Highly Imperceptible Black-Box Graph Injection Attacks with Reinforcement Learning

Maochang Zhao, Jing Zhang*

School of Cyber Science and Engineering, Southeast University, Nanjing, China
{mczhao, jingz}@seu.edu.cn

Abstract

Recent studies have revealed the vulnerability of graph neural networks (GNNs) to adversarial attacks. In practice, effectively attacking GNNs is not easy. Existing attack methods primarily focus on modifying the topology of the graph data. In many scenarios, attackers do not have the authority to manipulate the graph's topology, making such attacks challenging to execute. Although node injection attacks are more feasible than modifying the topology, current injection attacks rely on knowledge of the victim model's architecture. This dependency significantly degrades attack quality when there is inconsistency in the victim models. Moreover, the generation of injected nodes often lacks precise control over features, making it difficult to balance attack effectiveness and stealthiness. In this paper, we investigate a node injection attack under model-agnostic conditions and propose **T**argeted **E**vasion **A**ttack via **N**ode **I**njection (TEANI). Specifically, TEANI models the generation of adversarial nodes as a Markov process. Without considering the target model's structure, it guides the agent to select features that maximize attack effectiveness within a budget, based solely on the results of queries to a black-box model. Extensive experiments on real-world datasets and mainstream GNN models demonstrate that the proposed TEANI poses more effective and imperceptible threats than state-of-the-art attack methods.

Introduction

Graph Neural Networks (GNNs) (Kipf and Welling 2017; Veličković et al. 2018), as powerful tools for processing graph-structured data, have achieved significant success in many fields in recent years. Whether it is social network analysis (Pei, Chakraborty, and Sycara 2015), recommendation systems (Wang et al. 2019), or bioinformatics (yang zhang et al. 2024), GNNs have demonstrated excellent performance and broad application prospects. GNNs capture the relationships between nodes through a message-passing mechanism, enabling them to excel in various graph tasks. However, this also makes GNNs more susceptible to adversarial attacks. Recent studies (Zügner, Akbarnejad, and Günnemann 2018; Dai et al. 2018; Xu et al. 2019) have shown that by manipulating features, edges, and injecting nodes into the graph, attackers can effectively disrupt the classification performance of GNNs.

Early work primarily focused on misleading the victim model by modifying the structure of the graph, such as adding or deleting edges, or by altering node features to produce erroneous results (Wang and Gong 2019; Wang et al. 2024). Researchers, under varying degrees of background knowledge, have implemented targeted or untargeted attacks by perturbing the message passing mechanism within the node neighborhood or by disrupting the topology of the entire graph (Sharma et al. 2023). However, this type of graph modification attack (GMA) assumes that the attacker has the authority to alter the data. For instance, in a fraud detection system (Duan et al. 2024), the attacker's goal is to allow the target account to evade detection and carry out illegal activities unnoticed. However, attackers typically do not have the privilege to modify user features (e.g., transaction frequency), making GMA impractical in most scenarios.

In contrast, a more practical attack method is the Graph Injection Attack (GIA), which involves injecting new adversarial nodes and edges into the graph (e.g., adding new nodes with malicious features and connecting them to existing nodes) to achieve the attack (Wang et al. 2020). GIA does not require changing the topology and features of the original nodes; instead, it achieves the attack by injecting adversarial nodes. Due to its flexibility, GIA has been proven to be more effective and practical than GMA (Chen et al. 2022). In a fraud detection scenario, attackers can create fake accounts (i.e., adversarial nodes) that simulate the transaction and behavior patterns of normal users, and inject transaction edges between the fake accounts and the target account (i.e., connect the adversarial nodes to the target node) to conceal their fraudulent activities. Recently, numerous methods have used GIA to disrupt model performance. For example, contrastive learning methods (Liu, Huang, and Zhao 2024) use a pseudo-node generator and gradient optimization to generate pseudo-nodes with attack capabilities; poisoning attack methods (Sun et al. 2020) inject nodes into the graph during the training phase and execute attacks; reinforcement learning methods (Ju et al. 2023) utilize a node generator and edge selector to produce adversarial nodes and edges.

Although graph injection attacks have been extensively studied, in practical applications, attackers often do not have access to specific structural and parameter information of the

*Corresponding authors

victim models. More critically, to ensure the stealthiness of the attack, the number of node features must be controlled within a limited attack budget. Most current research only considers generating fake nodes with feature distributions similar to existing nodes and fails to prioritize features with stronger adversarial properties. Moreover, these studies often overlook the constraints of the feature budget, resulting in the generation of fake nodes with a number of features significantly exceeding the average, thus failing to achieve both effective attacks and high imperceptibility.

In this paper, we explore a new targeted evasion attack method in black boxes achieved through node injection. In this attack scenario, the attacker can only access the topology and feature information of the graph, and can only use the black box model for prediction during the inference stage. We propose a targeted evasion attack through node injection in a black box, namely TEANI. Influenced by the message passing mechanism, the reinforcement learning PPO algorithm (Schulman et al. 2017) is used to design adversarial nodes, which have an impact on the classification of target nodes in feature aggregation engineering. The generation process of injection nodes is defined as a Markov Decision Process (MDP), in which each action is refined into the selection of features within the node, and each action is decided based on the neighborhood information of the target node, ensuring that the features of the injection node have a greater impact on the classification of the target node. The main contributions of this paper are as follows:

- We propose an effective node injection attack method for GNNs, named TEANI, which achieves highly imperceptible targeted evasion attacks in a black-box scenario.
- TEANI does not require knowledge of the victim model's structure and efficiently explores the feature space by merely observing the target node's neighborhood and using proximal policy optimization to select highly adversarial feature combinations for the attack.
- We demonstrate through extensive experiments on several benchmark datasets that TEANI exhibits stronger attack capabilities against targets compared to other state-of-the-art methods.

