Lanjun Wang* Tianjin University Tianjin, China wanglanjun@tju.edu.cn

Weizhi Nie Tianjin University Tianjin, China weizhinie@tju.edu.cn Xinran Qiao* Tianjin University Tianjin, China qiaoxinran@tju.edu.cn

Yongdong Zhang University of Science and Technology of China Hefei, China zhyd73@ustc.edu.cn Yanwei Xie Tianjin University Tianjin, China xie_yw@tju.edu.cn

Anan Liu[†] Tianjin University Tianjin, China liuanan@tju.edu.cn

ABSTRACT

Social platforms such as Twitter are under siege from a multitude of fraudulent users. In response, social bot detection tasks have been developed to identify such fake users. Due to the structure of social networks, the majority of methods are based on the graph neural network(GNN), which is susceptible to attacks. In this study, we propose a node injection-based adversarial attack method designed to deceive bot detection models. Notably, neither the target bot nor the newly injected bot can be detected when a new bot is added around the target bot. This attack operates in a black-box fashion, implying that any information related to the victim model remains unknown. To our knowledge, this is the first study exploring the resilience of bot detection through graph node injection. Furthermore, we develop an attribute recovery module to revert the injected node embedding from the graph embedding space back to the original feature space, enabling the adversary to manipulate node perturbation effectively. We conduct adversarial attacks on four commonly used GNN structures for bot detection on two widely used datasets: Cresci-2015 and TwiBot-22. The attack success rate is over 73% and the rate of newly injected nodes being detected as bots is below 13% on these two datasets.

CCS CONCEPTS

Computing methodologies → Machine learning algorithms;
 Security and privacy → Human and societal aspects of security and privacy.

KEYWORDS

Black-box Adversarial Attack, Bot Detection, Social Media

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada.

ACM Reference Format:

Lanjun Wang, Xinran Qiao, Yanwei Xie, Weizhi Nie, Yongdong Zhang, and Anan Liu. 2023. My Brother Helps Me: Node Injection Based Adversarial Attack on Social Bot Detection. In *Proceedings of the 31st ACM International Conference on Multimedia (MM '23), October 29–November 3, 2023, Ottawa, ON, Canada*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/ 3581783.3612396

1 INTRODUCTION

As social media becomes an increasingly integral part of people's daily lives, public opinion is now more heavily influenced by its content than ever before [7, 34]. With millions of daily active users, Twitter is a thriving social media platform with significant influence in shaping public opinion. However, as well as bringing benefits, social media also poses a major threat [37]. The platform's immense power has led to the proliferation of a new type of users, known as social bots. These bots can amplify certain discussions at the expense of others and manipulate public opinion to achieve their own goals. Examples of such manipulation include extreme propaganda [1], promotion of political conspiracies [16, 21], and interference in elections [10, 15].

To address the various problems caused by social bots, social bot detection tasks have been developed. Existing social bot detection methods are usually divided into three categories: feature-based methods, text-based methods, and graph-based methods. Featurebased methods utilize user information for feature engineering and apply traditional classification algorithms to detect bots [13, 24]. Text-based methods use natural language processing techniques to process user tweets and user description texts for bot detection [3, 23]. Graph-based methods interpret the Twitter social network as a graph and use the concepts of network science and geometric deep learning for bot detection [11, 29].

Recent studies [12, 14] have shown that graph-based methods achieve state-of-the-art performance in social bot detection. These methods can detect novel bots and overcome the various challenges associated with social bot detection. Graph-based methods typically employ Graph Neural Networks (GNNs) to detect bots by interpreting users as nodes and relationships as edges. Most GNNs follow a message passing scheme and achieve significant performance in many tasks [4, 30, 39] by iteratively aggregating representations of representation learning nodes from their neighbors. Despite their

^{*}Both authors contributed equally to this research. [†]Corresponding Author

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

^{© 2023} Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-0108-5/23/10...\$15.00 https://doi.org/10.1145/3581783.3612396

success, GNNs have been found to be highly vulnerable to adversarial attacks [2, 31, 38, 41, 43]. Graph-based social bot detection methods rely on GNNs for processing social networks, making them similarly vulnerable to adversarial attacks.

However, there is no attempt at social bot detection tasks because of the following difficulties. Firstly, most of the existing adversarial attack methods are white-box attacks [5, 17], which require the attacker to master the victim model in advance. On the contrary, the information of the victim model is the key asset of the companies who manage social platforms, and thus it is infeasible for the attacker to access it. Therefore, the practical adversarial attacks on social bot detection are in a black-box manner. Secondly, too much modification on the social network can be noticed to lead to the failure of the attack, so it is necessary to maintain the imperceptibility of the attack method, which requires not perturbing too much information about the original network. According to the studies on GNN adversarial attacks [9, 33, 35, 45], we choose to leverage the single node injection method [35] to control the change of the whole network on one node. Furthermore, since this newly injected node is also a fake user in the social network, there is a task-specific imperceptible requirement: the newly injected node cannot be detected by the victim model. This is a different constraint from the classical adversarial attacks on GNNs which control the number of the perturbed nodes [35] or the access graph range of the attackers [27]. Thirdly, most of the existing node injection adversarial attacks against GNNs are carried out in the intermediate embedding space [35, 42], which leads to the attack generating an injected node in the form of embedding. However, for our specific adversarial attack on social bot detection, since the attacker needs to generate a new bot and inject it into the original social network to achieve the undetectable of the target bot, the original attributes of the injected bot have to be restored.

In this study, we address these challenges of black-box settings, imperceptibility, and attribute recovery in the adversarial attack on social bot detection. First, we set up a simple GNN structure based on embeddings from the original attribute space as a substitute model. This setting relies on the transferability of the adversarial samples to achieve the black-box attack. Then, a single-node injection adversarial attack approach, G-NIA [35], is applied to address the imperceptibility of the attack from the perturbation aspect. Moreover, to maintain the imperceptibility of the attack from the attribute aspect, we collect statistics on the attributes of human users, then convert them as a series of constraints, and apply these constraints to the newly injected node. Finally, we design an attribute recovery module to obtain the original features of the injected node from the embedding space.

In summary, this study has the following contributions:

• In order to fool bot detection methods, we propose a blackbox node injection-based adversarial attack method, which is to add a new bot around a target bot and to achieve both the target bot and the newly injected bot undetectable by the bot detection method. To the best of our knowledge, this is the first study on the node injection-based bot detection attack.

