [40], [41]. In particular, as one of the first methods to interpret GNN, GNNEXPLAINER [21] maximizes the mutual information between the distribution of possible subgraphs and the GNN's prediction to find the subgraph that is most influential for the prediction. PGExplainer [22] is proposed to generate an explanation for each instance with a global understanding of the target GNN model in an inductive setting. To capture more local information around the given node being explained, GraphLIME [24] is proposed to utilize predicted labels from the node and its neighbors. Meanwhile, in order to investigate what input patterns can result in a certain prediction, XGNN [23] is proposed to train a graph generator to find graph patterns to maximize a certain prediction of the target GNN model by formulating the graph generation process as reinforcement learning problem. The improved interpretability is believed to offer a sense of security by involving human in the decision-making process [23], [36]. However, given its data-driven nature, the interpretability itself is potentially susceptible to malicious manipulations [13], [42], [43]. In this work, our goal is to fool a GNN model as well as its interpretation methods. Note that there are other efforts devoted to connecting these two topics by attacking interpretation method on non graph-structured data [42]–[45]. To the best of our knowledge, this is the very first effort to investigate interactions between adversarial attacks and explainability by attacking both GNNs model and its explanations on graph-structured data for trustworthy GNNs [33], [46]–[48].

III. PRELIMINARIES

In this section, we investigate the potential of GNNEXPLAINER [21] to detect adversarial attacks through empirical study. Before that, we first introduce key notations and concepts used in this work.

A. Graph Neural Networks

Formally, let $G = (V, E)$ denote a graph where $V = \{v_1, ..., v_n\}$ is the set of n nodes and $E = \{e_1, ..., e_k\}$ is the edge set. We use $\mathbf{A} \in \{0, 1\}^{n \times n}$ to indicate the adjacency matrix of G, where the i, j-th element A_{ij} is 1 if node v_i and node v_j are connected in G, and 0 otherwise. We also use $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n] \in \mathbb{R}^{n \times d}$ to represent the node feature matrix, where $\mathbf{x}_i$ is the d-dimensional feature vector of the node v_i. Here we use $G = (\mathbf{A}, \mathbf{X})$ to represent the graph data. Without loss of generality, given a graph $G = (\mathbf{A}, \mathbf{X})$, we consider the problem of node classification task for GNNs to learn a function $f_\theta : V_L \to Y_L$ that maps the a part of nodes $V_L = \{v_1, v_2, ..., v_l\}$ to their corresponding labels $Y_L = \{y_1, y_2, ..., y_l\}$ [1]. The objective function of GNNs can be defined as,

$$\min_{\theta} \mathcal{L}_{\text{GNN}}(f_\theta(\mathbf{A}, \mathbf{X})) := \sum_{v_i \in V_L} \ell\left(f_\theta(\mathbf{A}, \mathbf{X})_{v_i}, y_i\right) \tag{1}$$
$$= -\sum_{v_i \in V_L} \sum_{c=1}^{C} \mathbb{I}[y_i = c] \ln(f_\theta(\hat{\mathbf{A}}, \mathbf{X})^c_{v_i})$$

where θ is the parameters of f_θ. $f_\theta(\mathbf{A}, \mathbf{X})^c_{v_i}$ denotes the c-th softmax output of node v_i and C is the number of class. y_i is the true label of node v_i and $\ell(\cdot, \cdot)$ is the cross-entropy loss function. In this work, we adopt a two-layer GCN model [1] with $\theta = (\mathbf{W}_1, \mathbf{W}_2)$ as

$$f_\theta(\mathbf{A}, \mathbf{X}) = \text{softmax}(\tilde{\mathbf{A}}\, \sigma(\tilde{\mathbf{A}}\, \mathbf{X}\, \mathbf{W}_1)\, \mathbf{W}_2) \tag{2}$$

where $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}(\mathbf{A} + \mathbf{I})\tilde{\mathbf{D}}^{-1/2}$ and $\tilde{\mathbf{D}}$ is the diagonal matrix of $\mathbf{A} + \mathbf{I}$ with $\tilde{\mathbf{D}}_{ii} = 1 + \sum_j \mathbf{A}_{ij}$. σ is the activation function such as ReLU.

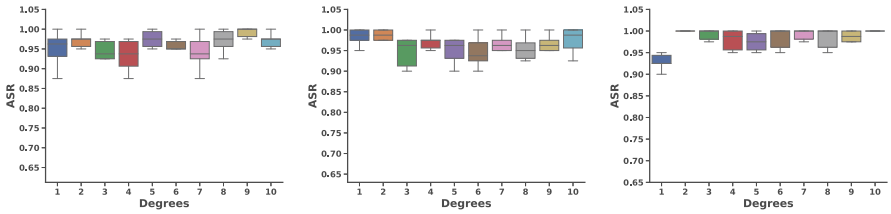
B. GNNEXPLAINER

In order to explain why a GNN model f_θ predicts a given node v_i's label as Y, the GNNEXPLAINER acts to provide a local interpretation $G_S = (\mathbf{A}_S, \mathbf{X}_S)$ by highlighting the relevant features $\mathbf{X}_S$ and the relevant subgraph structure $\mathbf{A}_S$ for its prediction [21]. To achieve this goal, it formalizes the problem as an optimization task to find the optimal explanation (G_S), which has the maximum *Mutual Information* (MI) with the GNN's prediction Y [21]:

$$\max_{(\mathbf{A}_S, \mathbf{X}_S)} MI\left(Y, (\mathbf{A}_S, \mathbf{X}_S)\right) := H(Y) - H(Y|\mathbf{A} = \mathbf{A}_S, \mathbf{X} = \mathbf{X}_S) \tag{3}$$

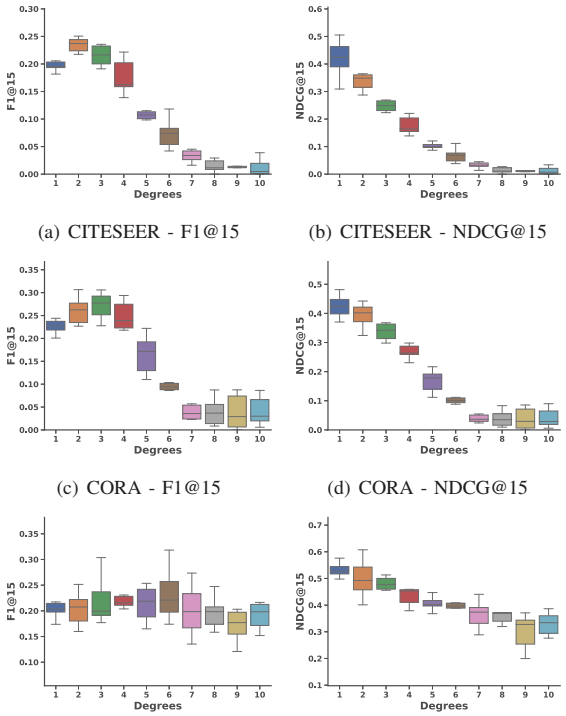
As the GNN model f_θ is fixed, the entropy term $H(Y)$ is also fixed. In other words, the explanation for v_i's prediction $\hat{y}_i$ is a subgraph $\mathbf{A}_S$ and the associated feature $\mathbf{X}_S$ that minimize the uncertainty of the GNN f_θ when the neural message-passing is limited to G_S as follows:

$$\begin{aligned} &\max_{(\mathbf{A}_S, \mathbf{X}_S)} MI\left(Y, (\mathbf{A}_S, \mathbf{X}_S)\right) \\ \to &\min_{(\mathbf{A}_S, \mathbf{X}_S)} H(Y|\mathbf{A} = \mathbf{A}_S, \mathbf{X} = \mathbf{X}_S) \\ \approx &\min_{(\mathbf{A}_S, \mathbf{X}_S)} -\sum_{c=1}^{C} \mathbb{I}[\hat{y}_i = c] \ln f_\theta(\mathbf{A}_S, \mathbf{X}_S)^c_{v_i} \end{aligned} \tag{4}$$



(a) CITESEER (b) CORA (c) ACM

Figure 2: Results of Attack Success Rate (ASR) under Nettack method on CITESEER, CORA, and ACM datasets.



(a) CITESEER - F1@15 (b) CITESEER - NDCG@15

(c) CORA - F1@15 (d) CORA - NDCG@15

(e) ACM - F1@15 (f) ACM - NDCG@15

Figure 3: Results of detecting the adversarial edges via GNNEXPLAINER under Nettack method on CITESEER, CORA, and ACM datasets.