Preliminaries

Let $G = (V, E, \mathbf{X})$ represent an undirected graph with N nodes, where $V = \{v_i \mid i = 1, 2, \ldots, N\}$ represents the set of nodes, and $E \subseteq V \times V$ represents the set of edges. $\mathbf{X} \in \mathbb{R}^{N \times F}$ is the feature matrix corresponding to the nodes in the graph, with each vector corresponding to a node, and F is the dimension of the feature. The adjacency matrix is denoted as $\mathbf{A} \subseteq \{0, 1\}^{N \times N}$. When $\mathbf{A}_{ij} = 0$ in the i-th row and j-th column of matrix $\mathbf{A}$, it indicates that there is an edge between node v_i and node v_j. Otherwise, the two nodes are not directly connected. The distance between nodes v and u in the graph is defined as their shortest path $d(v, u)$. Additionally, the k-order neighbors of node v are defined as $\mathcal{N}_G^k(v) = \{u \in \mathcal{V} \mid d(v, u) \leq k\}$.

The graph contains nodes categorized into C classes, with each node's label represented by a one-hot vector $\mathbf{y}_i$. The label matrix is denoted as $\mathbf{Y} \subseteq \{0, 1\}^{N \times C}$. The nodes are divided into a training set V_L and a test set V_U, where $V_L \cup V_U = V$. In the node classification task, only the labels of the training set V_L are accessible and used to train the GNNs model, while the labels of the test set V_U are used to evaluate the model's predictive performance.

Graph Neural Networks

GNNs effectively handle irregular data structures through message passing mechanisms (Kipf and Welling 2017; Veličković et al. 2018). The core of this mechanism lies in the exchange and updating of information between nodes and their neighbors. At the k-th layer, node v will collect messages from its first-order neighbors $\mathcal{N}_G^1(v)$ and perform aggregation operations. Taking GCN as an example (Kipf and Welling 2017), the process of message passing can be formalized as:

$$\mathbf{x}_v^{(k+1)} = \sigma\left(\sum_{u \in \mathcal{N}_G^1(v) \cup \{v\}} \frac{1}{\sqrt{d_v d_u}} \mathbf{x}_u^{(k)} \mathbf{W}^{(k)}\right) \quad (1)$$

where $\mathbf{x}_v^{(k)}$ represents the feature representation of node v at the k-th layer, d_v represents the degree of the node v, and $\mathbf{W}^{(k)}$ represents the weight matrix.

This aggregation process combines the features of neighboring nodes with the features of the node v itself to update the feature representation of the node v. This iterative feature updating method enables GNNs to gradually capture local and global information in the graph structure, thus performing well in various tasks of graph data.

Graph Node Injection Attack

Given a trained GNNs model $\mathcal{M}$, the attacker aims to alter the classification result of the target nodes $\mathcal{T}$ by injecting fake nodes into graph G. The graph after node injection is defined as $G' = (V', E', \mathbf{X}')$, where the adjacency matrix and feature matrix are defined as:

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A} & \hat{\mathbf{A}} \\ \hat{\mathbf{A}}^T & \mathbf{B} \end{bmatrix}, \mathbf{X}' = \begin{bmatrix} \mathbf{X} \\ \hat{\mathbf{X}} \end{bmatrix} \quad (2)$$

where $\hat{\mathbf{A}} \subseteq \{0, 1\}^{N \times N'}$ is the matrix representing the adjacency relationship between the injected nodes and the original graph, and N' is the number of injected nodes. $\mathbf{B}$ and $\hat{\mathbf{X}}$ respectively represent the adjacency matrix and the feature matrix of the injection nodes.

The attacker injects these carefully designed fake nodes into the original graph, which can cause the victim model $\mathcal{M}$ to misclassify the target nodes $\mathcal{T}$ during the testing phase. The objective function of this attack method can be formally expressed as:

$$\max_{G'} \sum_{v \in \mathcal{T}} \mathbb{I}(\mathcal{M}(v, G') \neq \mathbf{Y}_v) \quad (3)$$

where $\mathbb{I}(\cdot)$ denotes the indicator function, which returns 1 if the input is true, and 0 otherwise. To enhance the stealthiness of the attack, the node injection is constrained by an attack budget. Specifically, the constraints are a budget Δ_n for the number of injected nodes, a budget Δ_e for the degree of injected nodes, and a budget Δ_f for the features of injected nodes.