- We design an attribute recovery module to restore the node feature from the graph embedding space to the original feature space in order to make the adding node perturbation operable by the adversary.
- We attack four existing bot detection methods on two datasets (Cresci-2015 and TwiBot-22) to evaluate the generalizability and effectiveness of the attack models. The attack success rate is over 73% and the rate of newly injected nodes being detected as bots is below 13% on two datasets. Specifically, the newly injected node detection rate on Cresci-2015 is as low as 0.06%.

2 RELATED WORK

In this section, we introduce the vulnerability of bot detection and the adversarial attack on GNNs.

2.1 Vulnerability of Bot Detection

Existing bot detection methods are vulnerable to attack. The error of the bot detection model can be caused either by creating the bot scenarios or by carrying out the adversarial attack.

Torusdağ et al. [36] investigate the vulnerability of existing social bot detection systems by creating their own bot scenarios instead of relying on public datasets. They experimentally show that the existing social bot detection model is unable to detect their social bots, thus demonstrating the vulnerability of these models. However, their study only serves to verify the vulnerability of the bot detection model and does not offer further discussion on potential solutions or improvements. Kantartopoulos et al. [20] obtain good results in adversarial attacks for bot detection by poisoning the training dataset. Their attack involved randomly modifying the labels of the bot nodes in the training set and copying new nodes based on the information of existing nodes. While this method is an adversarial attack for bot detection, it is not within the same scope as ours. This random attack method cannot specify nodes for attack and lacks flexibility. Additionally, this attack method directly uses the poisoned data to train the bot detection model, which can only be used to enhance the robustness of the bot detection model. In contrast, we propose an adversarial attack method based on node injection. Our method involves adding a new node and a new relationship to the original social network, which causes the bot detection model to generate classification errors for both the target bot node and the newly injected bot node. This approach greatly improves the accuracy and benefits of the attack.

2.2 Adversarial Attack on GNN

Extensive studies have shown that graph neural networks (GNNs) are vulnerable to various adversarial attacks [19, 28, 33]. These attacks can perturb node attributes, graph structures, and labels [6, 40]. For instance, Nettack [46] is a targeted attack that aims to deceive specific nodes by modifying their properties and the structure of the gradient bootstrap graph. Meta-attack [44] is a non-targeted attack based on meta-learning, but this method degrades the overall performance of GNNs. Additionally, G-NIA [35] is a targeted attack that adds only one new node and one new edge at a time, resulting in minimal disturbance to the original graph structure.

In this study, we use a similar approach to G-NIA to conduct an undetectable adversarial attack against the social bot detection model. G-NIA alone is insufficient to achieve our goal since this method can only generate the embeddings of injected nodes and requires reading the parameters of the victim model for a white-box attack. As we cannot access the specific information of the targeted bot detection model, we design a substitute model for a blackbox attack. Additionally, to realize the attack on social networks, we need to obtain the attributes of injected nodes, including user descriptions, tweets, numerical metadata, and categorical metadata. To do so, we further propose an attribute recovery module to obtain the attributes of injected nodes from the generated embeddings.

3 METHODOLOGY

In this section, we delve into the specifics of our proposed method. In Sec. 3.1, we provide a definition of the problem we aim to address. Subsequently, in Sec. 3.2, we present the overall framework of our methodology. Finally, in Secs. 3.3-3.6, we specify the four primary modules in our framework.

3.1 **Problem Definition**

3.1.1 Objective. Let G = (V, E, A) represents a social network, where $V = \{1, 2, ..., k\}$ constitutes the set of k users, $E \subseteq V \times V$ defines the relationships of the users, and A represents the attributes of users.

The goal of the social bot detection task is to achieve accurate prediction of user types in the social graph, i.e., $f(h|G) = y_h$ and $f(b|G) = y_b$, where $f(\cdot|G)$ denotes the detection outcome of the detection model with respect to the social network *G*, *h* and *b* represent human and bot respectively, y_h and y_b indicate the node type is human and bot respectively.

Based on the difficulties of the adversarial attack against the bot detection task mentioned in Sec. 1, we adopt the single-node injection method to carry out the black-box attack. The problem is defined as follows:

DEFINITION 1 (SINGLE-NODE ADVERSARIAL ATTACK ON SOCIAL BOT DETECTION). Given a social network G, and a target bot b_t which the attacker wants to evade the bot detection $f(\cdot)$, the singlenode injection adversarial attack on bot detection is to obtain a new social network graph G', where G' is constituted by G and one new bot $b_{inj} = (v_{inj}, e_{inj}, a_{inj})$, that is, $G' = (V \cup v_{inj}, E \cup e_{inj}, A \cup a_{inj})$, such that $f(b_t|G') = y_h$ and $f(b_{inj}|G') = y_h$.

Specifically, this definition illustrates three requirements of this adversarial task: 1) the new social network graph G' only has one new node (i.e., v_{inj} with its attribute a_{inj}) and one new edge (i.e., e_{inj}) to connect the node to the original graph G, without altering any existing nodes or edges in G. This aims to achieve imperceptible perturbation on the social graph. 2) The original attribute a_{inj} is required to be attained, which keeps the adversarial attack operable in practice. 3) Although the task is only to shield the target bot b_t , as the new injected node b_{inj} is also a man-made bot, b_{inj} is also required to be recognized as a human (i.e., $f(b_{inj}|G') = y_h$) to escape the bot detection.

3.1.2 Threat Model. As introduced in Sec. 1, this study focuses on the black-box attack to keep it practical. That is, the adversary

does not know the structure and weight information of the victim model because the victim model is the key asset of the companies who manage social networks. Furthermore, the adversary knows the target bot which they manage, as well as the corresponding social networks which are published and can be crawled from social media.

