Efficient, Direct, and Restricted Black-Box Graph Evasion Attacks to Any-Layer Graph Neural Networks via Influence Function

Binghui Wang*
bwang70@iit.edu
Illinois Institute of Technology
Chicago, USA

Minhua Lin*
mfl5681@psu.edu
The Pennsylvania State University
State College, USA

Tianxiang Zhou*
tianxiangzhou@hust.edu.cn
Huazhong University of Science and Technology
Wuhan, China

Pan Zhou
panzhou@hust.edu.cn
Huazhong University of Science and Technology
Wuhan, China

Ang Li
angliece@umd.edu
University of Maryland
College Park, USA

Meng Pang
mengpang@ncu.edu.cn
Nanchang University
Nanchang, China

Hai Li
hai.li@duke.edu
Duke University
Durham, USA

Yiran Chen
yiran.chen@duke.edu
Duke University
Durham, USA

ABSTRACT

Graph neural network (GNN), the mainstream method to learn on graph data, is vulnerable to graph evasion attacks, where an attacker slightly perturbing the graph structure can fool trained GNN models. Existing work has at least one of the following drawbacks: 1) limited to directly attack two-layer GNNs; 2) inefficient; and 3) impractical, as they need to know full or part of GNN model parameters.

We address the above drawbacks and propose an influence-based *efficient, direct, and restricted black-box* evasion attack to *any-layer* GNNs. Specifically, we first introduce two influence functions, i.e., feature-label influence and label influence, that are defined on GNNs and label propagation (LP), respectively. Then we observe that GNNs and LP are strongly connected in terms of our defined influences. Based on this, we can then reformulate the evasion attack to GNNs as calculating label influence on LP, which is *inherently* applicable to any-layer GNNs, while no need to know information about the internal GNN model. Finally, we propose an efficient algorithm to calculate label influence. Experimental results on various graph datasets show that, compared to state-of-the-art white-box attacks, our attack can achieve comparable attack performance, but has a 5-50x speedup when attacking two-layer GNNs. Moreover, our attack is effective to attack multi-layer GNNs[1].

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

graph neural network, label propagation, attack, influence function

ACM Reference Format:
Binghui Wang, Minhua Lin, Tianxiang Zhou, Pan Zhou, Ang Li, Meng Pang, Hai Li, and Yiran Chen. 2024. Efficient, Direct, and Restricted Black-Box Graph Evasion Attacks to Any-Layer Graph Neural Networks via Influence Function. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM '24), March 4–8, 2024, Merida, Mexico.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3616855.3635826

1 INTRODUCTION

Learning with graph data, such as social networks, biological networks, financial networks, has drawn continuous attention recently. Graph neural network (GNN) has become the mainstream methodology for representation learning on graphs. GNN was first introduced in [28], which extended conventional neural network to process graph data. Then, various GNN methods have been proposed and achieved state-of-the-art performance in many graph-related tasks such as node classification [20, 36, 45], graph classification [15, 16], and link prediction [47]. However, recent works [8, 12, 25, 26, 31, 37, 39, 40, 42, 44, 54, 55] show that GNNs are vulnerable to graph evasion attacks—Given a target node and a trained GNN model, an attacker slightly perturbing the graph structure[2] (e.g., add new edges to or delete existing edges from the graph) can make the GNN model misclassify the target node. Existing attacks to GNNs can be roughly classified as *optimization-based* attacks [39, 42, 44, 54] and *reinforcement learning (RL)*-based attacks [8, 12, 31].

*These authors contribute equally to this paper

[1] Source code and full version is in the link: https://github.com/ventr1c/InfAttack

[2] An attacker can also perturb node features to perform the attack. However, structure perturbation is shown to be much more effective than node feature perturbation [54].

WSDM '24, March 4–8, 2024, Merida, Mexico

ACM ISBN 979-8-4007-0371-3/24/03…$15.00
https://doi.org/10.1145/3616855.3635826

In this paper, we focus on optimization-based attacks, as they are shown to be more effective [54]. Optimization-based attacks first formulate the graph evasion attack as a binary optimization problem, which is challenging to solve, and then design approximate algorithms to solve a tractable optimization problem. Although achieving promising attack performance, existing optimization-based attacks have one or more of the below key limitations:

- First, most of the existing attacks need to know the full/partial GNN model parameters, which is unrealistic in many real-world applications, e.g., when GNN models are confidential due to their commercial value and are deployed as an API. Thus, the practicability of the existing attacks are limited. Further, they are mainly designed to attack *two-layer GNNs*, while GNNs are multi-layer in essence. To attack multi-layer GNNs, they often first *indirectly* attack a surrogate two-layer GNN model, and then transfer the attack to the target multi-layer GNN. However, this strategy is not effective enough (See Figure 4(a) in Section 5).
- Second, they are not efficient, as they involve intensive computation, i.e., by multiplying GNN model parameters of different layers and with node feature matrix. If a GNN has many layers, such computation can be a bottleneck, especially for attackers who have limited computational resources or/and want to perform real-time attacks. For example, many fraud detection systems, such as detecting fake users in social networks and detecting anomalies from system logs, are updated frequently in order to reduce the loss caused by the evasion attacks' malicious activities. In these scenarios, efficiency is a major concern for the attack and an attacker performing efficient attacks is necessary, as otherwise the detection system may have already updated and identified the attack's malicious patterns before the attack is implemented.

Our work: We aim to address the above limitations in this paper. To this end, we propose an optimization-based evasion attack against any-layer GNNs based on influence function [21]—a completely different perspective from the existing works. Our influence-based attack is motivated by the strong connection between GNNs and label propagation (LP) [52]. Specifically, we first introduce two influence functions, i.e., feature-label influence and label influence, that are defined on GNNs and LP, respectively. Then, we prove that our label influence defined on LP is equivalent to feature-label influence on a particular well-known type of GNN, called Graph Convolutional Network (GCN) [20] (and its linearized version Simple Graph Convolutional (SGC) [41]). Based on this connection, we reformulate the evasion attack against GNNs to be related to calculating label influence on LP. As our influences are designed for any-layer GNNs, our attack is inherently applicable to attack any-layer GNNs. Note that label influence can be computed easily and we also design an efficient algorithm to compute it. Further, as our influence-based attack does not need to know any information about the GNN model (except the target node's neighboring information), it is a more practical (restricted black-box) attack. Finally, we evaluate our attack against GCN/SGC on three benchmark graph datasets. Compared to the state-of-the-art white-box attacks against two-layer GCN/SGC, our attack can achieve comparable attack performance but has a 5-50x speedup. Our attack is more effective to attack multi-layer GCN/SGC. For instance, our attack achieves a 93% attack success rate, when perturbing 4 edges per target node on Cora, while the surrogate model based attack only has 80% attack success rate. As a by product, our attack also shows promising transferability to attack other GNNs, and is more effective than existing black-box attacks.

Our contributions can be summarized as follows:

- We propose graph evasion attacks to GNNs based on influence function, which is a completely new perspective.
- Our attack is effective, direct, efficient, and practical.
- Our attack has promising transferability.