Experimentally, the objective function of GNNEXPLAINER can be optimized to learn adjacency mask matrix $\mathbf{M}_A$ and feature selection mask matrix $\mathbf{M}_F$ in the following manner [21]:

$$\min_{(\mathbf{M}_A,\mathbf{M}_F)} \mathcal{L}_{\text{Explainer}}(f_\theta, \mathbf{A}, \mathbf{M}_A, \mathbf{M}_F, v_i, \hat{y}_i) := -\sum_{c=1}^{C} \mathbb{I}[\hat{y}_i = c] \ln f_\theta(\mathbf{A} \odot \sigma(\mathbf{M}_A), \mathbf{X} \odot \sigma(\mathbf{M}_F))^c_{v_i} \tag{5}$$

where $\odot$ denotes element-wise multiplication, and σ is the sigmoid function that maps the mask to $[0,1]^{n\times n}$. After the optimal mask $\mathbf{M}_A$ is obtained, we can compute $\mathbf{A}_S = \mathbf{A} \odot \sigma(\mathbf{M}_A)$ and use a threshold to remove low values. Finally, top-L edges with the largest values can provide an explanation $\mathbf{A}_S$ for GNN's prediction at node v_i. The same operation $\mathbf{X}_S = \mathbf{X} \odot \sigma(\mathbf{M}_F)$ can be used to produce explanations by considering the feature information.

C. GNNEXPLAINER *as Adversarial Inspector*

In our work, we first hypothesize that if a model gives a wrong prediction to a test node v_i because of some adversarially inserted fake edges, these "adversarial edges" should make a great contribution to the model's prediction outcome. Therefore, if a GNNEXPLAINER can understand this wrong prediction outcome by figuring out the most influential edges, we are highly likely to find and locate the adversarial edges and finally exclude them from data. In this subsection, we first dispatch empirical studies to validate the above mentioned hypothesis. Note that in this work we concentrate on GNN's explanation on graph structural adversarial perturbations (for graph feature perturbations are similar, we leave it for future work). Thus, the objective function of GNNEXPLAINER for adversarial edges detection can be defined as:

$$\min_{\mathbf{M}_A} \mathcal{L}_{\text{Explainer}}(f_\theta, \mathbf{A}, \mathbf{M}_A, \mathbf{X}, v_i, \hat{y}_i) \rightarrow \max_{\mathbf{M}_A} \sum_{c=1}^{C} \mathbb{I}[\hat{y}_i = c] \ln f_\theta(\mathbf{A} \odot \sigma(\mathbf{M}_A), \mathbf{X})^c_{v_i} \tag{6}$$

where we find an optimal adjacency mask matrix $\mathbf{M}_A$, namely the influential subgraph for v_i's prediction. To check the inspection performance and verify our hypothesis, we check whether GNNEXPLAINER's output influential subgraph can detect out the adversarially inserted fake edges.

Inspection Performance. We conduct preliminary experiments on three real-world datasets (i.e., CITESEER, CORA, and ACM) under F1@15 and NDCG@15 metrics. The details of experimental settings can be found in Section VI-A. In these experiments, we choose the state-of-the-art graph attack method for GNN model, *Nettack* [14], to perturb the graph data. We choose 40 target (victim) nodes with each certain node's degrees, and then validate whether the adversarial edges can be found by GNNEXPLAINER for them. To extract the subgraph (G_S) for explanation, GNNEXPLAINER first computes the importance weights on edges via masked adjacency ($\mathbf{M}_A$),

and then uses thresholding to remove the edges with low values in graph. Finally, top L edges provide the explanation (G_S) for GNN's prediction at target node [21]. In other words, adversarial edges with higher importance weights are more likely to present at top ranks and be easily detected by people (such as system inspectors or designers). Here, we adopt the F1@15 and NDCG@15 to evaluate the detection rate on adversarial edges, where higher values of F1@15 and NDCG@15 indicate that the adversarial edges are more likely to be detected and noticeable. As shown in Figures 2 and 3, the *Nettack* attacker can perform effective attacks for nodes with different degrees on three datasets, achieving around 95% attack success rate (ASR). Meanwhile, we can observe that the detection performance via GNNEXPLAINER on these three datasets are quite high, especially for the nodes with low degree achieving around 0.4 under the NDCG@15 metric. It means that the adversarial edges are highly likely to be ranked top among all edges which contribute to the model's prediction. Our results indicate that GNNEXPLAINER has the potential to mark the adversarial edges in corrupted graph data for GNNs.

In other words, GNNEXPLAINER can generate a small subgraph (with some influential nodes) to reduce search space from millions of edges, and adversarial edges ranked top among all edges in the subgraph are likely to be inspected by domain experts. Thus, these observations indicate that GNNEXPLAINER has the potential to mark the adversarial edges in corrupted graph data for GNNs. Note that similar observations on another representative explainer (PGExplainer [22]) can be found in Section VI-C.

In addition, recent studies have shown that the model explanation techniques can be effectively used to diagnose model errors–model debugging in computer vision domain (e.g., diagnosing spurious image [28]) and NLP domain [27] (e.g., detecting gender bias in abusive language [26] and revealing model's vulnerability-adversarial example [25]). To the best of our knowledge, this work is the very first effort to investigate the capability of GNN explanations on model debugging for graph-structured data.

IV. PROBLEM STATEMENT

Given the node classification task, the attacker aims to attack a specific target node $v_i \in V_t$ by performing small perturbations on the graph $G = (\mathbf{A}, \mathbf{X})$ and obtains the corrupted graph $\hat{G} = (\hat{\mathbf{A}}, \hat{\mathbf{X}})$, such that the predicted label of the target node v_i can be manipulated [14], [17]. There are two main types of adversarial perturbations on the graph, including structure attacks to modify the adjacency matrix $\mathbf{A}$ and feature attacks to modify the feature matrix $\mathbf{X}$. Please note that we focus on the structure attacks that attackers only add fake edges to connect the target nodes with others under certain perturbation budget Δ. That is due to the fact that perturbing structure attack is more effective than modifying nodes' features [17], [49]. Note that a fixed perturbation budget Δ can be constrained as follows,

$$\|\mathbf{E}'\| = \|\hat{\mathbf{A}} - \mathbf{A}\|_0 \leq \Delta. \tag{7}$$

where $\mathbf{E}'$ denotes the added adversarial edges by attackers. In our work, in order to jointly attack a GNN model and its GNNEXPLAINER, we design a new attacking method which should achieve both: (1) misleading the GNN model f_θ to give a wrong prediction $\hat{y}_i$ on node v_i; (2) misleading the explanations of the GNN model such that the added fake edges do not appear in the output subgraph $\mathbf{A}_S$ given by the GNNEXPLAINER [21]. More formally, we state our attacking objective as:

Problem: Given $G = (\mathbf{A}, \mathbf{X})$, target (victim) nodes $v_i \subseteq V_t$ and specific target label $\hat{y}_i$, the attacker aims to select adversarial edges to composite a new graph $\hat{\mathbf{A}}$ which fulfills the following two goals:

- The added adversarial edges can change the GNN's prediction to a specific target label: $\hat{y}_i = \arg\max_c f_\theta(\hat{\mathbf{A}}, \mathbf{X})^c_{v_i}$;
- The added adversarial edges will not be included in the subgraph generated by GNNEXPLAINER: $\hat{\mathbf{A}} - \mathbf{A} \notin \mathbf{A}_S$.

V. THE PROPOSED FRAMEWORK

In this section, we first introduce the basic component of graph attack via inserting adversarial edges and then propose how to bypass the detection of GNNEXPLAINER. Finally, we present the overall framework GEAttack and the detailed algorithm.

A. Graph Attack

We first formulate the problem to attack a GNN model as one optimization problem to search optimal model structure $\hat{\mathbf{A}}$ which can let the model predict the victim node v_i as wrong label $\hat{y}_i$. In particular, given a well trained GNN model f_θ on the clean input graph G, we propose to achieve the attack on target node v_i with specific target label $\hat{y}_i$ by searching $\hat{\mathbf{A}}$ which let the model have minimum loss on v_i:

$$\min_{\hat{\mathbf{A}}} \mathcal{L}_{\text{GNN}}(f_\theta(\hat{\mathbf{A}}, \mathbf{X})_{v_i}, \hat{y}_i) := -\sum_{c=1}^{C} \mathbb{I}[\hat{y}_i = c] \ln(f_\theta(\hat{\mathbf{A}}, \mathbf{X})^c_{v_i}) \tag{8}$$

Note that since we minimize the negative likelihood probability of the target label $\hat{y}_i$, the optimization of above loss will promote the prediction probability of class $\hat{y}_i$ such that the prediction is maliciously manipulated from original prediction y_i to $\hat{y}_i$. Due to the perturbation budget, we can choose adversarial edges according to the top Δ elements in $\hat{\mathbf{A}}$.

