Black-box Adversarial Attack and Defense on Graph Neural Networks

Haoyang LI[1], Shimin DI[1†], Zijian LI[1], Lei CHEN[1], Jiannong CAO[2]
[1]The Hong Kong University of Science and Technology, Hong Kong SAR, China
[2]The Hong Kong Polytechnic University, Hong Kong SAR, China
{hlicg,sdiaa,zlicb,leichen}@cse.ust.hk, csjcao@comp.polyu.edu.hk

2022 IEEE 38th International Conference on Data Engineering (ICDE) | 978-1-6654-0883-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDE53745.2022.00081

***Abstract*—Graph neural networks (GNNs) have achieved great success on various graph tasks. However, recent studies have revealed that GNNs are vulnerable to adversarial attacks, including topology modifications and feature perturbations. Regardless of the fruitful progress, existing attackers require node labels and GNN parameters to optimize a bi-level problem, or cannot cover both topology modifications and feature perturbations, which are not practical, efficient, or effective. In this paper, we propose a black-box attacker PEEGA, which is restricted to access node features and graph topology for practicability. Specifically, we propose to measure the negative impact of various adversarial attacks from the perspective of node representations, thereby we formulate a single-level problem that can be efficiently solved. Furthermore, we observe that existing attackers tend to blur the context of nodes through adding edges between nodes with different labels. As a result, GNNs are unable to recognize nodes. Based on this observation, we propose a GNN defender GNAT, which incorporates three augmented graphs, i.e., a topology graph, a feature graph, and an ego graph, to make the context of nodes more distinguishable. Extensive experiments on three real-world datasets demonstrate the effectiveness and efficiency of our proposed attacker, despite the fact that we do not access node labels and GNN parameters. Moreover, the effectiveness and efficiency of our proposed defender are also validated by substantial experiments.**

***Index Terms*—Graph Neural Network, Adversarial Attack, Graph Defense**

I. Introduction

Graph tasks, such as graph isomorphism [1] and subgraph counting [2], are important and popular topics in the database area. Recently, many research works have shown that graph neural networks (GNNs) [3], [4], [5], [6], [7], [8], [9], [10], which learn node representations by aggregating the information from their neighbors, have achieved great success in the these tasks, including graph isomorphism [11], [12], subgraph counting [13], [14], [15], trajectory routing [16], [17], recommendation system [18], [19], knowledge graph completion [20], [21], and node classification [22], [23].

Despite the great success, many studies [24], [25], [26], [27], [28] have shown that the performance of GNNs on these tasks will be significantly reduced when the graph is modified. In real-world scenarios, it is common to run a GNN model on the modified graph with the consideration of data privacy or possible attacks. For example, the Internet platform may modify the graph to protect users' privacy before data publication [29], [30], like revising user profiles (perturbing features) and user links (modifying edge). Also, attackers may control or hack some users to poison the online social graphs by following/unfollowing users and revising profiles. Due to the unpredictability and destructiveness of the attacking behavior, exploring the vulnerability of GNNs under attackers and revealing attack patterns are crucial to improve the robustness and task performance of GNNs on modified graphs.

Currently, adversarial attackers on GNNs can be categorized into white-box attackers [31], [32], gray-box attackers [24], [25], and black-box attackers [33], [34], [35], [36], [26]. White-box attackers [31], [32] require the graph data set (including graph topology, node features, and labels) and the parameters of target GNNs to conduct attacks. Based on the gradients of GNNs, they modify the graph structures (i.e., topology modifications) and node features (i.e., feature perturbations) to make the attack more effective. Different from white-box attackers, gray-box attackers [24], [25] do not need the parameters of target GNNs, while they only require the graph data set. They train a surrogate model which simulates GNNs, and then generate adversarial attacks based on the surrogate model. However, white-box and gray-box attackers are not practical because it is scarcely realistic to obtain the parameters of target GNNs or a large amount of node labels in most of real-world applications [33], [34]. Moreover, the above white-box and gray-box attackers sacrifice their efficiency to gain effectiveness. Specifically, they formulate the attack problem as a bi-level optimization problem, where the upper-level attack selection relies on the lower-level GNN model training. Optimizing the underlying parameters is necessary to evaluate candidate attacks, which makes the whole optimization process highly time-consuming [37].

Therefore, to improve the practicability and the efficiency of attackers, black-box attackers are proposed [26], [33], [34], [35], [36]. Generally, black-box attackers assume that the parameters of GNNs and node labels are inaccessible, while they are restricted to access graph topology and node features. However, there are still several limitations in the settings of existing black-box attackers. First, they cannot handle both topology modifications and feature perturbations simultaneously, which is not flexible to deal with different attack scenarios. Second, beside graph topology and node features, several black-box attackers [26], [36] require some predictions of the target GNNs to optimize their reinforcement learning-based attack models, while the practicability

†Corresponding author

2375-026X/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDE53745.2022.00081

TABLE I: Summary of existing GNN attackers. 1 and 0 represent whether the input is required for the attacker. ✓and × indicate that the attacker can and cannot cover the attack type, respectively. The goal of attackers has two types: targeted attack and untargeted attacks. Attacker nodes denote these available nodes that can be accessed and attacked by attackers. Particularly, attackers that access all nodes can be easily extended to only access a subset of nodes. We discuss them in detail in Sec. II-B.

Type	Attack Model	Input Required					Attack Types				Objective Level
		GNN		Data Set			Attack Constraint		Conducted Attacks		
		Parameter	Prediction	Topology	Feature	Label	Goal	Attacker Nodes	Topology	Feature	
White-box	PGD [31]	1	1	1	1	1	Untargeted	All	✓	×	Bi-level
	MinMax [31]	1	1	1	1	1	Untargeted	All	✓	×	Bi-level
Grey-box	Nettack [25]	0	0	1	1	1	Targeted	Subset	✓	✓	Bi-level
	Metattack [24]	0	0	1	1	1	Untargeted	All	✓	✓	Bi-level
Black-box	RL-S2V [36]	0	1	1	1	0	Untargeted	All	✓	×	Single level
	ReWatt [26]	0	1	1	1	0	Untargeted	All	✓	×	Single level
	RWCS [33]	0	0	1	1	0	Untargeted	Subset	×	✓	Single level
	InfMax [34]	0	0	1	1	0	Untargeted	Subset	×	✓	Single level
	GF-Attack [35]	0	0	1	1	0	Targeted	All	✓	×	Single level
	PEEGA (ours)	0	0	1	1	0	Untargeted	All	✓	✓	Single level

of attackers decreases and they are difficult to converge with limited predictions. As shown in Tab. I we summarize the existing GNN attackers from three perspectives: the required inputs, the types of conducted attacks covered, and the number of objective levels. None of existing works can cover all these aspects.

To simulate real-world attacks, we propose a Practical, Effective and Efficient GNN adversarial Attacker, named PEEGA. PEEGA is a pure black-box attacker that is restricted to access the graph topology and node features, which is more practical and realistic in real-world applications. Specifically, we propose a novel way to qualify the influence of topology modifications and feature perturbations on the node representations. It first enables us to effectively attack both topology and features concurrently. Besides, the measurement is model-agnostic, where the performance of candidate attacks can be evaluated without training the underlying GNN parameters. As a result, the optimization in PEEGA can be formulated as a single-level problem, which can be efficiently solved.

Meanwhile, many GNN defenders [27], [28] have been proposed to protect GNNs from attacks and improve robustness of GNNs, i.e., achieving good performance on the poison graph (i.e., the graph has been attacked). Nevertheless, preprocessing-based defenders [32], [38] eliminate edges of dissimilar nodes, and attention-based defenders [4], [39], [40] restrict the negative impact of potential attacks by removing those edges with small attention weights. Such way ignores the case that the edges may be removed from the original graph by attackers. Graph-learning based defenders [41], [42], [43] target to improve performance of GNNs by modifying the poison graph. Such pure performance-driven way may make the modified graph deviate from the original graph. In summary, existing defenders do not well study attack patterns, such as the change of node contexts before and after attacks, and thus fail to defend against attackers effectively.

In this paper, we systematically investigate GNN attackers, and reveal their attack patterns. Specifically, existing attackers tend to add edges between nodes with different labels, which makes the context of nodes indistinguishable. To resist such attacks, we propose a simple but effective GNN defender based on graph augmeNtATions, named GNAT. Specifically, we propose three augmented graphs (topology graph, feature graph and ego graph) to make the context of nodes more distinguishable. Especially, it can be conducted in the black-box settings, such as lacking of data labels, and inaccessible GNN parameters. Overall, our contributions are summarized as follows:

TABLE II: Summary on Important Notations.

Symbols	Meanings
$G(\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{Y})$	The graph G with the node set $\mathcal{V}$, adjacency matrix $\mathbf{A}$, node features $\mathbf{X}$, and node label $\mathbf{Y}$.
$\hat{\mathbf{A}}, \hat{\mathbf{X}}$	The adjacency matrix and features after modification.
$\mathbf{A}_n$	The normalized adjacency matrix of $\mathbf{A}$.
$\mathcal{N}_v$	The neighbors of node v.
$\mathbf{h}_v, \mathbf{H}$	The representation of v and all nodes, and $\mathbf{H} \in \mathbb{R}^{\lvert\mathcal{V}\rvert \times d_h}$.
$\mathbf{Z}$	The label probability of all nodes, and $\mathbf{Z} \in [0,1]^{\lvert\mathcal{V}\rvert \times \lvert\mathcal{V}\rvert}$.
$\mathcal{L}_{gnn}, \mathcal{L}_{atk}$	The training loss of GNNs, and attack loss of GNN attackers.
$\mathcal{M}_\theta$	The GNN model $\mathcal{M}$ parameterized by θ.
$\lVert\cdot\rVert_p$	Lp norm distance.

- We propose a practical, effective, and efficient black-box attacker PEEGA. Although PEEGA requires least input parameters, we propose a novel way to measure the influence of adversarial attacks on the node representations. Thus, PEEGA can attack the graph topology and node features more effectively. Besides, the proposed influence measurement is unsupervised and model-agnostic, which avoids optimizing GNN parameters to achieve more efficiency.
- We provide insights that existing attackers degrade the performance of GNNs by making the context of nodes indistinguishable. To resist such attacks, we propose a simple but effective defender GNAT, which incorporates three augmented graphs to improve the robustness of GNNs.
- Experiments on three real-world datasets demonstrate that our GNN attacker PEEGA can efficiently achieve comparable attacking effectiveness with less inputs.
- Compared with raw GNNs and existing GNN defenders, GNAT can defend against various GNN attackers, including PEEGA. GNAT achieves the outstanding performance no matter it is trained on clean graphs or poison graphs. And GNAT only takes a bit more time than raw GNNs, but it is more efficient than all the other defender baselines.