3.2 Framework

The overall framework of our method is shown in Figure 1, which has four major modules: (1)substitute model, (2)embedding generation, (3)edge generation, and (4)attribute recovery. In detail, as the adversarial attack is in a black-box manner, we train a Relational Graph Convolutional Network (R-GCN) [32] as the ①substitute model, and leverage the transferability of the adversarial samples to fool some latest bot detectors. More details of victim models are in Sec. 4. Then, the node embeddings extracted from the ①substitute model as well as the weights of the (1)substitute model are used to generate the embedding of the injected node in (2)embedding generation. By leveraging the embedding of the injected node as well as the information used in (2), (3)edge generation obtains the injected edge. As illustrated in the definition, it is not enough to have the embedding of the new node, but the original attributes are required. Thus, (4) attribute recovery restores the embedding of the injected node output by (2)embedding generation into the original attributes. Finally, the new node can be created by the attacker and injected into the social graph to shield the target bot.

3.3 Substitute Model

To launch an attack without knowing the specific information of the victim models, we devise a substitute model for transfer-based attacks.

For each user v in the social graph G, the attribute a includes four types, which are the description D, a set of tweets T, a series of numerical properties N, and a series of categorical properties C, that is a = (D, T, N, C), where $D = \{d_i\}_{i=1}^{L}$ represents a user's description with L words, $T = \{t_i\}_{i=1}^{M}$ signify a user's M tweets, $N = \{n_i\}_{i=1}^{P}$ denotes a user's numerical properties, and $C = \{c_i\}_{i=1}^{Q}$ indicate a user's categorical properties.

For edges between users, given the following and being followed information, there are two types of edges: "friend" and "follow" [8], that is $R = \{r_f, r_o\}$. Due to these different types of edges in the social network, the social network is a heterogeneous graph. We represent the friend and follow neighborhoods of a user v as $E_f(v)$ and $E_o(v)$, respectively, and all neighbors of v is represented as $E_r(v) = E_f(v) \cup E_o(v)$.

The substitute model consists of two parts: individual attribute encoding and structure-based feature transformation. We mainly use four fully-connected layers for individual attribute encoding and an R-GCN for structure-based feature transformation. Specifically, the fully-connected layer networks are used to encode the users' semantic information from their descriptions and tweets, as well as both user numerical and categorical property information, respectively, as illustrated in below (a)-(d). Meanwhile, the R-GCN is to obtain the overall user representation by considering the heterogeneous social graph, which is specified as (e). In addition, we set the dimension of the total user embedding *D*. As the user



Figure 1: Framework of Single-node Adversarial Attack on Social Bot Detection

embedding is concatenated by attribute embeddings as defined in Equation (3), the dimension of each attribute type is $\frac{D}{4}$, that is, the output dimension of the fully connected layer networks is $\frac{D}{4}$.

The detailed processes are as follows.

a) User Description. We utilize pre-trained RoBERTa [25] to encode user descriptions. Initially, we transform the words in the user description using RoBERTa:

$$\overline{d} = RoBERTa(\{d_i\}_{i=1}^L), \quad \overline{d} \in \mathbb{R}^{D_s \times 1}$$
(1)

where \overline{d} represents the user description's representation, and D_s corresponds to the embedding dimension of RoBERTa. Next, we extract representation vectors for the user's description:

$$x_d = \phi(W_D \cdot \overline{d} + b_D), \quad x_d \in \mathbb{R}^{\frac{D}{4} \times 1}$$
(2)

where W_D and b_D are learnable parameters, ϕ is the activation function.

b) User Tweets. We follow the same process as Equation (1) and Equation (2) on each tweet. Then, by averaging the representations of all tweets, we obtain the user tweet representation $x_t \in \mathbb{R}^{\frac{D}{4} \times 1}$.

c) User Numerical Properties. We utilize numerical features that can be directly accessed through the Twitter API, as shown in Table 1. After performing z-score normalization, we obtain the representation of user numerical features $x_n \in \mathbb{R}^{\frac{D}{4} \times 1}$ using a fully connected layer.

d) User Categorical Properties. Similar to user numerical properties, we make use of directly available user categorical features from the Twitter API, which are displayed in Table 2. By employing one-hot encoding, concatenating, and transforming them with a fully-connected layer, we derive the representation for the user's categorical features $x_c \in \mathbb{R}^{\frac{D}{4} \times 1}$.

e) Overall User Embedding. We encode the user description, tweets, numerical, and categorical properties, then concatenate them to

Table 1: User numerical properties

Feature Name	Description
followers	number of followers
active days	number of active days
screen name length	screen name character count
followings	number of followings
status	number of public lists where this user belongs

Table 2: User categorical properties

Feature Name	Description
protected verified	protected or not verified or not
default profile image	the default profile image

serve as the user embedding. For each user $i \in V$, we represent the user embedding as:

$$x_i = [x_{di}; x_{ti}; x_{ni}; x_{ci}] \in \mathbb{R}^{D \times 1}$$
(3)

D. ...

We employ an R-GCN [32] on the heterogeneous graph to learn user representations. The overall user embeddings in Equation (3) are used directly as the initially hidden vectors for nodes within the graph:

$$x_i^{(0)} = x_i, \quad x_i^{(0)} \in \mathbb{R}^{D \times 1}$$
 (4)

Then, we apply the *l*-th R-GCN layer:

$$x_{i}^{(l+1)} = \Theta_{self} \cdot x_{i}^{(l)} + \sum_{r \in R} \sum_{j \in E_{r}(i)} \frac{1}{|E_{r}(i)|} \Theta_{r} \cdot x_{j}^{(l)}$$
(5)

where Θ is the projection matrix, $l = \{0, ..., L - 1\}$, and L is the number of edge types. We directly use the output of the R-GCN as the prediction label:

$$\widehat{y}_i = x_i^{(L)} \tag{6}$$

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada.

Here, we jointly train the attribute encoding and overall feature transformation in the substitute model. The loss function of the substitute model is constructed as follows:

$$L_s = -\sum_{i \in V} [y_i log(\widehat{y}_i) + (1 - y_i) log(1 - (\widehat{y}_i)] + \lambda \sum_{w \in \theta_s} w^2 \quad (7)$$

where y_i is the ground-truth label and θ_s are all parameters of the substitute model.

