Complex & Intelligent Systems (2024) 10:869–883
https://doi.org/10.1007/s40747-023-01200-6

ORIGINAL ARTICLE

Imperceptible graph injection attack on graph neural networks

Yang Chen[1,2] · Zhonglin Ye[1,2] · Zhaoyang Wang[1,2] · Haixing Zhao[1,2]

Received: 8 February 2023 / Accepted: 15 July 2023 / Published online: 9 August 2023

Abstract
In recent years, Graph Neural Networks (GNNs) have achieved excellent applications in classification or prediction tasks. Recent studies have demonstrated that GNNs are vulnerable to adversarial attacks. Graph Modification Attack (GMA) and Graph Injection Attack (GIA) are commonly attack strategies. Most graph adversarial attack methods are based on GMA, which has a clear drawback: the attacker needs high privileges to modify the original graph, making it difficult to execute in practice. GIA can perform attacks without modifying the original graph. However, many GIA models fail to take care of attack invisibility, i.e., fake nodes can be easily distinguished from the original nodes. To solve the above issue, we propose an imperceptible graph injection attack, named IMGIA. Specifically, IMGIA uses the normal distribution sampling and mask learning to generate fake node features and links respectively, and then uses the homophily unnoticeability constraint to improve the camouflage of the attack. Our extensive experiments on three benchmark datasets demonstrate that IMGIA performs better than the existing state-of-the-art GIA methods. As an example, IMGIA shows an improvement in performance with an average increase in effectiveness of 2%.

Keywords Graph neural networks · Graph injection attack · Attack invisibility · Homophily unnoticeability constraint

Introduction

In real life, complex relationships can be represented by graphs. For example, in social networks, we abstract individuals as nodes and relationships between individuals as edges [1, 2]. In citation networks, nodes represent papers and edges represent citation relations between papers [3]. Graphs are extensively utilized in diverse tasks, including node-level classification [4–6], graph-level classification [7, 8], and molecular prediction [9].

✉ Haixing Zhao
h_x_zhao@126.com

Yang Chen
chenyang2753@stu.qhnu.edu.cn

Zhonglin Ye
zhonglin_ye@foxmail.com

Zhaoyang Wang
z.y.wang@stu.qhnu.edu.cn

1 School of Computer Science, Qinghai Normal University, Xining, Qinghai, China

2 The State Key Laboratory of Tibetan Intelligent Information Processing and Application, Qinghai Normal University, Xining, Qinghai, China

In recent years, Graph Neural Networks (GNNs) have achieved outstanding results in various tasks [10–12]. GNNs overcome the problem that traditional deep learning models can't migrate to non-Euclidean distance data. However, recent studies have shown that GNNs are vulnerable to adversarial attacks leading to performance degradation [13–15]. The traditional Cyber [16] and DoS [17] attacks, they verify the stability of the model by theoretical analysis to get the system's steady state. The graph adversarial attack is a biased execution attack that degrades the performance of GNNS by modifying the node's self-attributes (including links and features) or injecting fake nodes. Various graph adversarial attack models are proposed based on training gradients, reinforcement learning and federation learning [18–22].

Graph Modification Attack (GMA) ignore an essential premise that the adversary has insufficient privileges in the real attack [18, 23–25]. In GMA models, the adversary has the privilege to modify the original data arbitrarily, which are difficult to implement and can be easily detected by defense models [19, 26]. For example, in some attacks on large online social or commercial networks (Facebook or Amazon), the adversary first attacks administrator accounts and then changes users' social connections (e.g., deleting or adding friends) or user' information (e.g., deleting or adding

preferences or postings). When the adversary performs the attack, the defense system can detect that the administrator accounts are executing abnormal commands, which leads to an alert to other administrators or security departments. FSP-GCN [27] and GraphNS [28] are two classical graph defense models, which detect anomalous nodes by measuring perturbed graph node similarity or node label differences. Therefore, it is challenging to implement in real attacks.

