GZOO: Black-Box Node Injection Attack on Graph Neural Networks via Zeroth-Order Optimization

Hao Yu, *Student Member, IEEE*, Ke Liang, Dayu Hu, Wenxuan Tu, Chuan Ma, *Member, IEEE*, Sihang Zhou, *Member, IEEE*, and Xinwang Liu, *Senior Member, IEEE*

***Abstract*—The ubiquity of Graph Neural Networks (GNNs) emphasizes the imperative to assess their resilience against node injection attacks, a type of evasion attacks that impact victim models by injecting nodes with fabricated attributes and structures. However, prevailing attacks face two primary limitations: (1) Sequential construction of attributes and structures results in suboptimal outcomes as structure information is overlooked during attribute construction and vice versa. (2) In black-box scenarios, where attackers lack access to victim model architecture and parameters, reliance on surrogate models degrades performance due to architectural discrepancies. To overcome these limitations, we introduce GZOO, a black-box node injection attack that leverages an adversarial graph generator, compromising both attribute and structure sub-generators. This integration crafts optimal attributes and structures by considering their mutual information, enhancing their influence when aggregating information from injected nodes. Furthermore, GZOO proposes a zeroth-order optimization algorithm leveraging prediction results from victim models to estimate gradients for updating generator parameters, eliminating the necessity to train surrogate models. Across sixteen datasets, GZOO significantly outperforms state-of-the-art attacks, achieving remarkable effectiveness and robustness. Notably, on the Cora dataset with the GCN model, GZOO achieves an impressive 95.69% success rate, surpassing the maximum 66.01% achieved by baselines.**

***Index Terms*—Black-box attacks, graph neural networks, node injection attacks, zeroth-order optimization.**

I. Introduction

GRAPH Neural Networks (GNNs), designed to learn representations from graph-structured data [1], [2], have achieved significant success in various real-world applications, including fraud transaction detection [3] and social recommender systems [4], [5]. However, these achievements have sparked concerns regarding the robustness of GNN-based models, particularly their vulnerability to evasion attacks [6], [7], [8], [9]. The primary objective of such attacks is to mislead victim GNNs into producing inaccurate predictions for specific nodes during the inference phase.

Current research on evasion attacks within the graph domain has predominantly focused on Graph Modification Attacks (GMAs), involving perturbations to graph attributes and structures [10]. Despite the promise of these methods, their practicality is limited by the necessity to modify existing graph structures, often an impractical proposition in real-world scenarios. For instance, in social network applications, modifying existing social connections or user attributes necessitates compromising either the client or server. Consequently, an alternative research direction has emerged, focusing on Node Injection Attacks (NIAs) [11], [12], [13]. NIAs involve generating malicious nodes (i.e., attributes and structures) and injecting them into existing graphs to compromise GNN performance, bypassing the need for alterations to established structures. In social networking applications, an adversary might create new accounts and establish connections with existing users to achieve malicious objectives. Therefore, the dual objectives of NIAs encompass: *1) crafting attributes of malicious nodes consistent with real-world scenarios* and *2) generating effective structures for malicious nodes to deceive the victim GNN*.

Limitations of Existing NIAs: Notwithstanding the notable achievements of NIAs, three critical limitations exist: (1) In real-world applications, node attributes exhibit diverse characteristics, ranging from continuous variables (e.g., salary) to binary-valued discrete attributes (e.g., gender) and multi-valued discrete attributes (e.g., education level). However, existing NIAs lack flexibility in constructing node attributes, confining themselves to binary-valued discrete attributes [14], representing only binary values such as 0 or 1, true or false, present or absent, and similar. (2) Current NIAs treat attribute and structure generation as sequential stages, thereby neglecting the crucial inter-relationship between attributes and structures [12]. The case study in Section IV illustrates the crucial role of the inter-relationship in misleading victim GNNs. For instance, in a situation where the adversary first generates attributes for malicious nodes and then constructs structures, the ability to generate optimal structures is constrained by the attributes generated earlier. (3) In black-box scenarios, where adversaries lack access to the architecture and parameters of victim models, existing NIAs resort to surrogate models [13] to mimic the functionality of victim models by training models on the dataset used to train victim models, disregarding potential architectural disparities

Received 24 December 2023; revised 13 August 2024; accepted 15 October 2024. Date of publication 21 October 2024; date of current version 26 November 2024. This work was supported by the National Natural Science Foundation of China under Grant 62325604 and under Grant 62276271. Recommended for acceptance by X. Yi. *(Corresponding author: Xinwang Liu.)*

Hao Yu, Ke Liang, Dayu Hu, Wenxuan Tu, and Xinwang Liu are with the School of Computer, National University of Defense Technology, Changsha 410073, China (e-mail: csyuhao@gmail.com; xinwangliu@nudt.edu.cn).

Chuan Ma is with the College of Computer Science, Chongqing University, Chongqing 400050, China, and also with the Zhejiang Lab, Hangzhou 311121, China.

Sihang Zhou is with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China.

Digital Object Identifier 10.1109/TKDE.2024.3483274

between surrogate and actual victim models. The first limitation affects the practicality of NIAs, and the latter two impact the attack performance.

Our Goals and Contributions: To comprehensively address these limitations, we present GZOO, an innovative framework employing an adversarial graph generator optimized through zeroth-order optimization to generate malicious nodes. Given that graphs comprise continuous or discrete attributes and discrete structures, GZOO utilizes the generator to produce both types of data and tailor the zeroth-order optimization to learn the generator's parameters within complex architectures. The generator comprises two key components: an attribute sub-generator and a structure sub-generator. The former is tasked with generating both discrete (including binary and multi-valued) and continuous attributes, while the latter employs learned sampling probabilities and the Gumbel-Softmax Trick to craft structures linking malicious nodes to the original graph. The main contributions can be summarized as follows:

- We introduce an adversarial graph generator that can take into account the mutual information between attributes and structures, and is designed to simultaneously craft optimal attributes and structures, accommodating both discrete (including binary and multi-valued) and continuous attribute spaces.
- We propose a zeroth-order optimization algorithm that uses model predictions to estimate gradients for updating generator parameters in black-box scenarios, eliminating the need for surrogate model training. We also provide a theoretical analysis of gradient estimation error to validate its effectiveness.
- Through extensive evaluations on sixteen benchmark datasets, covering node classification, graph classification, and link prediction tasks, we demonstrate the effectiveness and robustness of GZOO in comparison to state-of-the-art baselines.

The subsequent sections are organized as follows: Section II summarizes existing black-box NIAs. Section III introduces the problem of NIAs and the threat model of GZOO. Section IV presents the motivation behind GZOO. Section V outlines the proposed GZOO framework. Section VI assesses its attack performance. Section VII offers brief discussions, and Section VIII concludes this paper.

II. Related Work

The vulnerability of GNNs to evasion attacks is widely recognized [15], [16], [17]. Notably, among these attacks, NIAs stand out for their capability to inject malicious nodes into existing graphs without perturbing their original structures. Black-box NIAs, where the attacker lacks access to the architecture and parameters of the victim model, can be divided into two primary categories: surrogate-model-based attacks and gradient-free-based attacks.

Surrogate-model-based Attacks: Adversaries create surrogate models to mimic the behavior of victim models by training them on the same dataset. These surrogate models then provide gradients for generating attributes and structures of malicious nodes. SPEIT [18], the top solution in the KDD-CUP 2020 Graph Adversarial Attack & Defense competition, involves generating adjacency matrices and attributes using gradients from surrogate models. G-NIA [12] uses an optimization-based approach to sequentially generate attributes and construct injected edges for malicious nodes. TDGIA [13] employs a smooth optimization objective to create attributes and inject them around nodes vulnerable due to their topological position. PGD [19] optimizes the attributes of malicious nodes using gradients from surrogate models. However, architectural differences between surrogate and victim models can impair attack performance, and the effectiveness of surrogate-based attacks is significantly influenced by the surrogate model architecture [13].

Gradient-free-based Attacks: Adversaries do not use surrogate models to replicate the victim models' functionality. G ^{2} A2C [14] is the pioneering study on gradient-free black-box NIAs. It treats the NIA as a Markov decision process and employs a reinforcement learning framework to generate malicious nodes based on the outcomes of attacks on victim models. However, the sequential generation of attributes and injected edges in this approach faces difficulties due to the inter-dependencies between these components, which limits its effectiveness.

III. Problem Statement

A. Preliminaries on GNNs

Consider an attributed graph denoted as $\mathcal{G} = (\mathbf{A}, \mathbf{X})$, where the node set is represented as $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$. The matrices $\mathbf{A}$ and $\mathbf{X}$ correspond to the adjacency and attribute matrices, respectively. The binary adjacency matrix, $\mathbf{A} \subseteq \{0,1\}^{N\times N}$, defines node connections, where $\mathbf{A}_{ij}$ equals 1 if an edge links nodes v_i and v_j, and 0 otherwise. The attribute matrix, $\mathbf{X} \in \mathbb{R}^{N\times F}$, contains attributes for each node, with $\mathbf{X}_i$ representing the attributes of node v_i, which may be either continuous or discrete.

Our primary focus lies on the semi-supervised node classification task, where the label matrix of the attributed graph is denoted as $\mathbf{Y} \subseteq \{0,1\}^{N\times C}$, with C representing the total number of labels. For a GNN f parameterized by θ, the node classification task is formulated as follows:

$$\hat{\mathbf{Y}}_i = \underset{c\in[C]}{\arg\max}\ f_\theta(\mathbf{A}, \mathbf{X}_i)_c. \tag{1}$$

Here, $f_\theta(\mathbf{A}, \mathbf{X}_i)$ represents the logits of node v_i, and $[C]$ denotes the label set $\{1, \ldots, C\}$.

Given M labeled nodes, where $0 < M \ll N$, and the corresponding label matrix $\mathbf{Y}^L$, the GNN model f_θ comprises K layers and is optimized by minimizing a specific classification loss, such as the cross-entropy loss $\mathrm{CE}(\cdot,\cdot)$. The GNN $f_\theta(\cdot)$ employs a standard neural message-passing scheme [20], [21], [22], allowing for recursive updates of node representations at the kth layer. The update formula is expressed as follows:

$$\mathbf{H}_i^{(k)} = \sigma(\rho(\mathbf{H}_i^{(k-1)}, \{\mathbf{H}_j^{(k-1)}, \forall v_j \in \mathcal{N}(v_i)\})\mathbf{W}_k), \tag{2}$$

where $\mathbf{W}_k$ and $\mathbf{H}_i^{(k)}$ denote the learnable parameters and hidden representations of node v_i after the kth level of aggregation (where $k > 0$). Additionally, $\mathbf{H}_i^{(0)}$ equals $\mathbf{X}_i$ for all nodes v_i in the set $\mathcal{V}$. The function $\mathcal{N}(v_i)$ refers to the collection of

neighboring nodes of v_i, and $\sigma(\cdot)$ is an activation function, such as ReLU. Moreover, $\rho(\cdot)$ denotes an aggregation function for neighboring nodes, such as MEAN or SUM. At the Kth layer, the last dimensions of $\mathbf{W}_K$ and $\sigma(\cdot)$ are set to C and Softmax, respectively. The loss associated with node v_i is calculated as follows:

$$\mathcal{L}(\mathbf{A}, \mathbf{X}, \mathbf{Y}) = \mathrm{CE}(\mathbf{H}_i^{(k)}, \mathbf{Y}_i). \tag{3}$$

B. Node Injection Attack

Node Injection Attacks (NIAs) [11], [23] aim to strategically manipulate a GNN model, denoted as $f_{\theta^\star}$, that has undergone training on a specified graph $\mathcal{G} = (\mathbf{A}, \mathbf{X})$. The specific goal of this manipulation is to induce the misclassification of a targeted node v_i by crafting a modified graph $\mathcal{G}' = (\mathbf{A}', \mathbf{X}')$ while adhering to certain budgetary constraints. The construction of this modified graph, $\mathcal{G}'$, involves the generation of a set of malicious nodes, denoted as $\tilde{\mathcal{V}}$, accompanied by their corresponding attributes $\tilde{\mathbf{X}}$. The resulting $\mathbf{A}'$ and $\mathbf{X}'$ are formally expressed as follows:

$$\mathbf{X}' = \begin{bmatrix} \mathbf{X} \\ \tilde{\mathbf{X}} \end{bmatrix}; \quad \mathbf{A}' = \begin{bmatrix} \mathbf{A} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}} & \mathbf{O} \end{bmatrix},$$

where $\mathbf{O}$ is the adjacency matrix linking malicious nodes, and $\tilde{\mathbf{A}}$ denotes the adjacency matrix connecting the malicious nodes with the original ones.

The formalization of NIAs can be described by the optimization problem:

$$\max_{\mathcal{G}'} \mathcal{L}(\mathbf{A}', \mathbf{X}_i', \mathbf{Y}_i),$$
$$\text{s.t.}, |\tilde{\mathcal{V}}| \leq n_a,\ 1 \leq d_j \leq n_s,\ \langle \mathbf{X}_j', \mathbf{X} \rangle \leq \delta,\ \forall v_j \in \tilde{\mathcal{V}}. \tag{4}$$

Here, d_j represents the degree of the malicious node v_j. The integers n_a and n_s signify the maximum allowable numbers of malicious nodes and edges per node, respectively, known as attack budgets. Additionally, $|\cdot|$ denotes the cardinality of a set, and $\langle \cdot, \cdot \rangle$ measures the similarity between the generated attributes and the original nodes' attributes. This similarity assessment relies on metrics such as Kullback–Leibler divergence for continuous attributes or norm differences for discrete attributes. The symbol δ denotes the acceptable upper limit of this attribute dissimilarity.