II. Related Work

In this paper, we consider the node classification task where nodes have binary node features [25]. Formally,

$G(\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{Y})$ denotes a graph, where $\mathcal{V}$, $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, $\mathbf{X} \in \{0,1\}^{|\mathcal{V}|\times d_x}$, $\mathbf{Y} \in \{0,1\}^{|\mathcal{V}^{la}|\times|\mathcal{Y}|}$ denote nodes, adjacency matrix, node features, and node labels, respectively. Specifically, $\mathcal{V}^{la} \subset \mathcal{V}$ is the set of labeled training nodes, $\mathbf{A}[u][v] = 1$ indicates that there is an edge $e_{u,v}$ between node u and v, and $\mathcal{N}_v = \{u : \mathbf{A}[u][v] = 1\}$ denotes the neighbors of v. Generally, models for the node classification task [4] target to learn a classifier $f : \mathcal{V} \to \mathcal{Y}$ to map nodes to the label space $\mathcal{Y}$. The important notations are summarized in Tab. II.

A. Graph Neural Network

Taking $\mathbf{A}, \mathbf{X}$ in a graph G as inputs, GNNs [3], [5], [4], [44] first encode nodes $\mathcal{V}$ into d_h-dimensional vector space (i.e., $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times d_h}$) by recursively aggregating the information from its neighbors $u \in \mathcal{N}_v$. Taking one representative Graph Convolutional Network (GCN) [3] as an example, the *l+1-th* node representations $\mathbf{H}^{l+1} \in \mathbb{R}^{|\mathcal{V}|\times d_{l+1}}$ can be defined as:

$$\mathbf{H}^{l+1} = \sigma(\mathbf{A}_n \mathbf{H}^l \mathbf{W}^l),$$

where σ denotes a non-linear function (e.g., ReLU [45]), $\mathbf{H}^l \in \mathbb{R}^{|\mathcal{V}|\times d_l}$ denotes node representations in the *l-th* layer, $\mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l+1}}$ for $l \in \{1, \cdots, L-1\}$. $\mathbf{A}_n$ is a normalized adjacency matrix [3]. Initially, $\mathbf{H}^0 = \mathbf{X}$. After L-layer aggregations, we can obtain the final output $\mathbf{Z} \in \mathbb{R}^{|\mathcal{V}^{la}|\times|\mathcal{Y}|}$ as:

$$\mathbf{Z} = softmax(\mathbf{A}_n \sigma(\mathbf{A}_n \sigma(\cdots)\mathbf{W}^{L-1})\mathbf{W}^L), \tag{1}$$

where $\mathbf{W}^L \in \mathbb{R}^{d_{L-1}\times|\mathcal{Y}|}$. Then the loss of the GCN model can be defined as cross entropy loss:

$$\mathcal{L}_{gnn}(\mathcal{M}_\theta, G(\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{Y})) = -\sum_{v\in\mathcal{V}^{la}} \ln \mathbf{Z}[v][y_v], \tag{2}$$

where y_v is the given label of v from training data, $\mathcal{M}_\theta$ represents the GCN model parameterized by θ, and $\theta = \{\mathbf{W}^0, \cdots, \mathbf{W}^L\}$ denotes the set of transform parameter matrix. Following [24], [25], [41], we focus on GCN [3] in this paper. Additionally, several variants of GCN have been proposed to improve its efficiency [5], [46], learn the relative weight between each connected neighbors [4], or alleviate the over-smoothing problem [47]. More details can be referred to comprehensive surveys [48], [49], [50].

B. Adversarial Attack on GNNs

As introduced in Sec. II-A, GNNs take adjacent matrix $\mathbf{A}$ and node features $\mathbf{X}$ as inputs. Hence, existing GNN attackers generally conduct topology modifications on $\mathbf{A}$, such as adding and removing edges, and conduct feature perturbations on $\mathbf{X}$, such as modifying feature $\mathbf{x}_v$ of node v by $\hat{\mathbf{x}}_v = \mathbf{x}_v + \boldsymbol{\epsilon}_v$. These attacks influence representations of other nodes due to the message-passing manner of GNNs and thus affect their labels.

From the view of attacking scenarios, there are two constraints on the target nodes (victim nodes) and the available nodes that can be accessible for attacking (attacker nodes). Then, current GNN attackers can be divided into two categories [27], [28]: targeted attacks and untargeted attacks. The targeted attacks [25] aim to reduce the performance of a victim node by attacking a predefined set of attacker nodes. The untargeted attacks [24], [31], [32], [33], [34], on the other hand, aim to reduce the global performance of models on all test nodes by attacking predefined nodes or all nodes. As shown in Tab. I, most of the current attackers on untargeted attacks do not define the available attacker nodes in their papers. But these untargeted attackers including ours can be simply extended to this constraint.

From the view of attacking inputs, current attackers have three types: white-box, gray-box, and black-box. White-box attackers [31], [32] take the graph data set (including graph topology $\mathbf{A}$, node features $\mathbf{X}$, node labels $\mathbf{Y}$) and the parameters of the target GNN θ as inputs to generate topology modifications and feature perturbations. We formally define the goal of attackers as follows:

Definition 1 (Target of GNN attackers.)**.** *Given a modification budget δ, a GNN model $\mathcal{M}_\theta$ parameterized by θ, and the graph $G(\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{Y})$, the target of GNN attackers is to find the modified topology $\hat{\mathbf{A}}$ and node features $\hat{\mathbf{X}}$, which can minimize the attack loss $\mathcal{L}_{atk}$:*

$$\min_{\hat{\mathbf{A}},\hat{\mathbf{X}}} \ \mathcal{L}_{atk}(\mathcal{M}_{\theta^*}, \hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}}, \mathbf{Y})), \tag{3}$$
$$s.t. \begin{cases} \theta^* = \arg\min_\theta \ \mathcal{L}_{gnn}(\mathcal{M}_\theta, \hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}}, \mathbf{Y})) \\ \left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0 + \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_0 \le \delta \end{cases},$$

where $\|\cdot\|_0$ denotes L_0 norm, $\left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0$ counts the number of different elements in the same position between $\hat{\mathbf{A}}$ and $\mathbf{A}$, $\mathcal{L}_{gnn}(\cdot)$ measures the training loss of GNN $\mathcal{M}_\theta$ on the poison graph $\hat{G}$, $\mathcal{L}_{atk}(\cdot)$ measures the generalize loss of $\mathcal{M}_{\theta^*}$ on the test unlabeled nodes.

Assuming that a model with a high training error is more likely to generalize poorly on test nodes, current works [24], [25] set $\mathcal{L}_{atk} = -\mathcal{L}_{train}$ and minimize it by modifying graph topology given a limited budget.

Different from white-box attackers, the gray-box attackers [24], [25] assume that the GNN parameters θ is inaccessible. For example, Metattack [24] trains a surrogate model parameterized by θ' to fit the node labels, such as $f_{\theta'} : \mathcal{V} \to \mathcal{Y}$, then utilize $f_{\theta'}$ to replace $\mathcal{M}_\theta$ in Eq. (3). However, both white-box attackers and gray-box attackers require a large amount of node labels (i.e., $\mathbf{Y}$) to find effective attacks.

Black-box attackers take graph topology $\mathbf{A}$ and node features $\mathbf{X}$ [33], [34], [35], as well as limited predictions of the target GNNs [26], [36], as inputs. Therefore, black-box attackers cannot optimize the bi-level optimization problem in Eq. (3) due to the lack of node labels $\mathbf{Y}$ and the model parameters θ. RL-S2V [36] and ReWatt [26] employ the reinforcement learning framework, which needs the predictions of the target GNN as rewards. Although other black attackers [33], [34], [35] only needs $\mathbf{A}$ and $\mathbf{X}$ as inputs, they cannot conduct both feature perturbations and topology modifications simultaneously, which limit their practicability and attacking effectiveness.

C. Defense on GNNs

As introduced in Sec. II-B, GNN attackers attack a clean graph $G(\mathcal{V}, \mathbf{A}, \mathbf{X}, \mathbf{Y})$ to generate the poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}}, \mathbf{Y})$. GNN defenders are proposed to resist such attacks, which target to achieve good task performance on the poison graph. We formally define GNN defenders as follows.

Definition 2 (Target of GNN defenders)**.** *Given a GNN model $\mathcal{M}_\theta$ parameterized by θ and a poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}}, \mathbf{Y})$, the target of GNN defenders is to find a purified topology $\tilde{\mathbf{A}}$ and node features $\tilde{\mathbf{X}}$, which can minimize the training loss $\mathcal{L}_{gnn}$:*

$$\min_{\theta} \ \mathcal{L}_{gnn}(\mathcal{M}_\theta, \tilde{G}(\mathcal{V}, \tilde{\mathbf{A}}^*, \tilde{\mathbf{X}}^*, \mathbf{Y})) \tag{4}$$
$$s.t. \ \tilde{\mathbf{A}}^*, \tilde{\mathbf{X}}^* = \arg\min_{\theta, \tilde{\mathbf{A}}, \tilde{\mathbf{X}}} \mathcal{L}_{purify}(\mathcal{M}_\theta, \tilde{G}(\mathcal{V}, \tilde{\mathbf{A}}, \tilde{\mathbf{X}}, \mathbf{Y}); \hat{G})$$

where $\mathcal{L}_{purify}$ measures the loss of the purification algorithm, which takes GNN $\mathcal{M}_\theta$ and the poison graph $\hat{G}$ as inputs, and optimizes the purified graph $\tilde{G} = (\mathcal{V}, \tilde{\mathbf{A}}, \tilde{\mathbf{X}}, \mathbf{Y})$.

Generally, there are three main categories of GNN defenders. First, based on node features, preprocessing-based defenders [32], [38] compute the similarity of the connected nodes on the poison graph. Then, they eliminate those edges from the poison graph if the similarity is smaller than a threshold. They may suffer from the situations where features are seriously attacked. Second, attention-based defenders [4], [39], [40] use the attention mechanism [44] to measure the relative weight on each pair of connected nodes. Assuming the adversarial edges contribute less, they remove those edges with small attention values. Third, graph learning-based defenders [41], [42], [43] optimize the purified graph $\tilde{G}$ from $\hat{G}$ by evaluating the performance of a GNN model $\mathcal{M}_\theta$ on $\tilde{G}$. However, the above defenders ignore attack patterns somehow: 1) the first two cannot handle the case that edges may be deleted from the original graph, 2) the third one purely pursues the performance without studying the attack patterns for nodes with different labels.

III. Attack Model

In this section, we introduce our black-box attacker PEEGA. We first measure the impact of topology modifications and feature perturbations on node representations from self-view and global view. Then, we formulate a novel attack objective under the black-box setting, and propose an efficient greedy-based algorithm to solve it.

A. Measurement on the Difference of Node Representations

As introduced in Sec. I and Sec. II-B, white-box and grey-box attackers achieve better performance than black-box attackers, because the former can conduct both feature perturbations and topology modifications by leveraging more inputs (e.g., a large amount of node labels). Black-box attackers assume that GNN parameters and node labels cannot be accessed for the sake of practicability. But existing attackers fail to consider two attack types simultaneously and some of them require model predictions as inputs. Therefore, we target to conduct two types of attacks while only requiring $\mathbf{A}$ and $\mathbf{X}$ as inputs. To achieve this goal, we propose a model-agnostic and unsupervised way to measure the influence of attacks on the node representations.