To solve this optimization problem, we desire to apply gradient-based attack methods, such as [50] to figure out the each input edge's influence to the model output. However, due to the discrete property and cascading effects of graph data [14], [29], it is not straightforward to attack a GNN model via gradient-based attack method, like FGSM in continued space in computer vision tasks [50]. To address the graph attack problem, we first relax the adjacency matrix $\mathbf{A} \in \{0, 1\}^{n \times n}$ as continuous variable $\mathbb{R}^{n \times n}$. The calculated gradient information can help us approximately find the most "adversarial" edge in the current adjacency matrix, which is the element in the gradient that has the largest negative value. Once added this

founded edge into the input graph, the model is highly likely to give a false prediction.

B. GNNEXPLAINER *Attack*

The graph attack introduced in Section V-A is usually satisfactory in terms of successful attacking rate as we will show in Section VI. However, just like existing attack methods on graph data, since the adversarial edges are highly correlated with the target prediction $\hat{y}_i$, they are most likely included in the subgraph detected by the GNNEXPLAINER and then become noticeable to the inspector or system designers. Therefore, it is highly nontrivial to achieve attacking while bypassing the detection by GNNEXPLAINER.

As introduced in Section III-B, GNNEXPLAINER aims to identify an important small subgraph G_S that influences GNN's prediction the most for making GNN's explanations. It works by minimizing $\mathcal{L}_{\text{Explainer}}$ (Eq. 6) and selecting the top-L edges in the adjacency mask matrix $\mathbf{M}_A$ with the largest values. Therefore, to bypass the detection by GNNEXPLAINER, we propose a novel GNNEXPLAINER *attack* to suppress the possibility of adversarial edges being detected as follows:

$$\min_{\hat{\mathbf{A}}} \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{M}_A^T[i,j] \cdot \mathbf{B}[i,j] \tag{9}$$

where $\mathbf{B} = \mathbf{1}\mathbf{1}^T - \mathbf{I} - \mathbf{A}$. $\mathbf{I}$ is an identity matrix, and $\mathbf{1}\mathbf{1}^T$ is all-ones matrix. $\mathbf{1}\mathbf{1}^T - \mathbf{I}$ corresponds to the fully-connected graph. When t is 0, $\mathbf{M}_A^0$ is randomly initialized; while t is larger than 0, $\mathbf{M}_A^t$ is updated as follows:

$$\mathbf{M}_A^t = \mathbf{M}_A^{t-1} - \eta \nabla_{\mathbf{M}_A^{t-1}} \mathcal{L}_{\text{Explainer}}(f_\theta, \hat{\mathbf{A}}, \mathbf{M}_A^{t-1}, \mathbf{X}, v_i, \hat{y}_i). \tag{10}$$

There are several key motivations:

- The update of $\mathbf{M}_A^t$ mimics the gradient descent step in optimizing the loss function of GNNEXPLAINER in Eq. (5) and $\mathbf{M}_A^T$ corresponds to the adjacency mask matrix after T steps of update.
- The loss term represents the total value of the adjacency mask corresponding to the edges between node v_i and its direct neighbors $\mathcal{N}(v_i)$ since we focus on direct attack. Therefore, the adversarial edges we search among those neighbors tend to have a small value in the mask matrix $\mathbf{M}_A$; Since GNNEXPLAINER only select edges with large values to construct the subgraph, there is a higher chance that adversarial edges could bypass the detection.
- The penalty on existing edges in the clean graph is excluded by matrix $\mathbf{B}$ where $\mathbf{B}[i,j] = 0$ if edge (v_i, v_j) exists in the clean graph $\mathbf{A}$. In this way, the GNNEXPLAINER is still able to include normal edges in the subgraph. In other words, the GNNEXPLAINER works normally if not being attacked.

Note that this loss function essentially accumulates and penalizes the gradient of $\mathcal{L}_{\text{Explainer}}$ with respect to $\mathbf{M}_A^t$ along the optimization path $\mathbf{M}_A^0 \to \mathbf{M}_A^1 \to \cdots \to \mathbf{M}_A^T$. Each step of the gradient has a sophisticated dependency on the optimization variable $\hat{\mathbf{A}}$ and it requires the high-order gradient computation which is supported by deep learning frameworks such as Pytorch and TensorFlow.

C. GEAttack

After introducing the graph attack and GNNEXPLAINER attack, we finally obtain the GEAttack framework as follows:

$$\min_{\hat{\mathbf{A}}} \mathcal{L}_{\text{GEAttack}} := \mathcal{L}_{\text{GNN}}(f_\theta(\hat{\mathbf{A}}, \mathbf{X})_{v_i}, \hat{y}_i) + \lambda \sum_{v_i \in \mathcal{N}(v)} \mathbf{M}_A^T[i,j] \cdot \mathbf{B}[i,j] \tag{11}$$

where $\mathbf{M}_A^0$ is randomly initialized when t is 0, and for $t > 0$, $\mathbf{M}_A^t$ can be updated as follows:

$$\mathbf{M}_A^t = \mathbf{M}_{A^{t-1}} - \eta \nabla_{\mathbf{M}_A^{t-1}} \mathcal{L}_{\text{Explainer}}(f_\theta, \hat{\mathbf{A}}, \mathbf{M}_A^{t-1}, \mathbf{X}, v_i, \hat{y}_i) \tag{12}$$

The first loss term $\mathcal{L}_{\text{GNN}}$ guides the search of adversary edge such that the prediction of node v_i is attacked; the second loss term guides the search process to bypass the detection of GNNEXPLAINER; and λ is a hyperparameter which controls the balance between these two losses. The proposed algorithm **GEAttack** can be formulated as bi-level optimization problem as shown in Algorithm 1. It majorly runs two loops:

- In the inner loop, we mimic the optimization process of GNNEXPLAINER to obtain the adjacency mask $\mathbf{M}_A^T$ by T steps of gradient descent. Note that we maintain the computation graph of these updates such that the dependency of $\mathbf{M}_A^T$ on $\hat{\mathbf{A}}$ is maintained, which facilitates the gradient computation in the outer loop;
- In the outer loop, we compute the gradient of $\mathcal{L}_{\text{GEAttack}}$ with respect to $\hat{\mathbf{A}}$. Note that this step requires the backward propagation through all gradient descent updates in the inner loop and requires high-order gradient computation which is supported by the Automatic Differentiation Package in PyTorch and TensorFlow. In each iteration, we select one adversarial edge (set $\hat{\mathbf{A}}[i,j] = 1$) according to the largest value in this gradient since this update will decrease the loss maximally, similar to the greedy coordinate descent algorithm.

Algorithm 1 GEAttack

Input: perturbation budget: Δ; step-size and update iterations of GNNEXPLAINER: η, T; target node v_i; target label $\hat{y}_i$; graph $G = (\mathbf{A}, \mathbf{X})$, and a GNN model: f_θ.
Output: the adversarial adjacency matrix $\hat{\mathbf{A}}$.
$\mathbf{B} = \mathbf{1}\mathbf{1}^T - \mathbf{I} - \mathbf{A}$, $\hat{\mathbf{A}} = \mathbf{A}$, randomly initialize $\mathbf{M}_A^0$;
for $o = 1, 2, \ldots, \Delta$ **do** // outer loop over $\hat{\mathbf{A}}$;
  for $t = 1, 2, \ldots, T$ **do** // inner loop over $\mathbf{M}_A^t$;
    compute $\mathbf{P}^t = \nabla_{\mathbf{M}_A^{t-1}} \mathcal{L}_{\text{Explainer}}(f_\theta, \hat{\mathbf{A}}, \mathbf{M}_A^{t-1}, \mathbf{X}, v_i, \hat{y}_i)$;
    gradient descent: $\mathbf{M}_A^t = \mathbf{M}_A^{t-1} - \eta \mathbf{P}^t$;
  end for
  compute the gradient w.r.t. $\hat{\mathbf{A}}$: $\mathbf{Q}^o = \nabla_{\hat{\mathbf{A}}} \mathcal{L}_{\text{GEAttack}}$;
  select the edge between node pair (v_i, v_j) with the maximum element $\mathbf{Q}^o[i,j]$ as the adversarial edge, and update $\hat{\mathbf{A}}[i,j] = 1$ and $\mathbf{B}[i,j] = 0$;
end for
Return $\hat{\mathbf{A}}$.