2 RELATED WORK

Attacks to graph neural networks. Existing attacks to GNNs can be classified as graph *poisoning attacks* [7, 8, 24, 31, 32, 43, 44, 49, 50, 54, 55] and *evasion attacks* [8, 25, 42, 54]. In poisoning attacks, an attacker modifies the graph structure during the training process such that the trained GNN model has a low prediction accuracy on testing nodes. For instance, Xu et al. [44] developed a topology poisoning attack based on gradient-based optimization. Evasion attacks can be classified as untargeted attacks and targeted attacks, where the latter is more challenging. Given a target node and a trained GNN model, targeted attack means an attacker aims to perturb the graph structure such that the GNN model misclassifies the target node to be a target label, while untargeted attack misclassifies the target node to be an arbitrary label different from the target node's label. For instance, Dai et al. [8] leveraged reinforcement learning techniques to design non-targeted evasion attacks to both graph classification and node classification. Zügner et al. [54] proposed a targeted evasion attack, called Nettack, against two-layer GCN and achieved the state-of-the-art attack performance. Specifically, Nettack learns a surrogate linear model of GCN by removing the ReLU activation function and by defining a graph structure preserving perturbation that constrains the difference between the node degree distributions of the graph before and after attack. Our label influence-based attack is a targeted evasion attack.

Most of the existing GNN attacks are white/gray-box. Recently, two black-box attacks to GNNs [26, 39] have been proposed. For instance, Wang et al. [39] formulate the black-box attack to GNNs as an online optimization with bandit feedback. The original problem is NP-hard and they then propose an online attack based on (relaxed) bandit convex optimization which is proven to be sublinear to the query number. Our attack is a restricted black-box attack, where the attacker only needs to know the target node's neighbors.

Attacks to other graph-based methods. Besides attacking GNNs, other adversarial attacks against graph data include attacking graph-based clustering [6], graph-based collective classification [35, 37], graph embedding [1, 4, 5, 9, 30], community detection [22], etc. For instance, Chen et al. [6] proposed a practical attack against spectral clustering, which is a well-known graph-based clustering method. Wang and Gong [37] designed an optimization-based attack against the collective classification method, called linearized belief propagation, by modifying the graph structure.

Defending against graph perturbation attacks. Existing defenses against the graph perturbation attacks can be classified as empirical defenses [11, 33, 34, 42, 44, 51] and provable defenses [2, 3, 18, 23, 38]. The empirical defenses are shown to be easily broken by stronger/adaptive attacks [14, 27]. Provable defenses study certified robustness of GNNs against the worst-case graph perturbation attacks. For instance, Wang et al. [38] design a randomized smoothing-based provable defenses that achieves a tight certified robustness, when there are no assumptions about the GNN model. [38] achieves the state-of-the-art provable defense performance.

3 BACKGROUND AND PROBLEM DEFINITION

Graph Neural Network. Let $G = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ be a graph, where $u \in \mathcal{V}$ is a node, $(u, v) \in \mathcal{E}$ is an edge between u and v, and $\mathbf{X} = [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_n] \in \mathbb{R}^{n \times d}$ is the node feature matrix. We denote $\mathbf{A} = [\mathbf{a}_1; \mathbf{a}_2; \cdots; \mathbf{a}_n] \in \{0, 1\}^{n \times n}$ as the adjacency matrix, d_u and Γ_u as u's node degree and the neighborhood set of u (including self-loop (u, u)). We consider GNNs for node classification in this paper. In this context, each node u has a label y_u from a label set $\mathcal{Y} = \{1, 2, \cdots, C\}$. Given a set of $\mathcal{V}_L \subset \mathcal{V}$ labeled nodes $\{(\mathbf{x}_u, y_u)\}_{u \in \mathcal{V}_L}$ as the training set, GNN for node classification is to take the graph G and labeled nodes as input and learn a node classifier that maps each node $u \in \mathcal{V} \setminus \mathcal{V}_L$ to a class $y \in \mathcal{Y}$. In this paper, we focus on Graph Convolutional Network (GCN) [20], a widely used type of GNN, and its special case Simple Graph Convolution (SGC) [41].

GCN. GCN is motivated by spectral graph convolution [10]. Suppose GCN has K layers. We denote node v's representation in the k-th layer as $\mathbf{h}_v^{(k)}$, where $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. Then, GCN has the following form to update the node representation:

$$\mathbf{h}_v^{(k)} = \mathrm{ReLU}\Big(\mathbf{W}^{(k)}\big(\sum\nolimits_{u \in \Gamma_v} d_u^{-1/2} d_v^{-1/2} \mathbf{h}_u^{(k-1)}\big)\Big). \tag{1}$$

A node v's final representation $\mathbf{h}_v^{(K)} \in \mathbb{R}^{|\mathcal{Y}|}$ can capture the structural information of all nodes within v's K-hop neighbors. Moreover, the final node representations of training nodes are used for training the node classifier. Specifically, let $\Theta = \{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, \cdots, \mathbf{W}^{(K)}\}$ be the model parameters and v's output be $f_\Theta(\mathbf{A})_v = \mathrm{softmax}(\mathbf{h}_v^{(K)}) \in \mathbb{R}^{|\mathcal{Y}|}$, where $f_\Theta(\mathbf{A})_{v,y}$ indicates the probability of node v being class y. Then, Θ are learnt by minimizing the cross-entropy loss on the training nodes $\mathcal{V}_L$, i.e., $\Theta^* = \arg\min_\Theta - \sum_{v \in \mathcal{V}_L} \ln f_\Theta(\mathbf{A})_{v,y}$. With the learnt Θ^*, we can predict the label for each unlabeled nodes $u \in \mathcal{V} \setminus \mathcal{V}_L$ as $\hat{y}_u = \arg\max_y f_{\Theta^*}(\mathbf{A})_{u,y}$.

SGC. SGC is a linearized version of GCN. Specifically, its node representation is updated as follows:

$$\mathbf{h}_v^{(k)} = \mathbf{W}^{(k)}\big(\sum\nolimits_{u \in \Gamma_v} d_u^{-1/2} d_v^{-1/2} \mathbf{h}_u^{(k-1)}\big). \tag{2}$$

SGC has shown to have comparable node classification performance with GCN, but is much more efficient than GCN.

Label Propagation (LP). LP is a conventional semi-supervised node classification method without training. The key idea behind LP is that two nodes having a high similarity (e.g., connected nodes in a graph) are likely to have the same label. Thus, LP iteratively propagates labels among the graph to unlabeled nodes based on node-pair similarity. Let $\mathbf{y}_v \in \mathbb{R}^{|\mathcal{Y}|}$ be node v's initial label vector (For notation reason, one should note that y_v is v's categorical label). For instance, $\mathbf{y}_v$ can be v's one-hot label vector if v is a labeled node, and $\mathbf{y}_v = \mathbf{0}$, otherwise. Then, LP is formulated as follows:

$$\mathbf{y}_v^{(k)} = \sum\nolimits_{u \in \Gamma_v} d_u^{-1/2} d_v^{-1/2} \mathbf{y}_u^{(k-1)}, \quad \mathbf{y}_v^{(0)} = \mathbf{y}_v. \tag{3}$$

With K iterations, an unlabeled node u is predicted to be class c, if $c = \arg\max_i y_{u,i}^{(K)}$.