To lift the above limitation, researchers have proposed the Graph Injection Attack (GIA). Figure 1 shows the comparison of GMA and GIA strategies. Specifically, GMA degrades the performance of GNNs by generating perturbed edges or features, and GIA performs attacks by generating fake nodes with malicious information. Wang et al. [29] proposed the first GIA model that uses a greedy strategy to generate fake nodes. The literature [30] proposed AFGSM, which is a white-box attack using a fast gradient sign method to generate fake nodes. Sun et al. [31] proposed a global poisoning attack for NIPA based on a Q-learning strategy. The literature [32] presented NICKI based on an optimization policy that implements the misclassification of a specific node into a different class. Dai et al. [33] proposed the targeted universal attack (TUA), where TUA enhances the attackability of nodes by connecting the fake node, and the victim nodes connected to the attack nodes will be misclassified as the attacker-specified label. Though the above GIA models solve some problems, there are still two significant limitations. **(1) Features poorly invisible.** The camouflage features of the fake nodes are not considered. For example, a pair of fake node features never co-occurred in the original nodes and detectors can easily filter these anomalous nodes [34]. **(2) Structure poorly embedded.** Previous studies have shown that GIA lead to a significant decrease in the homophily of the graph structure (the homophily can reflect the reliability of the graph structure) [35–37]. Therefore, focusing only on changes in graph topological properties (i.e., node degree or betweenness) fails to guarantee the camouflage of fake nodes.

In this paper, we consider the invisibility of the structure and features and propose an *IM*perceptible *G*raph *I*njection *A*ttack (IMGIA), which can maintain the effectiveness and camouflage of the attack. Specifically, IMGIA can be divided into the following three processes: **(1) Feature generation.** IMGIA uses a normal distribution sampling to generate the fake node's initial features. This mechanism not only has low time consumption but also makes the feature distribution of the fake nodes the same as the original nodes. **(2) Link generation.** IMGIA obtains the best attack links between fake and original nodes using the mask learning method. Note that IMGIA does not need to manipulate the links between the original nodes but only needs to modify the links between the fake and original nodes. **(3) Graph optimization.** Homophily unnoticeability constraint is used to improve the camouflage of IMGIA, which can adjust the perturbation graph structure and features.

The main contributions of this paper are summarized as follows.

- We present a new graph injection attack: IMGIA. IMGIA performs without modifying the original graph.
- We promote the imperceptibility of GIA from the topological structure and features perspective, rarely discussed in previous studies.
- We have demonstrated empirically that IMGIA can achieve better performance and higher robustness than the previous GIA models.

The rest of this work is organized as follows. "Related work" section discusses the related work on graph modification and graph injection attacks. "Preliminary" section presents the knowledge about graph neural networks and graph injection attack. In "Methodology" section, each component of IMGIA is described in detail, including feature generation, link generation, and graph optimization. In "Experiments" section, we give the corresponding experiments, which mainly discuss the performance of IMGIA in detail. Finally, we conclude our work in "Conclusion" section.

Related work

Existing attack methods are classified into two main categories: GMA and GIA. In this section, we review works with GMA and then present some works on GIA.

Adversarial attack on GMA

GMA degrades the performance of GNNs in downstream tasks by perturbing the graph structure or node features [38].

The literature [19] proposed the classical graph adversarial attack, namely Nettack, which adds some unnoticeable perturbations (perturbed edges or features) into the graph to degrade the performance of GNNs. To ensure the unnoticeability of the perturbations, the feature co-occurrence and node degree distribution of the perturbed graph are similar to the original graph. The literature [23] proposed Metattack based on meta-learning, Metattack treats the graph structure matrix as hyperparameters manipulating the graph structure by the trained attack loss. Lin et al. [39] proposed a maximizing spectral distance attack SPAC, the core idea is to destroy the graph filter to achieve the attack. In addition, SPAC uses an approximation method to reduce the feature decomposition time. Liu et al. [40] proposed AtkSE, which uses edge discrete sampling to select the set of perturbation candidates and reduce the error of structural gradients. The literature [41] found that the node topology is lost using the surrogate attack