3.4 Embedding Generation

We use the embeddings of the target node x_{b_t} and its first-order neighbors x_n , to guide the generation of the injected node, where x_n is the average embedding of all first-order neighbors of the target node b_t . Since feature transformation maps the embedding space to the label space as shown in Equations (4)-(6), we adopt the column of the feature transformation weights to represent the label class u_{b_t} . The process of constructing u_{b_t} is as follows. Since there are two types of edges (i.e., "friend" and "follow") in the graph, the weight of the R-GCN is divided into W_f and W_o and then aggregated with a fully-connected layer f_W , that is $W = f_W(W_f, W_o)$. Finally, $u_{b_t} =$ $[W[:, y_b]; W[:, y_h]; W]$, where $W[:, y_b]$ represents the column of W with respect to the label of before attacking (i.e. bot, y_b), and $W[:, y_h]$ represents the column of W with respect to the expected label of after attacking (i.e. human, y_h).

Using the representations described above, we utilize two fullyconnected layers \mathcal{F}^x and a Multi-Layer Perceptron (MLP) \mathcal{G}^x to generate the embedding of the injected node, denoted as x_{inj} :

$$\begin{aligned} x_{inj} &= \mathcal{G}^{x}(\mathcal{F}^{x}(x_{b_{t}}, x_{n}, u_{b_{t}}, G; \theta_{x}^{*})) \\ \mathcal{F}^{x}(x_{b_{t}}, x_{n}, u_{b_{t}}, G; \theta_{x}^{*}) &= \sigma([x_{n}; x_{b_{t}}; u_{b_{t}}]W_{0}^{x} + b_{0}^{x})W_{1}^{x} + b_{1}^{x} \end{aligned}$$
(8)

where $\theta_x^* = \{W_0^x, b_0^x, W_1^x, b_1^x\}$ are trainable weights. The mapping function \mathcal{G}^x maps the output of \mathcal{F}^x to the designated embedding space of the original graph, making the embeddings similar to existing nodes.

3.5 Edge Generation

The injected edges serve to spread the attributes of the injected node to the target nodes. The injected edge is limited to the target node and its first-order neighbors, meaning that the injected node must be at least a second-order neighbor of the target node. To capture the coupling effect between network structure and node features, we jointly model the injected embeddings and edges.

Specifically, we use the injected embeddings to guide the generation of injected edges. To guide the generation of the injected edge e_{inj} , we include the generated injected embedding x_{inj} along with the information of the target bot and its neighbors as used in Equation (8). We employ two fully-connected layers \mathcal{F}^e and an MLP \mathcal{G}^e to generate e_{inj} as follows:

$$e_{inj} = \mathcal{G}^{e}(\mathcal{F}^{e}(x_{inj}, x_{b_{t}}, x_{n}, u_{b_{t}}, G; \theta_{e}^{*}))$$

$$\mathcal{F}^{e}(x_{inj}, x_{b_{t}}, x_{n}, u_{b_{t}}, G; \theta_{e}^{*}) = \sigma([x_{b_{t}}; x_{n}; x_{inj}; u_{b_{t}}]W_{0}^{e} + b_{0}^{e})W_{1}^{e} + b_{1}^{e}$$

(9)

where $\theta_e^* = \{W_0^e, b_0^e, W_1^e, b_1^e\}$ are trainable weights.

Once the injected embedding and edge are generated, we inject the injected node into the original graph G to obtain the perturbed graph G'. We then feed G' into the substitute model and compute the attack loss as follows:

$$L_{atk} = \sum_{b_t \in V_b} (S'_{b_t, y_b} - S'_{b_t, y_h})$$
(10)

where $S'_{b_t,y}$ denotes the predicted label probability of the substitute model in the new social network graph G' with respect to the target bot b_t on the corresponding label (i.e. y_h and y_b). Here we conduct joint training on the embedding generation module and edge generation module. The optimization process aims to minimize the attack loss L_{atk} , which guides the training process. We employ gradient descent to iteratively optimize L_{atk} until convergence.

3.6 Attribute Recovery

The social network bot detection model detects users by extracting features from the raw user data. However, the injected node embeddings generated in Sec. 3.4 represent the total user features that have been extracted by the feature extractor in the substitute model, which is not consistent with the bot detection model whose input is the original user features. To address this, we propose a module named attribute recovery to recover the generated injected node embeddings.

3.6.1 User Description & User Tweets. It is challenging to recover text features, so we directly set the text features for the user description and user tweets to 0 for the injected node. This means that the injected node does not contain any text information.

3.6.2 User Numerical Properties. To recover the numerical features of the injected node, we train a multi-layer perceptron (MLP) using the preprocessed numerical data $N = \{n_i\}_{i=1}^{P}$ and the numerical embeddings x_n extracted by the feature extractor of the substitute model. First, we calculate the loss per user at the original feature level:

$$l_{n1} = \sum_{i=1}^{P} |MLP_n(x_{n_i}) - n_i|$$
(11)

Next, we also constrain the effects of the numerical MLP at the feature level:

$$l_{n2} = \sum_{i=1}^{P} |\phi(W_N \cdot MLP_n(x_{n_i}) + b_N) - x_{n_i}|$$
(12)

Finally, we add the losses for each user *j* as a loss function of the numerical MLP:

$$L_n = \sum_{j \in V} (l_{n1} + \alpha l_{n2}) \tag{13}$$

where α is a constant, and here we set $\alpha = 0.01$.

To ensure that the recovered numerical features are realistic, we reverse the z-score normalization on the recovered features to obtain the original integer numerical features. Then, we constrain the numerical features to their respective appropriate ranges and perform z-score normalization again to obtain the final user numerical properties. The constraints are carefully designed based on the statistics of the corresponding dataset where the node is to be injected, in order to 1) achieve successful attacks to shield the target bot, 2) make the injected node imperceptible, as well as 3) the injection process operable by the attacker. Detailed setting criteria can be referred to in Sec. 4.1.2.

3.6.3 User Categorical Properties. Similar to the attribute recovery of the user numerical properties, we train another MLP using the data $C = \{c_i\}_{i=1}^{Q}$ obtained after data preprocessing and the categorical embeddings x_c processed by the feature extractor of the substitute model. For the categorical MLP, we calculate each user's losses at the original feature level only, similar to Equation (11), and then sum all users' losses as the loss function of categorical property recovery. We recover the user categorical properties to the one-hot encoding by adding a constraint during the attribute recovery process.

3.6.4 Edge Type. After generating the injected node with the injected embedding and recovering its attributes, we need to connect it to the target node in the subgraph. Since the injected node is a bot, we set the injected edge as "follow" to make it easier to establish.