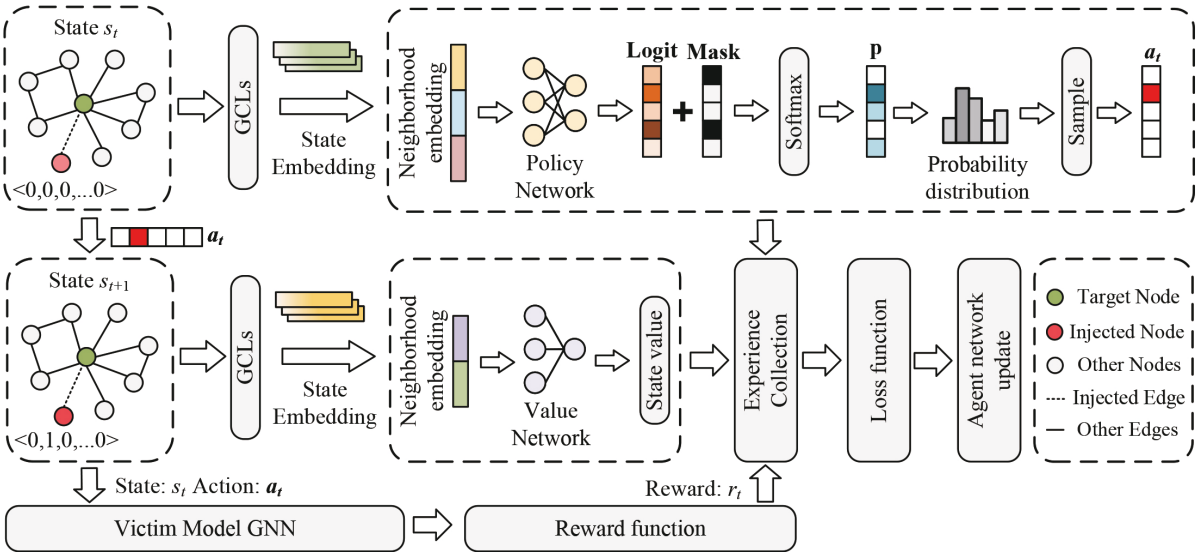


Figure 1: Overview of the application of the TEANI algorithm for injecting malicious nodes.

Methodology

In the context of adversarial attacks, evasion attacks pose a significant threat to GNNs. The effectiveness of evasion attacks largely depends on the selection of injected node features. However, finding suitable feature combinations in a high-dimensional feature space is extremely challenging, as the number of possible combinations grows exponentially with the dimensionality. Furthermore, the selected features must maximize the perturbation effect on the target node's classification result while minimally impacting the original graph structure, which imposes even higher demands on feature selection.

Feature selection is inherently a discrete process, and this discreteness makes it highly suitable for modeling as a MDP. Reinforcement learning offers a powerful approach to solving MDPs, effectively exploring the vast combinatorial space of feature selection. Specifically, in a black-box setting, we propose a reinforcement learning-based method for node injection evasion attacks. As shown in Figure 1, the agent interacts with the environment to optimize the selection of adversarial features that best mislead the model's classification of the target node, thereby maximizing the classification loss of the target node and causing the model to produce incorrect classification results.

Attack Environment

The node injection attack process is modeled as an MDP, where the attack process is defined as a tuple (S, A, R). Here, S is the state space, A is the action space, and R is the reward function. The components of the MDP are defined as follows:

State. At time t, the state $s_t \in S$ is defined as the intermediate graph at this moment, including the k-hop subgraph $\hat{G}_k(v_t)$ of the target node v_t and the adversarial node being generated. The subgraph $\hat{G}_k(v_t)$ contains the original k-order neighbors $\mathcal{N}_G^k(v_t)$ and the nodes V_{inj} injected before time t, as well as the topological information. The adversarial node v_{adv} being generated includes all selected features, which are also included in the state s_t.

Action. The action is to generate adversarial nodes within the budget Δ. We decompose the process of generating each node into a feature selection process that does not exceed the budget Δ_f times. During the generation of node v_{adv}, the trajectory of the MDP is represented as $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{T-1}, a_{T-1}, r_{T-1}, s_T)$. At time t, the action a_t selects a feature based on the state s_t. All the features selected during this process collectively form an adversarial node v_{adv}. r_t represents the reward value at time t, and the state s_T indicates the termination of the MDP, either when the budget Δ_f is exhausted or the target node is successfully attacked.

Reward. Since each action selects a feature, the learning trajectory is usually quite long. Therefore, we give a certain reward after each feature is generated, rather than providing feedback in the learning trajectory after the entire node is generated, which helps the agent find a better trajectory. To measure the impact of each action a_t on the attack, the reward r is set as the change in the classification loss of the target node v_t before and after the action is executed. Therefore, r is defined as:

$$r(v_t, a_t, G'_t) = \mathcal{L}(v_t, G'_{t+1}) - \mathcal{L}(v_t, G'_t) \tag{4}$$

where, G'_{t+1} and G'_t represent the graph after and before adding the action a_t (i.e., the feature selected at this moment) to the injected node v_{atk}, respectively. $\mathcal{L}(v_t, G)$ represents the classification loss of node v_t in graph G. Using classification loss as the reward for feature selection can provide more direct incentives for the agent. When the target node's classification is misjudged, we will give the agent an additional high reward r_{suc}, that is $r_t = r(v_t, a_t, G'_t) + r_{\text{suc}}$.

This helps to improve the relative value of successful attack actions and increase exploration efficiency.

Agent Design

The design of the reinforcement learning agent is divided into three parts: state embedding, feature selection network, and value prediction network.