C. Threat Model

In this study, consistent with prior investigations [12], [14], [24], we characterize NIAs as a form of evasion attack.

Objectives: (1) The adversary's primary objective is to inject malicious nodes into the original graph stealthily, leading the victim model to make inaccurate predictions for the target node during the inference phase. For example, in fraud detection, the aim is to deceive GNNs into misclassifying a fraudster as a legitimate user. Crucially, this attack occurs during the inference phase, with the victim model's parameters remaining static throughout the attack. (2) To maintain the attack inconspicuous, the adversary imposes constraints, such as ensuring that the attributes of malicious nodes closely resemble those of original nodes or confirming the absence of edges between malicious nodes, i.e., $\mathbf{O} = \mathbf{I}$.

Knowledge: (1) We consider a challenging scenario wherein the adversary operates without access to the architecture or parameters of the victim model. (2) The adversary possesses limited knowledge about the topology of the original graph during training, being constrained to sampling a subgraph during the execution of the attack. (3) Given that the attributes of nodes within the graph may represent discrete data, the adversary can assemble possible values for each attribute into a set denoted as $\mathcal{C} = \{\{a_{1,0}, \ldots, a_{1,c_1}\}, \ldots, \{a_{F,0}, \ldots, a_{F,c_F}\}\}$. Here, c_i represents the count of potential values for the ith attribute. These values can then be organized into a category matrix $\mathbf{C} \in \mathbb{Z}^{F \times M}$ encompassing all possible values, where M is the maximum count of potential values across all attributes, specifically $M = \max(c_1, \ldots, c_F)$. The computation for the ith row of $\mathbf{C}$ is expressed as:

$$\mathbf{C}_{ij} = \begin{cases} a_{i,j}, & \text{if } j \leq c_i; \\ 0, & \text{if } j > c_i. \end{cases} \tag{5}$$

Subsequently, a mask matrix is generated to identify the locations of the padding zeros, expressed as $\mathbf{M}_i^a = \mathbf{I}_{j \leq c_i}$.

IV. Motivation

Current NIAs typically involve a two-step process to construct attributes and structures. For instance, Tao et al. [12] first generate attributes and then use these attributes to fabricate structures. However, the performance of this sequential approach is limited, as pre-generated attributes may constrain crafting optimal structures to enhance the influence of malicious nodes. Consequently, this methodology overlooks the mutual information between attributes and structures.

The majority of GNNs follow a neural message-passing paradigm, where node embeddings are derived through the iterative aggregation and propagation of neighbor information. This leads to a plausible hypothesis that attributes and structures of malicious nodes are interrelated, collectively influencing the effectiveness of attacks. To substantiate this hypothesis, we introduce two mismatch scenarios after constructing attributes and structures of malicious nodes:

- *Attribute mismatch (attr-mis):* We randomly discard 50% of the attributes of malicious nodes to assess the significance of attributes.
- *Structure mismatch (str-mis):* We disrupt edges between the target node and the malicious node, randomly establishing a new edge between the malicious node and the 1-hop neighbors of the target node. The choice of 1-hop neighbors is intentional to ensure that malicious nodes can exert influence on target nodes.

A case study is conducted on two datasets, where malicious nodes are constructed using PGD [25] and SPEIT [18], and the victim model is GCN [26]. The dataset statistics for these two datasets are presented in Section VI-A, and the details about PGD and SPEIT are discussed in Section II. The results of this case study are presented in Fig. 1.

Fig. 1. A case study is conducted to evaluate the interaction between attributes and structures of malicious nodes generated by NIAs such as PGD and SPEIT.

Existing adversarial attacks employing neural networks to generate adversarial graphs generally follow a sequential process, first generating attributes and then deriving structures based on these attributes (e.g., G-NIA [12] and G^{2}A2C [14]). This implies that structures are derived from attributes. However, our findings reveal a mutual influence between edges and attributes. Thus, it is more effective to integrate the representations used for generating attributes and structures before the generation process, utilizing these fused representations for both tasks.

V. Methodology

In this section, we provide a detailed description of the proposed GZOO.

A. Overview

Fig. 2 provides a comprehensive overview of GZOO. GZOO employs the adversarial graph generator to create attributes and structures, using the learned generator parameters to produce malicious nodes for various targets. Given a graph $\mathcal{G}$ and the target node v_i, the generator g_ϕ is employed to generate attributes and structures of malicious nodes. The crafted malicious nodes are connected to the original graph, and the resulting adversarial graph is then input into the victim model, and its predictions are utilized in conjunction with zeroth-order optimization to update ϕ. This iterative process continues until there is a successful alteration of the label corresponding to v_i or the attack budget is fully utilized. The following subsections will explore the generator's architecture, loss functions, and optimization strategy.

B. Adversarial Graph Generator

To deceive the victim model $f_{\theta^\star}$ to misclassify the target node v_i, the generator g is harnessed to produce a set of malicious nodes denoted as $\tilde{\mathcal{V}} = \{v_1, v_2, \ldots, v_n\}$, along with a malicious attribute matrix $\tilde{\mathbf{X}}$. Subsequently, these newly generated nodes are seamlessly incorporated into the original graph, resulting in an altered adjacency matrix $\tilde{\mathbf{A}}$. To flexibly craft discrete attributes and simultaneously generate both attributes and structures, we bifurcate the model g_ϕ into two branches: one dedicated to generating continuous and discrete attributes, referred to as $\mathcal{G}^a$, and the other focused on structures, designated as $\mathcal{G}^s$. Acknowledging the interconnected nature of attributes and structure, g_ϕ avoids the simplistic approach of employing two entirely separate networks as independent generators. Instead, it leverages the backbone model to pre-fuse the information and then inputs the fused representations into their respective sub-generators, which utilize the mutual information of attributes and structures.

Backbone Model: Its objective is to generate fused representations for the two distinct sub-generators. Initially, a GCN layer is used to enable message propagation on the K-order subgraph $\mathcal{G}^K(v_i) = (\hat{\mathbf{A}}, \hat{\mathbf{X}})$:

$$\begin{aligned}\mathbf{H}_i^{a,1} &= \sigma(\rho(\hat{\mathbf{X}}_i, \{\hat{\mathbf{X}}_j, \forall v_j \in \mathcal{N}^{\hat{\mathbf{A}}}(v_i)\})\mathbf{W}^{a,0});\\ \mathbf{H}_i^{s,1} &= \sigma(\rho(\hat{\mathbf{X}}_i, \{\hat{\mathbf{X}}_j, \forall v_j \in \mathcal{N}^{\hat{\mathbf{A}}}(v_i)\})\mathbf{W}^{s,0}),\end{aligned} \tag{6}$$

where $\mathbf{H}_i^{a,1}$ and $\mathbf{H}_i^{s,1}$ denote latent representations used to generate attributes and structures, respectively, in the space $\mathbb{R}^{n\times d_1}$, and ρ represents the aggregation function. The subsequent step combines these representations to collaboratively synthesize malicious nodes:

$$\mathbf{H}^{a,2} = \mathbf{H}^{a,1} \| \mathbf{H}^{s,1}; \quad \mathbf{H}^{s,2} = \mathbf{H}^{s,1} \| \mathbf{H}^{a,1}, \tag{7}$$

where horizontal concatenation is represented by $\cdot\|\cdot$. Subsequently, this fused information is refined via (8).

$$\begin{aligned}\mathbf{H}_i^{a,3} &= \sigma(\rho(\mathbf{H}_i^{a,2}, \{\mathbf{H}_j^{a,2}, \forall v_j \in \mathcal{N}^{\hat{\mathbf{A}}}(v_i)\})\mathbf{W}^{a,1});\\ \mathbf{H}_i^{s,3} &= \rho(\mathbf{H}_i^{s,2}, \{\mathbf{H}_j^{s,2}, \forall v_j \in \mathcal{N}^{\hat{\mathbf{A}}}(v_i)\})\mathbf{W}^{s,1},\end{aligned} \tag{8}$$

where both matrix $\mathbf{W}^{a,1} \in \mathbb{R}^{2d_1\times(n_a d_2)}$ and $\mathbf{W}^{s,1} \in \mathbb{R}^{2d_1\times n_a}$ are trainable weights. $\mathbf{H}_i^{s,3}$ is processed using a softmax function with the mask matrix in (15). Here, n_a symbolizes the number of malicious nodes, while d_1 and d_2 refer to the dimensions of corresponding representations.

Attribute Sub-generator $\mathcal{G}^a$: Given that the dimensions of the attributes for malicious nodes are $n_a \times F$ and the dimensions of $\mathbf{H}^{a,3}$ are $n \times (n_a d_2)$, the attribute generation begins with reshaping the representation $\mathbf{H}^{a,3}$. We employ the reshape operator $\mathcal{M}_1$ to convert it into an appropriate dimension. The operator is defined as:

$$\mathbf{A} \in \mathbb{R}^{a\times(bc)} \rightarrow \dot{\mathbf{A}} \in \mathbb{R}^{a\times b\times c}. \tag{9}$$

Following this transformation, we introduce a mean operator $\mathcal{M}_2$ to aggregate representations of the n original nodes within the subgraph:

$$\dot{\mathbf{A}} \in \mathbb{R}^{a\times b\times c} \rightarrow \ddot{\mathbf{A}} \in \mathbb{R}^{b\times c} \text{ where } \ddot{\mathbf{A}}_{ij} = \frac{1}{a}\sum_{k=1}^{a} \dot{\mathbf{A}}_{kij}. \tag{10}$$

By applying both $\mathcal{M}_1$ and $\mathcal{M}_2$ to $\mathbf{H}^{a,3}$, we obtain $\mathbf{H}^{a,4}$ with dimensions $n_a \times d_2$, which is expressed as $\mathbf{H}^{a,4} = \mathcal{M}_2(\mathcal{M}_1(\mathbf{H}^{a,3}))$. We then proceed to synthesize attributes, considering both continuous and discrete types.

For continuous attributes, a linear layer with an activation function is employed, utilizing a weight matrix $\mathbf{W}^{a,2} \in \mathbb{R}^{d_2\times F}$ to generate the malicious attribute matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{n_a\times F}$, defined as:

$$\tilde{\mathbf{X}} = \sigma(\mathbf{H}^{a,4})\mathbf{W}^{a,2}, \tag{11}$$

where $\sigma(\cdot)$ denotes an activation function such as ReLU.

For discrete attributes, the sub-generator leverages the Gumbel-Softmax Trick [27], [28] to sample attributes from

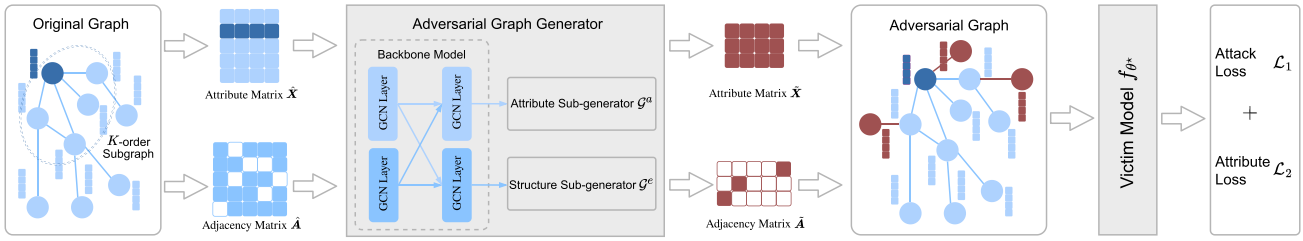


Fig. 2. A detailed explanation of GZOO highlights its use of an adversarial graph generator to synthesize attributes $\tilde{\mathbf{X}}$ and structures $\tilde{\mathbf{A}}$ tailored for malicious nodes. The integration of a zeroth-order optimization mechanism enables iterative refinement of the model's parameters, utilizing the K-order subgraph of the target node as input.

a specific discrete distribution. Assuming a random discrete variable X with k potential values, this distribution furnishes the probabilities associated with all k possible values and can be utilized for sampling to obtain a single possible value. This distribution is characterized by probabilities $\{\theta_1, \ldots, \theta_k\}$, satisfying $\sum_i \theta_i = 1$, and the probability mass function is given by $P(X = c_i) = \theta_i$. Adversaries have access to the category matrix $\mathbf{C}$ along with a corresponding mask matrix $\mathbf{M}^a$, and then calculate probabilities $\mathbf{P}^a \in \mathbb{R}^{n_a \times F \times M}$ for all possible values of discrete attributes:

$$\mathbf{H}^{a,5} = \sigma(\mathcal{M}_1(\sigma(\mathbf{H}^{a,4})\mathbf{W}^{a,3}))\mathbf{W}^{a,4};$$
$$\mathbf{P}^a_{ijk} = \mathbf{M}^a_{ijk} \cdot \frac{\exp(\mathbf{H}^{a,5}_{ijk})}{\sum_t \exp(\mathbf{H}^{a,5}_{ijt}) \cdot \mathbf{M}^a_{ijt}}, \tag{12}$$

where $\mathbf{W}^{a,3} \in \mathbb{R}^{d_2 \times (Fd_3)}$ and $\mathbf{W}^{a,4} \in \mathbb{R}^{d_3 \times M}$ are learnable weights, and $\mathbf{H}^{a,5}$ has dimensions $n_a \times F \times M$. The last dimension of $\mathbf{P}^a$ presents probabilities for each possible value of discrete attributes. To apply the Gumbel-Softmax Trick, we introduce a random matrix $\mathbf{G}$ where each row follows the $\mathrm{Gumbel}(0, 1)$ distribution, and a temperature parameter τ. A one-hot matrix for discrete attributes is then formed as:

$$\mathbf{H}^{a,6}_{ijk} = \frac{\exp(\log \mathbf{P}^a_{ijk} + \mathbf{G}_{ijk}/\tau)}{\sum_t \exp(\log \mathbf{P}^a_{ijt} + \mathbf{G}_{ijt}/\tau)}, \tag{13}$$

where the last dimension of $\mathbf{H}^{a,6}_{ijk}$ is as a one-hot vector, and the index of the 1 element indicates the selected value of the discrete attribute. Ultimately, GZOO obtains the complete set of discrete attributes through element-wise multiplication, known as the Hadamard product, between $\mathbf{H}^{a,6}_{ijk}$ and the category matrix $\mathbf{C}$:

$$\tilde{\mathbf{X}} = \mathbf{H}^{a,6} \circ \mathbf{C}, \tag{14}$$

where $\circ$ denotes the element-wise multiplication.