GNN vs. LP: Viewing Eqn (3) and Eqns (1) and (2), we observe that LP and GNNs have similar iterative processes: LP propagates node labels $\mathbf{y}_v$, while GNNs propagate node features $\mathbf{x}_v$. The key difference is that LP does not involve model parameters, while GNN involves multiplying the parameter matrix $\mathbf{W}^{(k)}$ in each k-th layer.

Problem Definition. We consider targeted evasion attacks[3] to GNNs. Suppose we are given a trained GNN model f_{Θ^*} for node classification. We assume v is the *target node* and c is the *target label*. We consider an attacker can perturb the graph structure (i.e., add new edges to or delete existing edges from the graph) in order to make f_{Θ^*} misclassify the target node v to be the target label c. We call the modified edges by the attacker as *attack edges*. In particular, we consider a practical *direct attack* [54], where an attacker can only modify the edge status between v and other nodes in the graph, while cannot modify the edge status among other nodes. We denote the perturbed graph as $\tilde{G}$ (with the perturbed adjacency matrix $\tilde{\mathbf{A}}$) after the attack and the attack budget as Δ, i.e., at most Δ edges can be perturbed for the target node. Then, the objective function of targeted evasion attacks to GNNs is formally defined as:

$$\begin{aligned}&\max\nolimits_{\tilde{\mathbf{A}}_v} \big(f_{\Theta^*}(\tilde{\mathbf{A}})_{v,c} - f_{\Theta^*}(\tilde{\mathbf{A}})_{v,y_v}\big) \Leftrightarrow \max\nolimits_{\tilde{\mathbf{A}}_v} \big([\tilde{\mathbf{h}}_v^{(K)}]_c - [\tilde{\mathbf{h}}_v^{(K)}]_{y_v}\big),\\ &s.t., \quad \sum\nolimits_s |\tilde{A}_{v,s} - A_{v,s}| \leq \Delta,\end{aligned} \tag{4}$$

where $\tilde{\mathbf{h}}_v^{(K)}$ is v's representation on the perturbed graph $\tilde{G}$.

A target node is called a success to attack the GNN model if the value of the attack's objective function is larger than 0, under the attack budget. Note that Equation (4) is a binary optimization problem and is challenging to solve in practice. Zügner et al. [54] proposed an optimization-based attack method, called Nettack, against two-layer GCN. Specifically, Nettack attacked a substitute GNN model (actually SGC) that removed the ReLU activation function in GCN. Nettack has achieved state-of-the-art attack performance. However, it is inefficient as it involves dense matrix multiplication (i.e., model parameters multiply node features); it also needs to know model parameters Θ^*, and can only attack two-layer GNNs.

4 INFLUENCE-BASED EVASION ATTACK

In this section, we propose our evasion attack against GNNs via influence function. In contrast to existing optimization-based attacks that only focus on two-layer GNNs, our attack is applicable to any-layer GNNs. Specifically, we first define two influence functions associated with GNNs and LP, respectively, and build an equivalence relation between GNNs and LP with the defined influences. Next, we reformulate the attack objective function as relating to label influence defined on LP. Finally, we design an efficient algorithm to calculate label influence and realize our attack.

4.1 Equivalence between GNNs and LP in terms of Influence

4.1.1 Motivation. Due to GNN's complex network structure, existing optimization-based evasion attacks can only attack two-layer GNNs *directly*. However, we note that LP has a similar iterative process to GNNs, but it has good properties, e.g., LP does not involve model parameters. Motivated by this, we aim to discover an

[3] As untargeted attacks are less powerful than targeted attacks, we only consider targeted attacks in this paper for simplicity.

equivalence relation between LP and GNNs, such that the challenging problem of attacking multi-layer GNNs can be converted to a relatively easier problem by leveraging good properties of LP. We notice that influence function [21, 46] is an appropriate tool to bridge the gap, and our purpose is to explore equivalent influence functions defined on LP and on GNNs, respectively. As the attacker's goal is to change the target node's label, we thus need to define influences associated with the node label. As LP propagates node labels, we can naturally design the *label influence* function (see Equation (6)). In addition, GNNs involve propagating node features. In order to also leverage node labels, we integrate both node features and labels and design the *feature-label influence* function (see Equation (5)). Next, we introduce our influence functions.

4.1.2 Influence function. Given two nodes u and v, an influence of u on v indicates how the output (e.g., final node representation in GNNs or estimated node label in LP) of v changes if the input of u is slightly perturbed. Inspired by [21, 46], we define the following feature-label influence on GNN and label influence on LP.

Definition 1 (Feature-label influence). *We define the feature-label influence of node u on node v associated with u's label on a K-layer GNN as follow:*

$$I_{fl}(v,u;K) = \Big\| \Big[\frac{\partial \mathbf{h}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)} \Big]_{y_u} \Big\|_1 = \mathbf{1}_{y_u}^T \cdot \frac{\partial \mathbf{h}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)}, \tag{5}$$

where $\mathbf{1}_{y_u} = [y_1, y_2, \cdots, y_n]$ is an indicator vector where $y_i = 1$ if $i = u$ and $y_i = 0$, otherwise; $\|\cdot\|_1$ is the vector ℓ_l-norm; T is a transpose; and $\mathbf{h}_u^{(0)} = \mathbf{x}_u$ is u's node features.

Definition 2 (Label influence). *We define the label influence of node u on node v after K iterations of label propagation as follows:*

$$I_l(v,u;K) = \frac{\partial y_v^{(K)}}{\partial y_u^{(0)}}. \tag{6}$$

Then, we have the following theorem showing the equivalence between GNN and LP in terms of influence.

Theorem 4.1. *If the GNN is a GCN/SGC, then:*

$$I_{fl}(v,u;K) = C \cdot I_l(v,u;K), \tag{7}$$

where $C = \rho \mathbf{1}_{y_u}^T [\prod_{l=1}^K \mathbf{W}^{(l)}] \mathbf{x}_u$ is constant related to GNN model parameters $\Theta = \{\mathbf{W}^{(l)}\}$ and u's node features $\mathbf{x}_u$.

Theorem 4.1 reveals that: given arbitrary node v, the feature-label influence defined on K-layer GCN/SGC of any other node u on the node v and the label influence defined on K-iteration LP of node u on the node v are equal (with a constant multiplier difference).