State Embedding. The state s_t includes information about the k-order neighbors of the target node v_t, the injected nodes V_{inj} and the adversarial node v_{adv} being generated. To better utilize this information, first extracts the topological and feature information of s_t through K layers of Graph Convolutional Layers (GCL). The k-th step in the embedding process of the state s_t is as follows:

$$\mathbf{H}_s^{(k+1)} = \sigma(f_{\text{GCL}}(\mathbf{A}_s, \mathbf{H}_s^{(k)}, \mathbf{W}^{(k)})) \tag{5}$$

where, $\mathbf{H}_s^{(k+1)}$ represents the embedding of the state s after the k-th layer of GCL. $\sigma(\cdot)$ denotes the activation function, and in this case, ReLU is chosen. $\mathbf{A_s}$ represents the topological connection information of all nodes in the state s, where the injected nodes V_{inj} are connected to the target node v_t by default. $\mathbf{W}^{(k)}$ represents the weights of the k-th layer of GCL.

Feature Selection Network. The focus of evasion attacks lies in the impact of the injected nodes V_{inj} on the target node v_t during the message passing and aggregation process. To better affect the classification performance of the victim model, the feature selection network aims to select a more malicious feature combination from numerous candidate positions to form the adversarial node v_{adv}.

To better capture the complex relationships and patterns in the topological information, the feature selection network first performs a readout operation on the state embedding $\mathbf{H}_s$ to represent it as a vector $\mathbf{n}_s$. This vector is then concatenated with the features of the target node v_t and the information of the adversarial node v_{adv}. The concatenated vector is passed through a Multi-Layer Perceptron (MLP) to obtain an F-dimensional vector. Subsequently, a mask $\mathbf{M}$ is generated based on the existing features in v_{adv} to mask the positions of the features already selected, and a softmax operation is applied to obtain the final probability distribution $\mathbf{p}$. The process is formalized as follows:

$$\mathbf{p} = \text{Softmax}(\sigma((\mathbf{n}_s \oplus \mathbf{H}_s[v_t] \oplus \mathbf{x}_{v_{\text{adv}}}) \cdot \mathbf{W}) + \mathbf{M}) \tag{6}$$

where, $\mathbf{W} \in \mathbb{R}^{(2h+F)\times F}$ is the weight matrix. The concatenation operation $\oplus$ merges multiple feature vectors into a single feature vector for further computation.

Treat the vector $\mathbf{p}$ as the logit of a one-hot probability distribution implemented through straight-through estimation (Bengio, Le´onard, and Courville 2013) to generate one-hot encoded samples $a_t \sim \pi_\theta(\cdot|s_t)$, i.e., single features. This distribution can be sampled using the reparameterization trick. The reparameterization trick allows discrete samples to be generated through simple operations during forward propagation, while handling gradient information during backpropagation.

Algorithm 1: The training algorithm of TEANI

Input: Graph G, target node set $\mathcal{T}$, attack budget Δ, victim model f_θ, number of experience collection $\mathcal{S}$
Output: Agent policy network π

```
Initialize agent parameters and initialize the experience buffer.
while epoch do
    while step < S do
        Perform state embedding for s_t.
        Sample action a_t from the feature selection network π_θ(·|s_t).
        Calculate the state value V_π(s_t).
        Obtain the reward r_t of the action a_t.
    end while
    Calculate R_t based on r_t and V_π(s_t).
    Sample a minibatch B from the experience buffer.
    Calculate the importance sampling ratio ρ_t(θ) and the loss function L.
    Update the agent network parameters.
end while
return policy network π
```

Value Prediction Network. The value prediction network $\mathcal{V}$ is responsible for evaluating the performance of the current strategy, providing estimates of the state value function, and helping to reduce the variance of the strategy gradient. First, the current state information is embedded to obtain $\mathbf{H}_s$. Note that the state embedding weights here are different from those in the feature selection. $\mathbf{H}_s$ is passed through an MLP, and the transformed information of the target node is extracted as $\mathbf{t}$:

$$\mathbf{t} = \text{Softmax}(\mathbf{H}_s \cdot \mathbf{W}_1)[v_t] \tag{7}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h\times C}$ represents the weight parameters of the MLP layer. Subsequently, $\mathbf{t}$ is concatenated with the state embedding of the node v_t and passed through an MLP layer. The state value is proportional to the classification loss value of the node v_t, so the output is used as the value score with the cross-entropy loss of the model's prediction results:

$$\mathcal{V}_\pi(s_t) = \mathcal{L}_{CE}(\mathcal{M}(v_t, G), (\mathbf{H}_s[v_t] \oplus \mathbf{t}) \cdot \mathbf{W}_2) \tag{8}$$

where $\mathbf{W}_2 \in \mathbb{R}^{(h+C)\times C}$ represents the learnable parameters.

Training Algorithm

The TEANI achieves experience collection and parameter updates through PPO-Clip. The experience buffer contains a tuple, i.e., $\{G'_t, a_t, R_t, d_t\}$, where d_t indicates whether the MDP process is complete. The advantage of taking action a in state s compared to the average behavior is evaluated using Generalized Advantage Estimation (GAE):

$$A_t = \sum_{k=0}^{T-t} (\gamma\lambda)^k \delta_{t+1} \tag{9}$$

where, γ is the discount factor, and δ represents the TD error, $\delta_t = r_t + \gamma\mathcal{V}_\pi(s_{t+1}) - \mathcal{V}_\pi(s_t)$. Additionally, the expected total reward R_t can be calculated as $R_t = A_t + \mathcal{V}_\pi(s_t)$.