Table I: The statistics of the datasets by considering the Largest Connected Component (LCC).

Datasets	Nodes	Edges	Classes	Features
CITESEER	2,110	3,668	6	3,703
CORA	2,485	5,069	7	1,433
ACM	3,025	13,128	3	1,870

VI. EXPERIMENT

In this section, we conduct experiments to verify the effectiveness of our attacking model.

A. Experimental Settings

*1) **Datasets**:* We conduct experiments on three widely used benchmark datasets for node classification, including CITESEER [1], CORA [1], and ACM [51], [52]. The processed datasets can be found in the github link[1]. Following [35], we only consider the largest connected component (LCC) of each graph data. The statistics of these three datasets are presented in Table I.

- CITESEER. CITESEER is a research paper citation network with nodes representing papers and edges representing their citation relationship. The node labels are based on the paper topics and the node attributes are bag-of-words descriptions about the papers.
- CORA. CORA is also a citation network where nodes are papers and edges are the citation relationship between the papers. The node attributes are also bag-of-words descriptions about the papers. The papers are divided into seven classes.
- ACM. This network is extracted from ACM dataset where nodes represent papers with bag-of-words representations as node attributes. The existence of an edge between two nodes indicates they are from the same author. The nodes are divided into three classes.

*2) **Baselines**:* To evaluate the effectiveness of the proposed attacking method, we compare it with the state-of-the-art attack algorithms. Since the problem for jointly attacking GNN and GNNEXPLAINER in this paper is a novel task, there are very few baselines we can compare with. We select following six baselines[1], which aim to perform targeted attack on a small set of test nodes.

- **Random Attack (RNA)**: The attacker randomly adds adversarial edges to connect the target node with one from candidate nodes whose label is specific target label until reaching the perturbation budget.
- **FGA** [29], [53]: This is a gradient-based attack method which aims to find adversarial edges by calculating the gradient of model's output on the adjacency matrix. Note that this method does not consider to fool the model to specific label.
- **FGA-T**: Similar to FGA attack, FGA-T is a targeted version of FGA attack which aims to attack the target node to specific target label.
- **Nettack** [14]: This method introduces the first study of adversarial attacks on graph data by preserving important graph characteristics.
- **IG-Attack** [17]: This baseline introduces an integrated gradients method that could accurately reflect the effect of perturbing edges for adversarial attacks on graph data.
- **FGA-T&E** (Joint Attack): Another baseline based on FGA-T method, but further incorporates the desire to evade the detection GNNEXPLAINER when generating adversarial edges. We first adopt GNNEXPLAINER to generate a small subgraph (explanation). Then, we exclude the potential nodes from the subgraph when generating the adversarial edges between the target node and the potential nodes.

As most baselines are not directly applicable in target attack with specific target label, we modify the attacking operations accordingly, such as modifying the same loss function, or constraining adversarial edges connecting with nodes who have the specific target label.

*3) **Evaluation Metrics**:* We evaluate the effectiveness of different attacking methods from two perspectives. The first are **Attack Success Rate (ASR)** [54] and **Attack Success Rate with Target label (ASR-T)**, which are the ratio of the successfully attacked nodes among all target nodes to any wrong label and specific (incorrect) target label.

In our preliminary experiments Section III-C, we have demonstrated that GNNEXPLAINER can act as an inspector for adversarial edges. Therefore, another type of evaluation metrics is the popular accuracy metrics for detection rate [55]: **Precision@K, Recall@K, F1@K**, and Normalized Discounted Cumulative Gain (**NDCG@K**). The first three metrics (Precision@K, Recall@K, F1@K) focus on how many adversarial edges are included in the Top-K list of subgraph generated via GNNEXPLAINER, while the last metric (NDCG@K) accounts for the ranked position of adversarial edges in the Top-K list. We set K as 5, 10, and 15. Note that adversarial edges with higher importance weights in masked adjacency ($\mathbf{M}_A$) are more likely to present at top ranks and be easily detected by people (such as system inspectors or designers). Hence, higher values of these metrics (Precision@K, Recall@K, F1@K, and NDCG@K) indicate that the adversarial edges are more likely to be detected and noticeable. Meanwhile, lower values of them also indicate adversarial edges are less likely to include into the subgraph (G_S) and more unnoticeable to human, where the GNNEXPLAINER can be attacked. Without any specific mention, we adopt the default parameter setting of GNNEXPLAINER in the author's implementation[3], and the size of subgraph L is set to 20. Note that we further analyse the impact of GNNEXPLAINER inspector on adversarial edge based on the various size of subgraph L at Section VI-E1.

*4) **Attacker Settings**:* In this experiments, we conduct the targeted attack by selecting a set of target nodes under white-box setting and only consider the adding fake edges when doing adversarial perturbations. Meanwhile, we conduct the

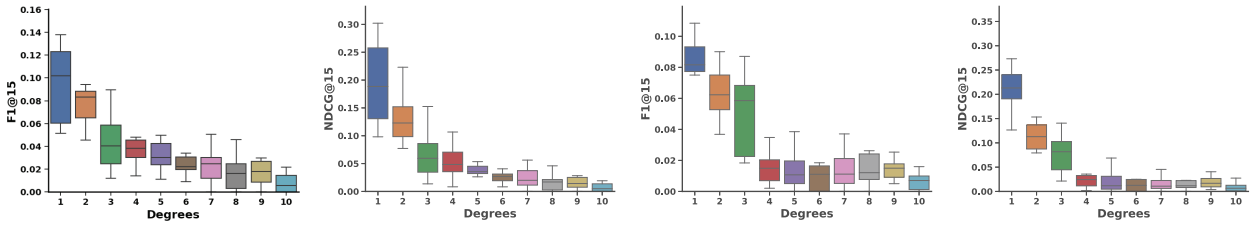
[1]https://github.com/DSE-MSU/DeepRobust/tree/master/deeprobust/graph

[3]https://github.com/RexYing/gnn-model-explainer

Table II: Results with standard deviations (± std) on three datasets using different attack algorithms. The higher ASR and ASR-T is, the stronger the attacker to attack target node is. The lower Precision/Recall/F1/NDCG@K is, the more unnoticeable the adversarial edges to be generated by attacker is.