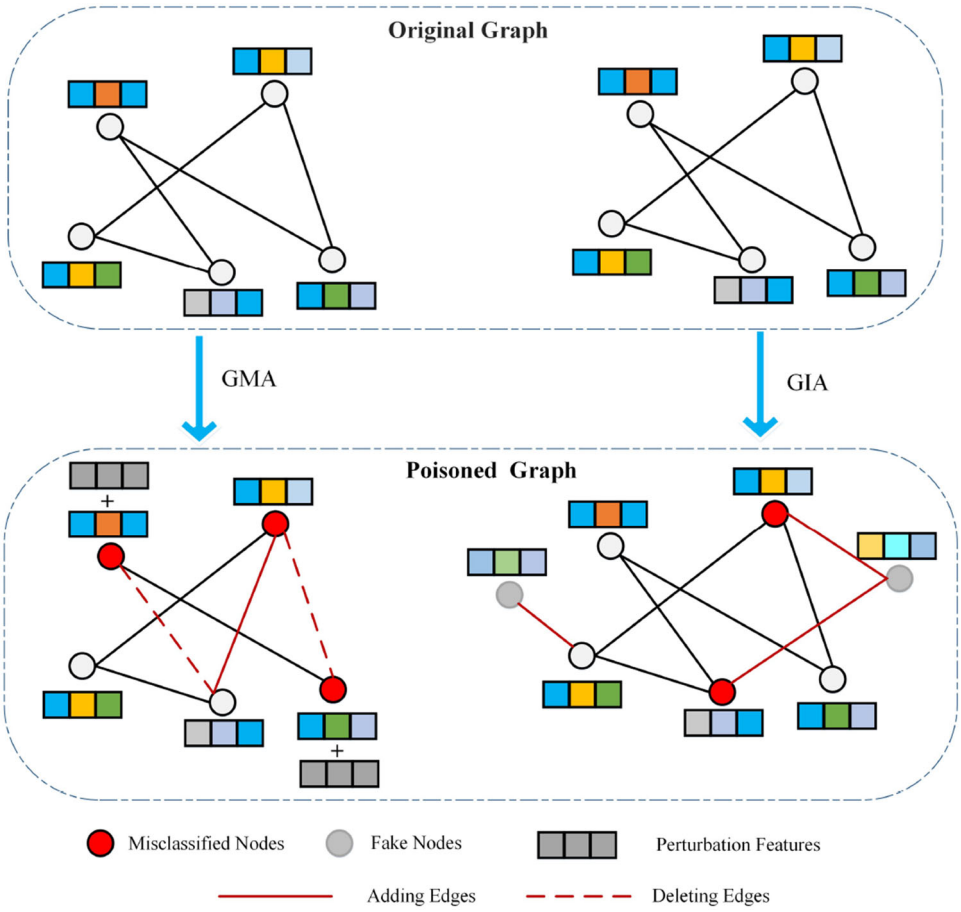

Fig. 1 Illustrative comparison of GMA (left) and GIA (right)

model, so the surrogate representation learning attack with isometric mapping (SRLIM) is proposed. To maintain node similarity during propagation, the node topology from the input layer to the embeddings is constrained by SRLIM using isometric mapping. Lin et al. [24] proposed EpoAtk, which uses gradient information to guide the adversary to modify the links. Specifically, EpoAtk uses three phases (generation, evaluation, and reorganization) to address the problem that gradient-based attacks will not get optimal solutions.

Although GMA can degrade the performance of GNNs by perturbing the graph structure and node features. Unfortunately, the implementation of GMA assumes that the adversary possesses elevated privileges, which is often difficult to attain during real-world attacks. Thus, our work focuses on practical GIA rather than GMA.

Adversarial attack on GIA

In the GIA scenario, the adversary cannot modify the structure or node features of the original graph. As a result, GIA more closely aligns with realistic attacks.

The literature [42] proposed a single node injection attack G-NIA, the experiment showed that injecting a single node can achieve efficient attacks in evasion attacks. Wang et al. [43] proposed a cluster attack (CLA). CLA calculates similarity metrics among victim nodes and injects perturbation nodes into the victim nodes of the same cluster to cause the GNNs to misclassify the targeted nodes. Zou et al. [44] analyzed the topological vulnerability of GNNs in GIA scenario and proposed the topological defective graph injection attack (TDGIA). TDGIA introduces a vulnerable topological edge selection strategy and designs a smooth feature optimization objective to generate the fake node edges amd features, respectively. Tao et al. [45] proposed the CANA framework in terms of both fidelity and diversity of self-networks centered on injection nodes, and the experimental results show that CANA significantly improves the attack performance. Ju et al. [21] studied the black-box graph injection attack and proposed GA2C, where GA2C queries the agent model based on the idea of reinforcement learning with a potential behavioral critic algorithm. The experimental results showed that GA2C can efficiently execute the attack with a low budget.