The parameter update process adjusts the difference between the new and old policies in the experience buffer using importance sampling. The importance sampling ratio $\rho_t(\theta)$ is the proportion of the probability of action a_t between the new policy $\pi_\theta(a_t|s_t)$ and the old policy $\pi_{\theta_{old}}(a_t|s_t)$. The clip function is used to limit the range of the importance sampling ratio to avoid instability caused by large policy updates:

$$\hat{\rho}_t(\theta) = \text{clip}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\right) \tag{10}$$

where ϵ represents the clipping coefficient. The loss function of the feature selection network is defined as:

$$\mathcal{L}_\pi(t) = -\min(\rho_t(\theta)A_t, \hat{\rho}_t(\theta)A_t) \tag{11}$$

Optimizing with the advantage function can effectively improve the expected return of the policy. We also use entropy regularization to encourage exploration, prevent the policy from prematurely converging to suboptimal solutions, and improve performance. The loss function of the Value Network is used to evaluate and optimize the prediction error of the value function. The formula is as follows:

$$\mathcal{L}_\mathcal{V}(t) = (\mathcal{V}_\pi(s_t) - R_t)^2 \tag{12}$$

The cosine similarity loss is used to encourage the generated injected nodes to maintain a certain similarity with the target node in the feature space, reducing the exploration space and improving stealth.

$$\mathcal{L}_{\text{sim}}(t) = (1 - \cos(\mathbf{A}_{\text{norm}}\mathbf{X}[v_t], \mathbf{X}[v_t]))^2 \tag{13}$$

where $\mathbf{A}_{\text{norm}}$ represents the regularization of the adjacency matrix. The overall loss function is expressed as:

$$\mathcal{L} = \sum_{t\in\mathcal{B}} (\mathcal{L}_\pi(t) + \mathcal{L}_\mathcal{V}(t) + \alpha\mathcal{L}_{\text{sim}}(t)) \tag{14}$$

where $\mathcal{B}$ represents a batch of experiences, and α is a hyperparameter.

Experiments

In this section, we evaluate TEANI on benchmark datasets and compare it with state-of-the-art methods, aiming to address the following questions: (**RQ1**) Can TEANI achieve strong attack performance on GNNs trained with different architectures? (**RQ2**) Can TEANI prioritize selecting features that are most adversarial to the target node during the attack? (**RQ3**) How does TEANI's attack effectiveness vary under different budget constraints?

Datasets. Following previous works (Ju et al. 2023; Liu, Huang, and Zhao 2024; Sun et al. 2020), we conducted experiments on three public datasets: Cora, Cora-ML (Mccallum et al. 2000), and Citeseer (Sen et al. 2008). All three datasets are citation networks, where nodes represent documents and edges represent citation links. The statistical details of these datasets are presented in Table 1.

Dataset	Cora	Citeseer	Cora-ML
Nodes	2078	3327	2995
Edges	5429	4732	8416
Features	1433	3703	2879
classes	7	6	7

Table 1: Statistics of the dataset.

SOTA Methods Used in Comparison. Since node injection attacks are an emerging research direction with only a few studies focusing on this field, we compared several state-of-the-art methods of node injection attacks in black box scenarios to demonstrate the effectiveness of TEANI. These methods include GCIA (Liu, Huang, and Zhao 2024), TDGIA (Zou et al. 2021), G-NIA (Tao et al. 2021), and G^2A2C (Ju et al. 2023), as well as a random method for comparison. GCIA generates adversarial nodes using gradient optimization through graph contrastive learning. TDGIA introduces a topology defect edge selection strategy and generates features for the injected nodes by optimizing the smoothness of features. G-NIA is also a single-node injection evasion attack method. G^2A2C is an injection attack based on the A2C reinforcement learning algorithm.

Parameter Settings

The experimental setup for TEANI is as follows: the state embedding network consists of two layers of GCL with a hidden dimension of 128. In the PPO, the learning rate is set to 10^{-4}, the minibatch size is 30, and the clipping coefficient ϵ is set at 0.1. In GAE, the discount factors γ and parameter λ are 0.95 and 0.99, respectively. All experiments are conducted on an Ubuntu server equipped with an Intel Xeon 8336C CPU and an RTX 4080 GPU.

In our experiments, in order to maintain higher imperceptibility, the feature budget for the injected nodes is limited. For the datasets used in the experiment, we typically set the mean of the injected node features to match the mean of the original graph. Specifically, for the TDGIA and G-NIA methods, we allow access to all data and labels, and permit the features of the injected nodes in the G^2A2C method to exceed the mean of the dataset.