4 EXPERIMENT

In this section, we conduct experiments to show the efficacy of our method. In Sec. 4.1, we introduce the experiment settings. Then, Sec. 4.2 shows the overall performance of our method. On this basis, we conduct ablation experiments, parameter analysis, and transferability analysis in Sec. 4.3, Sec. 4.4, and Sec. 4.5, respectively.

4.1 Experiment Settings

4.1.1 Datasets. We conduct experiments to evaluate the effectiveness of our method on two datasets: Cresci-2015 [8] and TwiBot-22 [12]. These datasets comprise heterogeneous graph data from social networks, and their statistics are shown in Table 3.

Table 3: Statistics of the datasets

Dataset	Human	Bot	Tweet	Edge
Cresci-2015	1,950	3,351	2,827,757	7,086,134
TwiBot-22	860,057	139,943	86,764,167	170,185,937

Due to NVIDIA RTX 3090Ti GPU memory limitations, we are unable to convert the complete social network graph provided by the TwiBot-22 dataset into a sparse matrix for quickly obtaining the first-order neighbors of the target as required in Equation 8 and Equation 9. Therefore, We use a community selection algorithm to select about 50,000 nodes to form five subgraphs for experiments. The statistics of the subgraphs are shown in Table 4.

Table 4: Statistics of the five subgraphs selected on TwiBot-22

Graph	Human	Bot	Edge
subgraph 1	45,081	4,491	993,930
subgraph 2	44,864	4,276	950,365
subgraph 3	46,079	4,278	953,733
subgraph 4	46,267	4,542	963,507
subgraph 5	45,553	4,496	983,006
whole graph	860,057	139,943	170,185,937

4.1.2 Constraint Settings. To make the injected node more realistic, we set constraints to adjust its attributes as introduced in Sec. 3.6. The constraints for injected nodes in the two datasets are shown in Table 5 and Table 6.

In order to make the user numerical properties of the injected node realistic and easy to implement, we apply the following constraintsetting criteria.

- The number of followers is set to 0, which indicates that the injected node does not need to be noticed by any others. That is to say, this indicates that the attacker does not need to buy any fan for this injected node.
- The maximum number of active days is set to 100, which indicates that the attacker does not need too much time to prepare for the shield. Noted this number is far lower than the average active days in the social networks.
- The screen name length and the number of followings to the maximum allowed by Twitter, as these are easily attainable attributes.
- The numbers of statuses are on the same orders of magnitudes as most users in the corresponding original social networks.

Table 5: Mean and standard deviation of the numerical fea-tures as well as the constraints on Cresci-2015

Feature Name	Mean	Std	Constraints
followers count	246	5,879	0
active days	3,201	461	100
screen name length	11	3	15
following count	386	561	5,000
status	3	25	500

Table 6: Mean and standard deviation of the numerical features as well as the constraints on TwiBot-22

Feature Name	Mean	Std	Constraints
followers count	41,230	602,078	0
active days	2,128	1,633	100
screen name length	14	7	15
following count	2,251	15,782	5,000
status	21,689	120,247	40,000

4.1.3 Victim Models. We evaluate the effectiveness of our method by attacking four GNNs that are used for bot detection, which are benchmarks given by the Twibot-22 [12] dataset. The four methods use Graph Convolutional Network (GCN) [22], Heterogeneous Graph Transformer (HGT) [18], Simple Heterogeneous Graph Neural Network (Simple-HGN) [26], and Relational Graph Convolutional Network (R-GCN) [32] respectively for social bot detection. These four methods leverage text information from user descriptions and tweets, user numerical and categorical property information, as well as heterogeneous graphs based on user relationships to learn user representations for bot detection tasks on

Twitter. Among them, the method based on R-GCN [14] is different from our substitute model. It jointly encodes the various original attributes of each user and inputs them to two R-GCNs. Then, it uses a fully-connected layer to get the prediction label. Meanwhile, as introduced in Sec. 3.3, we encode the four types of original attributes respectively, splice them together, and then input them to a network with one R-GCN to predict results directly.

4.1.4 *Evaluation Metrics.* To demonstrate the effectiveness of our method, we set two evaluation metrics as follows:

Attack success rate. We measure the success of the attack by evaluating whether the victim model can detect the target bot to the new social network graph G'. A successful attack occurs when the target bot node is classified as a human. That is to say, the higher the attack success rate, the better effectiveness of the attack.

New node detected as bot. We evaluate the imperceptibility of our injected node by measuring whether the victim model can detect it as a bot. A successful attack occurs when the injected node is classified as a human. That is to say, the lower rate of the new node detected as a bot, the better imperceptibility of the attack.

4.1.5 Implementation Details. When training the substitute model, the learning rate is set as 1e-2, and the epoch is set as 150. In the training for embedding generation and edge generation, we set a maximum of 500 epochs. If the misclassification rate of the validation set does not increase for 5 consecutive epochs, the training is stopped. Here, we set the batch size to 32 and the learning rate to 1e-5. For the attribute recovery, we set the initial learning rate as 1e-2, and the learning rate drops to 1e-5 with training. For the numerical property recovery loss in Equation 13, we set $\alpha = 0.01$. Each experiment is repeated five times to ensure reliability.

4.2 Overall Performance

We evaluate the effectiveness of our method on two datasets, Cresci-2015 and TwiBot-22, and the results are presented in Table 7 and Table 8, respectively.

Table 7: Overall	performance on	Cresci-2015 ((%)
------------------	----------------	---------------	-----

Method	attack success rate	new node become bot
GCN	95.68 ± 1.44	0.00 ± 0.00
HGT	94.79 ± 1.18	0.06 ± 0.12
Simple-HGN	95.74 ± 1.25	0.00 ± 0.00
R-GCN	95.74 ± 1.50	0.06 ± 0.12

Table 8: Overall performance on TwiBot-22 (%)

Method	attack success rate	new node become bot
GCN	93.97 ± 5.43	2.66 ± 5.09
HGT	89.37 ± 3.56	5.40 ± 10.80
Simple-HGN	74.94 ± 2.16	7.39 ± 14.78
R-GCN	73.73 ± 1.71	12.94 ± 19.19

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada.

Our method successfully achieves significant attack results on both datasets. However, it is noteworthy that the attack results on Cresci-2015 are better than those on TwiBot-22. This can be attributed to the increasing complexity of social networks and the advancements in camouflage technologies employed by bot users.