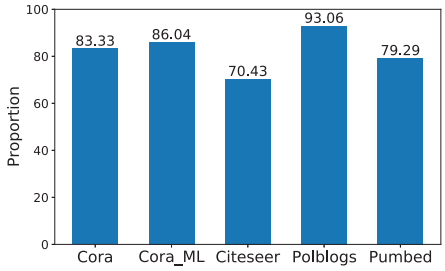


Fig. 1: The proportion of edges whose connected nodes have the same label.

As discussed in Sec. II-A, GNNs first derive node representations $\mathbf{H}$ by the neighbor aggregations and then use $\mathbf{H}$ to predict the label probability $\mathbf{Z}$ of nodes. It indicates that the classification performance is firmly correlated with the quality of node representations. Thus, we propose to evaluate the negative impact of feature perturbations and topology modifications on node representations when lacking of node labels. We next introduce how to measure the difference of node representations from self view and global view of nodes.

1) Self View: Intuitively, a node v tends to be misclassified if there is a big difference between its original representation $\mathbf{h}_v$ and the representation $\hat{\mathbf{h}}_v$ after the topology modifications $\hat{\mathbf{A}}$ and feature perturbations $\hat{\mathbf{X}}$. Let the representations of all nodes before and after graph attacks are denoted as $\mathbf{H} \in \mathbb{R}^{\mathcal{V} \times d_z}$ and $\hat{\mathbf{H}} \in \mathbb{R}^{\mathcal{V} \times d_z}$. The change of representation for every node $v \in \mathcal{V}$ can be calculated as:

$$Dif_1(\mathbf{H}, \hat{\mathbf{H}}) = \sum_{v \in \mathcal{V}} \left\| \hat{\mathbf{h}}_v - \mathbf{h}_v \right\|_p, \tag{5}$$

where $\|\cdot\|_p$ measures Lp-norm distance. Maximizing $Dif_1(\mathbf{H}, \hat{\mathbf{H}})$ can enlarge the difference between node representations before and after attacks.

2) Global View: In addition to measuring the difference of each node representation before and after $\hat{\mathbf{A}}, \hat{\mathbf{X}}$, it is also important to measure the difference between the representations of nodes with the same label. If the representation of a node is dissimilar to that of other nodes in the same category, the misclassification errors will increase. Even though we lack the node labels, we can maximize the representation difference between each pair of connected nodes. It is because the connected nodes tend to have the same label, i.e., the homophily property of graphs. As shown in Fig. 1, the proportion of connected nodes with same label on the five real-world datasets is more than 70.43%. Therefore, the 1-hop neighbors are likely to have the same label with nodes, and we can utilize their node representations as a guidance. Formally, the difference of representations among nodes and its neighbors can be calculated as:

$$Dif_2(\mathbf{A}, \mathbf{H}, \hat{\mathbf{H}}) = \sum_{v \in \mathcal{V}} \sum_{u \in \mathcal{N}_v} \left\| \hat{\mathbf{h}}_v - \mathbf{h}_u \right\|_p, \tag{6}$$

where the original topology $\mathbf{A}$ determines the neighbor information $\mathcal{N}_v$. Maximizing $Dif_2(\mathbf{A}, \mathbf{H}, \hat{\mathbf{H}})$ can enlarge the difference between the node representation after attack and the representations of its neighbors before attack.

B. PEEGA GNN Attacker

1) Problem Formulation: Unfortunately, the black-box setting assumes that node labels and GNN parameters are not available. This leads that the node representation $\mathbf{h}_v$ is unknown when we try to maximize $Dif_1(\mathbf{H}, \hat{\mathbf{H}})$ and $Dif_2(\mathbf{A}, \mathbf{H}, \hat{\mathbf{H}})$. Therefore, we propose an alternative objective to maximize the difference between node representations in Eq. (5) and (6). For simplicity, here we utilize two-layer GCN [3] as an example to represent the node difference as:

$$\begin{aligned}\left\|\hat{\mathbf{h}}_v - \mathbf{h}_u\right\|_p &= \left\|\hat{\mathbf{A}}_n^2[v]\hat{\mathbf{X}}\mathbf{W} - \mathbf{A}_n^2[u]\mathbf{X}\mathbf{W}\right\|_p \\ &\leq \|\mathbf{W}\|_p \cdot \left\|\hat{\mathbf{A}}_n^2[v]\hat{\mathbf{X}} - \mathbf{A}_n^2[u]\mathbf{X}\right\|_p, \end{aligned} \tag{7}$$

where we follow [25] to relax the non-linear function $\sigma(\cdot)$ to the linear one. The inequality in Eq. (7) can be driven from [51] when taking $\mathbf{W}$ as constant matrix. This is in line with the scenario of inaccessible and immutable GNN parameters. In Eq. (7), $\mathbf{A}_n^2\mathbf{X}$ can simulate the most important step of GNNs: aggregating the information from neighbors. Thus, $\mathbf{A}_n^2\mathbf{X}$ can serve as a surrogate model for black-box attackers. Modifying $\mathbf{A}$ and $\mathbf{X}$ in Eq. (7) could affect the upper bound of node difference. Overall, Eq. (7) enables a model-agnostic and unsupervised way to evaluate the influence of attacks. Formally, we define the black-box attack problem as follows:

Definition 3 (PEEGA Black-box Attack Problem)**.** *Given a graph modification budget δ, and a graph $G(\mathcal{V}, \mathbf{A}, \mathbf{X})$, the target of PEEGA black-box attacker is to find the topology modifications $\hat{\mathbf{A}}$ and feature perturbations $\hat{\mathbf{X}}$, which can maximize the difference of node representations before and after adversarial attacks. The problem is formulated as:*

$$\begin{aligned}\max_{\hat{\mathbf{A}},\hat{\mathbf{X}}} \; & \sum_{\mathrm{v}\in\mathcal{V}} \left\|\hat{\mathbf{A}}_n^2[\mathrm{v}]\hat{\mathbf{X}} - \mathbf{A}_n^2[\mathrm{v}]\mathbf{X}\right\|_p \\ & + \lambda \sum_{\mathrm{v}\in\mathcal{V}} \sum_{u\in\mathcal{N}_\mathrm{v}} \left\|\hat{\mathbf{A}}_n^2[\mathrm{v}]\hat{\mathbf{X}} - \mathbf{A}_n^2[u]\mathbf{X}\right\|_p, \\ s.t. \; & \left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0 + \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_0 \leq \delta \end{aligned} \tag{8}$$

where $\mathbf{A}_n$ denotes a normalized adjacency matrix [3] and λ is a trade-off hyper-parameter.

As shown in above formulation, PEEGA only requires the original graph topology $\mathbf{A}$ and node features $\mathbf{X}$ as inputs, while white-box and gray-box attackers require more inputs like node labels $\mathbf{Y}$ and GNN parameters θ (see Sec. II-B). Moreover, white-box and gray-box attackers formulate a bi-level optimization problem that optimizes GNN parameters and the attack loss iteratively, which is time-consuming. The above objective does not need to iteratively update the GNN parameters, which can be more efficiently solved. Furthermore, different from existing black-box attackers [33], [34], [35], [36], [26], PEEGA introduces the concept of difference measurement on node representations, which enables both topology modifications and feature perturbations under the black-box setting. Moreover, unlike some black-box attackers, PEEGA does not need model predictions as inputs (see Tab. I for more information).

Algorithm 1: PEEGA Attacker

Input: Graph $G(\mathcal{V}, \mathbf{A}, \mathbf{X})$, modification budget δ

$\hat{\mathbf{A}} \leftarrow \mathbf{A}$, $\hat{\mathbf{X}} \leftarrow \mathbf{X}$
while $\left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0 + \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_0 \leq \delta$ **do**
  $\mathbf{A}_t = -2\hat{\mathbf{A}} + \mathbf{1}$
  $\mathbf{X}_f = -2\hat{\mathbf{X}} + \mathbf{1}$
  $\mathbf{S}_t = \nabla_{\hat{\mathbf{A}}}\mathcal{L}(\hat{\mathbf{A}}, \hat{\mathbf{X}}) \odot \mathbf{A}_t$
  $\mathbf{S}_f = \nabla_{\hat{\mathbf{X}}}\mathcal{L}(\hat{\mathbf{A}}, \hat{\mathbf{X}}) \odot \mathbf{X}_f$
  $(u, v) \leftarrow$ entry with the highest score in $\mathbf{S}_t$
  $(o, i) \leftarrow$ entry with the highest score in $\mathbf{S}_f$
  if $\mathbf{S}_t[u][\mathrm{v}] < \mathbf{S}_f[o][i]$ **then**
    $\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} + \mathbf{X}_f[o][i]$
  else
    $\hat{\mathbf{A}} \leftarrow \hat{\mathbf{A}} + \mathbf{A}_t[u][v]$
Return: The poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}})$

2) Greedy-based Optimization: Although Def. 3 simplifies the classic bi-level formulation in Eq. (3), optimizing the single level problem Def. 3 is still a non-trivial task. Given the adjacency matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, node features $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}|\times d_x}$, and the budget δ, the number of candidate attack combinations is $\binom{|\mathcal{V}|^2 + d_x|\mathcal{V}|}{\delta}$, which leads to $O\left((|\mathcal{V}|^2 + d_x|\mathcal{V}|)^\delta\right)$ search space. It is clearly infeasible to obtain the best attack combinations by exhaustive search. Then, we propose a greedy-based optimization algorithm to solve the problem in Def. 3. With the help of gradients, our algorithm can give a good direction to find the next effective attack under the consideration of the selected attacks. First, we take the topology modifications and feature perturbations as discrete actions, and define them as follows:

Definition 4 (Candidate Attack Actions)**.** *Given the adjacency matrix $\mathbf{A} \in \{0, 1\}^{|\mathcal{V}|\times|\mathcal{V}|}$, the candidates of topology modifications can be denoted as $\mathbf{A}_t = -2\mathbf{A} + \mathbf{1}$, where $\mathbf{1}$ is an matrix with size of $|\mathcal{V}| \times |\mathcal{V}|$ and all elements are 1. Specifically, $\mathbf{A}_t[u][\mathrm{v}] = -1$ and $\mathbf{A}_t[u][\mathrm{v}] = 1$ indicate the attacker can delete or add the edge between node u and v, respectively. Similarly, given the node features $\mathbf{X} \in \{0, 1\}^{|\mathcal{V}|\times d_x}$, we define the candidates of feature perturbation as $\mathbf{X}_f$.*

Then, we define the score of candidate attack actions as:

$$\mathbf{S}_t = \nabla_{\hat{\mathbf{A}}}\mathcal{L}(\hat{\mathbf{A}}, \hat{\mathbf{X}}) \odot \mathbf{A}_t, \; \mathbf{S}_f = \nabla_{\hat{\mathbf{X}}}\mathcal{L}(\hat{\mathbf{A}}, \hat{\mathbf{X}}) \odot \mathbf{X}_f, \tag{9}$$

where $\mathbf{S}_t[u][v]$ and $\mathbf{S}_f[u][i]$ record the score of attacks in $\mathbf{A}_t[u][v]$ and $\mathbf{X}_f[u][i]$ respectively, $\odot$ denotes the element-wise multiplication, and $\mathcal{L}(\hat{\mathbf{A}}, \hat{\mathbf{X}})$ denotes the difference measurement in Eq. (8). An attack candidate with the highest gradient score indicates the direction of greatest change for maximizing

the goal in Eq. (8). Note that the gradients $\nabla_{\hat{\mathbf{A}}}\mathcal{L}(\hat{\mathbf{A}},\hat{\mathbf{X}})$ and $\nabla_{\hat{\mathbf{X}}}\mathcal{L}(\hat{\mathbf{A}},\hat{\mathbf{X}})$ cannot be directly computed because $\hat{\mathbf{A}}$ and $\hat{\mathbf{X}}$ are discrete. We employ the approximation technique presented in [24].