Performance Comparison

To evaluate the attack capability of TEANI and conduct a comparative analysis, we configured the attack as a single-node injection, meaning only one node is injected and connected to the target node through a single edge. Table 2 presents the attack results on different models. After injecting a fake node, all SOTA methods used for comparison caused varying degrees of degradation in the model's classification accuracy compared to the clean graph. Compared to SOTA methods, TEANI demonstrated superior attack capability, more effectively selecting the most suitable features to construct adversarial nodes. In response to **RQ1**: under black-box single-node injection attacks, TEANI exhibited stronger attack capabilities across different GNN architec-

Attacker	Cora			Citeseer			Cora-ML		
	GCN	GAT	APPNP	GCN	GAT	APPNP	GCN	GAT	APPNP
Clean	18.4	16.1	15.0	28.7	27.5	24.3	14.1	16.1	15.3
Random	19.5	19.8	16.4	31.1	30.5	29.8	15.7	17.7	17.6
GCIA	26.4	23.2	19.3	39.0	39.4	35.7	25.6	22.3	21.2
G-NIA	24.3	22.1	24.2	36.5	35.3	37.2	24.2	24.1	23.7
TDGIA	29.5	28.3	29.1	44.2	52.1	43.8	29.1	29.4	28.7
G^2A2C	36.3	33.4	36.4	49.4	55.6	48.0	34.8	33.6	34.1
TEANI	**59.7**	**58.0**	**39.5**	**64.2**	**70.4**	**48.5**	**47.6**	**37.1**	**36.4**
Avg. ↑	23.4	24.6	3.1	14.8	14.8	0.5	12.8	3.5	2.3

Table 2: Misclassification rates (%) of different GNN models after injection attacks (a single node connected to the target node via one edge). The best results are highlighted in **bold**.

tures, which also indicates that its feature selection network can effectively generate fake nodes.

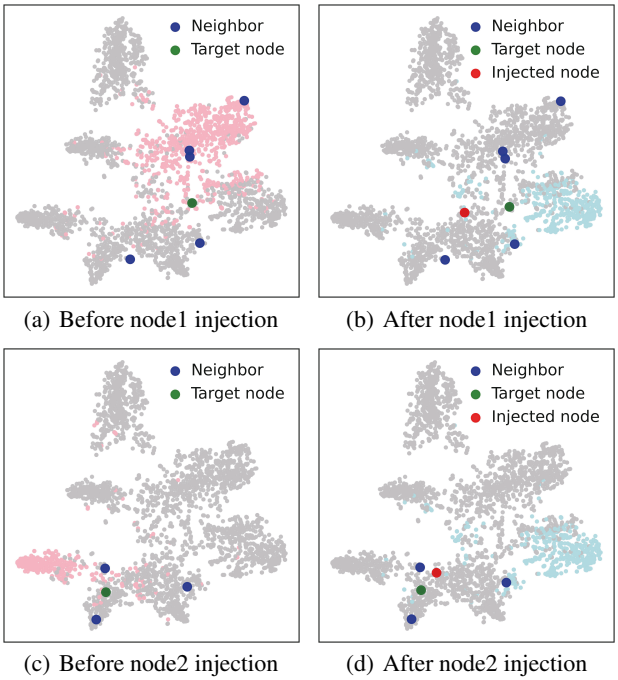
Case Study

To better illustrate the node attack process of TEANI, Figure 2 visualizes the comparison of node embeddings before and after node injection using T-SNE (van der Maaten and Hinton 2008) technology. The left side of the figure shows the representation of nodes in the clean graph before the injection, where the green node represent the target node, the blue nodes are the first-order neighbors of the target node, and the pink node represent all nodes within the target node's class. The right side displays the new embeddings after the injection, with the red node representing the newly injected adversarial node and light blue nodes representing all nodes in the new class of the target node.

As seen in the Figure 2, although the embedding of the target node undergoes slight changes after being connected to a fake node, these changes are enough to affect the victim model's classification of the target node. To address research question **RQ2**, node 1 was injected with 5 features, while node 2 had only 1 feature, far fewer than the average number of features in the original graph. This indicates that TEANI tends to prioritize features with higher attack potential when generating nodes, while also ensuring that the injection minimally impacts the target node's embedding. Additionally, the injected node's embedding is very close to the original embedding of the target node, which not only ensures the stealthiness of the injected node but also maintains high attack efficiency under conditions of high concealment.

Budget Analysis

To address **RQ3**, in this section, we conducted experiments and comparisons on the attack budget (i.e., the budget for the number of nodes Δ_n and the budget for node features Δ_f). The experimental results are presented in Figure 3 and Figure 4. The vertical axis represents the misclassification rate (%), while the horizontal axes in Figure 3 and Figure 4 represent the node budget Δ_n and feature budget Δ_f, respectively. Firstly, to verify the node budget, we performed attack experiments on different models and datasets, injecting one to five nodes. It can be observed that after injecting a



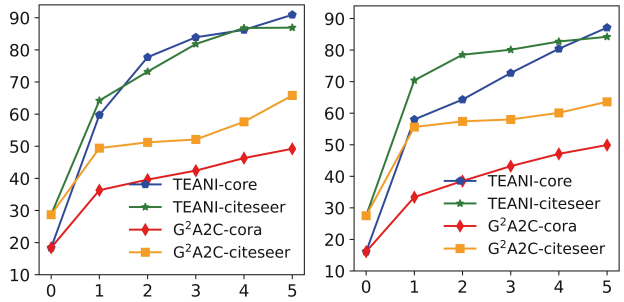
(a) Before node1 injection (b) After node1 injection

(c) Before node2 injection (d) After node2 injection

Figure 2: Node injection attack visualization.