4.2 Reformulate Evasion Attacks to Any-Layer GNNs as Calculating Label Influence on LP

Based on our influence functions and Theorem 4.1, we can first restate the challenging problem of attacking K-layer GNNs in Equation (4) in the form of feature-label influence, and further convert it to an equivalent problem related to label influence on LP. Before going into details, we first introduce the following lemma:

Lemma 1 (Xu et al.[46]). *Given a K-layer GCN. Assume all paths in the computation graph of the GCN model are activated (i.e., via ReLU) with the same probability of success ρ. Then,*

$$\frac{\partial \mathbf{h}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} = \rho \sum_{p=1}^{\Psi_{v\to u}} \prod_{l=K}^{1} a_{v_p^l, v_p^{l-1}} \cdot \mathbf{W}^{(l)}, \tag{8}$$

where $\Psi_{v\to u}$ is the total number of paths $[v_p^K, v_p^{K-1}, \cdots, v_p^1, v_p^0]$ of length $K+1$ from node v to the node u with $v_p^K = v$ and $v_p^0 = u$. For $l = 1, \cdots, K$, $v_p^{l-1} \in N\left(v_p^l\right)$, $a_{v_p^l, v_p^{l-1}} = d_{v_p^l}^{-\frac{1}{2}} d_{v_p^{l-1}}^{-\frac{1}{2}}$ is the normalized weight of the edge (v_p^l, v_p^{l-1}) in the path p. $\Theta = \{\mathbf{W}^{(l)}\}$ is the K-layer GCN model parameters.

Then, according to Equation (8) in Lemma 1, the target node v's final node representation $\tilde{\mathbf{h}}_v^{(K)}$ learnt on the perturbed graph $\tilde{G}$ can be expressed as $\tilde{\mathbf{h}}_v^{(K)} = \sum_{u \in \tilde{\Lambda}_v^{(K)}} \frac{\partial \tilde{\mathbf{h}}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)}$, where $\tilde{\Lambda}_v^{(K)}$ is the node set containing v's neighbors within K-hop on the perturbed graph $\tilde{G}$, i.e., after modifying the edge status between the target node v and other nodes in the clean graph G.

Then, the attack's objective function in Equation (4) is equivalent to the following objective function:

$$\max_{\tilde{A}_v} \Big(\Big[\sum_{u \in \tilde{\Lambda}_v^{(K)}} \frac{\partial \tilde{\mathbf{h}}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)} \Big]_c - \Big[\sum_{u \in \tilde{\Lambda}_v^{(K)}} \frac{\partial \tilde{\mathbf{h}}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)} \Big]_{y_v} \Big)$$
$$s.t., \quad \sum\nolimits_s |\tilde{A}_{v,s} - A_{v,s}| \le \Delta, \tag{9}$$

Finally, based on the following Assumption 1 and Theorem 4.1, we reach Theorem 4.2 that reformulates the evasion attack's objective function via label influence. We also conduct experiments (See Section 5.2) to verify that Assumption 1 holds in practice.

Assumption 1. *Given a target node v and a target label c. We assume that any node u, within the K-hop neighbor of v, has a negligible feature-label influence on v if u is not a label-c node. Formally,*

$$\Big[\frac{\partial \tilde{\mathbf{h}}_v^{(K)}}{\partial \mathbf{h}_u^{(0)}} \cdot \mathbf{h}_u^{(0)} \Big]_c \approx 0, \quad \forall u \in \tilde{\Lambda}_v^{(K)},\ y_u \ne c. \tag{10}$$

Theorem 4.2. *Let $\tilde{I}_l(v,u;K)$ be the label influence of node u on the target node v with K iterations of LP after the attack. Then, the attack's objective function in Equation (4) equals to the following objective function on label influence:*

$$\max_{\tilde{A}_v} \Big(\sum_{u \in \tilde{\Lambda}_v^{(K)}, y_u = c} \tilde{I}_l(v,u;K) - \sum_{z \in \tilde{\Lambda}_v^{(K)}, y_z = y_v} \tilde{I}_l(v,z;K) \Big),$$
$$s.t., \quad \sum\nolimits_s |\tilde{A}_{v,s} - A_{v,s}| \le \Delta, \tag{11}$$

where $\tilde{I}_l(v,u;K)$ is defined as:

$$\tilde{I}_l(v,u;K) = \sum_{p=1}^{\tilde{\Psi}_{v\to u}} \prod_{l=K}^{1} \tilde{d}_{v_p^l}^{-\frac{1}{2}} \tilde{d}_{v_p^{l-1}}^{-\frac{1}{2}}, \tag{12}$$

where $\tilde{\Psi}_{v\to u}$ is the total number of paths $[v_p^K, v_p^{K-1}, \cdots, v_p^1, v_p^0]$ of length $K+1$ from v to u on the perturbed graph $\tilde{G}$, where $v_p^K = v$ and $v_p^0 = u$. $\tilde{d}_u$ is u's degree on the perturbed graph $\tilde{G}$ and $\tilde{d}_{v_p^l}^{-\frac{1}{2}} \tilde{d}_{v_p^{l-1}}^{-\frac{1}{2}}$ is the normalized weight of the edge (v_p^l, v_p^{l-1}) in path p in $\tilde{G}$.

We have the following observations from Theorem 4.2.

- Our attack does not need to operate on model parameters Θ^*, different from existing attacks that involve dense multiplication on Θ^*. Thus, our attack is more efficient.

- Our attack can be applied to any layer GNN, as the label influence is defined for general K-iteration LP. However, most of the existing attacks can only directly attack two-layer GNNs. Thus, our attack is more practical.
- The only information our attack needs to know is the target node v's within K-hop neighbors, whose labels are y_v or c. In practice, if the labels of these node are unknown, we can estimate them via querying the GNN model, and treat the estimated labels as the true labels. Thus, our attack can be seen a restricted black-box attack.

Next, we show how to fast calculate the label influence and design our influence-based targeted evasion attack.

4.3 Efficient Calculation of Label Influence

According to Theorem 4.2, the attack's goal is to select the minimum set of nodes such that when changing the edge status between the target node v and these selected nodes, the difference between the two label influence terms will be maximized. Observing Equation (11), we note that the two label influence terms are defined on two sets of nodes: a set of nodes having the same label as the target label c, and a set of nodes having the same label as the target node's label y_v. Intuitively, if we add an edge between v and a label-c node, we can make v be close to label c; and if we remove an edge between v and a label-y_v node, we can make v away from label y_v. Thus, our idea to solve Equation (11) is as follows:

- First, we define a candidate set $\mathcal{N}_A \subset \{y_u = c, u \in \Lambda_v^{(K)}\}$ which contains label-c nodes that are *not* connected with v in the clean graph, as well as a candidate set $\mathcal{N}_B \subset \{y_z = y_v, z \in \Lambda_v^{(K)}\}$ which contains label-y_v nodes that are connected with v in the clean graph. We denote $\mathcal{S}$ as the final selected nodes from $\mathcal{N}_A$ and $\mathcal{N}_B$, and initialize $\mathcal{S} = \{\}$. For each node $u \in \mathcal{N}_A \cup \mathcal{N}_B \setminus \mathcal{S}$, we change the edge status between v and u and compute the gap between two label influence terms.
- Next, we record the node u^* that obtains the largest positive gap. Then, we modify the edge status between v and u^*, calculate the value of the attack's objective function, and update $\mathcal{S} = \mathcal{S} \cup \{u^*\}$.
- We repeat above steps at most Δ times and break if the value of attack's objective function is bigger than 0. Finally, we have the attack edges $\{(v, u^*), u^* \in \mathcal{S}\}$.