Datasets	Metrics (%)	FGA[2]	RNA	FGA-T	Nettack	IG-Attack	FGA-T&E	GEAttack
CITERSEER	**ASR**	86.79±0.08	55.52±0.08	99.56±0.01	99.11±0.01	91.54±0.05	98.74±0.02	**100±0.00**
	ASR-T	-	54.27±0.10	99.56±0.01	99.11±0.01	91.54±0.05	98.74±0.02	**100±0.00**
	Precision@5	17.37±0.03	12.47±0.03	17.54±0.04	12.00±0.02	13.45±0.02	17.39±0.03	**10.59±0.03**
	Precision@10	15.39±0.02	11.4±0.02	15.55±0.02	11.37±0.02	11.78±0.03	15.51±0.02	**10.13±0.03**
	Precision@15	13.45±0.01	9.96±0.01	13.44±0.02	10.21±0.01	10.21±0.01	13.31±0.01	**9.87±0.02**
	Recall@5	48.49±0.07	37.83±0.07	48.97±0.07	38.22±0.07	41.76±0.04	48.58±0.07	**34.55±0.08**
	Recall@10	66.99±0.06	55.34±0.07	67.33±0.06	57.08±0.06	57.88±0.07	67.27±0.06	**51.93±0.08**
	Recall@15	74.55±0.05	**63.80±0.05**	74.55±0.05	66.48±0.06	65.73±0.04	74.28±0.05	64.05±0.07
	F1@5	24.70±0.04	17.98±0.04	24.96±0.05	17.73±0.04	19.71±0.03	24.74±0.04	**15.77±0.04**
	F1@10	24.05±0.03	18.11±0.03	24.25±0.03	18.40±0.03	18.87±0.04	24.21±0.03	**16.45±0.04**
	F1@15	21.65±0.02	**16.44±0.02**	21.64±0.02	17.08±0.02	16.96±0.02	21.47±0.02	16.49±0.03
	NDCG@5	34.29±0.06	26.31±0.06	33.89±0.06	25.91±0.06	29.36±0.05	34.29±0.06	**23.40±0.07**
	NDCG@10	43.36±0.05	34.94±0.05	42.96±0.05	34.37±0.05	36.84±0.05	43.49±0.05	**31.06±0.06**
	NDCG@15	47.18±0.04	39.21±0.04	46.60±0.04	38.45±0.05	40.26±0.04	47.02±0.05	**36.11±0.05**
CORA	**ASR**	90.54±0.05	62.97±0.10	**100±0.00**	**100±0.00**	90.17±0.07	99.79±0.01	**100±0.00**
	ASR-T	-	62.58±0.10	**100±0.00**	**100±0.00**	90.17±0.07	99.79±0.01	**100±0.00**
	Precision@5	15.16±0.03	10.32±0.03	14.81±0.03	10.08±0.03	10.83±0.03	15.40±0.03	**8.30±0.03**
	Precision@10	18.86±0.02	**10.98±0.02**	19.01±0.02	13.13±0.02	14.94±0.04	18.98±0.02	11.71±0.02
	Precision@15	16.02±0.01	**10.47±0.01**	16.08±0.01	12.78±0.01	13.47±0.03	15.95±0.01	12.21±0.01
	Recall@5	35.45±0.08	26.16±0.07	35.05±0.08	27.03±0.07	28.55±0.06	35.76±0.08	**23.97±0.06**
	Recall@10	63.49±0.06	**44.23±0.08**	63.91±0.06	50.41±0.07	55.50±0.09	63.87±0.07	48.79±0.09
	Recall@15	72.65±0.05	**55.40±0.07**	72.75±0.05	63.83±0.06	67.66±0.04	72.45±0.05	65.03±0.06
	F1@5	20.42±0.05	14.09±0.04	20.05±0.04	14.16±0.04	15.05±0.04	20.70±0.04	**11.92±0.04**
	F1@10	28.00±0.03	**16.96±0.03**	28.22±0.03	20.12±0.03	22.71±0.06	28.18±0.03	18.39±0.03
	F1@15	25.30±0.02	**17.00±0.02**	25.38±0.02	20.64±0.02	21.79±0.04	25.21±0.02	20.06±0.02
	NDCG@5	24.61±0.06	19.4±0.05	24.63±0.06	19.24±0.06	19.77±0.05	25.22±0.06	**17.36±0.04**
	NDCG@10	38.75±0.05	**28.23±0.06**	39.14±0.05	30.43±0.05	32.61±0.07	39.33±0.05	28.62±0.05
	NDCG@15	43.15±0.04	**34.16±0.05**	43.41±0.04	36.47±0.04	38.05±0.05	43.46±0.04	35.60±0.03
ACM	**ASR**	67.50±0.07	63.66±0.13	**100±0.00**	98.00±0.03	98.82±0.02	**100±0.00**	**100±0.00**
	ASR-T	-	63.66±0.13	**100±0.00**	98.00±0.03	98.82±0.02	**100±0.00**	**100±0.00**
	Precision@5	18.96±0.06	14.23±0.05	19.28±0.06	20.03±0.03	18.85±0.08	18.65±0.06	**7.03±0.03**
	Precision@10	14.57±0.06	10.18±0.03	14.86±0.05	16.58±0.03	14.32±0.05	14.35±0.06	**7.90±0.00**
	Precision@15	11.57±0.05	**9.26±0.01**	11.88±0.05	12.98±0.03	11.69±0.05	11.31±0.05	9.61±0.02
	Recall@5	31.34±0.09	22.62±0.05	31.40±0.09	32.69±0.08	34.26±0.12	31.18±0.09	**20.89±0.07**
	Recall@10	36.57±0.11	**28.67±0.06**	36.67±0.11	40.78±0.09	40.65±0.12	36.38±0.11	30.09±0.05
	Recall@15	38.21±0.12	**34.05±0.05**	38.34±0.12	43.67±0.09	44.49±0.14	37.90±0.12	38.08±0.08
	F1@5	19.46±0.06	14.12±0.03	19.54±0.06	21.13±0.04	21.53±0.09	19.25±0.06	**10.23±0.04**
	F1@10	16.77±0.06	12.70±0.03	16.92±0.06	20.30±0.04	18.74±0.07	16.58±0.06	**11.73±0.01**
	F1@15	14.16±0.05	**12.75±0.02**	14.35±0.05	17.61±0.04	16.61±0.07	13.91±0.05	14.03±0.03
	NDCG@5	34.46±0.13	27.83±0.09	34.13±0.12	36.61±0.09	32.94±0.12	34.09±0.12	**14.27±0.05**
	NDCG@10	37.40±0.13	32.58±0.10	37.12±0.13	44.49±0.09	37.97±0.12	37.01±0.12	**19.34±0.04**
	NDCG@15	38.58±0.14	36.68±0.10	38.17±0.13	46.90±0.09	41.23±0.13	38.07±0.13	**24.43±0.06**

[2] FGA cannot evaluate ASR-T metric where the specific target label are not available.



(a) CITESEER - F1 (b) CITESEER - NDCG (c) CORA - F1 (d) CORA - NDCG

Figure 4: Results of detecting the adversarial edges via PGExplainer Inspector under Nettack on CITESEER and CORA datasets.

evasion attack, where attacking happens after the GNN model is trained or in the test phase. The model is fixed, and the attacker cannot change the model parameter or structure. The perturbation budget Δ of each target node is set to its degree. Following the setting of IG-Attack [17], we select in total 40 victim target nodes which contain the 10 nodes with top scores, 10 nodes with the lowest scores, and the remaining nodes are randomly selected. Note that we conducted direct attacks on the edges directly connected to the target node with specific target label. To obtain specific target label for each node, we first perform attack to fool the target nodes via basic FGA attack method. The changed label for each target node then is set to be specific target label if success. Note that we use these successfully attacked nodes to evaluate the final attacking performance.

*5) **Parameter Settings**:* For training the GNN model in each graph, we randomly choose 10% of nodes for training, 10% of nodes for validation and the remaining 80% of nodes for testing [49]. For each experiment, we report the average performance of 5 runs. The hyper-parameters of all the models are tuned based on the loss and accuracy on validation set. For the λ, we test the value of $\{0.001, 0.01, 1, 10, 20, 50, 100, 200, 500\}$. The value of step-size η was searched in $\{0.001, 0.01, 0.1, 0.5, 1\}$. The value of T was searched range from 1 to 10. Without any specific mention, we adopt the default parameter setting in the author's implementation. We implemented the proposed method on the basis of PyTorch.

B. Attack Performance Comparison

We first evaluate how the attack methods perform and whether the adversarial edges can be detected by GNNEXPLAINER. The results are demonstrated in Table II. According to the results, we have the following observations.

*1) **Attacking GNN Model**:* Our proposed attacker GEAttack works consistently comparable to or outperform other strong GNN attacking methods. In all three datasets (CITESEER, CORA, and ACM), our proposed attacker GEAttack achieves around 100% attacking success rate when doing adversarial attacks with and without target labels (ASR-T & ASR). It suggests that GEAttack can achieve similar attacking power compared to other strongest GNN attackers such as FGA-T and Nettack, while also outperforming other attackers such as IG-Attack and random attack (RNA).

*2) **Attacking GNNEXPLAINER**:* Our proposed attacker GEAttack consistently outperforms other methods when attacking the GNNEXPLAINER, except for the RNA method. In other words, our proposed GEAttack is much harder to be detected by GNNEXPLAINER than all other attacking methods, only except for the RNA attacker. Note that the RNA method is the strongest baseline with regard to evade the detection of GNNEXPLAINER, while having the worst performance on attacking the GNN model with the ASR-T & ASR metrics. That is due to the fact that RNA attacker randomly adds edges to the target node, so the added edge is expected to have low influence to model's prediction. From our experimental results, we could see, when excluding RNA attacker, our proposed GEAttack is the most strongest attacker for GNNEXPLAINER. Compared to the most successful GNN attackers (Nettack and FGA-T), GEAttack can let the GNNEXPLAINER have much lower Precision, Recall and F1 score, which suggests that the GNNEXPLAINER has much lower power to detect adversarial perturbations from GEAttack. For another baseline method, FGA-T&E which also tries to evade the GNNEXPLAINER (by considering only attack the edges that are not selected by GNNEXPLAINER), the GNNEXPLAINER detector still has high chance to figure out the adversarial perturbations. In conclusion, our proposed GEAttack can have good performance for attacking GNN models, which is comparable to other strongest attackers. At the same time, it is much harder to be detected by GNNEXPLAINER. The experimental results can verify that our proposed method can jointly attack both a GNN model and its explanations (GNNEXPLAINER).