However, most previous works focus on the effectiveness of the attack and neglected attack invisibility. The literature [35] demonstrated that GIA models can damage the homophily distribution of the original graph, which is easily detected by graph defense models. Therefore, a primary

Table 1 Notations frequently used in this paper and their corresponding descriptions

Notation	Description
$\mathcal{G}$	Original graph
$\mathcal{G}'$	Poisoned graph
V	Set of nodes of the clean graph
E	Set of edges of the clean graph
X	Feature matrix of the clean graph
A	Adjacency matrix of the clean graph
X'	Feature matrix of the perturbed graph
A'	Adjacency matrix of the perturbed graph
$f_\theta(\cdot)$	Node classifier
Y	Set of node labels
y	True label
$\tilde{y}$	Prediction label
$\widehat{y}$	The attacker-specified label
n	The number of clean graph nodes
n'	The number of perturbed graph nodes
m	The number of fake nodes
Δ_X	Feature budget
Δ_S	Link budget
Δ	Total Budget
λ	Homophily parameter
$\mathcal{L}_{atk}(\cdot)$	GNNs loss
$\mathcal{L}_{\text{Hom}}(\cdot)$	Homophily loss

focus of graph adversarial attack research is designing effective and unnoticeable attacks.

Preliminary

In this section, we introduce some knowledge about GNNs and GIA. Table 1 gives the frequently used notations.

Graph neural networks

Let $\mathcal{G}$ be an attribute graph that can be represented formally as $\mathcal{G} = (V, E, X)$. where $V = \{v_1, v_2, .., v_n\}$ represents the set of nodes, n represents the number of nodes, $E \subseteq V \times V$ represents the set of edges, $X \in \mathbb{R}^{n\times d}$ denotes the d-dimensional feature matrix. The adjacency relationship of nodes can be expressed as $A \in \{0, 1\}^{n\times n}$, $A_{i,j} = 1$ means there is a link between node i and j, and 0 otherwise.

In our work, we focus on the node-level classification task, where GNNs use known label nodes to predict the class of unlabeled nodes. We take the classical Graph Convolutional Network (GCN) [10] as an example, the two-layer GCN can be represented as

$$f(\mathcal{G}) = f_\theta(A, X) = softmax(\widehat{A}\sigma(\widehat{A}XW^{(0)})W^{(1)}). \quad (1)$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, $\tilde{D}$ is the diagonal degree matrix, and I is the identity matrix. $\sigma(\cdot)$ is the activation function, usually using ReLU. W denotes the weight parameter, and $f_\theta(\cdot)$ is the node classifier. $f(\mathcal{G}) \in \mathbb{R}^{n\times k}$, where $k = |Y|$ is denoted as the number of labels, and Y represents the set of node labels.

Graph injection attack model

The utility of IMGIA is that the adversary does not need to modify the original graph structure or node features, and the performance degradation of GNNs can be achieved by manipulating fake nodes. Specifically, the graph attack is divided into untargeted and targeted attacks. In this study, we mainly focused on untargeted attacks. Furthermore, we extended our model to targeted attacks. Formally, the objective function of the untargeted and the targeted attacks can be respectively expressed as

$$\begin{aligned} &\max_{A'\in\Phi(A'),X'\in\Phi(X')} \sum_{t\in V_{\text{test}}} \mathbb{I}\left(f_{\theta^*}\left(A', X'\right) \neq y_t\right). \\ &\text{s.t. } \theta^* = \arg\min_\theta \mathcal{L}_{\text{train}}\left(f_\theta\left(A', X'\right)\right), \mathcal{G}' - \mathcal{G} \leq \Delta \end{aligned} \quad (2)$$

$$\begin{aligned} &\max_{A'\in\Phi(A'),X'\in\Phi(X')} \sum_{t\in V_{tar}, y_t\neq\widehat{y}_t} \mathbb{I}\left(f_{\theta^*}\left(A', X'\right) = \widehat{y}_t\right). \\ &\text{s.t. } \theta^* = \arg\min_\theta \mathcal{L}_{\text{train}}\left(f_\theta\left(A', X'\right)\right), \mathcal{G}' - \mathcal{G} \leq \Delta \end{aligned} \quad (3)$$