As outlined above, the sub-generator $\mathcal{G}^a$ uses a linear layer to generate continuous attributes and a category matrix $\mathbf{C}$ to produce discrete attributes.

Structure Sub-generator $\mathcal{G}^s$: To establish connections between each malicious node and the original graph, GZOO aims to create n_s edges using $\mathcal{G}^s$. The structure sub-generator undergoes n_s iterations, employing the Gumbel-Softmax Trick, similar to the procedure utilized for discrete attribute generation. In the tth iteration, for a specific malicious node v_i, the structure sub-generator selects an additional node v_j from the subgraph to form the edge (v_i, v_j). The probabilities of selecting v_j are computed as follows:

$$\mathbf{P}^s_{ji} = \mathbf{M}^{s,t}_{ij} \cdot \frac{\exp(\mathbf{H}^{s,3}_{ij})}{\sum_k \exp(\mathbf{H}^{s,3}_{kj}) \cdot \mathbf{M}^{s,t}_{ij}}, \tag{15}$$

where $\mathbf{P}^s \in \mathbb{R}^{n_a \times n}$ represents the probabilities, and $\mathbf{M}^{s,t}$ is the mask matrix during the tth iteration. Subsequently, GZOO constructs a one-hot matrix $\mathbf{H}^{s,4}$, where each row is a one-hot vector indicating the chosen additional node v_j corresponding to the malicious node v_i:

$$\mathbf{H}^{s,4}_{ij} = \frac{\exp(\log \mathbf{P}^s_{ij} + \mathbf{G}_{ij}/\tau)}{\sum_k \exp(\log \mathbf{P}^s_{ik} + \mathbf{G}_{ik}/\tau)}. \tag{16}$$

During the first iteration, $\tilde{\mathbf{A}}$ is initialized to $\mathbf{0}$, and $\mathbf{M}^{s,1}$ is set as the identity matrix $\mathbf{I}$. Subsequently, the adjacency matrix $\tilde{\mathbf{A}}$ and the mask matrix $\mathbf{M}^{s,t}$ are updated as follows:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{A}} + \mathbf{H}^{s,4}_{ij}; \quad \mathbf{M}^{s,t} = \mathbf{M}^{s,t-1} \circ (1 - \mathbf{H}^{s,4}). \tag{17}$$

The update process ensures that each malicious node is connected to a different additional node during each iteration.

The sub-generator $\mathcal{G}^s$ constructs n_s edges by iteratively linking malicious nodes to distinct nodes within the K-order subgraph of the target node v_i.

C. Loss Function

The adversary's dual goals, which involve 1) causing the victim model to misclassify the target node v_i, and 2) maintaining the stealthiness of malicious nodes, have prompted the proposal of two distinct loss functions: the attack loss $\mathcal{L}_1$ and the attribute loss $\mathcal{L}_2$.

Attack Loss: The primary objective of the attack is to manipulate the victim model into an incorrect classification of the target node v_i. This goal is encapsulated by defining the attack loss as:

$$\mathcal{L}_1(\tilde{\mathbf{A}}, \tilde{\mathbf{X}}) = \max(f_{\theta^\star}(\mathbf{A}', \mathbf{X}'_i)_{\mathbf{Y}^U_i} - \max_{c \neq \mathbf{Y}^U_i} f_{\theta^\star}(\mathbf{A}', \mathbf{X}'_i)_c, -\kappa), \tag{18}$$

Here, $f_{\theta^\star}$ represents the victim model, while $\mathbf{A}'$ and $\mathbf{X}'_i$ represent the adversarial attribute and adjacency matrix, respectively. $\mathbf{Y}^U_i$ is the ground-truth label of the target node v_i. The factor κ serves as a margin to ensure that the victim model assigns higher probabilities to labels other than the ground-truth label. According to the form of (18), this loss can be employed to concurrently optimize both the attribute and structure sub-generators.

Attribute Loss: The attribute loss serves the purpose of ensuring that the attributes of malicious nodes remain relatively close to those within the original graph. For discrete attribute space, we utilize the norm measure as follows:

$$\mathcal{L}_2(\tilde{\mathbf{X}}) = (\|\tilde{\mathbf{X}}\|/\|\mathbf{X}\|)^2. \tag{19}$$

For continuous attribute space, we restrict the deviation between the mean values and standard deviations of malicious nodes and original nodes, expressed as:

$$\mathcal{L}_2(\tilde{\mathbf{X}}) = (\text{Mean}(\tilde{\mathbf{X}}) - \text{Mean}(\mathbf{X}))^2 + (\text{Std}(\tilde{\mathbf{X}}) - \text{Std}(\mathbf{X}))^2. \tag{20}$$

Note that this loss can only be employed to optimize the attribute sub-generator.

The total loss $\mathcal{L}$ is a combination of two losses with a balancing parameter α, given by:

$$\mathcal{L} = \mathcal{L}_1 + \alpha\mathcal{L}_2. \tag{21}$$

By minimizing $\mathcal{L}$, the model g_ϕ is guided to craft malicious nodes that effectively deceive the victim model while maintaining inconspicuous attributes to avoid detection.

D. Model Optimization

In black-box scenarios where the victim model's architecture and parameters are not accessible, zeroth-order optimization [29] is a viable method for optimizing the generator's parameters. This approach involves evaluating the victim model multiple times with noisy inputs to estimate gradients, allowing the generator to adjust its parameters without direct gradient information from the victim model. Given that graphs involve both continuous and discrete attributes, the optimization must also address the generator's components responsible for handling discrete variables.

The gradient with respect to ϕ, which represents the parameters of the generator g, can be expressed:

$$\frac{\partial\mathcal{L}}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\phi} + \alpha\frac{\partial\mathcal{L}_2}{\partial\phi}. \tag{22}$$

Given that the victim model $f_{\theta^\star}$ is solely involved in the attack loss, GZOO only requires the zeroth-order optimization algorithm to estimate $\partial\mathcal{L}_1/\partial\phi$. This gradient can be further represented as:

$$\frac{\partial\mathcal{L}_1}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\phi} + \frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{A}}}\cdot\frac{\partial\tilde{\mathbf{A}}}{\partial\phi}. \tag{23}$$

Attribute Sub-generator Optimization: Since the sub-generator employs different architectures to generate continuous and discrete attributes, distinct gradient estimation techniques are necessary for these two types of networks.

For continuous attributes, the sub-generator leverages (11) to craft attributes $\tilde{\mathbf{X}}$ for malicious nodes. The gradient $\partial\mathcal{L}_1/\partial\tilde{\mathbf{X}}$ can be estimated as:

$$\frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{X}}} = \frac{1}{B\cdot\eta}\sum_{i=1}^{B}[\mathcal{L}_1(\tilde{\mathbf{A}},\tilde{\mathbf{X}}+\eta\mathbf{U}_i^a) - \mathcal{L}_1(\tilde{\mathbf{A}},\tilde{\mathbf{X}})]\mathbf{U}_i^a. \tag{24}$$

Here, $\mathbf{U}_i^a$ are drawn from the standard multivariate normal distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$, B is the number of sampled directions, and η symbolizes the step size.

For discrete attributes, referring to (12), the gradient $\partial\tilde{\mathbf{X}}/\partial\phi$ can be further represented as $\partial\tilde{\mathbf{X}}/\partial\phi = \partial\tilde{\mathbf{X}}/\partial\mathbf{P}^a\cdot\partial\mathbf{P}^a/\partial\phi$. Thus, the first term of (23) is reformulated as

$$\frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{X}}}\cdot\frac{\partial\tilde{\mathbf{X}}}{\partial\mathbf{P}^a}\cdot\frac{\partial\mathbf{P}^a}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\mathbf{P}^a}\cdot\frac{\partial\mathbf{P}^a}{\partial\phi}. \tag{25}$$

The gradient $\partial\mathcal{L}_1/\partial\mathbf{P}^a$ can be estimated via:

$$\frac{\partial\mathcal{L}_1}{\partial\mathbf{P}^a} = \frac{1}{B\cdot\eta}\sum_{i=1}^{B}[\mathcal{L}_1(\tilde{\mathbf{A}},\mathcal{G}^a(\mathbf{P}^a+\eta\mathbf{U}_i^a)) - \mathcal{L}_1(\tilde{\mathbf{A}},\mathcal{G}^a(\mathbf{P}^a))]\mathbf{U}_i^a. \tag{26}$$

In addition, $\partial\mathbf{P}^a/\partial\phi$ can be efficiently calculated using the back-propagation algorithm.

Structure Sub-generator Optimization: The generator g employs the structure sub-generator to sample injected edges. According to (15), the second term of (23) is reformulated as

$$\frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{A}}}\cdot\frac{\partial\tilde{\mathbf{A}}}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\tilde{\mathbf{A}}}\cdot\frac{\partial\tilde{\mathbf{A}}}{\partial\mathbf{P}^s}\cdot\frac{\partial\mathbf{P}^s}{\partial\phi} = \frac{\partial\mathcal{L}_1}{\partial\mathbf{P}^s}\cdot\frac{\partial\mathbf{P}^s}{\partial\phi}. \tag{27}$$

The gradient $\partial\mathcal{L}_1/\partial\mathbf{P}^s$ can be estimated as

$$\frac{\partial\mathcal{L}_1}{\partial\mathbf{P}^s} = \frac{1}{B\cdot\eta}\sum_{i=1}^{B}[\mathcal{L}_1(\mathcal{G}^s(\mathbf{P}^s+\eta\mathbf{U}_i^s),\tilde{\mathbf{X}}) - \mathcal{L}_1(\mathcal{G}^s(\mathbf{P}^s),\tilde{\mathbf{X}})]\mathbf{U}_i^s. \tag{28}$$

Error Analysis of Gradient Estimation: Considering that (24), (26), and (28) all perform gradient estimation similarly, we analyze the gradient estimation error using (24) as an example. For simplicity, let

$$\kappa^a(\tilde{\mathbf{X}}+\eta\mathbf{U}_i) = \mathcal{L}_1(\tilde{\mathbf{A}},\tilde{\mathbf{X}}+\eta\mathbf{U}_i),$$

$$\mathbf{G} = {}^1/_\eta[\kappa^a(\tilde{\mathbf{X}}+\eta\mathbf{U}) - \kappa^a(\tilde{\mathbf{X}})]\mathbf{U},$$

and

$$\bar{\mathbf{G}} = {}^1/_{B\cdot\eta}\sum_{i=1}^{B}[\kappa^a(\tilde{\mathbf{X}}+\eta\mathbf{U}_i) - \kappa^a(\tilde{\mathbf{X}})]\mathbf{U}_i.$$

Theorem 1: Assume $\kappa^a : \mathbb{R}^{N\times F} \mapsto \mathbb{R}$ is differentiable, and its gradient $\nabla\kappa^a(\cdot)$ is L-Lipschitz .[1] Then, the mean squared estimation error of $\bar{\mathbf{G}}$ in (24) is upper bounded by:

$$\mathbb{E}[\|\bar{\mathbf{G}} - \nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2] \le 4\left(\frac{1}{n_a^2F^2} + \frac{1}{n_aFB} - \frac{(B-1)^2}{n_a^2F^2}\right)\nabla\kappa^a(\tilde{\mathbf{X}}) + \frac{2B+1}{B}L^2\eta^2, \tag{29}$$

[1] A function $W(\cdot)$ is L-Lipschitz if $\|W(\mathbf{w}_1) - W(\mathbf{w}_2\|_2 \le L\|\mathbf{w}_1 - \mathbf{w}_2\|_2$ for any $\mathbf{w}_1$ and $\mathbf{w}_2$.

which shows that increasing the sample size B and decreasing the step size η both reduce the gradient estimation error.