The basic idea of our algorithm is to greedily select attacks based on the score $\mathbf{S}_t$ and $\mathbf{S}_f$ until exceeding the attack budget δ. Our greedy-based algorithm has been summarized in Alg. 1. We first compute the attack candidates based on the existing poison topology $\hat{\mathbf{A}}$ and features $\hat{\mathbf{X}}$ (line 3-4). Then we compute the score of these attack candidates based on the multiplication of candidates and gradients (line 5-6). Finally, we select the candidate with the highest score (line 7-8) and add it into the poison topology $\hat{\mathbf{A}}$ or features $\hat{\mathbf{X}}$ (line 9-12)

Time Complexity. We analyze time complexity of Alg. 1. In line 5-12, it takes $O(d_x|\mathcal{V}|^2)$ to compute $\mathcal{L}(\hat{\mathbf{A}},\hat{\mathbf{X}})$ and $O(d_x|\mathcal{V}|^2)$ to compute the score of candidate attacks. Then, the total time complexity is $O(\delta d_x|\mathcal{V}|^2)$ after δ iterations.

IV. Defense Model

In this section, we first provide an interesting insight into the GNN attackers, i.e., tending to add edges among nodes with different labels, which blurs the context of nodes. Then, to defend such attacks, we propose a simple but effective graph augmentation method, especially it can handle the scenario of scarce labels.

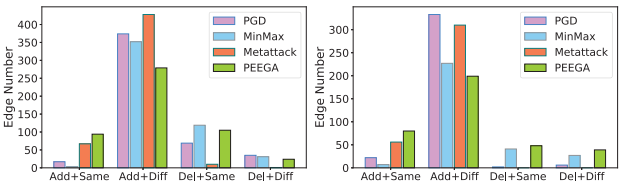
A. Insights on GNN Attackers

As discussed before, existing GNN attackers mainly conduct topology modifications and feature perturbations. To further explore defense, we analyze the attacks of several effective white-box, gray-box and black-box GNN attackers. Through experiments, we observe that existing attackers (including PEEGA) have a tendency in the topology modifications. Topology modifications are classified into four types: adding or deleting edges between nodes with the same or different labels. As illustrated in Fig. 2, GNN attackers tend to add edges between nodes with different labels in topology modifications. As a result, such attacks make the context of nodes indistinguishable, and thus GNNs fail to recognize nodes.

To further verify the above idea, we conduct an experiment to reveal that as the number of attacks increases, the context of nodes with different labels become more similar. Without loss of generality, we use Metattack [24] to attack a publication citation graph, namely Cora, with a variety of target budgets δ. For similarity and clarification, we set $\delta = r \cdot \|\mathbf{A}\|_0$, where $r \in \{0, 0.5, 1, 5\}$ denotes the perturbation rate and $\|\mathbf{A}\|_0$ is the number of edges. We then train a GCN [3] on the poison graphs for node classification task. Specifically, given graph $G(\mathcal{V},\mathbf{A},\mathbf{X},\mathbf{Y})$, the cross-label neighborhood similarity [52] between labels $y_i, y_j \in \mathcal{Y}$ is calculated by:

$$sim\_label(y_i,y_j) = \frac{1}{|\mathcal{V}_{y_i}|\cdot|\mathcal{V}_{y_j}|}\sum_{v\in\mathcal{V}_{y_i}}\sum_{u\in\mathcal{V}_{y_j}} cosine(\mathbf{c}_v,\mathbf{c}_u),$$

where node set $\mathcal{V}_{y_i}$ consists of nodes with label y_i, and $\mathbf{c}_v \in \mathbb{R}^{|\mathcal{Y}|}$ denotes normalized label histogram of v's 1-hop neighbors, i.e., $\mathbf{c}_v[y_i] = n_{v,y_i}/|\mathcal{N}_v|$ where n_{v,y_i} is the number of 1-hop neighbors of v with y_i. In particular, $sim\_label(y_i,y_j)$ denotes intra-label similarity if $y_i = y_j$ else it denotes inter-label similarity.



(a) Cora dataset. (b) Citeseer dataset.

Fig. 2: The edge difference between the poison graph and the origin graph under perturbation rate of 0.1, i.e., the budget $\delta = 0.1 \cdot \|\mathbf{A}\|_0$. **Add** and **Del** denote edge addition and deletion, respectively. **Same** and **Diff** denote whether the node pairs have the same or different label. **Add+Same** indicates edge addition between node pairs with the same labels.

As shown in Fig. 3, the clean graph has the higher intra-label similarity and lower inter-label similarity. It indicates that nodes with distinct labels have distinguishable neighborhood patterns, which enables GCN to recognize nodes accurately ($Acc = 0.83$). However, as r increases, the inter-label similarity of nodes increases, leading to indistinguishably neighbor patterns among nodes with distinct labels. As a consequence, the accuracy of node classification decreases.

B. Graph Augmentation Method

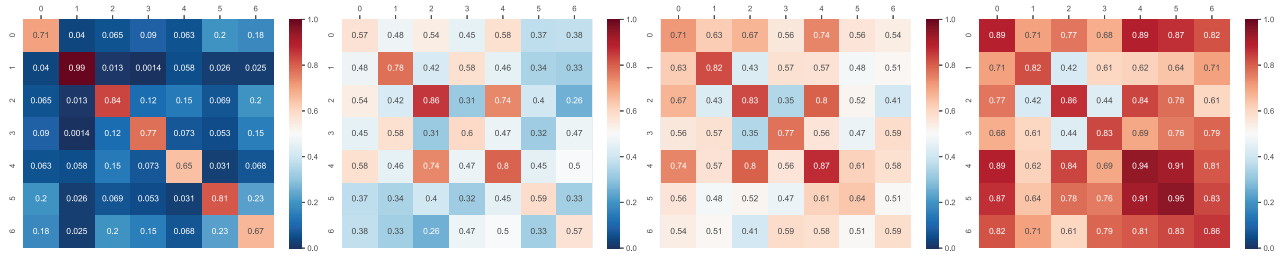
To against the specific topology modifications discussed in Sec. IV-A, we propose a simple but effective method based on data augmentation [53], [54], [55], [56]. Intuitively, adding edges between nodes with the same label can make nodes more distinguishable and decrease the attack loss. We conclude this intuition in the following theorem and put the proof in the online report[1] due to the space limit.

Theorem 1. *Assume that there is a poison graph $\hat{G}(\mathcal{V},\hat{\mathbf{A}},\hat{\mathbf{X}},\mathbf{Y})$, where each training node $\mathrm{v} \in \mathcal{V}^{la}$ has been connected with d nodes under every class $y \in \mathcal{Y}$, i.e., v totally has $d|\mathcal{Y}|$ neighbors (including a self-loop edge). Let the one-hot label vector $\mathbf{y}_\mathrm{v}$ be the feature of node v. For each node v, let $\hat{\mathbf{A}}'$ be the augmentation matrix of $\hat{\mathbf{A}}$, where $\hat{\mathbf{A}}'$ adds edges between v and other α nodes under the label y_v. If $\alpha > 0$, the loss on $\hat{G}'$ will be smaller than that on $\hat{G}$:*

$$\mathcal{L}_{gnn}(\mathcal{M}_\theta,\hat{G}'(\mathcal{V},\hat{\mathbf{A}}',\hat{\mathbf{X}},\mathbf{Y})) < \mathcal{L}_{gnn}(\mathcal{M}_\theta,\hat{G}(\mathcal{V},\hat{\mathbf{A}},\hat{\mathbf{X}},\mathbf{Y})).$$

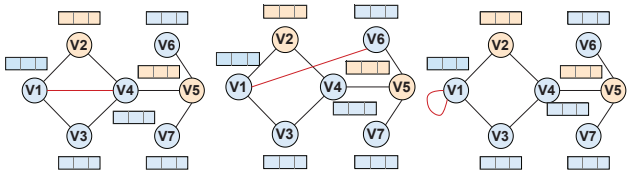
However, adding edges between nodes in one class is a non-trivial task because node labels are not available under the black-box setting. Previous works [57], [56] show that the node labels rely on their features or the graph topology. Inspired by the literature, we propose to evaluate whether the nodes in the poison graph are similar from three perspectives: 1) graph topology, 2) node features, 3) self-loops. As illustrated in Fig. 4, we propose three augmented graphs, including a topology graph, a feature graph and an ego graph.

[1]https://github.com/Refrainlhy/Proof/blob/main/proof.pdf



(a) Ptb_rate=0, Acc=0.83 (b) Ptb_rate=0.5, Acc=0.41 (c) Ptb_rate=1, Acc=0.31 (d) Ptb_rate=5, Acc=0.22

Fig. 3: The similarity among labels on Cora. **Ptb_rate** and **Acc** denote the perturbation rate and accuracy respectively.



(a) Topology Graph (b) Feature Graph (c) Ego Graph

Fig. 4: Three augmented graphs. Nodes with the same color have the same label, and GNAT adds red edges to form augmented graphs. v_1 is the example node of three graphs.

1) Topology Graph $\hat{G}^t$: Due to topology modifications, the 1-hop neighbors of nodes in the poison graph $\hat{G}$ are likely to have been disturbed. Assuming that nodes with the same labels tend to share the same neighbors [58], we add edges between nodes with its k_t-hop neighbors. Given the poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}})$, let $\hat{G}^t(\mathcal{V}, \hat{\mathbf{A}}^t, \hat{\mathbf{X}})$ be the k_t-hop neighbor augmented graph, where $\hat{\mathbf{A}}^t[v][u] = 1$ if v can reach u within k_t hops, i.e., $\hat{\mathbf{A}}^{k_t}[v][u] \neq 0$.

2) Feature Graph $\hat{G}^f$: To make attack more powerful with limited budget, existing attackers [31], [25], [24] tend to attack edges instead of node features (see Sec. V-D1 for more discussion). In other words, node features are likely to be trustworthy in the poison graph. Thus, we connect the nodes with similar feature. Given the poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}})$, we calculate the similarity of each node pair v and u by $cosine(\hat{\mathbf{x}}_v, \hat{\mathbf{x}}_u)$. Then, let $\hat{G}^f(\mathcal{V}, \hat{\mathbf{A}}^f, \hat{\mathbf{X}})$ be feature augmented graph, where $\hat{\mathbf{A}}^f[v][u] = 1$ if u is one of the top-k_f similar nodes of node v.