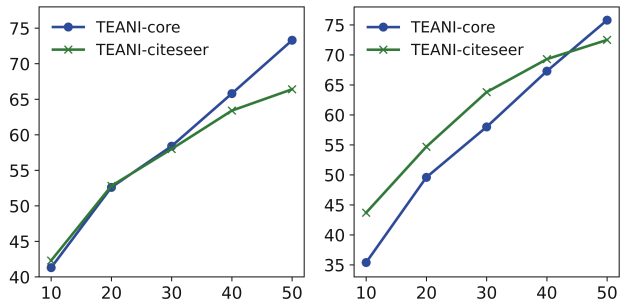
single node, the misclassification rate of the GCN model on the Cora dataset reached approximately 60%. After injecting five nodes, the classification performance of the GCN model was severely affected, with the number of correctly classified nodes falling below 10%. The same trend was observed on the GAT model. Under the same node budget Δ_n, the ability to disrupt the victim model's classification is significantly superior to that of the SOTA baselne methods.

Similarly, we also experimented with the feature budget Δ_f, as shown in Figure 4. We constrained the injected node's features to contain only Δ_f non-zero features. When $\Delta_f = 10$, meaning the injected node is a sparse vector with fewer than 10 non-zero features, the misclassification rate of the GCN model on the target nodes in Cora exceeds 40%. This indicates that TEANI tends to select more impactful



(a) GCN-Node (b) GAT-Node

Figure 3: Attack capabilities and comparison under different node budgets Δ_n.



(a) GCN-Feature (b) GAT-Feature

Figure 4: The impact of node feature budgets Δ_f on the accuracy of victim model classification.

features when attacking GNNs. When the feature budget was increased to 50, the classification accuracy of the GNN model significantly dropped to around 30%. Compared to other benchmark methods, TEANI can generate more powerful attacks with a smaller feature budget Δ_f.

Related Work

Adversarial attacks on GNNs mainly consist of two parts. One involves attacks on the existing structure of the graph, which are carried out by modifying attributes (Wang et al. 2024; Zügner, Akbarnejad, and Günnemann 2018), labels (Zhang et al. 2020), or the topology (Sharma et al. 2023; Lin et al. 2020; Xu et al. 2019; Dai et al. 2018; Wang and Gong 2019) to achieve the attack objectives. The other method is through the injection of new data (including nodes and edges) to accomplish the attack. This type of attack includes poisoning attacks (Liu, Huang, and Yi 2022; Sun et al. 2020) prior to training as well as evasion attacks (Ju et al. 2023; Tao et al. 2021; Wang et al. 2020; Liu, Huang, and Zhao 2024) during the inference stage.

For poisoning attacks in a black-box setting, early work (Bojchevski and Günnemann 2019) effectively solved the twolayer optimization problem related to poisoning attacks using eigenvalue perturbation theory. NIPA (Sun et al. 2020) uses Gaussian noise-injected feature means as the features of pseudo-nodes. These nodes are connected to the graph and assigned toxic labels through hierarchical Q-learning, resulting in effective attacks in non-targeted poisoning scenarios. LafAK (Zhang et al. 2020) is an adversarial label flipping attack method. By manipulating a subtle portion of the training labels, it can poison the training set and efficiently generate attacks using a gradient-based optimizer.

For evasion attacks in a black-box setting, GCIA (Liu, Huang, and Zhao 2024) implements a graph injection attack using contrastive learning. It encodes the target node's neighborhood and employs a pseudo-node generator that utilizes gradient optimization to create pseudo-nodes. Chang et al. (Chang et al. 2019) modeled graph embedding as a specialized graph signal processing task and proposed GF-Attack, which requires only the adjacency matrix and the feature matrix. TDGIA (Zou et al. 2021) introduces a topological defect edge selection strategy, selecting connections between original nodes and injected nodes. It generates features for the injected nodes by optimizing for smooth features. G-NIA (Tao et al. 2021) focuses on attacks with a limited node budget, achieving effective evasion in single-node injection scenarios. G^2A2C (Ju et al. 2023) optimizes the node injection strategy using the A2C algorithm, fitting the distribution of the target node's neighbors through a policy network and sampling new node features and adversarial edges to execute targeted evasion attacks.

Conclusion

This paper studies the method of targeted node evasion attacks in black-box scenarios for Graph Neural Networks, with a particular focus on enhancing the imperceptibility of the attacks. We propose an efficient method, TEANI, that enables attacks without the need to understand the structure or gradient information of GNNs, thus discarding the traditional reliance on surrogate models. This approach avoids the potential drawbacks caused by structural differences in the victim models, thereby improving the generalization and applicability of the attacks. We model the process of generating injected nodes as a Markov Decision Process and use Proximal Policy Optimization to address the feature selection problem in generating fake nodes. TEANI leverages a feature selection network to choose features that are more conducive to effective attacks within a limited feature budget while better controlling the number of features, making the injected fake nodes more difficult to detect. Extensive experiments conducted on three widely recognized benchmark datasets and different GNN architectures demonstrate that our proposed method can prioritize features with higher attack performance within a very limited budget. Compared to the state-of-the-art baseline methods, TEANI shows excellent performance, highlighting its effectiveness when facing diverse GNN architectures and datasets.

Acknowledgments

This work was sponsored by the National Natural Science Foundation of China (No. 62076130) and the Start-up Research Fund of Southeast University (No. RF1028623059).

References

Bengio, Y.; Le´onard, N.; and Courville, A. 2013. Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation. *Computer Science*.