where y_t represents the label of node t. $\widehat{y}_t$ is the attacker-specified label. Δ is the attack budget. $\mathcal{L}_{\text{train}}$ usually uses the cross-entropy function, i.e., $\mathcal{L}_{\text{train}}\left(f_\theta\left(A', X'\right)\right) = \sum_{v\in V_{\text{train}}} -y_v \log \tilde{y}_v$, $\tilde{y}_v$ is the prediction label. $\mathbb{I}(\cdot)$ is an indicator function that returns 1 when the parameter is true and 0 otherwise. $\Phi(A')$ and $\Phi(X')$ are the feasible domains of the adjacency matrix $A^{'}$ and the feature matrix $X^{'}$, respectively. Poisoned graph $\mathcal{G}^{'}$ can be expressed as $\mathcal{G}^{'} = (V^{'}, E^{'}, X^{'})$, where $V' = \{v_1, v_2, \ldots, v_n, \ldots, v_{n'}\}$ is the set of poisoned graph nodes, $n'(n' = n + m)$ is the number of perturbed graph nodes, m is the number of fake nodes. $X' = \begin{bmatrix} X \\ X_m \end{bmatrix}$ is set of the poisoned graph feature. $X_m \in \mathbb{R}^{m\times d}$ is the set of features of the fake node. $A' = \begin{bmatrix} \mathrm{A} & A_m \\ \mathrm{A}_m^T & B_m \end{bmatrix}$ is the adjacency matrix of the poisoned graph. $B_m \in \mathbb{R}^{m\times m}$ is the unit matrix, $A_m \in \mathbb{R}^{n\times m}$ represents the adjacency matrix of the fake nodes and the original nodes.

Equations 2 and 3 show that the victim nodes of the untargeted attack are all the nodes in the test set, and GNNs misclassify the nodes indicating that the attack is successful.

In the targeted attack, the victim nodes are the specified nodes in the test set, and GNNs need to not only misclassify the nodes but also classify them to the attacker-specified labels. Furthermore, the aim of the untargeted attack is to decrease the classification accuracy of GNNs, and the targeted attack focuses on maximizing the accuracy of classifying victim nodes to the attacker-specified label.

Methodology

In this section, we describe the building blocks of IMGIA in detail, and the pipeline is shown in Fig. 2. Specifically, we use a normal distribution sampling to generate the fake node features. This method has a low time and space cost, and we will describe this item in detail in "Feature generation" section. Fake node links are obtained using the mask learning mechanism, which is described in detail in "Link generation" section. GIA is known to harm the homophily of the graph, which can lead to the destruction of imperceptibility. Database administrators or homology defenders can easily detect and remove fake nodes. We use homophily unnoticeability constraint to improve attack imperceptibility, and we will describe this item in detail in "Graph optimization" section.

Feature generation

In previous GIA models, fake node features are often generated using Generative Adversarial Networks (GAN) [29] and Graph Autoencoders (GA) [32] mechanisms. However, these models usually use GNNs to optimize model performance, which significantly improves model complexity and runtime. Taking GAN as an example, the generator and discriminator components require constant feedback during feature generation. If the number of feedback is low, the generated features are more different from the original features. GAN has been successful in modeling continuously distributed data. It is less effective in discrete graph data due to the difficulty in optimizing the model distribution to match the target data distribution. Moreover, they are not suitable for high-dimensional and small training datasets.

To address the above issues, we chose a simple and robust feature generation method, named normal distribution sampling. Specifically, the original features are first fitted to a normal distribution, and then the fake node features are sampled from it. No other operations are required for continuous features. For binary features, we need to binarize the sampled feature values, which are 0 when the Gaussian sample is less than 0.5 and 1 otherwise. The above operations are mathematically expressed as

$$\mathrm{X}'(x) = \mathrm{Sample}(x). \quad \text{s.t. } x \sim U, \left\|\mathrm{X}' - \mathrm{X}\right\| \leq \Delta_X. \tag{4}$$

$$\mathrm{X}'(x) = \begin{cases} 1, & \mathrm{Sample}(x) > 0.5. \\ 0, & \mathrm{Sample}(x) \leq 0.5. \end{cases} \quad \text{s.t. } x \sim U, \|X' - X\| \leq \Delta_X. \tag{5}$$

where U represents the fitted normal distribution, Δ_X represents the feature budget. If the training set contains 1 with probability p and 0 with probability $1-p$, the fitted normal distribution with mean p and variance $p(1-p)$. The probability of IMGIA sampling to 1 is $\frac{1}{2}[1 - erf(\frac{\frac{1}{2}-p}{\sqrt{2p(1-p)}})]$.