3) Ego Graph $\hat{G}^e$: Intuitively, the feature of each node indicates its label. Thus, we can emphasize the own feature of nodes by self-loops to alleviate the negative effect of modified edges. Given the poison graph $\hat{G}(\mathcal{V}, \hat{\mathbf{A}}, \hat{\mathbf{X}})$, let $\hat{G}^e(\mathcal{V}, \hat{\mathbf{A}}^e, \hat{\mathbf{X}})$ be the ego augmented graph, where $\hat{\mathbf{A}}^e = \hat{\mathbf{A}} + k_e \cdot \mathbf{I}$, $\mathbf{I} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is an identity matrix, and k_e is a hyperparameter.

After above graph augmentation, we utilize three augmented graphs $\{\hat{G}^t, \hat{G}^f, \hat{G}^e\}$ to jointly train a GCN model $\mathcal{M}_\theta$. Inputting three graphs into $\mathcal{M}_\theta$, we can obtain three specific representations $\{\mathbf{Z}^t, \mathbf{Z}^f, \mathbf{Z}^e\}$ from three different but correlated graph views. We then average these three representation and derive the final representation by $\mathbf{Z} = (\mathbf{Z}^t + \mathbf{Z}^f + \mathbf{Z}^e)/3$. Finally, given the training label $\mathbf{Y}$, we utilize $\mathbf{Z}$ to optimize the GCN model $\mathcal{M}_\theta$ following Eq. (2).

TABLE III: The statistics of three datasets.

	Cora	**Citeseer**	**Polblogs**
#Nodes	2485	2110	1222
#Edges	5069	3668	16714
#Node Classes	7	6	2
#Train Nodes	248	211	121
#Valid Nodes	249	211	123
#Test Nodes	1988	1688	978
d_x	1433	3703	1222

V. Experiments

In this section, we compare our black-box attacker PEEGA and defender GNAT with state-of-the-art baselines. We introduce the experimental settings in Sec. V-A, then show experimental results of effectiveness and efficiency in Sec. V-B and Sec. V-C, respectively. Moreover, we provide the ablation study to analyze our key component in Sec. V-D and present the parameter sensitivity of our model in Sec. V-E.

A. Experimental Setting

All codes are implemented by PyTorch [59] and experiments are run on a CentOS 7 machine with a 20-core Intel(R) Xeon(R) Silver 4210 CPU @ 2.20GHz, 8 NVIDIA GeForce RTX 2080 Ti GPUs (11G), and 92G of RAM.

1) Task and Datasets: In this paper, we follow [24], [25], [41] to focus on the node classification task [24], [25], [3], [41], [42] and evaluate our attacker and defender on three benchmark datasets, Cora [3], Citeseer [60], and Polbogs [61] provided by DeepRobust [62]. Specifically, Cora and Citeseer are citation networks where nodes denote publications, edges denote citations, and features are bag-of-words representations of publications. Polbogs is a blog network, where nodes denote blogs, edges denote citation, and features are the one-hot format generated by their id. The statistics of these three datasets are shown in Tab. III. For every dataset, we follow the setting in [24], [25], [41], [62] that randomly choose 10% of nodes for training, 10% of nodes for validation, and the remaining 80% of nodes for testing. And we report the average accuracy of 10 runs.

2) Baselines: To evaluate the effectiveness of our black-box attacker and our defender, we first utilize the state-of-the-art attackers and our proposed attacker to generate the poison graphs on the three benchmark datasets, and conduct

TABLE IV: Node classification performance (Accuracy±Std) on Cora dataset under 0.1 perturbation rate.

Attackers \ GNNs		Raw GNNs		GNN Defenders					
		GCN	GAT	GCN-Jaccard	GCN-SVD	RGCN	Pro-GNN	SimPGCN	GNAT
Clean Graph		83.36±0.19	84.01±0.35	82.33±0.29	78.33±0.33	83.74±0.14	83.26±0.27	83.39±0.52	(85.52±0.15)
White-box	PGD	80.96±0.83	84.41±0.50	80.52±0.55	77.52±0.57	78.18±0.52	82.39±0.76	81.45±1.09	(84.77±0.20)
	MinMax	78.89±1.34	80.69±1.42	78.84±0.31	77.41±0.70	78.21±0.67	82.57±0.41	77.19±0.83	(83.89±0.19)
Gray-box	Metattack	**72.83±0.29**	**75.56±0.38**	**75.99±0.78**	**73.69±0.51**	**72.47±0.78**	**80.26±0.28**	**75.18±1.20**	**(81.44±0.13)**
Black-box	GF-Attack	83.72±0.42	83.88±0.40	82.28±0.48	78.21±0.22	83.53±0.39	82.22±0.18	82.42±0.23	(85.41±0.12)
	PEEGA	75.31±0.75	77.79±0.82	76.06±0.06	77.02±0.06	75.64±0.07	81.99±0.73	76.51±0.30	(83.12±0.43)

TABLE V: Node classification performance (Accuracy±Std) on Citeseer dataset under 0.1 perturbation rate.

Attackers \ GNNs		Raw GNNs		GNN Defenders					
		GCN	GAT	GCN-Jaccard	GCN-SVD	RGCN	Pro-GNN	SimPGCN	GNAT
Clean Graph		72.03±0.61	73.75±0.41	72.46±0.55	70.01±0.25	72.13±0.66	73.26±0.57	73.12±0.68	(76.39±0.12)
White-box	PGD	70.89±0.84	72.65±0.43	71.17±0.97	68.18±1.36	70.15±1.43	72.35±0.33	73.32±1.06	(76.36±0.15)
	MinMax	70.46±0.97	72.14±0.74	70.53±0.26	68.24±0.32	67.51±0.90	71.53±0.67	72.51±1.26	(75.54±0.13)
Gray-box	Metattack	67.33±0.91	70.70±0.14	69.23±1.01	68.99±0.77	67.86±0.54	72.63±0.89	72.77±0.49	(75.57±0.42)
Black-box	GF-Attack	71.95±0.80	72.93±0.89	72.19±0.55	70.21±0.17	71.75±0.75	73.03±0.46	73.44±0.99	(76.21±0.25)
	PEEGA	**66.20±0.37**	**69.37±0.42**	**67.17±0.40**	**67.46±0.32**	**67.12±0.37**	**71.14±0.24**	**72.21±0.16**	**(75.27±0.19)**

evaluations by training our defender and the state-of-the-art GNN defenders on these poison graphs.

First, we compare the proposed PEEGA with advanced GNN attackers on the scenario of untargeted attacks and black-box settings (i.e., only $\mathbf{A}$ and $\mathbf{X}$ are available). In particular, some baselines in Tab. I are not included in comparisons because: the gray-box attacker Nettack [25] is designed specifically for targeted attacks; the black-box attackers RL-S2V [36] and ReWatt [26] require GNN predictions; the experimental implementations of black-box attackers InfMax[2][34] and RWCS[3][33], access the GNN parameters and node labels to estimate the importance of node features. The baselines of attackers in this paper are listed as follows:

- **PGD** [31]: Projected gradient descent (PGD) first pre-trains the target GNN model, fixes its parameters, and then modifies the graph topology based on gradient projection.
- **MinMax** [31]: Different from PGD, MinMax modifies graph topology and optimizes the target GNN iteratively.
- **Metattack** [24]: Metattack utilizes meta-gradients to solve the bi-level optimization problem to optimize graph topology. Here we use its most destructive variant, Meta-Self.
- **GF-Attack** [63]: GF-Attack can be intrinsically extended to the scenario of untargeted attacks by computing the score of all attack candidates and then select attacks accordingly.
- **PEEGA**: PEEGA is our proposed pure black-box attacker that only takes adjacency matrix $\mathbf{A}$ and node feature $\mathbf{X}$ as inputs, and maximizes the difference between node representations from the self-view and global view.

Additionally, we choose seven GNNs as baselines to evaluate poison graphs. Specifically, GCN [3] and GAT [4] are raw GNNs. GCN-Jaccard [32], RGCN [39], GCN-SVD [64], Pro-GNN [41], and SimPGCN [42] are GNN defenders, which are proposed to resist adversarial attacks for raw GNNs.

- **GCN** [3]: Graph Convolutional Network (GCN) learns node representation by a set of spectral layers, which is the representative one among various GNNs.
- **GAT** [4]: Graph Attention Network (GAT) incorporates the attention mechanism to learn the relative weight between each pair of connected nodes to resist attacks.
- **GCN-Jaccard** [32]: GCN-Jaccard proposes removing edges if the connected nodes' Jaccard similarity is lower than a pre-defined threshold.
- **RGCN** [39]: RGCN models node representations as Gaussian distributions to defense against adversarial attacks. It also incorporates attention mechanism [44] to alleviate the negative impact of adversarial edges.
- **GCN-SVD** [64]: GCN-SVD approximates the poison graph topology into low rank, which is based on the observation that attackers will increase the rank of graph topology.
- **Pro-GNN** [41]: Pro-GNN optimizes the GNN parameters and learn graph structure jointly by preserving the smoothness, low-rank, and sparsity properties of graphs.
- **SimPGCN** [42]: SimPGCN resists adversarial attacks by preserving node similarity and learning node representations adaptively from graph structure and node features.
- **GNAT**: GNAT is the GNN defender proposed in this paper. To resist attacks, we propose three augmented graphs to make the context more distinguishable.

3) Hyper-parameter Setting: For attacker baselines, we use their default hyper-parameter setting. For PEEGA, we tune λ from $\{0, 0.05, 0.01, 0.015, 0.02, 0.025, 0.03\}$, and tune norm p from $\{1, 2, 3\}$. We set the modification budget $\delta = r \cdot \|\mathbf{A}\|_0$, where $r \in \{0.05, 0.1, 0.15, 0.2\}$ is the perturbation rate. Following [24], we compute the objective on training nodes.

Moreover, for defenders, we use the default setting of GCN, GAT, SimPGCN, and Pro-GNN in the authors' implementation. For RGCN, we tune the number of hidden units from $\{16, 32, 64, 128\}$. For GNN-Jaccard, we tune the similarity threshold for removing dissimilar edges from $\{0.01, 0.02, 0.03, 0.04, 0.05, 1\}$. For GCN-SVD, we tune the reduced rank of poison graphs from $\{5, 10, 15, 50, 100, 200\}$. For GNAT, we utilize GCN as the training model and tune k_t, k_f, and k_e in three augmented graphs from $\{0, 1, 2\}$, $\{0, 5, 10, 15, 20\}$, and $\{0, 5, 10, 15, 20\}$, respectively.

[2]https://github.com/TheaperDeng/GNN-Attack-InfMax
[3]https://github.com/Mark12Ding/GNN-Practical-Attack

TABLE VI: Node classification performance (Accuracy±Std) on Polblogs dataset under 0.1 perturbation rate.