4.3 Ablation Study

To demonstrate the validity of our adversarial attack method, we conduct ablation experiments on our method. Since the substitute model and attribute recovery module are essential for data conversion, they cannot be removed. Therefore, we perform ablation on the embedding generation and edge generation modules. We choose the TwiBot-22 dataset, which is more representative of the current social network environment.

In the ablation experiment for embedding generation, we directly assign the embedding of the target node to the injected node to explore the importance of the embedding generation module in adversarial attacks. The experimental results are shown in Table 9. It is obvious that the attack effect decreases after removing the embedding generation module. This shows that the embedding generation module plays a vital role in both the attack success rate and the probability of the new node becoming a bot.

Table 9: Ablation study for embedding generation on TwiBot-22 (%)

Mathad	attack success rate		new node become bot		
Method	ours	assign	ours	assign	
GCN	93.97 ± 5.43	85.99 ± 2.75	2.66 ± 5.09	39.24 ± 1.32	
HGT	89.37 ± 3.56	84.36 ± 2.66	5.40 ± 10.80	26.86 ± 1.69	
Simple-HGN	74.94 ± 2.16	74.10 ± 1.19	7.39 ± 14.78	37.54 ± 1.15	
R-GCN	73.73 ± 1.71	74.91 ± 2.17	12.94 ± 19.19	35.29 ± 1.82	

In the ablation experiment for edge generation, we randomly select a node in the first-order subgraph of the target node to connect with the injected node. The experimental results are shown in Table 10. Overall, our approach achieves better performance, but the improvement is modest. This may be due to the failure to select all first-order nodes around the target bot during dataset construction and subgraph selection, resulting in a limited selection range of edge generation. We consider that in practical application if all the first-order neighbors can be selected to construct the subgraph, our method can achieve better results.

Table 10: Ablation study for edge generation on TwiBot-22(%)

Mathad	attack success rate		new node become bot		
Methou	ours	random	ours	random	
GCN	94.51 ± 1.57	93.80 ± 2.19	2.66 ± 5.09	2.17 ± 4.12	
HGT	89.37 ± 3.56	86.48 ± 1.02	5.40 ± 10.80	4.34 ± 8.67	
Simple-HGN	74.94 ± 2.16	75.52 ± 1.15	7.39 ± 14.78	6.37 ± 12.74	
R-GCN	73.73 ± 1.71	74.39 ± 1.46	12.94 ± 19.19	8.37 ± 10.83	

4.4 Parameter Analyse

4.4.1 Substitute model. In this experiment, we investigate the impact of the substitute model on the performance of the attack. Besides R-GCN used in the overall performance as shown in Sec. 3.3,

Mothed attack success rate		new node become bot				
Method	1	2	3	1	2	3
GCN	95.68 ± 1.44	95.86 ± 1.66	95.62 ± 1.60	0.00 ± 0.00	0.00 ± 0.00	0.12 ± 0.15
HGT	94.79 ± 1.18	87.40 ± 10.19	95.66 ± 2.19	0.06 ± 0.12	28.23 ± 37.35	13.55 ± 12.16
Simple-HGN	95.74 ± 1.25	95.80 ± 1.74	94.57 ± 1.59	0.00 ± 0.00	28.64 ± 38.83	1.66 ± 1.41
R-GCN	95.74 ± 1.50	95.62 ± 1.39	95.62 ± 1.59	0.06 ± 0.12	3.37 ± 6.74	1.66 ± 1.07

Table 11: The number of R-GCNs in the substitute model on Cresci-2015 (%)

Table 12: The number of R-GCNs in the substitute model on TwiBot-22 (%)

Method	attack success rate			new node become bot		
	1	2	3	1	2	3
GCN	93.97 ± 5.43	95.08 ± 2.79	95.60 ± 1.50	2.66 ± 5.09	4.82 ± 7.27	0.40 ± 0.49
HGT	89.37 ± 3.56	85.67 ± 6.46	86.16 ± 5.76	5.40 ± 10.80	3.04 ± 4.60	0.04 ± 0.09
Simple-HGN	74.94 ± 2.16	72.07 ± 2.97	72.76 ± 5.05	7.39 ± 14.78	1.25 ± 1.63	0.00 ± 0.00
R-GCN	73.73 ± 1.71	79.40 ± 5.05	78.83 ± 1.42	12.94 ± 19.19	3.20 ± 2.74	$\textbf{0.04} \pm \textbf{0.09}$

we implement the framework based on a GCN substitute model. We use a GCN layer to directly replace the R-GCN in the substitute model. Since GCN can only handle homogeneous graphs, we do not add edge-type information during input. The outputs of GCN are consistent with the output of R-GCN, which are predictive labels.

Table 13: GNN types in substitute model on Cresci-2015 (%)

Method	attack success rate		new node become bot	
	R-GCN	GCN	R-GCN	GCN
GCN	95.68 ± 1.44	96.51 ± 1.42	0.00 ± 0.00	0.00 ± 0.00
HGT	94.79 ± 1.18	94.02 ± 0.51	0.06 ± 0.12	8.46 ± 13.29
Simple-HGN	95.74 ± 1.25	96.75 ± 1.24	0.00 ± 0.00	11.12 ± 22.25
R-GCN	95.74 ± 1.50	96.16 ± 0.82	0.06 ± 0.12	11.54 ± 23.08

Table 14: GNN types in substitute model on TwiBot-22 (%)

Method	attack success rate		new node become bot	
	R-GCN	GCN	R-GCN	GCN
GCN	93.97 ± 5.43	80.22 ± 10.87	2.66 ± 5.09	66.79 ± 36.76
HGT	89.37 ± 3.56	84.13 ± 5.22	5.40 ± 10.80	75.91 ± 32.64
Simple-HGN	74.94 ± 2.16	76.89 ± 1.96	7.39 ± 14.78	79.74 ± 32.17
R-GCN	73.73 ± 1.71	74.61 ± 2.98	12.94 ± 19.19	71.93 ± 35.65

As shown in Table 13 and Table 14, we can observe that the substitute model has a significant impact on the adversarial attack effect. Specifically, R-GCN achieves the highest overall attack success rate, and the probability of the injected node being detected as a bot is lower, especially on the TwiBot-22 dataset. This may be because R-GCN incorporates edge information when processing information, resulting in the weight of R-GCN containing more informative features than GCN.