After the above feature generation process, the feature distribution of the poisoned graph has a high similarity to the original graph. Therefore, the use of normal distribution sampling can improve the camouflage of the fake node feature.

Link generation

For link generation, many works utilize gradient learning or meta-learning methods. These methods are typically less efficient, as they tend to generate only one perturbed edge per iteration. Besides, the literature [23] found that meta-learning is expensive in terms of both computation and storage.

IMGIA uses a mask learning mechanism to generate links for fake nodes. This mechanism sets the mask as a hyperparameter and iteratively optimizes it to obtain the final link mask. The mask learning mechanism not only improves the effectiveness of the attack but also has the advantage of low complexity and memory. In graph adversarial learning tasks, fake node generation can be represented as a two-layer optimization problem.

$$\begin{aligned} &\max \mathcal{L}_{atk}\left(y_t, f_{\theta^*}\left(S, X'\right)\right) \\ &\text{s.t.} \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{\text{train}}\left(f_\theta\left(A, X\right)\right). \end{aligned} \tag{6}$$

where y_t represents the real label of node t. $\mathcal{L}_{atk}(\cdot)$ represents the train loss, which usually uses a cross-entropy function. $S \in \mathbb{R}^{n' \times n'}$ represents the link mask. Specifically, $S = \begin{bmatrix} \mathrm{A} & A_1 \\ \mathrm{A}_1^T & B_m \end{bmatrix}$, $A_1 \in \mathbb{R}^{n \times m}$ denoted as an all-1 matrix, A represents the original graph adjacency matrix. X' is considered a constant and not modified in this section. Note that IMGIA does not modify the structure and features of the original graph.

When the mask learning is complete, we sort A_1 and filter to get the final links. First, sorting A_1 is sorted from largest to smallest to get A_1^{sort}, then the top Δ_S node pairs are selected, and finally the value of Δ_S node pairs in A_m is set to 1. The above procedure can be expressed formally as

$$A_{m,(i,j)} = 1 \quad s.t. \quad S_{1,(i,j)} \in S_{1,\Delta_S}^{sort}. \tag{7}$$

where Δ_S denotes the budget of the modified link.