Attackers \ GNNs		Raw GNNs		GNN Defenders				
		GCN	GAT	GCN-SVD	RGCN	Pro-GNN	SimPGCN	GNAT\f^1
Clean Graph		(95.79±0.20)	95.22±0.41	94.84±0.07	95.34±0.15	95.33±0.12	95.56±0.17	95.70±0.09
White-box	PGD	85.78±0.76	92.09±1.93	89.12±1.27	81.52±1.06	87.08±0.11	84.04±1.44	(89.43±0.33)
	MinMax	77.38±2.83	87.02±0.96	87.58±2.33	81.16±1.54	87.68±0.94	72.06±3.26	(88.62±0.31)
Gray-box	Metattack	80.32±0.34	88.44±0.37	89.98±0.22	80.43±0.29	93.46±0.17	77.24±0.25	(93.31±0.21)
Black-box	GF-Attack	94.94±0.05	96.19±0.22	94.32±0.23	95.37±0.15	95.42±0.08	94.87±0.24	(95.62±0.12)
	PEEGA	**72.57±0.88**	**81.15±0.65**	**80.23±0.26**	**74.18±0.22**	**75.26±0.66**	**71.51±0.88**	(**82.61±0.21**)

[1] GCN-Jaccard and our feature augmented graph cannot be directly applied on Polblogs because its node features are an identity matrix, where the similarity measurement is not applicable.

B. Effectiveness Evaluation

We report the performance of GNN attackers and GNN defenders in Tab. IV, V, and VI. The row of clean graph reports the performance of raw GNNs and GNN defenders trained on the clean graph. Others report the performance of a specific GNN (in the column) trained on the poison graph produced by the GNN attacker (in the row). First, attackers target to make the raw GNNs or GNN defenders achieve the low accuracy, i.e., the lower performance, the more powerful an attacker is. Bold number indicates the lowest accuracy among attack models under the same GNN defender (the more bold number has, the stronger attacker's ability is). Second, GNN defenders target to achieve high accuracy on the clean or poison graphs. Under one attacker, we mark the highest accuracy among GNN defenders with "()", i.e., the more number of "()", the stronger ability of the GNN model is.

1) Evaluation on the Performance of Attackers: Note that under one GNN defender, the lower accuracy indicates the better attack performance of a GNN attacker. First, almost all GNNs achieve the highest accuracy on the clean graph, which indicates that GNN attackers can reduce the performance of GNN more or less. Compared with white-box and gray-box attackers, the black-box attacker GF-Attack marginally reduces performance. That is because GF-Attack attacks graphs by maximizing the higher rank value of node representations, which is not directly degrade the performance of GNNs. It is worth noting that Metattack and PEEGA generally outperform white-box attackers MinMax and PGD. That is because, unlike Metattack and PEEGA that greedily select attacks, MinMax and PGD first compute the score of each candidate attack and then choose all attacks at one time, which does not consider the dependency among attacks.

Furthermore, PEEGA only needs $\mathbf{A}$ and $\mathbf{X}$ as model inputs while Metattack requires one more input: the node label $\mathbf{Y}$. But PEEGA achieves comparable results in Cora, and the best attacking performance on Citeseer and Polbogs. This is because PEEGA measures the node difference before and after attack (see Sec. III-A), which is most related to model predictions. Note that PEEGA treats the node features at each dimension uniformly, while Metattack can measure the importance of node features in each dimension via the guidance of node labels. In Cora, only a part of node features may be relevant to the node labels, i.e., these unproductive features may introduce some noises into PEEGA. As for Polbogs, it only has an identity matrix as node features, with each node having only one critical identical feature. As a result, PEEGA outperforms Metattack significantly.

TABLE VII: The running time (seconds) of attackers under perturbation rate of 0.1. **Bold number** means the best one and underline number means the second better one.

	Cora	Citeseer	Polblogs
PGD	$\underline{28.87 \pm 1.80}$	$\underline{26.18 \pm 0.86}$	$\mathbf{8.13 \pm 0.07}$
MinMax	50.52 ± 3.11	47.34 ± 1.28	$\underline{12.74 \pm 0.32}$
Metattack	439.09 ± 7.89	378.42 ± 4.22	630.61 ± 11.93
GF-Attack	890.77 ± 10.85	1245.53 ± 40.63	252.97 ± 11.01
PEEGA	$\mathbf{18.76 \pm 0.80}$	$\mathbf{15.42 \pm 0.37}$	18.17 ± 0.36

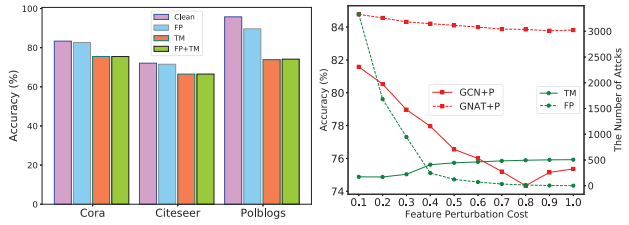
2) Evaluation on the Performance of Defenders: Note that under one GNN attacker, the higher accuracy indicates better defense performance. GCN-Jaccard and GCN-SVD perform worse among GNN defenders because simply preprocessing poison graphs cannot recover graphs from crafted poison graphs [41] and make learned node representations more distinguishable. Because of the error propagation problem, RGCN, Pro-GNN, and SimPGCN cannot perform better. They build a GNN model on the poison graph and then utilize the trained GNN to defend against attacks by re-weighting edges or reconstructing the graph structures. However, they cannot prevent the negative effect of attacks on the start of GNN training, which propagates error to the process of edge re-weighting and graph structure reconstruction [65], [66]. In this paper, Sec. IV-A analyzes the tendencies of GNN attackers, i.e., adding edges between nodes of difference classes to make nodes indistinguishable. Therefore, GNAT proposes three augmentation graphs (i.e., adds edges into the poison graphs) to against such attacks. As a result, GNAT is able to recognize nodes with different labels, and consistently outperforms defense baselines under the clean graph and the poison graphs produced by different GNN attackers.

C. Efficiency Evaluation

1) Evaluation on the Efficiency of Attackers: Tab. VII illustrates the poison graph generating time of each attack model on the three datasets under the perturbations rate of 0.1. GF-Attack is much slower than others methods because it utilizes SVD to evaluate the effect of each candidate edge, which is time-consuming. As discussed in Sec. II-B, the white-box attackers (PGD and MinMax) and gray-box attackers (Metattack) formulate the bi-level objective, which require updating the GNN parameters in low-level. But PEEGA formulates a single-level optimization goal (see Def. 3) and proposes an efficient greedy-based algorithm, thereby it is generally more efficient than PGD, MinMax, and Metattack. In particular, PEEGA is a little slow in Polbogs because it targets to enlarge the difference between node representations

TABLE VIII: The training time (seconds) evaluation of defense models on the three datasets. **Bold number** means the best one and underline number means the second better one.

	Cora	Citeseer	Polblogs
GCN	$\mathbf{0.56 \pm 0.16}$	$\mathbf{0.49 \pm 0.13}$	$\mathbf{0.55 \pm 0.08}$
GAT	2.02 ± 0.42	1.89 ± 0.35	2.31 ± 0.30
GCN-Jaccard	1.20 ± 0.19	1.11 ± 0.30	1.49 ± 0.40
GCN-SVD	7.01 ± 1.10	7.73 ± 0.67	5.43 ± 0.47
RGCN	1.14 ± 0.07	1.12 ± 0.06	1.12 ± 0.11
Pro-GNN	1326.22 ± 25.15	878.11 ± 10.09	330.07 ± 4.49
SimPGCN	2.82 ± 0.20	2.27 ± 0.28	2.45 ± 0.26
GNAT	0.98 ± 0.10	0.87 ± 0.08	0.81 ± 0.07



(a) The accuray of different attacks. (b) Evalutaion on Cora.

Fig. 5: Ablation study on different attacks in PEEGA.

and its neighbors. But Polblogs is denser than the other two graphs, which takes more time.

2) Evaluation on the Efficiency of Defenders: Tab. VIII illustrates the training time of defenders on the three datasets. We only report their running time on the clean graphs since similar observations are obtained in poison graphs. We observe that our approach is slightly slower than GCN. That is because GNAT creates three augmented graphs and train them on the GCN model. But GNAT is efficient than other baselines due to their customized designs. Specifically, GAT and RGCN incorporate the attention mechanism to measure the weights among connected nodes, SimPGCN needs to construct the similarity graph topology matrix in the training process, and GCN-SVD uses SVD to approximate a low-rank graph topology. Moreover, Pro-GNN jointly optimizes GNN parameters and learns graph structure, which is heavily time-consuming.

D. Ablation Study

1) Evaluation on Different Attacks in PEEGA: To evaluate the effect of topology modifications and feature perturbations, we use GCN to evaluate PEEGA, i.e., GCN+P, under the perturbation rate $r = 0.1$. We mainly compare three variants: FP, TM, and TM+FP, implying that PEEGA attacks graphs exclusively via the feature perturbation (FP), topology modification (TM), and their combinations (FP+TM). As illustrated in Fig. 5 (a), TM and TM+FP produce almost the same effective results, indicating that FP contributes little in generating effective attacks. We infer that such observation is caused by the settings of existing attackers. In the attack budget δ, existing works assume that the costs of feature perturbations and edge modifications are equal, i.e., $\left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0 + \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_0 \leq \delta$. However, each edge modification affects the message passing of node features, whereas each feature perturbation affects only one feature dimension. Thus, feature perturbations are less effective than topology modifications under the same cost.

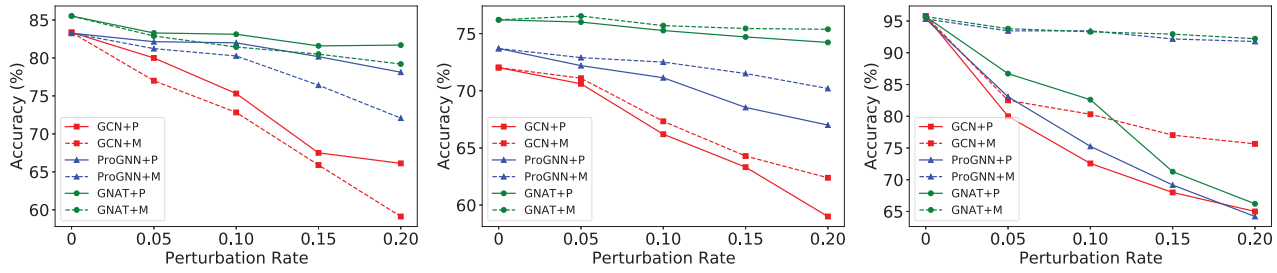
TABLE IX: Ablation study of defenders. Bold number indicates the best performance.