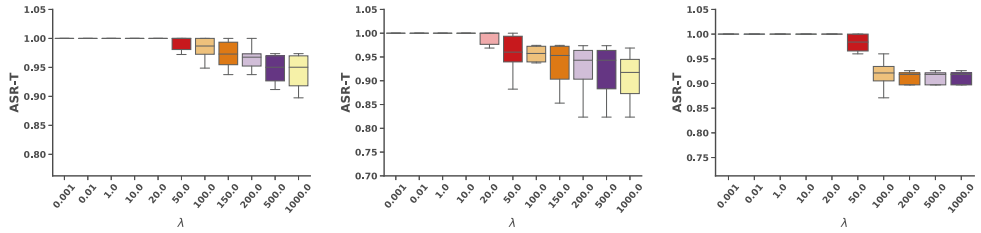
C. Jointly attacking GNNs and PGExplainer

In this section, in order to evaluate the effectiveness of our proposed attacking method on both GNNs and its explanations, we apply our proposed method to another representative explainer for the GNNs model (PGExplainer [22]), which adopts a deep model to parameterize the generation process of explanations in the inductive setting. As shown in Figure 4, we first conducted empirical studies to validate that PGExplainer has the potential to mark the adversarial edges in corrupted graph data for GNNs over CITESEER and CORA datasets, which has similar observations on GNNEXPLAINER in Section III.

To perform jointly attacking, we adopt a similar manner to the search of adversarial edges via the gradient computation of PGExplainer. Table III shows the overall attack performance comparison on CITESEER dataset. We do not show the results on CORA and ACM datasets since similar observations can be made. In general, we find that our proposed attacker GEAttack achieves the highest attacking success rate (ASR/ASR-T) compared with baselines. Meanwhile, as for attacking PGExplainer, our proposed attacker GEAttack also consistently outperforms other methods under Precision/Recall/F1/NDCG metrics when attacking the PGExplainer, except for the RNA method. Note that as RNA attacker randomly adds edges to the target node for jointly attacking, these adversarial edges might have a low influence to the model's prediction and could easily lead to evade the detection from Explainer, while making it difficult to attack the GNN model under the ASR-T & ASR metrics. These observations demonstrate that both GNNs model and its explanations are vulnerable to adversarial attacks, and our proposed method can jointly attack both a GNN model and its explanations.

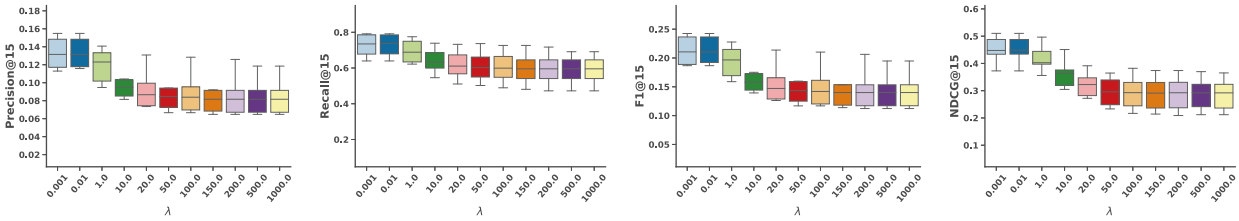
D. Balancing the Graph Attack and GNNEXPLAINER *Attack - λ*

In the previous subsection, we have demonstrated the effectiveness of the proposed method. In this subsection, we study the effect of model components between Graph Attack



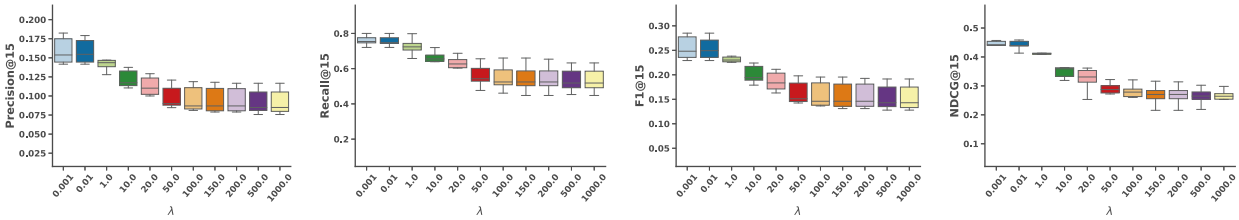
(a) CITESEER - ASR-T (b) CORA - ASR-T (c) ACM - ASR-T

Figure 5: Effect of λ under Attack Success Rate with Target label (ASR-T) on CITESEER, CORA, and ACM datasets.



(a) CITESEER - Precision@15 (b) CITESEER - Recall@15 (c) CITESEER - F1@15 (d) CITESEER - NDCG@15

Figure 6: Effect of λ under detection rate (Precision/Recall/F1/NDCG@15) on CITESEER dataset.

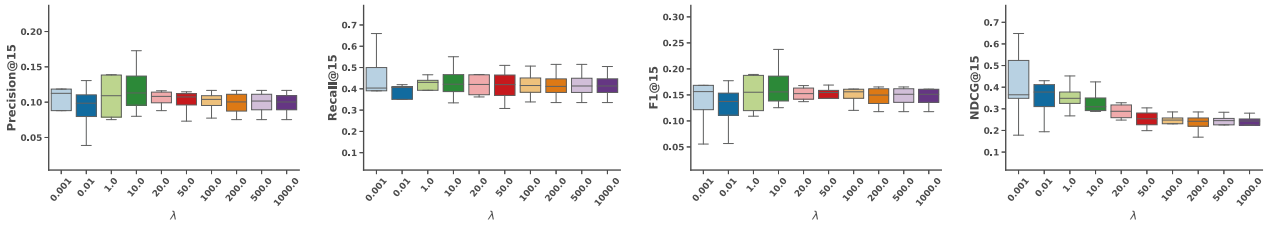


(a) CORA - Precision@15 (b) CORA - Recall@15 (c) CORA - F1@15 (d) CORA - NDCG@15

Figure 7: Effect of λ under detection rate (Precision/Recall/F1/NDCG@15) on CORA dataset.

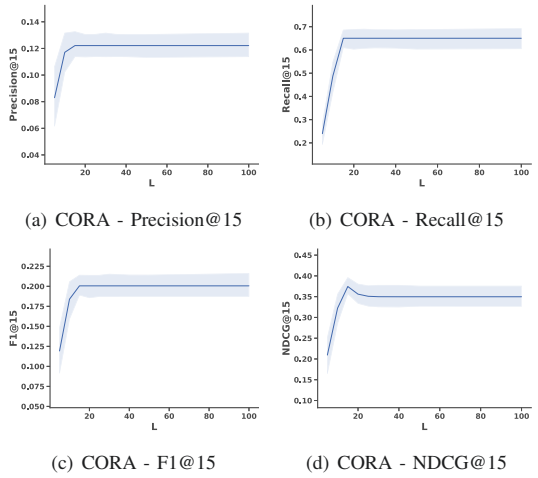
Table III: Results with standard deviations (±std) on CITESEER dataset using different attacking algorithms.