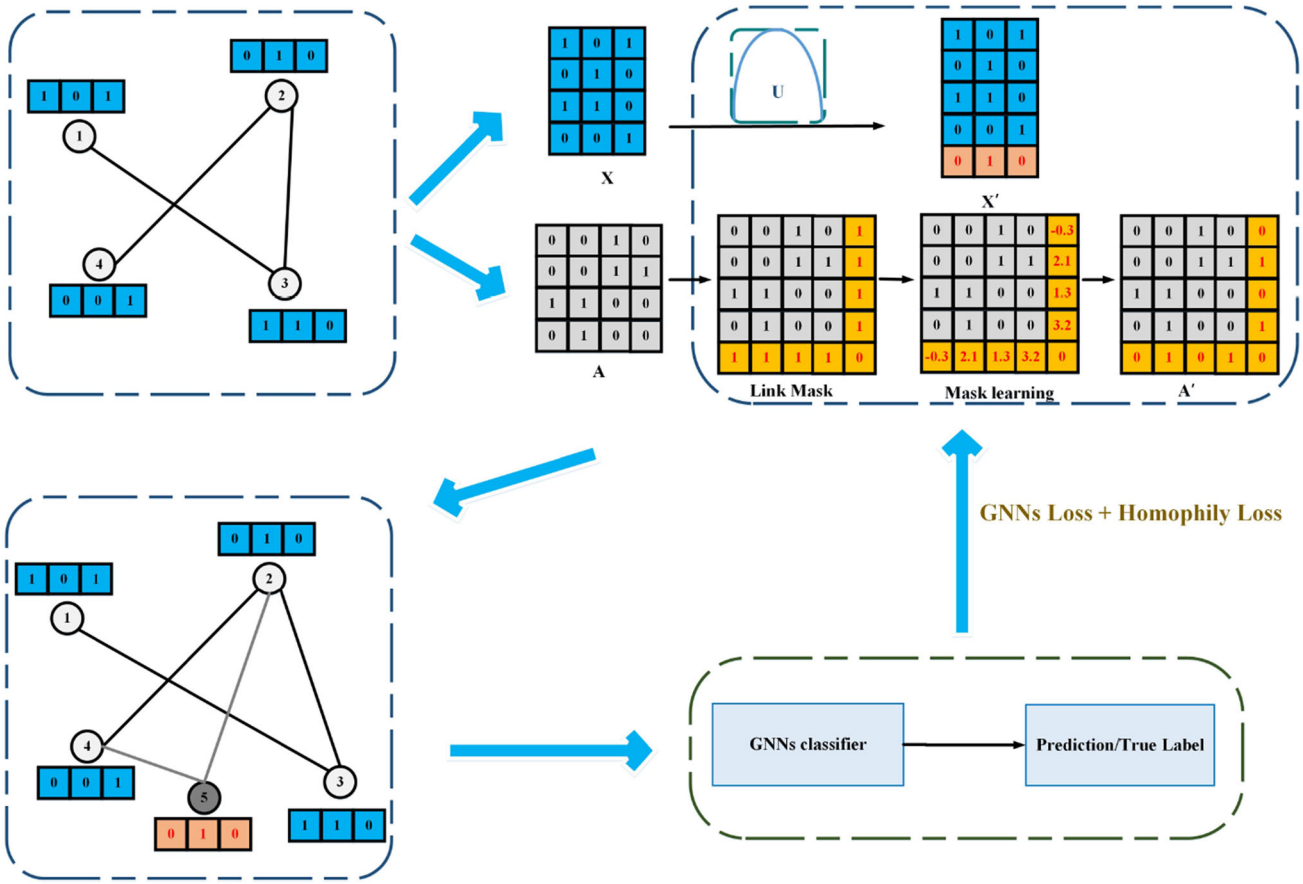

Fig. 2 Illustration of IMGIA. We first generate fake node features and links using the normal distribution sampling and mask learning mechanisms respectively, and then use the homophily unnoticeability constraint to adjust both the graph structure and fake node features

Graph optimization

The high flexibility of GIA can result in the destruction of the original graph's homophily distribution and significantly damage the similarity of neighboring nodes, consequently negatively impacting its invisibility. How can the effect of GIA on homology be reduced? To answer this question, we use the homophily unnoticeability constraint to optimize the graph (including graph structure and node features). By achieving homophily unnoticeability, IMGIA can mitigate the damage of GIA on graph homogeneity. Here, we give the definition of node homophily,

$$h_u = \text{sim}\left(r_u, X'_u\right), \quad r_u = \sum_{j \in \mathcal{N}_u} \frac{1}{\sqrt{d_j}\sqrt{d_u}} X_j. \tag{8}$$

where $sim(\cdot)$ is the cosine similarity, d_u is the degree of node u, and $\mathcal{N}_u$ denotes the set of neighbors of node u. Equation 8 shows that the homophily of node u is expressed the similarity between the features of node u and the aggregated features of its neighbors.

Intuitively, we can derive the definition of homophily of all fake nodes by the node homophily definition.

$$H\left(\mathcal{G}, \mathcal{G}'\right) = \frac{1}{m} \sum_{u \in V'/V} h_u. \tag{9}$$

Our goal is to maximize the homophily of the perturbed graph, so the overall goal of homophily unnoticeability is set as

$$\begin{aligned} \min \mathcal{L}_{Hom}(\mathcal{G}, \mathcal{G}') = -H(\mathcal{G}, \mathcal{G}') + \Omega(\left\|\mathcal{G}'\right\|) \\ s.t. \left\|\mathcal{G}' - \mathcal{G}\right\| \leq \Delta. \end{aligned} \tag{10}$$

where $\Omega(\cdot)$ is L1 norm used to coordinate the number of modified structures and features. Δ is the total budget, i.e., $\Delta = \Delta_S + \Delta_F$.

Overall objective and algorithm

Combining the adversarial attack and homophily unnoticeability constraint objectives, the overall objective function of IMGIA is as follows:

$$\min_{\mathcal{G}\in\Phi(\mathcal{G})} \mathcal{L}_{atk}\left(f_{\theta^*}\left(\mathcal{G}'\right)\right) + \lambda\mathcal{L}_{\text{Hom}}\left(\mathcal{G},\mathcal{G}'\right) \quad \text{s.t. } \left\|\mathcal{G}'-\mathcal{G}\right\| \leq \Delta. \tag{11}$$

where $\mathcal{L}_{atk}\left(y_t, f_{\theta^*}\left(S, X'\right)\right) = -\mathcal{L}_{atk}\left(f_{\theta^*}\left(\mathcal{G}'\right)\right)$, $\Phi(\mathcal{G})$ denotes the set of permissible perturbation graphs and $\lambda(\lambda \geq 0)$ is the homophily parameter controlling the scale of homophily unnoticeability.