	Cora	Citeseer	Polblogs
GNAT-t	82.28 ± 0.19	74.13 ± 0.10	79.72 ± 0.56
GNAT-f	71.16 ± 0.09	72.20 ± 0.37	-
GNAT-e	76.29 ± 0.64	68.09 ± 0.53	73.42 ± 0.76
GNAT-t+f	82.68 ± 0.21	74.38 ± 0.20	-
GNAT-t+e	82.75 ± 0.76	73.83 ± 0.25	$\mathbf{82.61 \pm 0.21}$
GNAT-f+e	78.99 ± 0.24	73.68 ± 0.15	-
GNAT-t+f+e	$\mathbf{83.12 \pm 0.43}$	$\mathbf{75.27 \pm 0.19}$	-
GNAT-tf	80.08 ± 0.01	73.06 ± 0.05	-
GNAT-te	80.16 ± 0.64	70.36 ± 0.85	76.34 ± 0.33
GNAT-fe	71.83 ± 0.33	72.18 ± 0.12	-
GNAT-tfe	82.91 ± 0.13	73.82 ± 0.54	-

To further verify this idea, we conduct an ablation study to investigate the influence of different feature perturbation cost, such that $\left\|\hat{\mathbf{A}} - \mathbf{A}\right\|_0 + \beta \left\|\hat{\mathbf{X}} - \mathbf{X}\right\|_0 \leq \delta$ where $\beta \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$, and normalize the feature perturbation score $\mathbf{S}_f = \mathbf{S}_f / \beta$. In Fig. 5 (b) (check the green line and right y-axis), as β increases, the number of feature modifications (the green dot-line) decreases, and the number of topology modifications (the green solid-line) increases. Then, GCN and GNAT are evaluated on the poison graphs produced by PEEGA with various β, i.e., GCN+P and GNAT+P. We observe that the accuracy of GCN (the red solid-line) decreases initially and then increases. This indicates that when the feature perturbation cost is appropriate, the combinations of feature and topology attacks are the most effective. Additionally, GNAT (the red dot-line) performs consistently as the best one, indicating that it can defense against a wide variety of feature perturbations and topology modifications. We only report experimental results on Cora since similar results are made on other data sets.

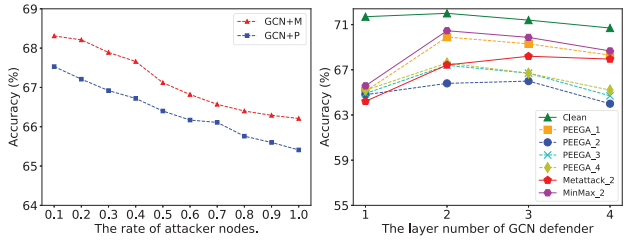
2) Evaluation on Augmented Graphs: Here we evaluate the effectiveness of different augmented graphs in GNAT. We poison graphs by PEEGA at 0.1 perturbation rate. In Sec. IV-B, we propose three augmented graphs, i.e., topology graph (t), feature graph (f), and ego graph (e), to make nodes more distinguishable. We create three type variants of GNAT: (1) Single graph: GCN is trained on the single graph, i.e., GNAT-t, GNAT-f, and GNAT-e. (2) Multiple graphs: GCN is trained on multiple graphs, i.e., GNAT-t+f, GNAT-t+e, GNAT-f+e, and GNAT-t+f+e. (3) Merged graph: to verify the help of correlated views, we merge multiple augmented graphs into one graph that contains all edges in the multiple graphs. We obtain four variants, i.e., GNAT-tf, GNAT-te, GNAT-fe, and GNAT-tfe.

Evaluations on three datasets are shown in Tab. IX. Note that the feature graph (f) for Polblogs is not available, since the node features of Polblogs are an identity matrix and the cosine similarity between nodes is all zero. First, the variants trained on multiple graphs are better than their subset version on any data set. For example, GNAT-f+e is better than GNAT-f and GNAT-e, and GNAT-t+f+e is better than GNAT-t+f, GNAT-t+e, and GNAT-f+e. This indicates that any proposed augmented graph can improve the performance of GNNs. Moreover, the variants trained in merged graphs are worse than



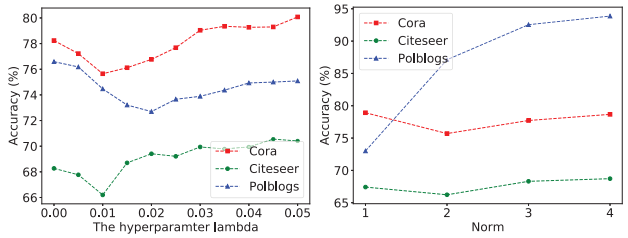
(a) Cora dataset. (b) Citeseer dataset. (c) Polblogs dataset.

Fig. 6: The evaluation of attackers and defenders under different perturbation rate.



(a) Effect of contraint nodes. (b) Effect of layer number.

Fig. 7: Hyperparamter evaluation on Citeseer dataset.



(a) The hyperparamter λ. (b) The norm p.

Fig. 8: Hyperparamter evaluation of PEEGA.

their corresponding variants in multiple graphs. For example, GNAT-te is lower than GNAT-t+e on three datasets. It indicates that providing different but correlated graph views instead of merging then into one graph can improve performance.

E. Parameter Sensitivity Evaluation

1) Perturbation Rate Evaluation on Attack and Defense Model: Recall that the attacking budget $\delta = r \cdot \|\mathbf{A}\|_0$ is controlled by the perturbation rate r. Here we study the influence of different perturbation rates $r \in \{0, 0.05, 0.1, 0.15, 0.20\}$ as shown in Fig. 6. For simplicity and clarification, we only include the representative GCN and the best baselines in Tab. IV, V, and VI, i.e., the GNN attacker Metattack and GNN defender Pro-GNN. Specifically, we first utilize PEEGA and Metattack to generate poison graphs, and then train GCN, Pro-GNN, and GNAT on them. We utilize [GNN]+[Attack] to denote each variant. For example, GCN+M and GCN+P indicate that GCN is trained on the poison graph generated by Metattack and PEEGA, respectively.

Generally, the performance of all GNN models decreases with the increase of the perturbation rate. In the attack view, PEEGA can reduce performance more than Metattack on Citeseer and Polblogs, i.e., the solid-line (PEEGA) is under the dot-line (Metattack) in the same color (one defender). In Cora dataset, PEEGA has a little inferior attack performance than Metattack, i.e., the solid-line (PEEGA) is above the dot-line (Metattack) in the same color (one defender). As discussed in Sec. V-B1, Metattack can measure the relevance of node features in each dimension regarding node labels, whereas PEEGA is not. In the defense view, GNAT can generally achieve better performance on all datasets, i.e., green line (GNAT) is above the blue line (ProGNN) and red line (GCN). Note that GNAT+P does perform well on the Polbogs dataset. That is because feature augmented graph is not available on Polbogs, which decreases the power of GNAT. In summary, GNAT performs more stable than GCN and Pro-GNN on different perturbation rates.

2) Parameter Sensitivity on the Rate of Attacker Nodes: As discussed in Sec. II-B and Tab. I, some of existing attackers set the predefined nodes that attackers have the access to modify. Here we test the effectiveness of PEEGA on the scenario of predefined available attacker nodes. The rate of accessible nodes ranges from 0.1 to 1. We first use PEEGA and Metattack to attack the accessible nodes, and then use GCN to defend against PEEGA (GCN+P) and Metattack (GCN+M). As shown in Fig. 7 (a), as the rate of accessible nodes increases, the accuracy of GCN under both attackers decreases, which indicates the power of attackers will increase with more accessible nodes. Moreover, at the same rate, PEEGA (blue dot-line) outperforms Metattack (red dot-line), which demonstrates its effectiveness.

3) Parameter Sensitivity on Layer Number: In Eq. (7), we use $\mathbf{A}_n^2\mathbf{X}$ to surrogate GNNs. To show the generality of PEEGA, we evaluate the variants of PEEGA to surrogate the information propagation with different hops of neighborhoods, i.e., $\mathbf{A}_n\mathbf{X}$, $\mathbf{A}_n^2\mathbf{X}$, $\mathbf{A}_n^3\mathbf{X}$, and $\mathbf{A}_n^4\mathbf{X}$. We first use PEEGA with different layers (i.e., PEEGA_l), Metattack [24], and MinMax [31] to poison graphs, and then use GCN with various layers (x-axis) to evaluate them. Note that for clarity, we only plot Metattack and MinMax with 2 layers, because these two variants can achieve the best or similar performance compared with their other variants, respectively. As shown in Fig. 7 (b), the PEEGA variants with $l = 2, 3, 4$ generally achieve the better or competitive performance compared with Metattack and MinMax in attacking different GCNs.

Moreover, PEEGA_2 performs the best among PEEGA

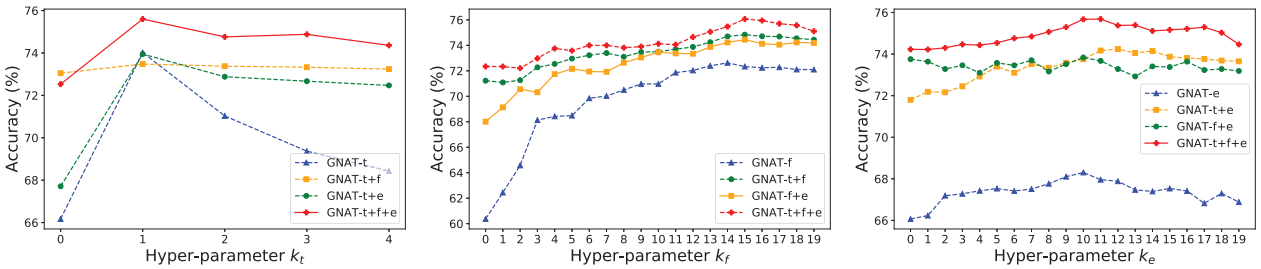


Fig. 9: Hyperparamter evaluation of our proposed defender GNAT on the Citeseer dataset.

variants. That is because the node label is largely determined by its local structure, as evidenced by GCN's accuracy on clean graphs. It indicates that node representations learned from the two hops can distinguish nodes from others with different labels, while exploring neighbors in more hops due to the over-smoothing problem [67], [68], [69]. Correspondingly, PEEGA_2 is the most effective because it destroys the best node representations that can reveal node labels. We can observe that PEEGA_1 performs poorly. This is because PEEGA_1 only considers the 1-hop neighbors of each node, which is insufficient for conducting effective attacks.

4) Parameter Sensitivity on λ and p : We use GCN to evaluate the effect of the trade-off parameter λ between self-view and global view, and the norm of distance p in Eq. (8). In Fig. 8 (a), as λ increases, the performance of GCN on three datasets first decreases and then increases. It indicates that enlarging representation differences between nodes and its neighbors can increase the attack power, since it potentially let nodes with the same label have different representations. Besides, larger λ will overvalue neighbors, which may cause nodes with different labels to become more distinct and thus reduce attack power, due to the fact that not all neighbors have the same label with each node. Particularly, the best λ on Cora and Citeseer is smaller than Polbogs, because the proportion of edges whose connected nodes have the same label is smaller than that in Polbogs. Besides, in Fig. 8 (b), the best choices for p on Cora and Citeseer are both 2, while the best p for Polbogs is 1. It is because node features in Polbogs are an identity matrix, where each node have a unique dimension. Therefore, the semantic meaning of node representation is more sensitive to the aggregated feature difference.