However, note that when modifying the edge status between v and u^*, the normalized weight for all edges containing u^* in all paths $\tilde{\Psi}_{v \to u}$ in Equation (12) should be recalculated. When the candidate set has a large size or/and the number of recalculated edge weights is large, calculating the exact label influence will have a large computational complexity. To solve the problem, we propose an approximate algorithm to efficiently compute the label influences. More details are in Supplementary Material.

Algorithm 1 in the full report illustrates how we efficiently calculate the label influences via depth first search (DFS), and Algorithm 2 in the report shows the details of implementing our attack.

5 EVALUATION

5.1 Experimental Setup

Datasets. Following [13, 44, 54], we use three benchmark graphs (i.e., Cora, Citeseer, and Pubmed) [29] to evaluate our attack. In these graphs, each node represents a documents and each edge indicates a citation between two documents. Each document treats the bag-of-words feature as the node feature vector, and has a label. Table 5 in the full version shows basic statistics of these graphs.

Training nodes and target nodes. We use the public training nodes to train GNN models, and target nodes to evaluate attacks against the trained GNN models. For the target nodes, we employ a random sampling technique to select 100 nodes that are correctly classified by each GNN model as the target nodes. Similar to Nettack [54], for each target node, we choose the predicted label by the GNN model with a second largest probability as the target label.

Compared attacks. We compare our influence-based attack with the state-of-the-art Nettack [54] for attacking two particular GNNs: GCN and SGC. Note that Nettack is mathematically designed to only attack two-layer GNNs and cannot directly attack multi-layer GNNs. To attack multi-layer GNNs, Nettack needs to be performed via an indirect way: It first attacks a surrogate two-layer GNN to generate the attack edges, and then transfers these attack edges to attack the target multi-layer GNNs. When computing the label influence, our attack needs to know the labels of unlabeled nodes in the graph. When our attack knows the the true labels, we denote it as **Ours-KL**. When the true labels are unknown, our attack first queries the learnt GNN model to estimate labels for unlabeled nodes and then uses the estimated labels as the true labels. We denote this variant as **Ours-UL**. As a comparison, we also test our attack that is implemented based on exact label influence calculation, and denote the corresponding two methods with known and unknowns labels as **Ours (exact)-KL** and **Ours (exact)-UL**, respectively.

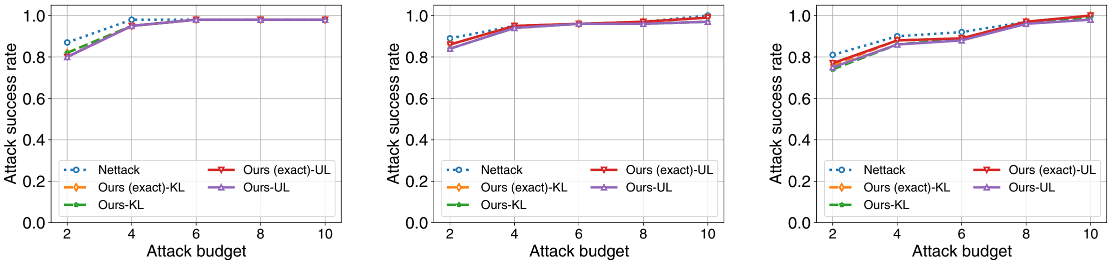
Evaluation metric. For graph perturbation attacks, we adopt attack success rate and running time as the metrics. Given a target GNN model, a set of target nodes, target label, and an attack budget Δ, attack success rate is the fraction of target nodes that are misclassified by the target GNN to be the target label when the number of attack edges per target node is at most Δ. Running time is reported on average across all the target nodes.

Implementation. We train all GNNs using the public source code. We test Nettack using the source code (https://github.com/danielzuegner/nettack). We implement our attack in PyTorch. All experiments are conducted on an A6000 GPU with 48G memory. *Due to space limitation, we only show comparison results with GCN, and all results are in the full version: https://github.com/ventr1c/InfAttack.*

5.2 Results on Attacking Two-layer GNNs

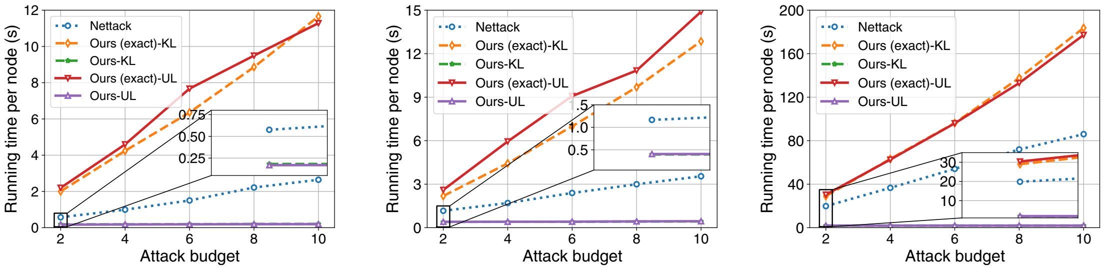
Results on attacking two-layer GCN/SGC. We compare all attacks in terms of effectiveness (i.e., attack success rate) and efficiency (i.e., running time) against two-layer GCN/SGC. Figure 1 and Figure 7 in the full version show the attack success rate against GCN and SGC on the three graphs, respectively. Moreover, Figure 2 and Figure 8 in the full version show the running time of all attacks against GCN and SGC on the three graphs, respectively. We have the following key observations.

- *Our attacks based on approximate label influence have similar performance with those based on exact label influence, but is much more efficient.* Specifically, the difference of the attack success rate between the two is less than 2% in all cases. This shows that our proposed efficient algorithm for label influence calculation



(a) Cora (b) Citeseer (c) Pubmed

Figure 1: Attack success rate vs. attack budget per target node on a two-layer GCN of all compared attacks on the three graphs.



(a) Cora (b) Citeseer (c) Pubmed

Figure 2: Running time vs. attack budget per target node on two-layer GCN of all compared attacks on the three graphs.

is effective enough. Moreover, our attacks based on approximate label influence are 1-2 orders of magnitude more efficient than those based on exact label influence.