4.4.2 Number of *R*-GCNs in the Substitute Model. In this experiment, we investigate the effect of the number of R-GCNs in the substitute model on the performance of the adversarial attack method, and the results are presented in Table 11 and Table 12.

The experimental results show that the number of R-GCNs in the substitute model has little effect on the adversarial attack effect. Therefore, we choose only one R-GCN as the structure of feature transformation in the substitute model, which requires the least amount of computation while carrying out the effective attack.

4.5 Transferability Analyse

Sec. 4.2 demonstrates the transferability between models because we train on the substitute model and test on victim models. In this experiment, we investigate the transferability between data, namely whether the attack model can be trained on a subset of social networks and used to attack nodes on the complete social network. Since Twibot-22 has been divided into 5 subgraphs as shown in Table 4, we train the attack model on one subgraph to attack bot nodes on others. The results are shown in Table 15.

Table 15: Transferability analysis on TwiBot-22 (%)

Method	attack success rate		new node become bot	
	same	others	same	others
GCN	93.97 ± 5.43	94.03 ± 3.49	2.66 ± 5.09	7.72 ± 20.99
HGT	89.37 ± 3.56	88.29 ± 5.19	5.40 ± 10.80	8.27 ± 24.09
Simple-HGN	74.94 ± 2.16	74.92 ± 1.88	7.39 ± 14.78	8.44 ± 24.06
R-GCN	73.73 ± 1.71	74.09 ± 2.19	12.94 ± 19.19	16.65 ± 20.46

The experimental results demonstrate that our adversarial attack method exhibits good transferability. By training on one social network, it can achieve good attack performance on different social networks. This shows that our model is general and not limited to a specific social network.

5 CONCLUSION

In this study, we propose an adversarial attack framework to hide bot users in social networks. We employ the single-node injection method to conduct a black-box attack, and subsequently perform attribute recovery on the injected node embedding. Our attack makes neither the target bot node nor the injected bot node detected as a bot by the victim model. Experimental results on the Cresci-2015 and TwiBot-22 datasets demonstrate the effectiveness of our method in achieving a high attack success rate and making the injected node undetectable.

ACKNOWLEDGMENTS

This work is supported in part by the National Natural Science Foundation of China (U21B2024, 62202329).

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada.

REFERENCES

- Jonathon M Berger and Jonathon Morgan. 2015. The ISIS Twitter Census: Defining and describing the population of ISIS supporters on Twitter. (2015).
- [2] Aleksandar Bojchevski and Stephan Günnemann. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In ICML, Vol. 97. 695–704.
- [3] Chiyu Cai, Linjing Li, and Daniel Zeng. 2017. Detecting Social Bots by Jointly Modeling Deep Behavior and Content Information. In CIKM. 1995–1998.
- [4] Qi Cao, Huawei Shen, Jinhua Gao, Bingzheng Wei, and Xueqi Cheng. 2020. Popularity Prediction on Social Platforms with Coupled Graph Neural Networks. In WSDM. 70–78.
- [5] Giuseppe Castiglione, Gavin Ding, Masoud Hashemi, Christopher Srinivasa, and Ga Wu. 2022. Scalable Whitebox Attacks on Tree-based Models. CoRR (2022).
- [6] Liang Chen, Jintang Li, Jiaying Peng, Tao Xie, Zengxu Cao, Kun Xu, Xiangnan He, and Zibin Zheng. 2020. A Survey of Adversarial Learning on Graphs. *CoRR* (2020).
- [7] Weilong Chen, Chenghao Huang, Weimin Yuan, Xiaolu Chen, Wenhao Hu, Xinran Zhang, and Yanru Zhang. 2022. Title-and-Tag Contrastive Vision-and-Language Transformer for Social Media Popularity Prediction. In ACM MM. 7008–7012.
- [8] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. 2015. Fame for sale: Efficient detection of fake Twitter followers. Decision Support Systems 80 (2015), 56–71.
- [9] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial Attack on Graph Structured Data. In *ICML*, Vol. 80. 1123–1132.
- [10] Ashok Deb, Luca Luceri, Adam Badawy, and Emilio Ferrara. 2019. Perils and Challenges of Social Media and Election Manipulation Analysis: The 2018 US Midterms. In WWW. 237–247.
- [11] Ashkan Dehghan, Kinga Siuta, Agata Skorupka, Akshat Dubey, Andrei Betlen, David Miller, Wei Xu, Bogumil Kaminski, and Pawel Pralat. 2022. Detecting Bots in Social-networks using Node and Structural Embeddings. In DATA. 50–61.
- [12] Shangbin Feng, Zhaoxuan Tan, Herun Wan, Ningnan Wang, Zilong Chen, Binchi Zhang, Qinghua Zheng, Wenqian Zhang, Zhenyu Lei, Shujie Yang, Xinshun Feng, Qingyue Zhang, Hongrui Wang, Yuhan Liu, Yuyang Bai, Heng Wang, Zijian Cai, Yanbo Wang, Lijing Zheng, Zihan Ma, Jundong Li, and Minnan Luo. 2022. TwiBot-22: Towards Graph-Based Twitter Bot Detection. CoRR (2022).
- [13] Shangbin Feng, Herun Wan, Ningnan Wang, Jundong Li, and Minnan Luo. 2021. SATAR: A Self-supervised Approach to Twitter Account Representation Learning and its Application in Bot Detection. In *CIKM*. 3808–3817.
- [14] Shangbin Feng, Herun Wan, Ningnan Wang, and Minnan Luo. 2021. BotRGCN: Twitter bot detection with relational graph convolutional networks. In ASONAM. 236–239.
- [15] Emilio Ferrara. 2017. Disinformation and social bot operations in the run up to the 2017 French presidential election. *First Monday* 22, 8 (2017).
- [16] Emilio Ferrara. 2020. What types of COVID-19 conspiracies are populated by Twitter bots? First Monday 25, 6 (2020).
- [17] Chao Hu, Ruishi Yu, Binqi Zeng, Yu Zhan, Ying Fu, Quan Zhang, Rongkai Liu, and Heyuan Shi. 2023. HyperAttack: Multi-Gradient-Guided White-box Adversarial Structure Attack of Hypergraph Neural Networks. *CoRR* (2023).
- [18] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous Graph Transformer. In WWW. 2704–2710.
- [19] Jinyuan Jia, Binghui Wang, Xiaoyu Cao, and Neil Zhenqiang Gong. 2020. Certified Robustness of Community Detection against Adversarial Structural Perturbation via Randomized Smoothing. In WWW. 2718–2724.
- [20] Panagiotis Kantartopoulos, Nikolaos Pitropakis, Alexios Mylonas, and Nicolas Kylilis. 2020. Exploring adversarial attacks and defences for fake twitter account detection. *Technologies* 8, 4 (2020), 64.
- [21] Franziska B Keller, David Schoch, Sebastian Stier, and JungHwan Yang. 2020. Political astroturfing on Twitter: How to coordinate a disinformation campaign. *Political communication* 37, 2 (2020), 256–280.
- [22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.
- [23] Jürgen Knauth. 2019. Language-Agnostic Twitter-Bot Detection. In RANLP. 550–558.
- [24] Sneha Kudugunta and Emilio Ferrara. 2018. Deep neural networks for bot detection. *Information Sciences* 467 (2018), 312–322.
- [25] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. CoRR (2019).
- [26] Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jianguo Jiang, Yuxiao Dong, and Jie Tang. 2021. Are we really making much progress?: Revisiting, benchmarking and refining heterogeneous graph neural networks. In KDD. 1150–1160.
- [27] Jiaqi Ma, Shuangrui Ding, and Qiaozhu Mei. 2020. Towards More Practical Adversarial Attacks on Graph Neural Networks. In *NeurIPS*.
- [28] Yao Ma, Suhang Wang, Tyler Derr, Lingfei Wu, and Jiliang Tang. 2021. Graph Adversarial Attack via Rewiring. In KDD. 1161–1169.
- [29] Phu Pham, Loan TT Nguyen, Bay Vo, and Unil Yun. 2022. Bot2Vec: A general approach of intra-community oriented representation learning for bot detection in different types of social networks. *Information Systems* 103 (2022), 101771.