5) Parameter Sensitivity on Augmented Graphs: We evaluate the effect of k_t (topology graph), k_f (feature graph), and k_e (ego graph) of GNAT. Specifically, we use PEEGA to generate poison graphs under 0.1 perturbation rate. Since similar observations are made on other graphs, we only report results on Citeseer. The default setting of $\{k_t, k_f, k_e\}$ is $\{2, 15, 10\}$. When we evaluate one parameter, we set others as default. As shown in Fig. 9, as three parameters increase, the accuracy of GNAT trained on their single graph or combinations first increases and then decreases. It demonstrates that our defender can make nodes more distinguishable by connecting nodes with the same labels. Besides, as k_t and k_f increase, nodes will connect more dissimilar nodes, which introduces noise. As k_e increases, the GNN will emphasize the own feature of nodes and overlook the information from its local structure. Therefore, the accuracy will decrease.

VI. Conclusion and Future Work

In this paper, we propose PEEGA, a pure black-box GNN attacker that only accesses graph topology and node features when attacking graphs. To enable both topology modifications and feature perturbations, we propose to qualify the negative impact of attacks on node representations and then formulate a single-level objective. Moreover, we observe that effective attackers degrade GNN performance by obscuring node context. We then propose GNAT, a GNN defender based on graph augmentation, which improves the robustness of GNNs by making node context more distinguishable. The effectiveness and efficiency of our proposed attacker and defender are shown by experiments on three real-world datasets.

As for future works, this work can be further improved. Due to the sequential attack process in Alg. 1, the time complexity increases linearly with the size of budget. Inspired by PTDNet [70], Gumbel-Softmax sampling [71], which samples attacks in a parallel manner, is a potential solution to make the attack process more efficient. Recently, such technique has been employed into another bi-level optimization problem in GNNs [72], i.e., automated GNN designs [73], [74], [75]. Second, the proposed defender GNAT only adds edges into the poison graph to defend against adversarial attacks. We may remove some noises in the poison graph introduced by attackers. Leveraging the knowledge of adding and removing can further boost the performance of GNAT. We leave these improvements in the future work.

VII. Acknowledgement

This work is partially supported by National Key Research and Development Program of China Grant No. 2018AAA0101100, the Hong Kong RGC GRF Project 16202218, CRF Project C6030-18G, C1031-18G, C5026-18G, AOE Project AoE/E-603/18, RIF Project R6020-19, Theme-based project TRS T41-603/20R, China NSFC No. 61729201, Guangdong Basic and Applied Basic Research Foundation 2019B151530001, Hong Kong ITC ITF grants ITS/044/18FX and ITS/470/18FX, Microsoft Research Asia Collaborative Research Grant, HKUST-NAVER/LINE AI Lab, Didi-HKUST joint research lab, HKUST-Webank joint research lab grants.

REFERENCES

[1] B. D. McKay and A. Piperno, "Practical graph isomorphism, ii," *Journal of symbolic computation*, vol. 60, pp. 94–112, 2014.

[2] P. Ribeiro, P. Paredes, M. E. Silva, D. Aparicio, and F. Silva, "A survey on subgraph counting: Concepts, algorithms and applications to network motifs and graphlets," *arXiv preprint arXiv:1910.13011*, 2019.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[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] H. Liu, S. Lu, X. Chen, and B. He, "G3: when graph neural networks meet parallel graph processing systems on gpus," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2813–2816, 2020.

[7] R. Zhu, K. Zhao, H. Yang, W. Lin, C. Zhou, B. Ai, Y. Li, and J. Zhou, "Aligraph: a comprehensive graph neural network platform," *arXiv preprint arXiv:1902.08730*, 2019.

[8] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, "Parameterized explainer for graph neural network," *arXiv preprint arXiv:2011.04573*, 2020.

[9] H. Li and L. Chen, "Cache-based gnn system for dynamic graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 937–946.

[10] W. Zhang, Y. Shen, Y. Li, L. Chen, Z. Yang, and B. Cui, "Alg: Fast and accurate active learning framework for graph convolutional networks," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 2366–2374.

[11] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" *arXiv preprint arXiv:1810.00826*, 2018.

[12] C. T. Duong, T. D. Hoang, H. Yin, M. Weidlich, Q. V. H. Nguyen, and K. Aberer, "Efficient streaming subgraph isomorphism with graph neural networks," *Proceedings of the VLDB Endowment*, vol. 14, no. 5, pp. 730–742, 2021.

[13] X. Liu and Y. Song, "Graph convolutional networks with dual message passing for subgraph isomorphism counting and matching," *arXiv preprint arXiv:2112.08764*, 2021.

[14] X. Liu, H. Pan, M. He, Y. Song, X. Jiang, and L. Shang, "Neural subgraph isomorphism counting," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 1959–1969.

[15] Z. Chen, L. Chen, S. Villar, and J. Bruna, "Can graph neural networks count substructures?" *arXiv preprint arXiv:2002.04025*, 2020.

[16] Z. Xing and S. Tu, "A graph neural network assisted monte carlo tree search approach to traveling salesman problem," *IEEE Access*, vol. 8, pp. 108 418–108 428, 2020.

[17] M. Prates, P. H. Avelar, H. Lemos, L. C. Lamb, and M. Y. Vardi, "Learning to solve np-complete problems: A graph neural network for decision tsp," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4731–4738.

[18] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "Graph neural networks for recommender systems: Challenges, methods, and directions," *arXiv preprint arXiv:2109.12843*, 2021.

[19] Z. Wang, H. Zhao, and C. Shi, "Profiling the design space for graph neural networks based collaborative filtering."

[20] Y. Zhang and Q. Yao, "Knowledge graph reasoning with relational directed graph," *arXiv preprint arXiv:2108.06040*, 2021.

[21] Anonymous, "Message function search for hyper-relational knowledge graph," in *Submitted to The Tenth International Conference on Learning Representations*, 2022, under review. [Online]. Available: https://openreview.net/forum?id=CQzlxFVcmw1

[22] W. Zhang, X. Miao, Y. Shao, J. Jiang, L. Chen, O. Ruas, and B. Cui, "Reliable data distillation on graph convolutional network," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1399–1414.

[23] L. Wei, H. Zhao, and Z. He, "Designing the topology of graph neural networks: A novel feature fusion perspective," *arXiv preprint arXiv:2112.14531*, 2021.

[24] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," *arXiv preprint arXiv:1902.08412*, 2019.

[25] 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.

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

[27] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, L. He, and B. Li, "Adversarial attack and defense on graph data: A survey," *arXiv preprint arXiv:1812.10528*, 2018.

[28] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv e-prints*, pp. arXiv–2003, 2020.

[29] X.-Y. Li, C. Zhang, T. Jung, J. Qian, and L. Chen, "Graph-based privacy-preserving data publication," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[30] S. De Capitani Di Vimercati, S. Foresti, G. Livraga, and P. Samarati, "Data privacy: definitions and techniques," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 20, no. 06, pp. 793–817, 2012.

[31] 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," *arXiv preprint arXiv:1906.04214*, 2019.

[32] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples on graph data: Deep insights into attack and defense," *arXiv preprint arXiv:1903.01610*, 2019.

[33] J. Ma, S. Ding, and Q. Mei, "Black-box adversarial attacks on graph neural networks with limited node access," *arXiv preprint arXiv:2006.05057*, 2020.

[34] J. Ma, J. Deng, and Q. Mei, "Near-black-box adversarial attacks on graph neural networks as an influence maximization problem," 2021. [Online]. Available: https://openreview.net/forum?id=sbyjwhxxT8K

[35] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang, "A restricted black-box adversarial framework towards attacking graph embedding models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3389–3396.

[36] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *International conference on machine learning*. PMLR, 2018, pp. 1115–1124.

[37] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, and J. Tang, "Adversarial attacks and defenses on graphs," *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 19–34, 2021.

[38] X. Xu, Y. Yu, B. Li, L. Song, C. Liu, and C. Gunter, "Characterizing malicious edges targeting on graph neural networks," 2018.

[39] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 1399–1407.

[40] X. Zhang and M. Zitnik, "Gnnguard: Defending graph neural networks against adversarial attacks," *arXiv preprint arXiv:2006.08149*, 2020.

[41] 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.

[42] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 148–156.

[43] A. Zhang and J. Ma, "Defensevgae: Defending against adversarial attacks on graph data via a variational graph autoencoder," *arXiv preprint arXiv:2006.08900*, 2020.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[45] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[46] W. Huang *et al.*, "Adaptive sampling towards fast graph representation learning," in *Advances in neural information processing systems*, 2018, pp. 4558–4567.

[47] W. Huang, Y. Rong, T. Xu, F. Sun, and J. Huang, "Tackling over-smoothing for general graph convolutional networks," *arXiv preprint arXiv:2008.09864*, 2020.

[48] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, 2020.
[49] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
[50] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.
[51] T. Rowland, "Operator norm." [Online]. Available: https://mathworld.wolfram.com/OperatorNorm.html
[52] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" *arXiv preprint arXiv:2106.06134*, 2021.
[53] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the Web Conference 2021*, 2021, pp. 2069–2080.
[54] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," *arXiv preprint arXiv:2006.06830*, 2020.
[55] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
[56] 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.
[57] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.
[58] S. Nandanwar and M. N. Murty, "Structural neighborhood based classification of nodes in a network," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1085–1094.
[59] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.
[60] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.
[61] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, 2005, pp. 36–43.
[62] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," *arXiv preprint arXiv:2005.06149*, 2020.
[63] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang, "A restricted black-box adversarial framework towards attacking graph embedding models," 2019.
[64] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank) defending against adversarial attacks on graphs," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 169–177.
[65] B. Fatemi, L. E. Asri, and S. M. Kazemi, "Slaps: Self-supervision improves structure learning for graph neural networks," *arXiv preprint arXiv:2102.05034*, 2021.
[66] Y. Wang, S. Mukherjee, H. Chu, Y. Tu, M. Wu, J. Gao, and A. H. Awadallah, "Adaptive self-training for few-shot neural sequence labeling," *arXiv preprint arXiv:2010.03680*, 2020.
[67] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," *arXiv preprint arXiv:1907.10903*, 2019.
[68] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun, "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3438–3445.
[69] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 338–348.
[70] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 779–787.
[71] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.
[72] Z. Wang, S. Di, and L. Chen, "Autogel: An automated graph neural network with explicit link information," *Advances in Neural Information Processing Systems*, vol. 34, 2021.
[73] Z. Huan, Y. Quanming, and T. Weiwei, "Search to aggregate neighborhood for graph neural network," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 552–563.
[74] L. Wei, H. Zhao, Q. Yao, and Z. He, "Pooling architecture search for graph classification," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 2091–2100.
[75] X. Wang, Z. Zhang, and W. Zhu, "Automated graph machine learning: Approaches, libraries and directions," *arXiv preprint arXiv:2201.01288*, 2022.