- *Our attacks with true labels and with estimated labels have similar performance.* Specifically, the difference of the attack success rate between Ours-KL and Our-UL is negligible, i.e., less than 2% in all cases, and the running time of both Ours-KL and Our-UL are almost the same. One reason is that the trained GNN model has accurate predictions on the unlabeled nodes, and thus most of the estimated labels match the true labels. One should note that Ours-UL knows very limited knowledge about the GNN model and thus it is a very practical attack.
- *Our attacks achieve comparable performance with Nettack.* Nettack achieves state-of-the-art attack performance against two-layer GCN. Our attacks have a slightly lower attack success rate than Nettack when the attack budget is small, e.g., less than 4. This is possibly because our attack use some approximations on Assumption 1, and when the attack space is small, Assumption 1 negatively affects the attack effectiveness to some extent. However, when the attack budget is larger than 4, our attacks obtain almost the same performance with Nettack.
- *Our attacks are much more efficient than Nettack.* Specifically, our attacks have a 5-50x speedup over Nettack across the three graphs. As the attack budget increases (from 2 to 10) or the graph size increases (from Cora to Pubmed), our attacks achieve better efficiencies. The reasons are two-fold. First, Nettack needs to multiply GNN model parameters in different layers, while our attacks do not. Second, Nettack involves multiplying the node hidden features, while ours is performed by calculating the label influence. Node hidden features are often high-dimensional, while label influence only needs scalar edge weights products.

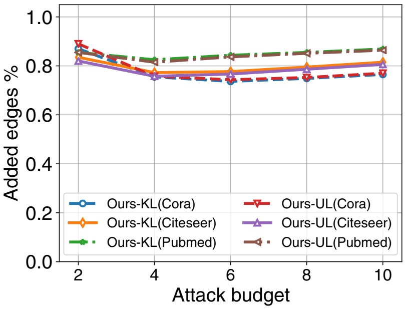
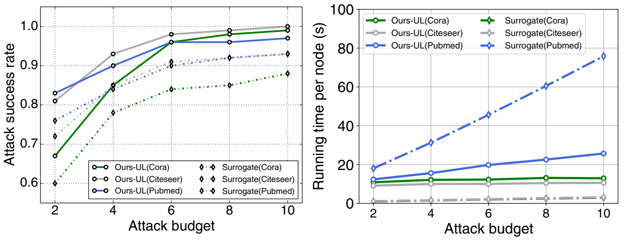


Figure 3: Fraction of the added edges among all attack edges generated by our attacks against a two-layer GCN vs. attack budget per target node.

Analysis of the attack edges. We further analyze the properties of the attack edges. Figure 3 and Figure 9 in the full version show the fraction of the added edges generated by our attacks against two-layer GCN and two-layer SGC, respectively. First, Ours-KL and Ours-UL generate almost the same fraction of added edges in all attack budgets and all graphs. This again verifies the similar characteristics between Ours-KL and Ours-UL. Second, the fraction of added edges is larger than 0.5 in all cases. This indicates that when performing the targeted attack, adding new edges between the target node and the nodes with the target label could be more effective than removing existing edges between the target node and the nodes having the same label as the target node.

Analysis of the factors that affect the attack performance. We consider the following three factors, i.e., *node degree*, *node centrality*, and *graph size*, that could affect the target node's attack performance. Here, we adopt the normalized closeness centrality



(a) Attack success rate

(b) Running time

Figure 4: (a) Attack success rate and (b) Running time of our attacks against four-layer GCN on the three graphs.

(NCC) as the metric to measure node centrality. Specifically, the NCC of a node is the average length of the shortest path between the node and all other nodes in the graph. We have the following conclusions: **1)** *Nodes with smaller degrees are easier to attack.* Given a fixed attack budget (e.g., 4 in our experiment), we observe that in all the three datasets, 100% of the target nodes with degree <= 4 attack successfully, while at most 83% and 85% of the target nodes with degree > 4 attack successfully against 2-layer GCN and 2-layer SGC, respectively. **2)** *Nodes with larger centrality are easier to attack.* We assume the attack budget is 4 per node. Specifically, 100% and 98% of the 50 target nodes with the largest NCC successfully attack 2-layer GCN and 2-layer SGC in the three datasets, while < 90% and < 80% of the 50 target nodes with the smallest NCC can successfully perform the attack. **3)** *No obvious relationship between graph size and attack success rate.* Specifically, graph size: Pubmed > Citeseer > Cora. When attacking 2-layer GCN and the attack budget is 6, we have the attack success rate: Cora (0.98) > Citeseer (0.96) > Pubmed (0.87). However, when the attack budget is 10, we have the attack success rate: Pubmed (1.00) > Citeseer (0.99) > Cora (0.98).

5.3 Results on Attacking Multi-layer GNNs

Attack performance on four-layer GCN/SGC. In this experiment, we evaluate our attacks against multi-layer GCN/SGC. We denote Nettack that attacks a surrogate two-layer GNN model first and then transfers to attacking the target model as **Surrogate**. Figure 4(a) and Figure 10(a) in the full version show the attack success rate of our attack vs. attack budget against four-layer GCN and four-layer SGC on the three graphs, respectively. First, similarly, our attacks with both known label and unknown label are effective and achieve close attack performance, and we thus show results with unknown label for simplicity. When the attack budget is 6, our attacks achieve an attack success rate of ≥ 90% in all cases. Second, our attacks are more effective than the *indirect* surrogate model based attacks. Specifically, our attacks have more than 10% higher attack success rate than the surrogate model based attacks in almost all cases.

Moreover, Figure 4(b) and Figure 10(b) in the full version shows the running time of our attacks vs. attack budget against four-layer GCN and four-layer SGC on the three graphs, respectively. Our attack is efficient. For instance, it takes our attacks less than 25*s* on average to attack a target node on the largest Pubmed in all cases. However, the surrogate model costs about 85*s*, validating that our attack is much more efficient.

Table 1: Transferability of our attacks against two-layer GCN to other GNNs. Attack budget per target node is 6.

Dataset	Source	Target			
Cora	GCN	GCN	SGC	GAT	JK-Net
	No attack	0	0.01	0.03	0.02
	Ours-KL	0.98	0.82	0.66	0.67
	Ours-UL	0.98	0.84	0.65	0.70
Citeseer	GCN	GCN	SGC	GAT	JK-Net
	No attack	0	0.01	0.01	0.03
	Ours-KL	0.96	0.78	0.70	0.63
	Ours-UL	0.96	0.78	0.72	0.63
Pubmed	GCN	GCN	SGC	GAT	JK-Net
	No attack	0	0.03	0.04	0.05
	Ours-KL	0.89	0.80	0.80	0.79
	Ours-UL	0.88	0.80	0.80	0.77

Table 2: Attack results of Ours-KL on OGB-arxiv.

Attack budget	2	4	6	8	10	Time
Nettack	OOM	OOM	OOM	OOM	OOM	-
Ours-GCN	0.65	0.72	0.73	0.73	0.82	40.1s
Ours-SGC	0.90	0.93	0.95	0.95	0.96	40.7s

Transferring our attack to other GNNs. In this experiment, we study the transferability of our attacks, i.e., whether the attack edges generated by our attacks against GCN/SGC can be also effective for other GNNs. Specifically, we use our attacks to generate the attack edges for each target node by attacking the source GNN (GCN or SGC), change the graph structure based on the attack edges, and adopt a target GNN to classify each target node on the perturbed graph. We select two additional representative GNNs, i.e., GAT [36] and JK-Net [46], as the target GNN. If a target node is also misclassified by the target GNN to be the target label, we say the attack edges generated by the source GNN are transferable.