Metrics (%)	FGA	RNA	FGA-T	Nettack	IG-Attack	FGA-T&E	GEAttack
ASR	88.89±0.06	55.19±0.04	99.24±0.01	97.20±0.18	98.93±0.01	98.76±0.01	**99.34±0.03**
ASR-T	-	51.74±0.06	99.24±0.01	96.91±0.11	98.42±0.02	98.81±0.01	**99.34±0.03**
Precion@5	6.30±0.03	4.53±0.03	6.62±0.04	7.05±0.05	7.89±0.02	6.71±0.04	**4.59±0.02**
Precion@10	7.04±0.03	4.12±0.02	6.42±0.03	6.51±0.04	7.00±0.02	5.89±0.03	**4.82±0.02**
Precion@15	6.77±0.03	4.10±0.02	6.47±0.02	6.45±0.03	6.52±0.02	5.66±0.02	**4.65±0.01**
Recall@5	20.5±0.11	14.05±0.08	21.71±0.13	22.6±0.12	27.65±0.06	19.93±0.14	**14.16±0.09**
Recall@10	33.7±0.15	21.07±0.09	31.90±0.15	32.89±0.14	37.62±0.08	29.02±0.16	**23.05±0.10**
Recall@15	40.39±0.14	27.37±0.12	39.71±0.16	40.50±0.16	43.73±0.10	35.14±0.16	**28.60±0.11**
F1@5	9.16±0.05	6.44±0.04	9.65±0.06	10.27±0.06	11.91±0.03	9.38±0.06	**6.56±0.03**
F1@10	11.09±0.05	6.51±0.03	10.13±0.05	10.36±0.05	11.07±0.03	9.22±0.05	**7.42±0.03**
F1@15	11.07±0.04	6.79±0.03	10.61±0.04	10.65±0.05	10.72±0.03	9.19±0.04	**7.47±0.02**
NDCG@5	14.59±0.08	9.57±0.06	15.54±0.10	15.84±0.08	20.24±0.04	13.56±0.10	**10.45±0.07**
NDCG@10	20.07±0.09	12.55±0.06	19.78±0.11	20.07±0.09	24.47±0.05	17.15±0.11	**14.24±0.07**
NDCG@15	22.65±0.09	14.85±0.07	22.87±0.11	23.07±0.09	26.76±0.06	19.38±0.11	**16.45±0.07**



(a) ACM - Precision@15 (b) ACM - Recall@15 (c) ACM - F1@15 (d) ACM - NDCG@15

Figure 8: Effect of λ under detection rate (Precision/Recall/F1/NDCG@15) on ACM dataset.



(a) CORA - Precision@15 (b) CORA - Recall@15

(c) CORA - F1@15 (d) CORA - NDCG@15

Figure 9: Effect of size of subgraph L under detection rate (Precision/Recall/F1/NDCG@15) on CORA dataset.

and GNNEXPLAINER Attack, which is controlled by λ. When λ is close to 0, GEAttack is degraded to Graph Attack model, while it focuses on GNNEXPLAINER Attack for larger values of λ.

The ASR-T performance change of GEAttack on CITESEER, CORA, and ACM datasets is illustrated in Figure 5. As we can see from figures, the ASR-T of GEAttack can maintain 100% successfully attacked nodes until 50 and 20 for CITESEER (ACM) and CORA datasets, respectively. However, larger values of λ can greatly hurt the ASR-T performance. For instance, ASR-T performance of GEAttack can reduce to 95% when λ is set to 50. Moreover, for the detection performance on the GNNEXPLAINER, the Precision/Recall/F1/NDCG@15 performance changes of GEAttack on CITESEER, CORA, and ACM datasets are illustrated in Figure 6, 7 and 8, respectively. From the figures, we first observe that when the value of λ becomes large, the detection rate on CITESEER and CORA datasets consistently has the same trend under Precision/Recall/F1/NDCG@15 metrics. In addition, the detection ratio maintains stable when the value of λ is larger than 50. This observation suggests that a larger value of λ is more likely to encourage GEAttack for selecting the adversarial edges as more unnoticeable as possible. Note that more results regarding the effect of λ are shown in Figure 11 and 12.

To summarize, larger values of λ can hurt attack Graph Attack, while benefiting to GNNEXPLAINER Attack and vise versa. These observations demonstrate that there may indeed exist the trade-off relation between attacking GNN model and attacking the GNNEXPLAINER. However, selecting a proper λ can facilitate us to achieve good attacking performance for the two adversarial goals simultaneously.

E. Parameter Analysis

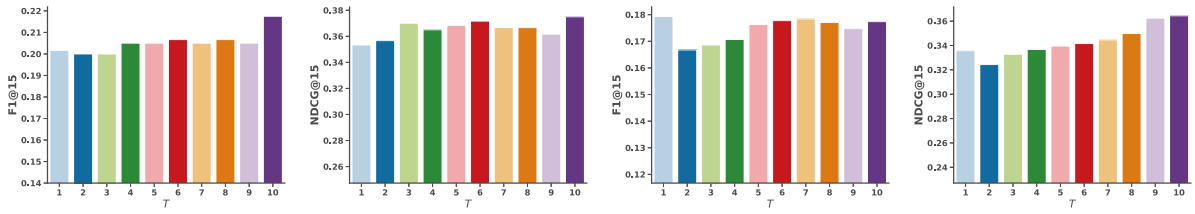
In this subsection, we study effect of model hyper-parameters for understanding the proposed method, including the size of subgraph L and the number of update iterations T.

*1) **Effect of Subgraph Size** L:* In this subsection, we further study the impact of GNNEXPLAINER inspector for adversarial edges based on the size of subgraph L. Figure 9 shows the detection rate of GEAttack with varied size of subgraph L. As we can see, when the size of subgraph increases, the performance tends to increase first. And GEAttack can not keep increasing when the size of subgraph is larger than around 20.

*2) **Effect of the Number of Update Iterations** T:* In this subsection, we explore the sensitivity of hyper-parameter T for $GEAttack$. T is the number step of updating GNNEXPLAINER, which may influence the learning of $\mathbf{M}_A^t$. The results are given on Figure 10 on CORA and ACM datasets. We do not show the results under attack success rate (ASR-T) as the performance almost achieves 100% and do not change too much. From the figure, we can observe that our proposed GEAttack method can achieve good performance under small value of T (i.e., less than 3), which indicates that sub-optimal solution of GNNEXPLAINER can provide enough gradient signal regarding $\mathbf{M}_A^t$ to guide the selection of adversarial edges for jointly attacking graph neural networks and GNNEXPLAINER.

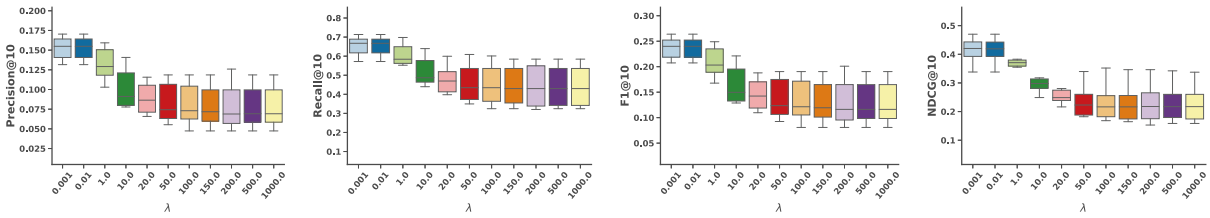
VII. CONCLUSION

In this paper, we first dispatched empirical studies to demonstrate that GNNs' explanations can act as an inspection tool and have the potential to detect the adversarial perturbations for graph data. After that, we introduced a new problem: *Whether a graph neural network and its explanations can be jointly attacked by modifying graph data with malicious desires?* To



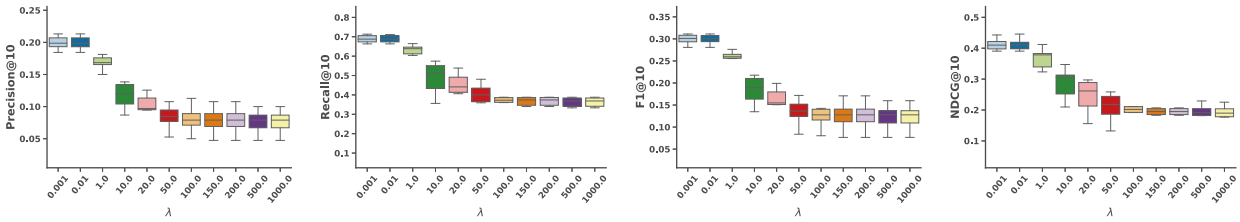
(a) CORA - F1@15 (b) CORA - NDCG@15 (c) ACM - F1@15 (d) ACM - NDCG@15

Figure 10: Effect of T under detection rate (F1/NDCG@15) on CORA and ACM datasets.



(a) CITESEER - Precision@10 (b) CITESEER - Recall@10 (c) CITESEER - F1@10 (d) CITESEER - NDCG@10

Figure 11: Effect of λ under detection rate (Precision/Recall/F1/NDCG@10) on CITESEER dataset.