Proof: Recall that the dimension of $\tilde{\mathbf{X}}$ is $n_a \times F$. Based on S5 from Tu et al. [30], we have:

$$\mathbb{E}_{\mathbf{U}}[\mathbf{G}] = {}^{1}/_{n_a F}\nabla\kappa^a(\tilde{\mathbf{X}}) + \epsilon, \tag{30}$$

where $\|\epsilon\|_2 \leq {}^{\eta L}/_{2}$. Applying Lemma 4.1-b from Gao et al. [31], we obtain:

$$\mathbb{E}_{\mathbf{U}}[\|\mathbf{G}\|_2^2] \leq {}^{L^2\eta^2}/_{2} + {}^{2}/_{n_a F}\|\nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2. \tag{31}$$

Given the i.i.d. nature of samples $\{\mathbf{U}_i\}$ and (30), define:

$$\mathbf{V} =: \mathbb{E}[\mathbf{G}_i] = {}^{1}/_{n_a F}\nabla\kappa^a(\tilde{\mathbf{X}}) + \epsilon. \tag{32}$$

Furthermore, we have:

$$\mathbb{E}[\|\bar{\mathbf{G}}\|_2^2] = \|\mathbf{V}\|_2^2 + {}^{1}/_{B}\mathbb{E}[\|\mathbf{G}_0\|_2^2] - {}^{1}/_{B}\|\mathbf{V}\|_2^2, \tag{33}$$

using the fact that $\mathbb{E}[\mathbf{G}_i] = \mathbb{E}[\mathbf{G}_0] = \mathbf{V} \quad \forall i$. From (32), it follows that:

$$\|\mathbf{V}\|_2^2 \leq 2{}^{1}/_{n_a^2 F^2}\|\nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2 + {}^{1}/_{2}\eta^2 L^2. \tag{34}$$

Additionally, from (30), we have for any i:

$$\mathbb{E}[\|\mathbf{G}_i\|_2^2] \leq {}^{L^2\eta^2}/_{2} + {}^{2}/_{n_a F}\|\nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2. \tag{35}$$

Substituting (34) and (35) into (33), we obtain:

$$\begin{aligned}&\mathbb{E}[\|\bar{\mathbf{G}}\|_2^2]\\ &\leq 2\left(\frac{1}{n_a^2F^2} + \frac{1}{n_aFB}\right)\|\nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2 + \frac{B+1}{2B}L^2\eta^2.\end{aligned} \tag{36}$$

Finally, the estimation error is bounded as follows:

$$\begin{aligned}&\mathbb{E}[\|\bar{\mathbf{G}} - \nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2]\\ &\leq 2\mathbb{E}[\|\bar{\mathbf{G}} - \mathbf{V}\|_2^2] + 2\|\mathbf{V} - \nabla\kappa^a(\tilde{\mathbf{X}})\|_2^2\\ &\leq 2\mathbb{E}[\|\bar{\mathbf{G}}\|_2^2] + 2\left\|\frac{1}{n_aF}\nabla\kappa^a(\tilde{\mathbf{X}}) + \epsilon - \nabla\kappa^a(\tilde{\mathbf{X}})\right\|_2^2\\ &\leq 4\left(\frac{1}{n_a^2F^2} + \frac{1}{n_aFB} - \frac{(B-1)^2}{n_a^2F^2}\right)\\ &\quad\times\nabla\kappa^a(\tilde{\mathbf{X}}) + \frac{2B+1}{B}L^2\eta^2.\end{aligned} \tag{37}$$

□

Through gradient estimation within the zeroth-order optimization, we effectively calculate the gradients with respect to ϕ for both attribute and structure sub-generators. Subsequently, we utilize the gradient descent algorithm to optimize the parameters ϕ of the generator. Thus, GZOO can effectively achieve its objectives in a black-box setting.

VI. Experimental Evaluation

This section delves into four key research questions to comprehensively evaluate the effectiveness, robustness, sensitivity to hyper-parameters and attack budgets, and stealthiness of the proposed GZOO attack:

- **RQ I**: Does GZOO outperform existing state-of-the-art NIAs in terms of attack effectiveness?
- **RQ II**: Does GZOO demonstrate robustness against various defenses, and prove effective across datasets with multi-valued discrete attribute spaces?
- **RQ III**: Does GZOO deceive victim models in other tasks, e.g., link prediction and graph classification?
- **RQ IV**: Does GZOO exhibit stability in its performance under varying hyperparameters?
- **RQ V**: Does the inner mechanism of GZOO, such as the number of layers used to craft continuous attributes, improve GZOO's performance?
- **RQ VI**: Can GZOO inject malicious nodes into the original graph without raising suspicion?

A. Experimental Setup

Implementation: The hyperparameters for GZOO are configured as follows: The embedding dimensions, d_1, d_2, and d_3, are set to 64, 128, and 4, respectively. The Gumbel-Softmax temperature τ is fixed at 10^{-3}. The step size for gradient estimation η, is set to 10^{-5}. The loss balancing factor α, is set to 1.0. The generator g_ϕ is optimized using the Adam algorithm with a learning rate of 10^{-4} . All experiments are repeated three times, with mean values and standard deviations reported.

Datasets: Our experiments scope sixteen diverse datasets, including Cora, Citeseer [32], Amazon Photo, Amazon Computer [33], ENZYMES, PROTEINS [34], PPI [35], Coauthor CS, Coauthor Physics [33], Reddit [35], OGBN-Products [36], Wiki CS [37], Pubmed [32], Flickr [38], OGBL-Collab [39], and MNISTSuperpixels [40]. For brevity, we abbreviate some datasets as follows: AmzP (Amazon Photo), AmzC (Amazon Computer), CaCS (Coauthor CS), CaPhy (Coauthor Physics), OGBNPro (OGBN-Products), WikiCS (Wiki CS), and MNIST-S (MNISTSuperpixels). These datasets span multiple domains, including citation networks, social networks, product networks, and protein networks.

Certain datasets, such as Cora, Citeseer, AmzP, AmzC, ENZYMES, and PROTEINS, feature binary-valued discrete attribute spaces, while others, like PPI, CaCS, and CaPhy, have multi-valued discrete attribute spaces. Datasets including Pubmed, OGBNPro, WikiCS, Reddit, Flickr, OGBL-Collab, and MNIST-S exhibit continuous attribute spaces. This diverse selection enables a thorough evaluation of the attack performance across different domains. We use the dataset implementations provided by the DGL library [41]. For the PPI dataset, DGL introduces reverse edges and removes duplicates, diverging from the original paper [35]. For the Reddit and OGBNPro datasets, we apply subgraph sampling and select the largest connected components, following the methods of Tao et al. [12] and Ju et al. [14] to ensure fair comparisons. Further details about these datasets are provided in Table I.

GNN Model: In our experiments, we utilize six widely adopted GNNs as victim models: GCN [26], SGC [42], APPNP [43], ChebNet [44], SGFormer [45], and PC-Net [46]. Additionally, we employ GAT [47] as the surrogate model for NIAs in the black-box setting.

Baselines: For a comprehensive evaluation, we utilize five state-of-the-art NIAs: G ^{2} A2C [14], TDGIA [13], G-NIA [12],

TABLE I
DATASET STATISTICS

Datasets	$\lvert\mathcal{G}\rvert$	N	$\lvert\mathbf{A}\rvert$	C	F	# Target†
Datasets with Binary-valued Discrete Attribute Space						
Cora	1	2,708	5,429	7	1433	801
Citeseer	1	3,327	4,732	6	3,703	696/1,000
AmzP	1	7,650	119,043	8	745	4,966
AmzC	1	13,752	245,778	10	767	8,309
ENZYMES	600	33	124	6	3	56
PROTEINS	1,113	39	146	2	3	609
Datasets with Multi-valued Discrete Attribute Space						
PPI*	24	1,767	34,085	2	50	956
CaCS	1	18,333	81,894	15	6,805	13,409
CaPhy	1	34,493	247,962	5	8,415	26,005
Datasets with Continuous Attribute Space						
Reddit*	1	10,004	37,014	41	602	1,701
OGBNPro*	1	10,494	77,656	35	100	1,604
WikiCS	1	11,701	216,123	10	300	4,632
Pubmed	1	19,717	44,338	3	500	763
Flickr	1	89,250	899,756	7	500	1,000
OGBL-Collab	1	235,868	1,285,465	2	128	1,000
MNIST-S	70,000	75	1,393	10	1	21,309

†: In node classification tasks, this denotes the number of target nodes, while in graph classification and link prediction tasks, it refers to the number of target graphs and target edges, respectively.
∗: The complete PPI dataset consists of 24 graphs. Given the focus of this paper on the semi-supervised node classification task, we choose to evaluate the attack performance of NIAs in experiments using the first graph.
⋆: In the Reddit and OGBNPro datasets, we employ subgraph sampling and substitute the entire graph with the largest connected component of the entire graph for evaluating attack performance. This operation aims to ensure a fair comparison with NIAs proposed by both Tao *et al.* [12] and Ju *et al.* [14].

SPEIT [18], and PGD [23], [25]. Detailed descriptions of these attacks are provided in Section II. To compare GZOO with attacks that require gradients from the victim model, we employ a two-layer GAT as the surrogate model. In constrained scenarios, we use default settings with $n_a = 1, n_s = 1$, and $\delta = 1.0$, unless otherwise specified.

Defenses: To address vulnerabilities posed by NIAs, we employ three established robust GNNs: RGAT [23], GNNGuard [48], and RobustGCN [49], as recommended by recent surveys [6]. Additionally, we include three defense mechanisms: MAGNet [50], ProGNN [51], and GTrans [52]. RGAT and GNNGuard utilize graph purification to remove dissimilar neighbors during message passing, while RobustGCN improves hidden representation stability by redesigning its loss function to counter NIAs. MAGNet uses a graph autoencoder to detect corrupted attributes, ProGNN adapts and refines graph data at test time to defend against adversarial attacks, and GTrans jointly learns a structural graph and a robust GNN from the perturbed graph, guided by properties like low-rank and sparsity.

Evaluation Metrics: To assess attack effectiveness, we utilize Attack Success Rate (ASR) as our metric. GZOO, characterized as an evasion attack, employs ASR to represent the misclassification rate, indicating the proportion of target nodes incorrectly classified after an attack. In general, a higher ASR indicates better performance of NIAs.

B. RQ I: Effectiveness Evaluation

In this subsection, we analyze GZOO's effectiveness compared to five state-of-the-art attacks across eight datasets. Evaluations use six commonly employed GNNs: GCN, SGC, APPNP, ChebNet, SGFormer, and PC-Net. Experimental results for these GNNs in Table II lead to two key observations:

1) GZOO exhibits superior performance, attaining the highest ASR across all datasets and significantly outperforming other attacks. For example, on the Cora dataset, GZOO achieves an ASR of 95.69%, well above the 66.01% ASR of the next-best attack, SPEIT. This exceptional performance is due to GZOO's ability to simultaneously generate the attributes and structures of malicious nodes, thereby leveraging the interrelationships between edges and attributes to maximize the impact of the malicious nodes.
2) GNNs trained on datasets with binary-valued discrete attributes show greater resistance to NIAs compared to those trained on datasets with continuous attributes. For instance, GZOO achieves an ASR close to 100.0% on datasets such as Reddit, OGBNPro, WikiCS, and Pubmed, significantly outperforming its results on Cora, Citeseer, AmzP, and AmzC datasets. This observation corroborates the findings of Ju et al. [14]. The underlying reason is that, when attacking datasets with binary-valued discrete attributes, GZOO optimizes a probability matrix $\mathbf{P}^a$ (as described in (12)) used for attribute sampling via the Gumbel-Softmax Trick. This probability matrix is highly sensitive to changes, especially when probabilities are near 0.5, making the attack loss function $\mathcal{L}_a$ non-smooth in binary attribute spaces. As a result, optimizing this probability matrix is more complex than optimizing continuous variables.

C. RQ II: Robustness Evaluation

In this subsection, we evaluate GZOO's robustness against six GNNs equipped with defenses, i.e., RGAT, GNNGuard, RobustGCN, MAGNet, ProGNN, and GTrans, and examine its performance across multiple datasets with multi-valued discrete attribute spaces.