- [30] Bastian Rieck, Christian Bock, and Karsten M. Borgwardt. 2019. A Persistent Weisfeiler-Lehman Procedure for Graph Classification. In *ICML*, Vol. 97. 5448– 5458.
- [31] Jitao Sang, Xian Zhao, Jiaming Zhang, and Zhiyu Lin. 2022. Benign Adversarial Attack: Tricking Models for Goodness. In ACM MM. 6883–6889.
- [32] Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In ESWC, Vol. 10843. 593–607.
- [33] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant G. Honavar. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In WWW. 673–683.
- [34] Yunpeng Tan, Fangyu Liu, Bowei Li, Zheng Zhang, and Bo Zhang. 2022. An Efficient Multi-View Multimodal Data Processing Framework for Social Media Popularity Prediction. In ACM MM. 7200–7204.
- [35] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single Node Injection Attack against Graph Neural Networks. In CIKM. 1794–1803.
- [36] M Buğra Torusdağ, Mucahid Kutlu, and Ali Aydın Selçuk. 2020. Are we secure from bots? Investigating vulnerabilities of botometer. In UBMK. 343–348.
- [37] Christian von der Weth, Ashraf M. Abdul, Shaojing Fan, and Mohan S. Kankanhalli. 2020. Helping Users Tackle Algorithmic Threats on Social Media: A Multimedia Research Agenda. In ACM MM. 4425–4434.
- [38] Lina Wang, Kang Yang, Wenqi Wang, Run Wang, and Aoshuang Ye. 2020. MGAAttack: Toward More Query-efficient Black-box Attack by Microbial Genetic Algorithm. In ACM MM. 2229–2236.
- [39] Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. 2019. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning. In IJCAI. 1928–1934.
- [40] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. 2020. Adversarial Attacks and Defenses in Images, Graphs and Text: A Review. Int. J. Autom. Comput. 17, 2 (2020), 151–178.
- [41] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *IJCAI*. 3961–3967.
- [42] Qiuling Xu, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. 2021. Towards Feature Space Adversarial Attack by Style Perturbation. In AAAI. 10523–10531.
- [43] Xing Xu, Jiefu Chen, Jinhui Xiao, Zheng Wang, Yang Yang, and Heng Tao Shen. 2020. Learning Optimization-based Adversarial Perturbations for Attacking Sequential Recognition Models. In ACM MM. 2802–2822.
- [44] Xiao Yang, Yinpeng Dong, Wenzhao Xiang, Tianyu Pang, Hang Su, and Jun Zhu. 2021. Model-Agnostic Meta-Attack: Towards Reliable Evaluation of Adversarial Robustness. *CoRR* (2021).
- [45] Zheng Yuan, Jie Zhang, Yunpei Jia, Chuanqi Tan, Tao Xue, and Shiguang Shan. 2021. Meta Gradient Adversarial Attack. In *ICCV*. 7728–7737.
- [46] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2019. Adversarial Attacks on Neural Networks for Graph Data. In IJCAI. 6246–6250.

A ALTERNATIVE CONSTRAINTS

In Sec. 4.1.2, we set the followers count to zero, which is to minimize the costs associated with buying followers, making the attack easier to carry out. However, it should be noted that setting the followers count to zero does not mean that it must be zero. According to [13] showing that the number of followers is positively correlated with the probability that a node is detected as human, if the injected node can successfully attack without followers, then it can successfully attack with any number of followers. Thus, we relax the restriction on zero follower count and conduct more experiments. In the following experiments, we set the follower count as 600 and reduce the status to 18,000 for TwiBot-22. The rest of the settings are the same as Table 6.

Table 16: Analysis of followers count on TwiBot-22 (%)

Method	attack success rate	new node become bot
GCN	93.27 ± 1.38	10.31 ± 11.28
HGT	89.00 ± 4.00	5.62 ± 11.24
Simple-HGN	74.37 ± 2.70	18.14 ± 30.44
R-GCN	76.53 ± 2.40	3.34 ± 6.05

MM '23, October 29-November 3, 2023, Ottawa, ON, Canada.

As shown in Table 16, increasing the followers count still keeps the success rate of the attacks high. At the same time, increasing the followers count can significantly reduce the status requirements of the injection nodes. This shows that attackers can set the constraints on their own according to the actual situation.