(a) CORA - Precision@10 (b) CORA - Recall@10 (c) CORA - F1@10 (d) CORA - NDCG@10

Figure 12: Effect of λ under detection rate (Precision/Recall/F1/NDCG@10) on CORA dataset.

address this problem, we presented a novel attacking method (**GEAttack**) to jointly attack a graph neural network and its explanations (e.g., GNNEXPLAINER and PGExplainer). Our thorough experiments on three real-world datasets shown the superiority of the proposed GEAttack over a set of competitive baselines. Then, we furthermore performed model analysis to better understand the behavior of GEAttack.

Currently we only consider detecting adversarial edges via GNNEXPLAINER and PGExplainer [22], while there exist other adversarial perturbations, like modifying features and injecting fake nodes. In the future, we would like to extend the proposed model for performing attacks via other types of adversarial perturbations. Moreover, inspired by recent success on self-supervised learning [56], [57], we would like to investigate an adversarial self-supervised learning framework to defend against such joint attacks on both GNNs and their explanations via enhancing the robustness of the GNNs model and explanations model simultaneously.

ACKNOWLEDGMENT

The research described in this paper has been partly supported by NSFC (Project No. 62102335), an internal research fund from The Hong Kong Polytechnic University (Project No. P0036200, P0042693, and P0043302), a General Research Fund from the Hong Kong Research Grants Council (Project No. PolyU 15200021 and PolyU 15207322), and Hong Kong Research Grant Council under RIF R5060-19. Han Xu, Wei Jin, Xiaorui Liu, and Jiliang Tang are supported by the National Science Foundation (NSF) under grant numbers IIS1714741, CNS1815636, IIS1845081, IIS1907704, IIS1928278, IIS1955285, IOS2107215, and IOS2035472, and the Army Research Office (ARO) under grant number W911NF-21-1-0198. Suhang Wang is supported by NSF under grant number IIS-1909702, ARO under grant W911NF21-1-0198, and Department of Homeland Security (DHS) CINA under grant E205949D.

REFERENCES

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

[2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

[3] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 112–121.

[4] T. Derr, Y. Ma, W. Fan, X. Liu, C. Aggarwal, and J. Tang, "Epidemic graph convolutional network," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 160–168.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in neural information processing systems*, 2017, pp. 1024–1034.

[6] W. Fan, Y. Ma, Q. Li, J. Wang, G. Cai, J. Tang, and D. Yin, "A graph neural network framework for social recommendations," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[7] B. Chen, X. Zhao, Y. Wang, W. Fan, H. Guo, and R. Tang, "Automated machine learning for deep recommender systems: A survey," *CoRR*, vol. abs/2204.01390, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.01390

[8] W. Fan, Y. Ma, D. Yin, J. Wang, J. Tang, and Q. Li, "Deep social collaborative filtering," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 305–313.

[9] J. Wu, W. Fan, J. Chen, S. Liu, Q. Li, and K. Tang, "Disentangled contrastive learning for social recommendation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2022, pp. 4570–4574.

[10] W. Fan, Q. Li, and M. Cheng, "Deep modeling of social relations for recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[11] X. Zhao, H. Liu, W. Fan, H. Liu, J. Tang, and C. Wang, "Autoloss: Automated loss function search in recommendations," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021, pp. 3959–3967.

[12] W. Fan, T. Derr, Y. Ma, J. Wang, J. Tang, and Q. Li, "Deep adversarial social recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 1351–1357.

[13] H. Xu, Y. Ma, H. Liu, D. Deb, H. Liu, J. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, 2020.

[14] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2847–2856.

[15] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of the 35th International Conference on Machine Learning, PMLR*, vol. 80, 2018.

[16] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of The Web Conference 2020*, 2020, pp. 673–683.

[17] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples for graph data: deep insights into attack and defense," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 4816–4823.

[18] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: an optimization perspective," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019, pp. 3961–3967.

[19] W. Fan, T. Derr, X. Zhao, Y. Ma, H. Liu, J. Wang, J. Tang, and Q. Li, "Attacking black-box recommendations via copying cross-domain user profiles," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1583–1594.

[20] J. Chen, W. Fan, G. Zhu, X. Zhao, C. Yuan, Q. Li, and Y. Huang, "Knowledge-enhanced black-box attacks for recommendations," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 108–117.

[21] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," in *Advances in neural information processing systems*, 2019, pp. 9244–9255.

[22] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," in *Proceedings of 34th Conference on Neural Information Processing Systems*, 2020.

[23] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, ser. KDD '20, 2020, p. 430–438.

[24] Q. Huang, M. Yamada, Y. Tian, D. Singh, D. Yin, and Y. Chang, "Graphlime: Local interpretable model explanations for graph neural networks," *arXiv preprint arXiv:2001.06216*, 2020.

[25] M. T. Ribeiro, S. Singh, and C. Guestrin, "Semantically equivalent adversarial rules for debugging nlp models," in *ACL*, 2018.

[26] P. Lertvittayakumjorn, L. Specia, and F. Toni, "Find: Human-in-the-loop debugging deep text classifiers," in *EMNLP*, 2020.

[27] P. Lertvittayakumjorn and F. Toni, "Explanation-based human debugging of nlp models: A survey," *Transactions of the Association for Computational Linguistics*, 2021.

[28] J. Adebayo, M. Muelly, I. Liccardi, and B. Kim, "Debugging tests for model explanations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 700–712, 2020.

[29] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv preprint arXiv:2003.00653*, 2020.

[30] W. Fan, Y. Ma, H. Xu, X. Liu, J. Wang, Q. Li, and J. Tang, "Deep adversarial canonical correlation analysis," in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 352–360.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[32] Y. Wan, H. Xu, X. Liu, J. Ren, W. Fan, and J. Tang, "Defense against gradient leakage attacks via learning to obscure data," *arXiv preprint arXiv:2206.00769*, 2022.

[33] W. Fan, X. Zhao, X. Chen, J. Su, J. Gao, L. Wang, Q. Liu, Y. Wang, H. Xu, L. Chen *et al.*, "A comprehensive survey on trustworthy recommender systems," *arXiv preprint arXiv:2209.10117*, 2022.

[34] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach," in *Proceedings of The Web Conference 2020*, 2020, pp. 673–683.

[35] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[36] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[37] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[38] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[39] S. Kim, J. Yi, E. Kim, and S. Yoon, "Interpretation of nlp models through input marginalization," in *EMNLP*, 2020.

[40] E. Dai and S. Wang, "Towards self-explainable graph neural network," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 302–311.

[41] ——, "Towards prototype-based self-explainable graph neural network," *arXiv preprint arXiv:2210.01974*, 2022.

[42] J. Heo, S. Joo, and T. Moon, "Fooling neural network interpretations via adversarial model manipulation," in *Advances in Neural Information Processing Systems*, 2019, pp. 2925–2936.

[43] X. Zhang, N. Wang, H. Shen, S. Ji, X. Luo, and T. Wang, "Interpretable deep learning under fire," in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[44] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3681–3688.

[45] N. Liu, H. Yang, and X. Hu, "Adversarial detection with model interpretation," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1803–1811.

[46] H. Liu, Y. Wang, W. Fan, X. Liu, Y. Li, S. Jain, Y. Liu, A. K. Jain, and J. Tang, "Trustworthy ai: A computational perspective," *ACM Transactions on Intelligent Systems and Technology (ACM TIST)*, 2022.

[47] H. Zhang, B. Wu, X. Yuan, S. Pan, H. Tong, and J. Pei, "Trustworthy graph neural networks: Aspects, methods and trends," *arXiv preprint arXiv:2205.07424*, 2022.

[48] E. Dai, T. Zhao, H. Zhu, J. Xu, Z. Guo, H. Liu, J. Tang, and S. Wang, "A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability," *arXiv preprint arXiv:2204.08570*, 2022.

[49] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 66–74.

[50] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[51] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1243–1253.

[52] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[53] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," *arXiv preprint arXiv:1809.02797*, 2018.

[54] Y. Ma, S. Wang, T. Derr, L. Wu, and J. Tang, "Attacking graph convolutional networks via rewiring," *arXiv preprint arXiv:1906.03750*, 2019.

[55] J. Han, J. Pei, and M. Kamber, *Data mining: concepts and techniques*. Elsevier, 2011.

[56] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15 920–15 933, 2021.

[57] M. Kim, J. Tack, and S. J. Hwang, "Adversarial self-supervised contrastive learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2983–2994, 2020.