Attacks against GNNs with Defenses: This assessment, conducted across eight datasets, mirrors the evaluation in Section VI-B. The results are detailed in Table III and reveal two key observations:

1) GZOO achieves the highest ASRs in most scenarios. For example, it reaches a 45.85% ASR against the RGAT model on the Cora dataset, significantly outperforming the 18.95% ASR of SPEIT. This advantage arises because GZOO does not rely on a surrogate model to approximate the victim model's behavior. Instead, it employs zeroth-order optimization to directly update the generative model's parameters, effectively utilizing information from the victim model to craft more impactful malicious nodes.
2) On the AmzP and AmzC datasets, GZOO attains only the second-best attack performance. This outcome is due to RobustGCN's introduction of Gaussian noise into latent representations during the message-passing process, which increases the discrepancy between the estimated and true gradients. As a result, the influence of the generated malicious nodes on target nodes is reduced. Furthermore, the higher average number of edges per target node in these datasets further diminishes the impact of the

TABLE II
EFFECTIVENESS OF GZOO COMPARED TO STATE-OF-THE-ART ATTACKS USING GCN, SGC, APPNP, CHEBNET, SGFORMER, AND PC-NET AS VICTIM MODELS (%)

Victim Model	Attacks	Datasets with Binary-valued Discrete Attribute Space				Datasets with Continuous Attribute Space			
		Cora	Citeseer	AmzP	AmzC	Reddit	OGBNPro	WikiCS	Pubmed
GCN	G-NIA [12]	02.80 ± 1.08	04.55 ± 0.54	01.17 ± 0.37	00.94 ± 0.08	24.36 ± 1.26	84.21 ± 2.86	05.70 ± 2.40	39.93 ± 3.15
	G^2A2C [14]	25.40 ± 0.48	35.51 ± 2.56	19.06 ± 1.86	21.27 ± 2.73	86.66 ± 0.05	93.83 ± 3.70	94.90 ± 1.35	60.94 ± 2.41
	PGD [23]	57.97 ± 1.11	58.60 ± 0.41	03.47 ± 0.30	03.53 ± 0.24	27.97 ± 1.16	72.41 ± 0.79	30.06 ± 0.84	69.22 ± 1.07
	SPEIT [18]	66.01 ± 0.92	64.12 ± 0.41	16.41 ± 1.65	14.18 ± 2.26	28.69 ± 0.88	48.88 ± 1.58	53.00 ± 1.43	86.87 ± 0.85
	TDGIA [13]	09.61 ± 0.42	29.77 ± 3.32	01.06 ± 0.22	01.30 ± 0.25	63.96 ± 1.45	85.82 ± 0.30	12.23 ± 1.47	37.02 ± 1.23
	GZOO (**ours**)	$\mathbf{95.69 \pm 1.94}$	$\mathbf{96.33 \pm 0.50}$	$\mathbf{47.93 \pm 3.46}$	$\mathbf{54.22 \pm 2.01}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{99.98 \pm 0.03}$	$\mathbf{99.88 \pm 0.04}$	$\mathbf{99.97 \pm 0.05}$
SGC	G-NIA [12]	03.21 ± 0.85	04.82 ± 1.37	00.98 ± 0.09	00.95 ± 0.16	28.59 ± 2.94	74.57 ± 1.45	05.56 ± 0.75	26.56 ± 3.37
	G^2A2C [14]	35.91 ± 0.29	28.41 ± 0.67	14.55 ± 1.61	12.09 ± 2.31	97.30 ± 1.79	96.93 ± 0.50	89.43 ± 1.63	81.34 ± 0.04
	PGD [23]	58.28 ± 0.45	59.53 ± 0.17	01.24 ± 0.93	00.98 ± 0.94	33.86 ± 2.81	76.27 ± 1.95	25.62 ± 9.69	66.30 ± 7.38
	SPEIT [18]	65.65 ± 0.51	63.90 ± 2.22	15.18 ± 1.46	12.75 ± 0.60	34.52 ± 1.38	53.49 ± 1.03	51.51 ± 2.73	84.39 ± 0.76
	TDGIA [13]	10.91 ± 1.23	23.80 ± 2.07	01.01 ± 0.12	00.85 ± 0.03	73.61 ± 0.26	91.09 ± 0.95	13.08 ± 0.24	41.95 ± 0.12
	GZOO (**ours**)	$\mathbf{96.87 \pm 0.93}$	$\mathbf{98.05 \pm 0.39}$	$\mathbf{45.67 \pm 2.34}$	$\mathbf{33.28 \pm 3.51}$	$\mathbf{99.95 \pm 0.08}$	$\mathbf{99.92 \pm 0.05}$	$\mathbf{99.72 \pm 0.13}$	$\mathbf{99.97 \pm 0.05}$
APPNP	G-NIA [12]	01.67 ± 0.35	02.88 ± 1.04	00.58 ± 0.14	00.53 ± 0.04	17.41 ± 3.10	51.47 ± 3.24	02.76 ± 0.46	30.06 ± 1.88
	G^2A2C [14]	04.12 ± 0.64	23.51 ± 3.31	02.60 ± 0.35	05.95 ± 1.24	96.67 ± 1.87	97.22 ± 0.62	84.29 ± 0.03	79.42 ± 0.16
	PGD [23]	64.55 ± 0.36	64.24 ± 0.47	07.44 ± 0.61	06.69 ± 1.02	55.40 ± 1.04	86.09 ± 1.09	47.48 ± 1.86	76.58 ± 0.82
	SPEIT [18]	64.92 ± 1.20	63.47 ± 0.44	05.10 ± 1.07	03.56 ± 1.93	21.32 ± 0.64	35.58 ± 0.48	42.66 ± 0.31	77.74 ± 3.29
	TDGIA [13]	16.64 ± 0.01	35.90 ± 1.78	01.25 ± 0.03	01.21 ± 0.02	91.55 ± 0.66	92.41 ± 0.19	21.12 ± 0.24	50.01 ± 0.41
	GZOO (**ours**)	$\mathbf{91.68 \pm 1.58}$	$\mathbf{95.10 \pm 0.97}$	$\mathbf{29.74 \pm 0.00}$	$\mathbf{23.59 \pm 0.94}$	$\mathbf{99.94 \pm 0.08}$	$\mathbf{99.96 \pm 0.04}$	$\mathbf{99.79 \pm 0.11}$	$\mathbf{99.97 \pm 0.06}$
ChebNet	G-NIA [12]	04.38 ± 1.12	09.91 ± 0.84	02.63 ± 0.21	03.30 ± 0.20	23.70 ± 2.86	64.53 ± 3.09	06.13 ± 1.21	22.89 ± 8.33
	G^2A2C [14]	18.02 ± 5.12	36.01 ± 4.18	16.77 ± 2.24	22.45 ± 1.34	93.37 ± 6.05	98.97 ± 0.85	74.20 ± 0.28	60.00 ± 3.07
	PGD [23]	63.34 ± 1.28	65.41 ± 1.05	06.03 ± 0.81	06.37 ± 0.45	53.77 ± 1.18	80.13 ± 2.11	48.13 ± 2.23	77.80 ± 2.99
	SPEIT [18]	63.21 ± 1.63	64.50 ± 1.02	05.90 ± 0.61	06.83 ± 1.10	15.90 ± 1.01	27.20 ± 1.30	43.43 ± 1.10	75.06 ± 2.25
	TDGIA [13]	82.53 ± 1.24	86.31 ± 3.15	06.83 ± 0.49	06.93 ± 1.02	95.20 ± 0.60	92.97 ± 1.10	30.93 ± 0.81	47.59 ± 3.10
	GZOO (**ours**)	$\mathbf{93.58 \pm 0.74}$	$\mathbf{93.26 \pm 2.42}$	$\mathbf{29.75 \pm 7.57}$	$\mathbf{24.53 \pm 1.79}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{99.90 \pm 0.10}$	$\mathbf{100.0 \pm 0.00}$
SGFormer	G-NIA [12]	15.55 ± 8.35	17.50 ± 7.93	01.87 ± 0.35	03.00 ± 0.85	19.15 ± 2.47	30.17 ± 2.82	06.95 ± 0.49	23.58 ± 8.19
	G^2A2C [14]	09.79 ± 2.60	40.95 ± 9.77	39.45 ± 2.47	06.03 ± 1.85	90.17 ± 5.82	75.13 ± 0.60	74.05 ± 0.07	59.97 ± 3.44
	PGD [23]	47.54 ± 5.50	54.47 ± 6.22	08.77 ± 0.70	07.60 ± 1.65	67.63 ± 1.55	80.50 ± 1.28	49.70 ± 3.08	75.33 ± 2.24
	SPEIT [18]	35.32 ± 2.46	45.83 ± 6.14	05.93 ± 1.56	05.57 ± 0.93	09.70 ± 0.70	12.87 ± 2.24	31.97 ± 3.06	66.17 ± 2.45
	TDGIA [13]	52.38 ± 9.69	69.61 ± 5.17	08.30 ± 0.56	07.53 ± 1.18	94.37 ± 1.75	91.93 ± 0.40	32.40 ± 4.90	48.01 ± 4.44
	GZOO (**ours**)	$\mathbf{74.33 \pm 7.90}$	$\mathbf{71.26 \pm 6.73}$	$\mathbf{48.67 \pm 3.40}$	$\mathbf{51.43 \pm 5.80}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{99.80 \pm 0.20}$	$\mathbf{100.0 \pm 0.00}$
PC-Net	G-NIA [12]	05.57 ± 0.37	10.14 ± 0.77	03.23 ± 0.59	02.73 ± 0.42	19.82 ± 3.16	64.88 ± 4.76	06.93 ± 0.64	29.39 ± 2.69
	G^2A2C [14]	11.30 ± 1.65	26.04 ± 7.73	04.50 ± 0.95	03.97 ± 1.21	93.58 ± 6.86	91.64 ± 5.07	87.40 ± 6.36	61.96 ± 0.86
	PGD [23]	60.37 ± 3.35	61.77 ± 0.55	04.23 ± 0.87	03.65 ± 0.38	22.78 ± 1.63	57.65 ± 3.26	36.00 ± 1.15	70.21 ± 1.40
	SPEIT [18]	60.10 ± 3.41	60.80 ± 0.53	05.20 ± 0.95	03.82 ± 0.33	06.83 ± 1.65	16.79 ± 1.75	31.27 ± 0.31	67.29 ± 2.41
	TDGIA [13]	68.45 ± 7.04	78.46 ± 7.05	04.77 ± 0.47	03.23 ± 0.65	86.61 ± 2.54	89.22 ± 1.53	18.90 ± 1.87	40.05 ± 2.99
	GZOO (**ours**)	$\mathbf{70.46 \pm 4.94}$	$\mathbf{80.63 \pm 2.71}$	$\mathbf{17.67 \pm 1.78}$	$\mathbf{21.48 \pm 8.02}$	$\mathbf{99.97 \pm 0.06}$	$\mathbf{99.73 \pm 0.15}$	$\mathbf{99.83 \pm 0.06}$	$\mathbf{99.95 \pm 0.08}$

malicious nodes. These factors collectively contribute to GZOO's suboptimal performance on these datasets.

Attacks on Datasets with Multi-valued Discrete Attribute Space: We evaluate GZOO's performance on datasets with multi-valued discrete attributes, i.e., PPI, CaCS, and CaPhy datasets. Five baselines, which are only tailored for datasets with binary discrete attributes, are not applicable to these datasets. Results in Table IV reveal following observations:

GZOO effectively deceives GNNs trained on datasets with multi-valued discrete attributes. For example, GZOO achieves a 79.31% ASR against the GCN model on the PPI dataset. This success is due to the design of the attribute generator $\mathcal{G}^a$ within the adversarial graph generative model, which generates both binary-valued and multi-valued discrete attributes.

D. RQ III: Evaluations on Other Tasks

In this subsection, we evaluate GZOO's performance on both link prediction and graph classification tasks. For link prediction, we introduce one malicious node between two nodes connected by an edge to disrupt the victim model's predictions, using the Citeseer, OGBL-Collab, and Flickr datasets. For graph classification, we insert one malicious node into the original graph to mislead the victim model, employing the PROTEINS, ENZYMES, and MNIST-S datasets. The experimental results are summarized in Table V, which reveal:

1) Link prediction tasks are more challenging than node and graph classification tasks. This increased difficulty likely arises from the graph homophily, where nodes connected by edges exhibit highly similar attributes. Consequently, modifying latent representations of target nodes by attaching a single malicious node proves to be difficult.
2) GZOO attains the highest ASRs in all link prediction task evaluations. For instance, on the Citeseer dataset with GCN as the victim model, GZOO achieves a 15.29% ASR, substantially exceeding the 5.93% ASR of G^2 A2C.
3) GZOO achieves the highest ASRs in most graph classification tasks. For example, GZOO achieves an average ASR of 54.26% on the PROTEINS dataset with GCN as the victim model, exceeding the 33.51% ASR of G-NIA. This highlights the effectiveness of zeroth-order optimization in training the adversarial graph generator.

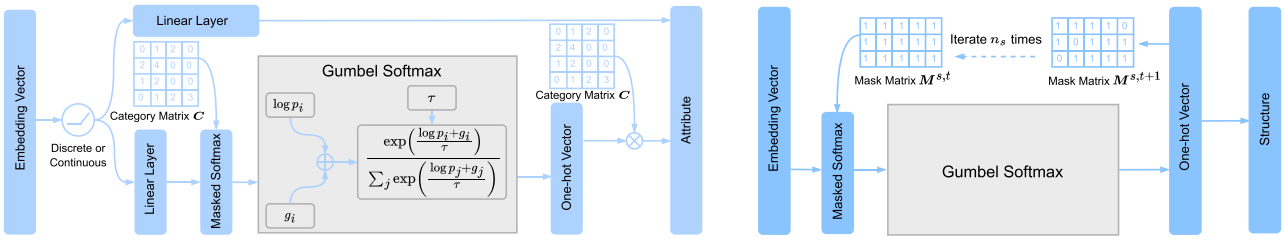
E. RQ IV: Sensitivity Analysis

In this subsection, we investigate the performance of GZOO under varying hyperparameters.