Bojchevski, A.; and Günnemann, S. 2019. Adversarial Attacks on Node Embeddings via Graph Poisoning. In *International Conference on Machine Learning*.

Chang, H.; Rong, Y.; Xu, T.; Huang, W.; Zhang, H.; Cui, P.; Zhu, W.; and Huang, J. 2019. A Restricted Black-Box Adversarial Framework Towards Attacking Graph Embedding Models. In *AAAI Conference on Artificial Intelligence*.

Chen, Y.; Yang, H.; Zhang, Y.; Ma, K.; Liu, T.; Han, B.; and Cheng, J. 2022. Understanding and Improving Graph Injection Attack by Promoting Unnoticeability. In *International Conference on Learning Representations (ICLR)*.

Dai, H.; Li, H.; Tian, T.; Huang, X.; Wang, L.; Zhu, J.; and Song, L. 2018. Adversarial Attack on Graph Structured Data. In *International Conference on Machine Learning*.

Duan, Y.; Zhang, G.; Wang, S.; Peng, X.; Ziqi, W.; Mao, J.; Wu, H.; Jiang, X.; and Wang, K. 2024. CaT-GNN: Enhancing Credit Card Fraud Detection via Causal Temporal Graph Neural Networks. arXiv:2402.14708.

Ju, M.; Fan, Y.; Zhang, C.; and Ye, Y. 2023. Let Graph Be the Go Board: Gradient-Free Node Injection Attack for Graph Neural Networks via Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

Lin, X.; Zhou, C.; Yang, H.; Wu, J.; Wang, H.; Cao, Y.; and Wang, B. 2020. Exploratory Adversarial Attacks on Graph Neural Networks. In *2020 IEEE International Conference on Data Mining (ICDM)*.

Liu, G.; Huang, X.; and Yi, X. 2022. Adversarial Label Poisoning Attack on Graph Neural Networks via Label Propagation. In *European Conference on Computer Vision*.

Liu, X.; Huang, J.-J.; and Zhao, W. 2024. GCIA: A Black-Box Graph Injection Attack Method Via Graph Contrastive Learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Mccallum, A. K.; Nigam, K.; Rennie, J.; and Seymore, K. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2): 127–163.

Pei, Y.; Chakraborty, N.; and Sycara, K. 2015. Nonnegative matrix tri-factorization with graph regularization for community detection in social networks. In *International Joint Conference on Artificial Intelligence*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; and Eliassi-Rad, T. 2008. Collective Classification in Network Data. *Ai Magazine*.

Sharma, K.; Verma, S.; Medya, S.; Bhattacharya, A.; and Ranu, S. 2023. Task and Model Agnostic Adversarial Attack on Graph Neural Networks. In *AAAI Conference on Artificial Intelligence*.

Sun, Y.; Wang, S.; Tang, X.; Hsieh, T.-Y.; and Honavar, V. 2020. Adversarial Attacks on Graph Neural Networks via Node Injections: A Hierarchical Reinforcement Learning Approach. In *The Web Conference*.

Tao, S.; Cao, Q.; Shen, H.; Huang, J.; Wu, Y.; and Cheng., X. 2021. Single Node Injection Attack against Graph Neural Networks. In *ACM International Conference on Information and Knowledge Management*.

van der Maaten, L.; and Hinton, G. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations (ICLR)*.

Wang, B.; and Gong, N. Z. 2019. Attacking Graph-based Classification via Manipulating the Graph Structure. arXiv:1903.00553.

Wang, H.; Zhao, M.; Xie, X.; Li, W.; and Guo, M. 2019. Knowledge Graph Convolutional Networks for Recommender Systems. In *The World Wide Web Conference*.

Wang, X.; Chang, H.; Xie, B.; Bian, T.; Zhou, S.; Wang, D.; Zhang, Z.; and Zhu, W. 2024. Revisiting Adversarial Attacks on Graph Neural Networks for Graph Classification. *IEEE Transactions on Knowledge and Data Engineering*, 36(5): 2166–2178.

Wang, X.; Cheng, M.; Eaton, J.; Hsieh, C.-J.; and Wu, F. 2020. Attack Graph Convolutional Networks by Adding Fake Nodes. arXiv:1810.10751.

Xu, K.; Chen, H.; Liu, S.; Chen, P.-Y.; Weng, T.-W.; Hong, M.; and Lin, X. 2019. Topology Attack and Defense for Graph Neural Networks: An Optimization Perspective. In *International Joint Conference on Artificial Intelligence*.

yang zhang; Wei, Z.; Yuan, Y.; Li, C.; and Huang, W. 2024. EquiPocket: an E(3)-Equivariant Geometric Graph Neural Network for Ligand Binding Site Prediction. In *International Conference on Machine Learning*.

Zhang, M.; Hu, L.; Shi, C.; and Wang, X. 2020. Adversarial Label-Flipping Attack and Defense for Graph Neural Networks. In *2020 IEEE International Conference on Data Mining (ICDM)*.

Zou, X.; Zheng, Q.; Dong, Y.; Guan, X.; Kharlamov, E.; Lu, J.; and Tang, J. 2021. TDGIA: Effective Injection Attacks on Graph Neural Networks. In *ACM SIGKDD Conference on Knowledge Discovery & Data Mining*.

Zügner, D.; Akbarnejad, A.; and Günnemann, S. 2018. Adversarial Attacks on Neural Networks for Graph Data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.