Table 1 and Table 7 in the full version show the attack success rate of transferring of our attacks against two-layer GCN and two-layer SGC to attack other GNNs on the three graphs, where the attack budget per target node is 6. Note that we also show the attack performance for target GNNs without attack, i.e., the prediction error of target GNNs on the target nodes in the clean graph. We have the observations: First, our attacks against GCN (or SGC) have the best transferability to SGC (or GCN). This is because SGC is a special case of GCN and they share similar model architectures. Second, our attacks are also effective against GAT and JK-Net. Specifically, on all the three graphs, our attacks can increase the classification errors by at least 60% when attacking GAT and JK-Net. This indicates that all the attack edges generated by our attack on the source GNN can be transferred to attack the target GNNs. Such good transferability further demonstrates the advantages of using (label) influence to perform the target evasion attacks.

6 DISCUSSIONS

Attack performance on a large-scale dataset. We conduct experiments on the large-scale dataset OGB-arxiv [17] to demonstrate the superior efficiency of our proposed attack method compared to

Table 3: Comparing our attack vs. IG-FGSM [42] on Cora.

Model	budget	2	4	6	8	10	Time
GCN	IG-FGSM	0.29	0.75	0.89	0.92	0.94	62s
	Nettack	0.85	0.96	0.97	0.97	0.97	1.5s
	Ours	0.90	0.93	0.95	0.95	0.96	0.1s

Table 4: Comparing our attack vs. black-box attack [39].

GCN	budget	2	4	6	GCN	budget	2	4	6
Cora	Ours	0.81	0.94	0.98	Citeseer	Ours	0.86	0.94	0.95
	[39]	0.68	0.79	0.80		[39]	0.84	0.91	0.92

baselines. We use a 2-layer GCN/SGC as the target GNN, achieving a clean accuracy of approximately 60% on test nodes. The attack method is set as Ours-KL. Note that time is denoted as the average running time of compared attack methods across the five attack budgets. We assess the attack performance on 100 target nodes that are accurately classified by GCN/SGC. The comparison results between Nettack and our attack are presented in Table 2. We observe that **1)** Nettack encounters an out-of-memory (OOM) error on our platform due to the need for storing dense model weights and involving intensive matrix-matrix multiplication. **2)** Our attacks achieve highly promising attack success rates while maintaining efficiency. Specifically, with an attack budget of 6, the attack success rates of Ours-KL against GCN and SGC are 73% and 95%, respectively. These results demonstrate the significant advantages of our attack method over baselines on large graphs.

Comparing with more attack baselines. To further demonstrate the effectiveness of our attack, we compare Ours-KL with a more recent attack IG-FGSM [42], where we use the same setting as that in Section 5.2 (i.e., 2-layer GCN/SGC as the target GNN, 100 target nodes). The comparison results on Cora are reported in Table 3 and Table 8 in the full report. Note that time is denoted as the average running time of compared attack methods across the five attack budgets. We observe that: **1)** Nettack not only outperforms IG-FGSM when the attack budget is small, but also is far more efficient than IG-FGSM. Specifically, when the attack budget is 2, the attack success rates of IG-FGSM agasint GCN and SGC are only 0.29 and 0.32, respectively, which are significantly lower than that of Nettack, 0.85 and 0.65. Moreover, the running times of IG-FGSM against GCN and SGC are 62s and 55s, respectively, which are much lower than that of Nettack, 1.5s and 2.5s, respectively. **2)** Our method is even more efficient than Nettack. Specifically, when attacking GCN and SGC, the running time of our method is only 0.1s, which is lower than other two methods, further validating the efficiency of our method.

Comparing with black-box attacks. In the paper, we mainly compare our attack with the white-box attack. Here we also compare with the stringent black-box attacks proposed in [39]. To best explore the attack capability, we do not restrict the number of queries in [39], and obtain the optimal attack successful rate for a given attack budget. Table 4 and Table 9 in the full version show the comparison results on Cora and Citeseer (Note that [39] cannot run on Pubmed due to limited GPU memory) on attacking 2-layer GCN/SGC. We can see that our attack is more effective than [39], especially when the attack budget is small. One key reason is that our attack utilizes the strong connection between GNN and LP, while [39] performs the attack based on the query feedback, i.e., the target node's confidence score after querying the black-box GNN.

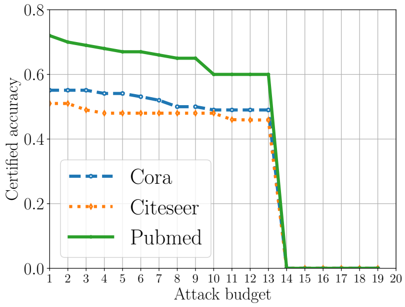


Figure 5: Certified accuracy [38] vs. attack budget of GCN.

Defending against our attack. As shown in Section 2, existing empirical defenses [11, 13, 19, 42, 48, 51, 53] are broken [27] when the adversary knows the defense mechanism. Hence, we propose to defend our attack via provable defenses and choose the state-of-the-art randomized smoothing-based provable defense [38]. Specifically, given a target node, a model is provably robust for the target node if the model correctly predicts the same label for the target node when the attacker *arbitrarily* modifies a bounded number of (e.g., at most R) edges in the graph, where R is called *certified radius*. Hence, provably robust models can defend against the worst-case attack (including our attack). Accordingly, certified accuracy under R means the fraction of target nodes that are predicted accurately by modifying any R edges. That is, if a model achieves a larger certified accuracy at a given budget, it shows better provable robustness. We conduct experiments on defending two-layer GCN against the worst-cast attack via [38]. Results on the three datasets are shown in Figure 5. We can see that, when any 4 edges are allowed to be modified, the certified accuracy achieved by the method [38] on the three datasets are about 0.50, 0.55, and 0.70, respectively. However, the method [38] cannot provably defend against the worst-case attack when the attack budget is larger than 13, which implies the need to design more powerful provable defenses.

7 CONCLUSION

We propose an influence-based evasion attack against GNNs. Specifically, we first build the connection between GNNs and label propagation (LP) via carefully designed influence functions. Next, we reformulate the attack against GNNs to be related to label influence on LP. Then, we design an efficient algorithm to calculate label influences. Our attack is applicable to any-layer GNNs and does not need to know the GNN model parameters. Experimental results demonstrate that our attack achieves comparable performance against state-of-the-art white-box attacks, and has a 5-50x speedup when attacking two-layer GCNs. Our attack is also effective to attack multi-layer GNNs and is transferable to other GNNs.

Acknowledgments. We thank the anonymous reviewers for their constructive feedback. Wang is supported by the Cisco Research Award, National Science Foundation under grant Nos. NSF-2216926 and NSF-2241713. Pang is supported in part by the Natural Science Foundation of Jiangxi Province of China (20232BAB212025) and High-level and Urgently Needed Overseas Talent Programs of Jiangxi Province (20232BCJ25024).