Attack Budget: It consists of parameters: the number of malicious nodes (n_a), the number of edges injected per malicious node (n_s), and the attribute shift limit (δ). We assess the impact of these parameters on GZOO's attack performance using the AmzC and AmzP datasets. The values for n_a and n_s range from

TABLE III
ROBUSTNESS OF GZOO COMPARED TO STATE-OF-THE-ART ATTACKS USING SIX VICTIM MODELS(%)

Victim Model	Attacks	Datasets with Binary-valued Discrete Attribute Space				Datasets with Continuous Attribute Space			
		Cora	Citeseer	AmzP	AmzC	Reddit	OGBNPro	WikiCS	Pubmed
RGAT	G-NIA [12]	05.76 ± 0.80	14.65 ± 0.40	02.49 ± 0.03	04.28 ± 0.07	07.57 ± 1.92	10.92 ± 0.37	04.54 ± 0.12	24.44 ± 3.90
	G^2A2C [14]	07.09 ± 0.50	23.52 ± 4.08	03.14 ± 0.01	03.95 ± 0.52	18.74 ± 0.34	33.05 ± 2.07	06.15 ± 0.35	25.25 ± 1.99
	PGD [23]	11.66 ± 0.38	25.98 ± 0.08	03.18 ± 0.12	04.19 ± 0.02	04.10 ± 0.89	07.71 ± 0.87	09.66 ± 0.30	26.44 ± 3.85
	SPEIT [18]	18.95 ± 0.24	34.26 ± 1.93	03.39 ± 0.20	04.49 ± 0.35	06.62 ± 0.46	16.86 ± 0.73	12.85 ± 0.27	47.60 ± 0.19
	TDGIA [13]	06.28 ± 0.47	20.38 ± 1.51	02.78 ± 0.21	03.62 ± 0.10	05.02 ± 0.63	08.91 ± 0.57	05.04 ± 0.27	10.81 ± 0.61
	GZOO (**ours**)	**45.85 ± 1.27**	**71.05 ± 2.52**	**05.95 ± 0.87**	**07.93 ± 3.64**	**74.34 ± 0.66**	**96.43 ± 0.02**	**30.58 ± 1.79**	**82.62 ± 1.65**
GNNGuard	G-NIA [12]	06.29 ± 2.45	11.01 ± 0.41	04.36 ± 0.30	04.24 ± 0.12	65.09 ± 2.93	71.65 ± 0.74	14.19 ± 2.62	53.51 ± 0.85
	G^2A2C [14]	47.29 ± 0.59	43.17 ± 0.41	44.23 ± 2.41	41.68 ± 1.69	96.53 ± 1.22	78.84 ± 0.09	76.75 ± 1.05	77.96 ± 2.00
	PGD [23]	66.96 ± 0.72	67.98 ± 0.57	33.81 ± 1.08	20.57 ± 1.87	36.14 ± 0.00	87.78 ± 0.16	79.32 ± 0.91	90.53 ± 0.56
	SPEIT [18]	67.07 ± 0.45	66.83 ± 1.05	35.80 ± 1.21	26.89 ± 0.39	24.82 ± 3.85	48.32 ± 2.67	74.61 ± 1.20	90.92 ± 0.84
	TDGIA [13]	52.92 ± 3.82	71.89 ± 1.53	06.97 ± 0.55	07.51 ± 0.79	64.41 ± 5.89	89.94 ± 0.18	61.04 ± 1.35	57.44 ± 2.41
	GZOO (**ours**)	**98.11 ± 0.12**	**98.51 ± 0.26**	**64.86 ± 0.60**	**72.54 ± 2.76**	**100.0 ± 0.00**	**99.82 ± 0.06**	**99.81 ± 0.08**	**99.73 ± 0.00**
RobustGCN	G-NIA [12]	04.62 ± 0.79	07.67 ± 1.49	04.70 ± 1.15	02.06 ± 0.20	13.59 ± 2.56	47.53 ± 3.03	07.56 ± 1.24	27.15 ± 2.39
	G^2A2C [14]	16.04 ± 1.20	23.51 ± 2.27	**07.99 ± 1.38**	**08.75 ± 2.23**	96.10 ± 1.36	93.95 ± 0.64	94.75 ± 0.09	81.00 ± 0.64
	PGD [23]	54.63 ± 1.47	58.20 ± 2.10	04.35 ± 0.94	02.25 ± 0.38	12.48 ± 0.46	40.84 ± 1.24	21.66 ± 0.76	65.76 ± 2.26
	SPEIT [18]	65.97 ± 0.55	63.46 ± 0.37	04.62 ± 1.83	02.52 ± 1.30	03.13 ± 1.20	09.70 ± 3.06	34.84 ± 0.97	79.99 ± 1.26
	TDGIA [13]	10.99 ± 3.06	34.10 ± 1.60	04.73 ± 1.11	01.94 ± 0.30	48.33 ± 1.58	77.83 ± 0.82	11.83 ± 0.71	32.04 ± 1.13
	GZOO (**ours**)	**76.29 ± 2.09**	**72.52 ± 3.72**	06.46 ± 0.21	05.31 ± 0.51	**99.05 ± 0.86**	**96.17 ± 1.25**	**96.20 ± 3.08**	**94.75 ± 3.60**
MAGNet	G-NIA [12]	02.06 ± 0.78	04.48 ± 1.10	01.65 ± 0.64	OOM	OOM	69.15 ± 8.41	OOM	22.67 ± 0.71
	G^2A2C [14]	26.95 ± 0.08	25.75 ± 0.78	12.50 ± 0.14	12.90 ± 0.71	93.15 ± 2.19	90.75 ± 1.34	67.10 ± 0.28	75.29 ± 4.45
	PGD [23]	62.89 ± 0.31	62.83 ± 1.32	07.70 ± 0.99	05.45 ± 0.78	46.70 ± 1.13	78.85 ± 0.07	49.70 ± 1.56	88.04 ± 0.25
	SPEIT [18]	63.26 ± 0.13	61.44 ± 1.41	08.55 ± 0.49	05.60 ± 0.99	16.00 ± 0.00	34.90 ± 0.57	47.05 ± 0.64	88.11 ± 0.16
	TDGIA [13]	75.41 ± 2.81	89.71 ± 1.34	12.95 ± 2.47	09.05 ± 2.47	93.30 ± 0.14	94.80 ± 0.14	40.55 ± 1.20	49.02 ± 0.46
	GZOO (**ours**)	**89.74 ± 0.21**	**93.34 ± 0.42**	**15.55 ± 3.75**	**29.05 ± 1.77**	**99.90 ± 0.00**	**99.95 ± 0.07**	**97.50 ± 1.98**	**100.0 ± 0.00**
ProGNN	G-NIA [12]	01.94 ± 0.80	04.27 ± 1.08	01.75 ± 0.49	OOM	OOM	74.55 ± 7.00	OOM	22.76 ± 2.43
	G^2A2C [14]	40.52 ± 1.22	27.88 ± 0.59	19.95 ± 2.62	26.95 ± 1.20	96.70 ± 2.12	93.60 ± 7.07	95.05 ± 2.76	77.84 ± 1.95
	PGD [23]	63.02 ± 0.57	61.95 ± 0.47	09.75 ± 0.78	05.80 ± 0.14	49.75 ± 0.35	84.40 ± 0.71	49.05 ± 2.90	80.24 ± 0.88
	SPEIT [18]	62.21 ± 0.49	60.48 ± 0.46	08.70 ± 1.13	05.75 ± 0.78	14.05 ± 0.49	33.55 ± 0.21	43.50 ± 2.69	79.33 ± 0.33
	TDGIA [13]	84.30 ± 3.22	86.04 ± 0.49	10.65 ± 0.35	06.65 ± 2.62	94.35 ± 0.49	91.65 ± 1.06	34.45 ± 0.92	49.42 ± 0.09
	GZOO (**ours**)	**91.48 ± 0.67**	**92.97 ± 0.75**	**31.00 ± 3.25**	**27.00 ± 4.81**	**100.0 ± 0.00**	**99.90 ± 0.14**	**99.70 ± 0.28**	**99.87 ± 0.18**
GTrans	G-NIA [12]	07.66 ± 0.27	16.67 ± 2.97	03.60 ± 1.98	01.20 ± 0.28	15.70 ± 2.26	59.13 ± 1.50	08.60 ± 0.14	25.04 ± 0.92
	G^2A2C [14]	11.69 ± 0.83	25.71 ± 0.66	07.15 ± 3.89	03.45 ± 0.07	98.05 ± 0.92	88.27 ± 0.20	95.25 ± 0.35	63.55 ± 0.50
	PGD [23]	60.66 ± 0.03	56.65 ± 1.82	09.85 ± 2.62	06.90 ± 0.57	41.15 ± 2.19	87.04 ± 4.19	45.25 ± 2.33	40.40 ± 1.39
	SPEIT [18]	55.65 ± 1.04	55.49 ± 0.67	09.40 ± 1.41	06.05 ± 0.64	10.70 ± 1.13	**100.0 ± 0.00**	39.80 ± 2.97	37.56 ± 2.16
	TDGIA [13]	72.96 ± 4.95	68.63 ± 6.33	12.35 ± 0.78	06.85 ± 1.91	94.25 ± 3.32	87.38 ± 3.71	30.35 ± 5.73	36.73 ± 1.68
	GZOO (**ours**)	**96.40 ± 4.12**	**99.12 ± 0.39**	**72.85 ± 1.77**	**54.05 ± 1.77**	**100.0 ± 0.00**	99.88 ± 0.18	**100.0 ± 0.00**	**99.88 ± 0.17**

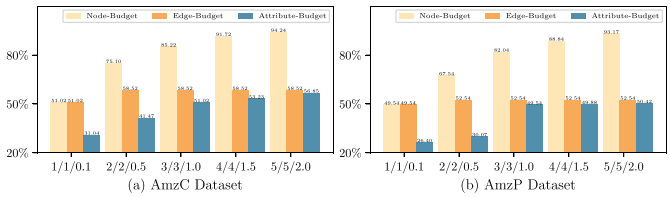


(a) Attribute sub-generator $\mathcal{G}^a$ (b) Structure sub-generator $\mathcal{G}^s$

Fig. 3. The architecture of the attribute and structure sub-generators utilizes representations produced by the backbone model as a shared input. The attribute sub-generator $\mathcal{G}^a$ is designed to generate both continuous and discrete attributes.

TABLE IV
ASRS OF GZOO ON PPI, CACS, AND CAPHY DATASETS (%)

Victim Model	PPI	CaCS	CaPhy
GCN	79.31 ± 3.04	12.02 ± 1.13	31.72 ± 2.63
GAT	56.32 ± 3.27	47.71 ± 2.70	32.77 ± 0.33
SGC	38.56 ± 0.26	18.87 ± 0.86	40.06 ± 2.29
APPNP	08.34 ± 2.12	04.05 ± 0.45	15.00 ± 0.68



(a) AmzC Dataset (b) AmzP Dataset

Fig. 4. Performance analysis of GZOO on AmzC and AmzP datasets under different node/edge/attribute attack budgets.

1 to 5, while δ varies between 0.1 and 2.0. In these evaluations, the other two parameters remain constant. Results in Fig. 4 indicate that:

1) As the budgets for nodes and features increase, the attack's performance improves. This is because larger budgets amplify the impact of malicious nodes on target nodes, enhancing GZOO's performance.
2) The attack's performance remains stable as the edge budget increases. This stability arises because a higher edge

TABLE V
PERFORMANCE OF GZOO ON LINK PREDICTION AND GRAPH CLASSIFICATION TASKS COMPARED TO STATE-OF-THE-ART ATTACKS (%)

Tasks	Attacks	GCN Citeseer	GCN OGBL-Collab	GCN Flickr	SGC Citeseer	SGC OGBL-Collab	SGC Flickr	APPNP Citeseer	APPNP OGBL-Collab	APPNP Flickr
Link Prediction	G-NIA [12]	01.58 ± 0.45	OOM	OOM	02.02 ± 0.57	OOM	OOM	00.86 ± 0.30	OOM	OOM
	G^2A2C [14]	05.93 ± 0.86	18.68 ± 2.58	18.67 ± 2.52	05.20 ± 2.46	17.24 ± 2.85	19.47 ± 3.78	05.07 ± 1.00	17.91 ± 1.56	23.20 ± 2.26
	PGD [23]	00.81 ± 0.42	15.36 ± 3.00	37.22 ± 3.98	01.08 ± 0.27	15.13 ± 3.87	20.47 ± 1.14	00.82 ± 0.24	14.87 ± 0.40	28.49 ± 3.01
	SPEIT [18]	00.92 ± 0.27	14.52 ± 2.05	39.31 ± 3.49	00.71 ± 0.39	17.22 ± 2.34	17.18 ± 1.79	01.18 ± 0.52	19.22 ± 2.01	35.76 ± 2.05
	TDGIA [13]	00.63 ± 0.55	08.58 ± 2.14	13.11 ± 1.54	00.87 ± 0.51	09.75 ± 1.51	14.51 ± 1.63	00.84 ± 0.46	04.60 ± 0.53	17.38 ± 3.56
	GZOO (**ours**)	$\mathbf{15.29 \pm 2.80}$	$\mathbf{32.44 \pm 5.81}$	$\mathbf{43.51 \pm 6.60}$	$\mathbf{10.10 \pm 1.15}$	$\mathbf{30.40 \pm 1.98}$	$\mathbf{58.91 \pm 1.72}$	$\mathbf{10.72 \pm 2.23}$	$\mathbf{46.24 \pm 2.54}$	$\mathbf{41.10 \pm 1.56}$
		PROTEINS	ENZYMES	MNIST-S	PROTEINS	ENZYMES	MNIST-S	PROTEINS	ENZYMES	MNIST-S
Graph Classification	G-NIA [12]	33.51 ± 2.98	87.65 ± 1.10	83.43 ± 1.33	33.11 ± 0.90	82.43 ± 2.42	83.00 ± 1.82	09.40 ± 0.69	73.24 ± 1.28	72.13 ± 2.12
	G^2A2C [14]	22.76 ± 1.73	91.89 ± 0.57	$\mathbf{100.0 \pm 0.00}$	26.56 ± 2.73	90.09 ± 2.60	$\mathbf{100.0 \pm 0.00}$	10.02 ± 0.91	76.46 ± 2.62	71.65 ± 2.33
	PGD [23]	12.03 ± 3.40	15.23 ± 5.60	06.60 ± 1.42	07.44 ± 2.65	17.07 ± 4.81	04.40 ± 0.70	01.11 ± 0.32	04.99 ± 3.09	05.33 ± 2.50
	SPEIT [18]	11.81 ± 4.04	13.41 ± 4.80	05.00 ± 0.57	07.29 ± 2.80	15.97 ± 1.75	05.93 ± 0.76	01.36 ± 0.38	05.26 ± 3.12	04.57 ± 0.81
	TDGIA [13]	04.27 ± 0.77	07.43 ± 2.23	06.75 ± 2.05	02.96 ± 1.15	15.27 ± 4.27	05.40 ± 1.04	00.99 ± 0.53	04.71 ± 3.09	05.70 ± 0.40
	GZOO (**ours**)	$\mathbf{54.26 \pm 0.42}$	$\mathbf{100.0 \pm 0.00}$	97.90 ± 0.44	$\mathbf{65.76 \pm 1.64}$	$\mathbf{100.0 \pm 0.00}$	99.40 ± 0.17	$\mathbf{27.21 \pm 1.33}$	$\mathbf{100.0 \pm 0.00}$	$\mathbf{100.0 \pm 0.00}$

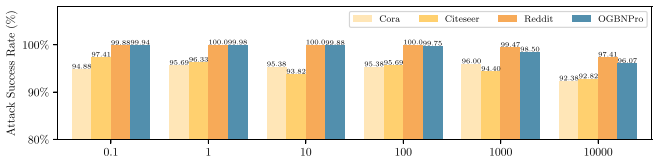


Fig. 5. GZOO's performance with varying levels of the loss balance factor α across four datasets.

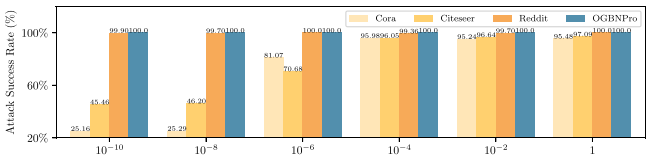


Fig. 6. The impact of the temperature parameter τ.

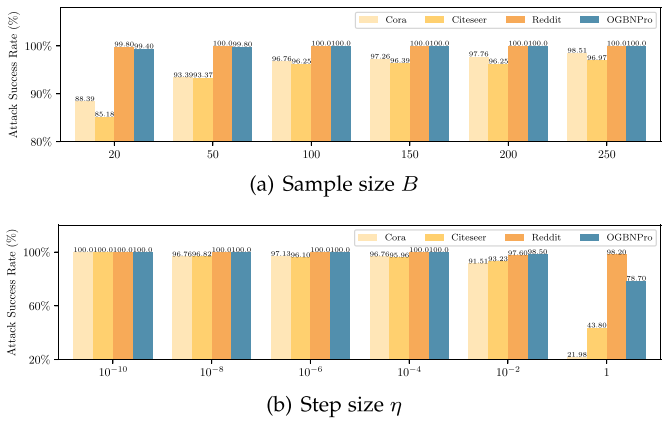


Fig. 7. The impact of zeroth-order optimization hyperparameters across various datasets.

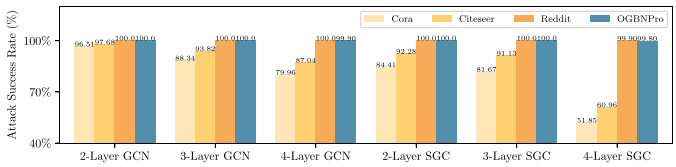


Fig. 8. The impact of the number of layers in the victim model.

budget means malicious nodes connect to more nodes in the original graph, which dilutes their influence due to the message-passing mechanism.

Loss Balance Factor α: It is used to balance attack and attribute losses as described in (21). We examine the effect of α on four datasets varying α in the range of $\{0.1, 1, 10, 10^2, 10^3, 10^4\}$ with GCN as the victim model. The experimental results are shown in Fig. 5.

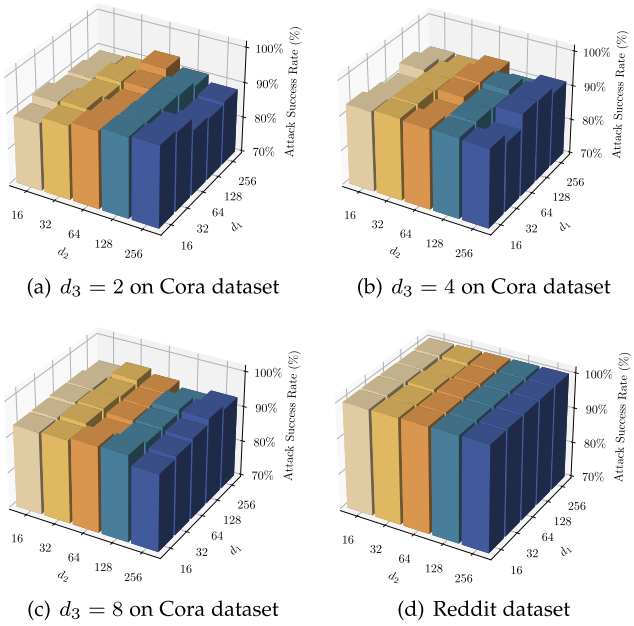
Our results indicate that GZOO's performance remains stable at lower values of the balancing factor α. However, when α is excessively large, GZOO's performance slightly diminishes. This occurs because a high α limits the extent to which the attributes of malicious nodes can deviate from the mean attributes of nodes in the original graph, thereby reducing the attack potential of the malicious nodes.

Temperature Parameter τ: We investigate the effect of the temperature parameter τ in (13) and (15), choosing τ values from $\{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$ for the Cora, Citeseer, Reddit, and OGBNPro datasets with GCN as the victim model. Results, shown in Fig. 6, indicate that GZOO's attack performance remains stable as long as τ is not excessively small (i.e., greater than 10^{-6}). For datasets with continuous attributes, such as Reddit and OGBNPro, the Gumbel-Softmax Trick is less critical, resulting in minimal impact from variations in τ.

Sample Size B and Step Size η: We assess the impact of hyperparameters in zeroth-order optimization on attack performance across the Cora, Citeseer, Reddit, and OGBNPro datasets. Specifically, we examine sample sizes in the range of $\{20, 50, 100, 150, 200, 250\}$ and step sizes in the range of $\{10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}$, using GCN as the victim model. The results, presented in Fig. 7, demonstrate that GZOO's attack performance remains stable as long as neither B is too small nor η is too large.

Number of Layers in Victim Models: We evaluate the effect of the number of layers in victim models on attack performance, with layer counts set to $\{2, 3, 4\}$ and using GCN and SGC as the victim models. Results in Fig. 8 reveal that increasing the number of layers enhances the models' resistance to attacks. This is due to the larger neighborhood of the message-passing mechanism in deeper models, which reduces the impact of individual malicious nodes on the latent representations of target nodes.

Dimension of Node Representations: We select dimensions for node representations as $d_1 \in \{16, 32, 64, 128, 256\}$, $d_2 \in$



(a) $d_3 = 2$ on Cora dataset (b) $d_3 = 4$ on Cora dataset

(c) $d_3 = 8$ on Cora dataset (d) Reddit dataset

Fig. 9. The effect of node representation dimensions on the Cora and Reddit datasets.

TABLE VI
THE EFFECT OF THE NUMBER OF LAYERS ON GENERATING CONTINUOUS ATTRIBUTES(%)

Datasets	1 Layer	2 Layers	3 Layers
Reddit	100.0 ± 0.00	99.90 ± 0.01	100.0 ± 0.01
OGBNPro	99.98 ± 0.03	100.0 ± 0.01	100.0 ± 0.02
WikiCS	99.88 ± 0.04	99.90 ± 0.02	99.50 ± 0.03
Pubmed	100.0 ± 0.05	100.0 ± 0.01	100.0 ± 0.04

$\{16, 32, 64, 128, 256\}$, and $d_3 \in \{2, 4, 8\}$. To prevent excessive size of $\mathbf{W}^{a,3} \in \mathbb{R}^{d_2 \times (Fd_3)}$, we ensure that d_3 is smaller than d_2. Note that d_3 applies only to discrete attributes, such as those in the Cora dataset. Thus, our evaluations consider the impact of d_1, d_2, and d_3 on the Cora dataset, and the effect of d_1 and d_2 on the Reddit dataset, using the GCN model as the victim. Results, shown in Fig. 9, indicate that GZOO's performance remains stable.

F. RQ V: Ablation Analysis

In this subsection, we evaluate the effectiveness of the inner mechanisms in GZOO.

Number of Layers for Continuous Attributes: To evaluate the effect of linear layer on attack performance, we test configurations with one, two, and three layers. The dimensionality of latent representations remained consistent across these configurations. Using GCN as the victim model, we conduct experiments on four datasets with continuous attributes: Reddit, OGBNPro, WikiCS, and Pubmed. The results, detailed in Table VI, indicate that the number of layers has a minimal effect on attack performance. Thus, we chose to use a single layer for generating continuous attributes of malicious nodes.

TABLE VII
THE EFFECTIVENESS OF REPRESENTATION CONCATENATION(%)

Datasets	GZOO	GZOO *w/ S*	GZOO *w/ A*
Cora	95.69 ± 1.94	93.88 ± 1.01	91.88 ± 0.03
Citeseer	96.33 ± 0.50	93.39 ± 0.98	90.68 ± 0.40
Reddit	100.0 ± 0.00	98.00 ± 1.30	96.00 ± 0.52
OGBNPro	99.98 ± 0.03	99.00 ± 1.20	97.00 ± 0.48

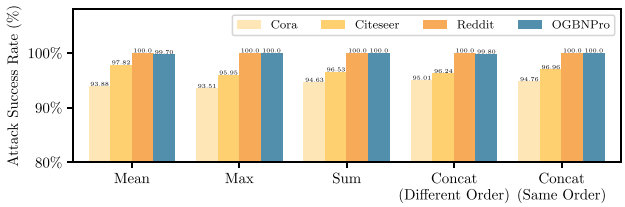


Fig. 10. The impact of methods of representation fusion.

Effectiveness of Representation Concatenation: We design the generator to simultaneously generate both attributes and structures of malicious nodes, addressing the limitations of existing methods that generate these elements sequentially (i.e., attributes first, then structures based on those attributes). Given that attributes and structures influence each other, fusing relevant information before their generation is a logical approach, as depicted in Fig. 2 and (7).

To validate this approach, we compare two variants: "GZOO *w/ S*" and "GZOO *w/ A*". "GZOO *w/ S*" uses fused information for structure generation while relying on original attribute information. In contrast, "GZOO *w/ A*" uses fused information for attributes and original information for structures. Experimental results in Table VII demonstrate the advantage of using fused information, particularly highlighting its significant impact on the generation of structures and overall attack performance (see Section IV).

Methods of Representation Fusion: We assess several representation fusion methods, including Mean, Max, Sum, and various concatenation orders. "Concat *with Same Order*" represents the same order of concatenation in (7). Experimental results in Fig. 10 indicate that the method and order of concatenation have a minimal impact.

G. RQ VI: Stealthiness Analysis

In this subsection, we evaluate the stealthiness of GZOO, emphasizing the importance of ensuring that injected malicious nodes remain undetected. While the concept of stealthiness is well-defined, quantitatively assessing it presents a significant challenge. To address this, we draw inspiration from the concept of node-centric homophily proposed by Chen et al. [23]. We employ this concept to gauge the similarity among neighbors and leverage changes in node-centric homophily after GZOO attacks for evaluating the stealthiness of NIAs. The distribution of node-centric homophily before and after attacks on the Cora dataset, utilizing the GCN model, is presented in Fig. 11.

TABLE VIII
TIME (S) AND SPACE (MB) CONSUMPTION ANALYSIS

Attacks	GCN								SGC							
	Time				Space				Time				Space			
	Cora	Citeseer	Reddit	OGBNPro	Cora	Citeseer	Reddit	OGBNPro	Cora	Citeseer	Reddit	OGBNPro	Cora	Citeseer	Reddit	OGBNPro
G-NIA [12]	00153.49	00170.72	34107.45	04030.13	03122.65	07299.88	09314.78	05078.22	00122.97	00153.05	46969.08	10846.33	03122.37	07300.38	09306.38	05077.32
G^2A2C [14]	00202.64	00086.54	00175.65	00238.94	00168.59	00407.35	00200.81	00079.03	00176.30	00124.65	00241.57	00243.74	00167.51	00406.25	00207.31	00081.11
PGD [23]	00403.95	00404.96	00674.24	00562.39	00166.84	00436.17	00395.94	00253.11	00427.11	00430.97	00735.76	00527.13	00166.62	00435.46	00396.23	00253.11
SPEIT [18]	00437.11	00426.72	00780.20	00683.10	00171.05	00438.55	00421.02	00253.11	00461.31	00458.80	00826.76	00664.05	00171.02	00437.80	00419.37	00253.11
TDGIA [13]	00604.58	00603.47	00919.52	00797.16	00171.75	00450.01	00434.09	00253.11	00640.41	00769.32	01239.94	01189.56	00171.75	00449.29	00433.02	00253.11
GZOO (**ours**)	01345.38	01114.44	00238.49	00241.83	00140.80	00377.30	00238.17	00083.41	02124.17	02004.74	00221.84	00152.87	00151.66	00420.92	00242.91	00083.63

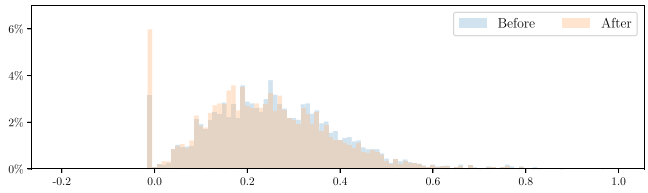


Fig. 11. The distribution of homophily before and after the GZOO attack on the Cora dataset.