Ethical considerations: Our work studies the vulnerability of graph neural networks, and it could probably have both a negative and a positive impact. From the negative side, our findings will inspire attackers to perform malicious activities against real-world systems. For instance, a malicious user in a social network (e.g, Twitter) can leverage our attack to make him avoid being detected by the malicious user detection system. Then, he can perform malicious activities, e.g., spreading fake news and distributing phishing attacks among the social network. From the positive side, our work will inspire following works to design more robust graph neural networks against adversarial attacks. All datasets and codes we used in the paper are publicly available. Our work is mainly for research purpose and complies with ethical standards. Therefore, it does not have any negative ethical impact on society.

REFERENCES

[1] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *ICML*.

[2] Aleksandar Bojchevski and Stephan Günnemann. 2019. Certifiable Robustness to Graph Perturbations. In *NeurIPS*.

[3] Aleksandar Bojchevski, Johannes Klicpera, and Stephan Günnemann. 2020. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *ICML*.

[4] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, and Junzhou Huang. 2020. A Restricted Black-Box Adversarial Framework Towards Attacking Graph Embedding Models.. In *AAAI*.

[5] Jinyin Chen, Yangyang Wu, Xuanheng Xu, Yixian Chen, Haibin Zheng, and Qi Xuan. 2018. Fast gradient attack on network embedding. *arXiv* (2018).

[6] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. 2017. Practical attacks against graph-based clustering. In *CCS*.

[7] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. 2023. Unnoticeable backdoor attacks on graph neural networks. In *WWW*.

[8] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *ICML*.

[9] Quanyu Dai, Qiang Li, Jian Tang, and Dan Wang. 2018. Adversarial network embedding. In *AAAI*.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*.

[11] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. 2020. All you need is low (rank) defending against adversarial attacks on graphs. In *WSDM*.

[12] Houxiang Fan, Binghui Wang, Pan Zhou, Ang Li, Zichuan Xu, Cai Fu, Hai Li, and Yiran Chen. 2021. Reinforcement learning-based black-box evasion attacks to link prediction in dynamic graphs. In *HPCC*.

[13] Simon Geisler, Tobias Schmidt, Hakan Şirin, Daniel Zügner, Aleksandar Bojchevski, and Stephan Günnemann. 2021. Robustness of graph neural networks at scale. *NeurIPS*.

[14] Simon Geisler, Daniel Zügner, and Stephan Günnemann. 2020. Reliable graph neural networks via robust aggregation. In *NeurIPS*.

[15] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *ICML*.

[16] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NIPS*.

[17] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *NeurIPS*.

[18] Hongwei Jin, Zhan Shi, Venkata Jaya Shankar Ashish Peruri, and Xinhua Zhang. 2020. Certified Robustness of Graph Convolution Networks for Graph Classification under Topological Attacks. In *NeurIPS*.

[19] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph structure learning for robust graph neural networks. In *KDD*.

[20] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

[21] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*.

[22] Jia Li, Honglei Zhang, Zhichao Han, Yu Rong, Hong Cheng, and Junzhou Huang. 2020. Adversarial attack on community detection by hiding individuals. In *WWW*.

[23] Minhua Lin, Teng Xiao, Enyan Dai, Xiang Zhang, and Suhang Wang. 2023. Certifiably Robust Graph Contrastive Learning. In *NeurIPS*.

[24] Xuanqing Liu, Si Si, Xiaojin Zhu, Yang Li, and Cho-Jui Hsieh. 2019. A unified framework for data poisoning attack to graph-based semi-supervised learning. *arXiv preprint arXiv:1910.14147* (2019).

[25] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. In *NeurIPS*.

[26] Jiaming Mu, Binghui Wang, Qi Li, Kun Sun, Mingwei Xu, and Zhuotao Liu. 2021. A Hard Label Black-box Adversarial Attack Against Graph Neural Networks. In *CCS*.

[27] Felix Mujkanovic, Simon Geisler, Stephan Günnemann, and Aleksandar Bojchevski. 2022. Are Defenses for Graph Neural Networks Robust?. In *NeurIPS*.

[28] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* (2008).

[29] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Rad. 2008. Collective classification in network data. *AI magazine* (2008).

[30] Mingjie Sun, Jian Tang, Huichen Li, Bo Li, Chaowei Xiao, Yao Chen, and Dawn Song. 2018. Data poisoning attack against unsupervised node embedding methods. *arXiv* (2018).

[31] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *The Web Conference*.

[32] Tsubasa Takahashi. 2019. Indirect Adversarial Attacks via Poisoning Neighbors for Graph Convolutional Networks. In *IEEE BigData*.

[33] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. 2020. Transferring Robustness for Graph Neural Network Against Poisoning Attacks. In *WSDM*.

[34] Shuchang Tao, Huawei Shen, Qi Cao, Liang Hou, and Xueqi Cheng. 2021. Adversarial Immunization for Certifiable Robustness on Graphs. In *WSDM*.

[35] MohamadAli Torkamani and Daniel Lowd. 2013. Convex adversarial collective classification. In *ICML*.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

[37] Binghui Wang and Neil Zhenqiang Gong. 2019. Attacking Graph-based Classification via Manipulating the Graph Structure. In *CCS*.

[38] Binghui Wang, Jinyuan Jia, Xiaoyu Cao, and Neil Gong. 2021. Certified robustness of graph neural networks against adversarial structural perturbation. In *KDD*.

[39] Binghui Wang, Youqi Li, and Pan Zhou. 2022. Bandits for Structure Perturbation-based Black-box Attacks to Graph Neural Networks with Theoretical Guarantees. In *CVPR*.

[40] Binghui Wang, Meng Pang, and Yun Dong. 2023. Turning Strengths into Weaknesses: A Certified Robustness Inspired Attack Framework against Graph Neural Networks. In *CVPR*.

[41] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. 2019. Simplifying graph convolutional networks. In *ICML*.

[42] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. 2019. Adversarial examples on graph data: Deep insights into attack and defense. In *IJCAI*.

[43] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *({USENIX} Security 21)*.

[44] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology attack and defense for graph neural networks: An optimization perspective. In *IJCAI*.

[45] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks?. In *ICLR*.

[46] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. In *ICML*.

[47] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. In *NeurIPS*.

[48] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. In *NeurIPS*.

[49] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. (2021).

[50] Zijie Zhang, Zeru Zhang, Yang Zhou, Yelong Shen, Ruoming Jin, and Dejing Dou. 2020. Adversarial Attacks on Deep Graph Matching. In *NeurIPS*, Vol. 33.

[51] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *KDD*.

[52] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*.

[53] Jun Zhuang and Mohammad Al Hasan. 2022. Defending Graph Convolutional Networks against Dynamic Graph Perturbations via Bayesian Self-supervision. In *AAAI*.

[54] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *KDD*.

[55] Daniel Zügner and Stephan Günnemann. 2019. Adversarial attacks on graph neural networks via meta learning. In *ICLR*.