Our analysis reveals that the distribution of node-centric homophily undergoes minimal changes after the attacks. The Jensen–Shannon divergence [53] between these two distributions is measured at 0.0065. This result demonstrates that the attribute loss in (21) effectively ensures that attributes of injected malicious nodes do not deviate significantly from the attributes of nodes within the original graph, contributing to the stealthiness of the attack.

VII. DISCUSSION

Time and Space Consumption: 1) The time and space complexities of the generator are analyzed as follows. For a K-order subgraph $\mathcal{G}^K(v_i)$ of the target node v_i, with n nodes and e edges, the time complexity for the backbone model is $\mathcal{O}(nd_1F + nn_ad_1d_2 + ed_1 + en_ad_2)$, while the space complexity is $\mathcal{O}(e + d_1F + nd_1 + n_ad_1d_2 + nn_ad_2)$. For the attribute generator $\mathcal{G}^a$, the time complexity is $\mathcal{O}(nn_ad_2 + nd_2d_3FM + n_aFM)$, and the space complexity is $\mathcal{O}(nn_ad_2 + n_aFM)$. The structure generator $\mathcal{G}^s$ has a time complexity of $\mathcal{O}(nn_an_s)$ and a space complexity of $\mathcal{O}(nn_an_s)$. 2) Zeroth-order optimization, involving B evaluations of the victim model, is computationally intensive. To address this, we batch the target nodes for attack and adjust the generator's parameters to produce malicious nodes for each batch. This method enables us to use the learned parameters to generate malicious nodes periodically, thereby misleading the victim model's predictions and removing successfully attacked nodes from the target list. 3) Evaluations on the Cora, Citeseer, Reddit, and OGBNPro datasets with GCN and SGC as victim models, detailed in Table VIII, illustrate GZOO's efficiency with continuous attributes. For datasets with discrete attributes, the use of the Gumbel-Softmax Trick to sample attributes according to generated probabilities increases optimization time.

Practical Threat: We present a realistic scenario in which companies operating social networks, such as Facebook and Twitter, use user preference prediction models to recommend advertisements and receive compensation from advertisers. Competitors could exploit these models by creating fake accounts through node injection attacks, potentially causing substantial financial losses for the companies.

VIII. CONCLUSION

In this paper, we have introduced GZOO, an effective and robust framework designed to execute NIAs in black-box settings. This framework employs an adversarial graph generator with the ability to consider the mutual information between attributes and structures when simultaneously crafting attributes and structures for malicious nodes. By bypassing the need for surrogate model training, we have developed an algorithm based on zeroth-order optimization to update the parameters of the generator iteratively. The thorough evaluations and experimental results provided clear evidence of GZOO's superiority over existing state-of-the-art attacks. Future research directions will revolve around enhancing GZOO's efficacy and exploring strategies to bolster its defenses against more potent NIAs.

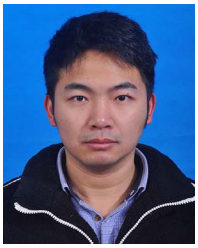
REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.
[2] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Feb. 2022.
[3] C. Liu, L. Sun, X. Ao, J. Feng, Q. He, and H. Yang, "Intention-aware heterogeneous graph attention networks for fraud transactions detection," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 3280–3288.
[4] W. Fan et al., "A graph neural network framework for social recommendations," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 5, pp. 2033–2047, May 2022.
[5] K. Liang et al., "A survey of knowledge graph reasoning on graph types: Static, dynamic, and multi-modal," *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 1–20, 2024.
[6] L. Sun et al., "Adversarial attack and defense on graph data: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 7693–7711, Aug. 2023.
[7] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 8317–8328.
[8] S. Geisler, T. Schmidt, H. Sirin, D. Zügner, A. Bojchevski, and S. Günnemann, "Robustness of graph neural networks at scale," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2021, pp. 7637–7649.
[9] J. Ma, S. Ding, and Q. Mei, "Towards more practical adversarial attacks on graph neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 4756–4766.
[10] J. Li, T. Xie, L. Chen, F. Xie, X. He, and Z. Zheng, "Adversarial attack on large scale graph," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 82–95, Jun. 2023.
[11] X. Wang, J. Eaton, C. Hsieh, and S. F. Wu, "Attack graph convolutional networks by adding fake nodes," 2018, *arXiv: 1810.10751*.
[12] S. Tao, Q. Cao, H. Shen, J. Huang, Y. Wu, and X. Cheng, "Single node injection attack against graph neural networks," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2021, pp. 1794–1803.

[13] X. Zou et al., "TDGIA: Effective injection attacks on graph neural networks," in *Proc. ACM Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 2461–2471.
[14] M. Ju, Y. Fan, C. Zhang, and Y. Ye, "Let graph be the go board: Gradient-free node injection attack for graph neural networks via reinforcement learning," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2023, pp. 4383–4390.
[15] H. Chang et al., "Adversarial attack framework on graph embedding models with limited knowledge," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 5, pp. 4499–4513, May 2023.
[16] K. Liang et al., "Knowledge graph contrastive learning based on relation-symmetrical structure," *IEEE Trans. Knowl. Data Eng.*, vol. 36, no. 1, pp. 226–238, Jan. 2024.
[17] K. Liang et al., "Mines: Message intercommunication for inductive relation reasoning over neighbor-enhanced subgraphs," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 10 645–10 653.
[18] Q. Zheng, Y. Fei, Y. Li, Q. Liu, M. Hu, and Q. Sun, "KDD CUP ML track 2 adversarial attacks and defense on academic graph 1st place solution," 2020. [Online]. Available: https://github.com/Stanislas0/KDD_CUP_2020_MLTrack2_SPEIT
[19] Q. Zheng et al., "Graph robustness benchmark: Benchmarking the adversarial robustness of graph machine learning," in *Proc. Int. Conf. Neural Inf. Process. Syst. Datasets Benchmarks*, 2021, pp. 1–14.
[20] L. Waikhom and R. Patgiri, "Graph neural networks: Methods, applications, and opportunities," 2021, *arXiv:2108.10733*.
[21] M. Liu et al., "Self-supervised temporal graph learning with temporal and structural intensity alignment," 2023, *arXiv:2302.07491*.
[22] M. Liu, Y. Liu, K. Liang, S. Wang, S. Zhou, and X. Liu, "Deep temporal graph clustering," 2023, *arXiv:2305.10738*.
[23] Y. Chen et al., "Understanding and improving graph injection attack by promoting unnoticeability," in *Proc. Int. Conf. Learn. Representations*, 2022, pp. 1–42.
[24] J. Fang, H. Wen, J. Wu, Q. Xuan, Z. Zheng, and C. K. Tse, "GANI: Global attacks on graph neural networks via imperceptible node injections," 2022, *arXiv:2210.12598*.
[25] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–23.
[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–14.
[27] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–12.
[28] Z. Tang et al., "Automatic sparse connectivity learning for neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7350–7364, Oct. 2023.
[29] P. Chen, H. Zhang, Y. Sharma, J. Yi, and C. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. ACM Workshop Artif. Intell. Secur.*, 2017, pp. 15–26.
[30] C. Tu et al., "Autozoom: Autoencoder-based zeroth order optimization method for attacking black-box neural networks," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2019, pp. 742–749.
[31] X. Gao, B. Jiang, and S. Zhang, "On the information-adaptive variants of the ADMM: An iteration complexity perspective," *J. Sci. Comput.*, vol. 76, no. 1, pp. 327–363, 2018.
[32] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, 2008.
[33] J. J. McAuley, C. Targett, Q. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Informat. Retrieval*, 2015, pp. 43–52.
[34] S. Ivanov, S. Sviridov, and E. Burnaev, "Understanding isomorphism bias in graph data sets," 2019, *arXiv: 1910.12091*.
[35] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[36] W. Hu et al., "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 22118–22133.
[37] P. Mernyei and C. Cangea, "Wiki-CS: A wikipedia-based benchmark for graph neural networks," 2020, *arXiv: 2007.02901*.
[38] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. K. Prasanna, "Graph-saint: Graph sampling based inductive learning method," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–19.
[39] K. Wang, Z. Shen, C. Huang, C. Wu, Y. Dong, and A. Kanakia, "Microsoft academic graph: When experts are not enough," *Quant. Sci. Stud.*, vol. 1, no. 1, pp. 396–413, 2020.
[40] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5425–5434.
[41] M. Wang et al., "Deep graph library: A graph-centric, highly-performant package for graph neural networks," 2019, *arXiv: 1909.01315*.
[42] F. Wu, A. H. S. Jr, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2019, pp. 6861–6871.
[43] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–15.
[44] M. He, Z. Wei, and J. Wen, "Convolutional neural networks on graphs with chebyshev approximation, revisited," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2022, pp. 7264–7276.
[45] Q. Wu et al., "Simplifying and empowering transformers for large-graph representations," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2023, pp. 64753–64773.
[46] B. Li, E. Pan, and Z. Kang, "PC-Conv: Unifying homophily and heterophily with two-fold filtering," in *Proc. Conf. Assoc. Advance. Artif. Intell.*, 2024, pp. 13 437–13 445.
[47] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–12.
[48] X. Zhang and M. Zitnik, "GNNGuard: Defending graph neural networks against adversarial attacks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2020, pp. 9263–9275.
[49] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2019, pp. 1399–1407.
[50] B. Zhou et al., "Robust graph representation learning for local corruption recovery," in *Proc. ACM Int. Conf. World Wide Web*, 2023, pp. 438–448.
[51] W. Jin, T. Zhao, J. Ding, Y. Liu, J. Tang, and N. Shah, "Empowering graph representation learning with test-time graph transformation," in *Proc. Int. Conf. Learn. Representations*, 2023, pp. 1–27.
[52] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. ACM Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 66–74.
[53] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Trans. Inf. Theory*, vol. 37, no. 1, pp. 145–151, Jan. 1991.

Hao Yu (Student Member, IEEE) received the BEng degree in computer science from Inner Mongolia University, Hohhot, China, in 2019, and the MASc degree in cyberspace science and technology from the Beijing Institute of Technology, Beijing, China, in 2022. He is currently working toward the PhD degree with the National University of Defense Technology (NUDT). His current research focuses on AI security and Federated Learning. He has authored several papers in top-level journals and conferences, such as *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Dependable and Secure Computing*, and *ACM Multimedia*, etc.

Ke Liang received the BSc degree from Beihang University (BUAA) and the MSc degree from the Pennsylvania State University (PSU). He is currently working toward the PhD degree with the National University of Defense Technology (NUDT). Before joining NUDT. His research interests include knowledge graphs, graph learning, and healthcare AI. He has published several papers in top-level journals and conferences, e.g., *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Neural Networks and Learning Systems*, SIGIR, AAAI, ICML, etc.



Dayu Hu received the BSc degree from Northeastern University (NEU). He is currently working toward the PhD degree with the National University of Defense Technology (NUDT). Before joining NUDT. His current research interests include graph learning and bioinformatics. He has published several papers and served as PC member/ Reviewer in highly regarded journals and conferences, such as *ACM Multimedia*, AAAI, TNNLS, TKDE, TCBB, etc.

Wenxuan Tu is currently working toward the PhD degree in College of Computer, National University of Defense Technology (NUDT), China. His research interests include unsupervised graph learning, deep graph clustering, and image semantic segmentation. He has published several papers in highly regarded journals and conferences, such as AAAI, ICML, MM, *IEEE Transactions on Image Processing*, *Information Sciences*, etc.

Chuan Ma (Member, IEEE) received the BS degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2013 and the PhD degree from the University of Sydney, Australia, in 2018. From 2018 to 2022, he worked as a lecturer in the Nanjing University of Science and Technology, and now he is a principal investigator with Zhejiang Lab. He has published more than 40 journal and conference papers, including a best paper, in WCNC 2018, and a best paper award in IEEE Signal Processing Society 2022. His research interests include stochastic geometry, wireless caching networks and distributed machine learning, and now focuses on the Big Data analysis and privacy-preserving.

Sihang Zhou (Member, IEEE) received the PhD degree from the School of Computer, National University of Defense Technology (NUDT), China. He is now lecturer with the College of Intelligence Science and Technology, NUDT. His current research interests include machine learning and medical image analysis. Dr. Zhou has published 40+ peer-reviewed papers, including *IEEE Transactions on Image Processing*, *IEEE Transactions on Neural Networks and Learning Systems*, *IEEE Transactions on Medical Imaging*, *Information Fusion, Medical Image Analysis*, AAAI, MICCAI, etc.

Xinwang Liu (Senior Member, IEEE) received the PhD degree from the National University of Defense Technology (NUDT), China. He is now professor of School of Computer, NUDT. His current research interests include kernel learning and unsupervised feature learning. Dr. Liu has published 100+ peer-reviewed papers, including those in highly regarded journals and conferences, such as *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transactions on Image Processing*, ICML, NeurIPS, ICCV, CVPR, etc. He serves as the associated editor of TNNLS, TCYB, and Information Fusion Journal.