Algorithm 1 shows the attack process of IMGIA.

Algorithm 1 IMGIA

Require: Original graph $\mathcal{G} = (V, E, X)$, Pre-training GNNs model $f_{\theta^*}(\cdot)$, Number of fake nodes m.
Ensure: Poisoned graph $\mathcal{G}^{'} = (V^{'}, E^{'}, X^{'})$.
1: Initialization: iteration parameter T, link mask S
2: Generate fake node features using the normal distribution sampling by Eq. 4 or Eq. 5
3: for $i = 1$ to T
4: Update the link mask by Eq. 6
5: Adjust the fake node features and links by Eq. 10
6: Filter the link mask to get fake node link by Eq. 7

Time complexity

In this section, we analyze the time complexity of IMGIA, and the pre-training model is taken as an example of GCN. Specifically, IMGIA contains three modules: feature generation, link generation and graph optimization. (1) In the feature generation module, we use the normal distribution sampling to generate fake node features which does not use the GCN pre-trained model, so the time complexity is low. The cost of the fake node generation module is $\mathcal{O}(md)$. (2) In the link generation module, the masked learning mechanism with GCN pre-training is used to generate fake node features. The GCN pre-trained model includes forward and backward propagations, the cost is $\mathcal{O}(n_{epo}d\left\|X'\right\|)$. Where d represents the matrix dimension, n_{epo} represents the number of training. Moreover, the cost of iteratively updating the link mask is $\mathcal{O}(T(n')^2)$, T is the iteration number. Therefore, the cost of the fake node generation module is $\mathcal{O}(n_{epo}d\left\|X'\right\| + T(n')^2)$. (3) The graph optimization module is to adjust the features and links of the fake nodes, so the time complexity can be expressed as $\mathcal{O}(T(md+mn))$.

Thus, the overall time complexity of IMGIA is $\mathcal{O}(md + n_{epo}d\left\|X'\right\| + T((n')^2 + md + mn))$.

Experiments

In this section, we will first introduce the corresponding settings, including statistics of the datasets, baselines, and GNNs and IGMIA parameter settings. Finally, the corresponding experimental results and analysis are shown to validate the performance of IGMIA.

Datasets

Our work focuses on the node-level classification task. To illustrate the adaptability of IMGIA, we conducted node-level classification experiments on three different types of datasets (Cora, Citeseer, Cora-ML). The statistics of the datasets are summarized in Table 2.

Baselines

We have verified the performance of IMGIA using several state-of-the-art GIA models as the baselines. The baseline details are as follows.

- GNIA [42]. G-NIA demonstrated the effectiveness of injecting a single fake node, and we use a generalized version of GNIA with multiple fake node attacks injected.
- NIPA [31]. NIPA uses hierarchical Q-learning to generate adversarial edges, and adds some Gaussian noise to the original node features to obtain the fake node features.
- TDGIA [44]. TDGIA uses heuristics to select fake node perturbed edges and uses optimization methods to smooth the features of the fake nodes.
- AFGSM [30]. AFGSM uses an approximate greedy strategy to generate fake node edges and features. Since AFGSM is a targeted attack, here we perform the attack using its untargeted attack version.
- GAFNC [46]. GAFNC uses GAN to generate fake node features and uses edge mask learning to generate fake node links.
- IMGIA-E1 and IMGIA-E2. IMGIA-E1 and IMGIA-E2 are extended models based on IMGIA. In the IMGIA-E1 version, the fake node features are obtained by mask learning and the links are randomly generated. In the IMGIA-E2 version, the fake node features and links are obtained by mask learning. Note that IMGIA-E1 and IMGIA-E2 still use the homophily unnoticeability constraint to adjust the graph structure and node features.

Parameter settings and evaluation metric

Parameter settings In this section, we introduce some experimental parameters. In our experiments, we use two types of GNNs, including normal GNNs (GCN [10], SGC [47] and GraphSAGE [12]) and robust GNNs (RobustGCN [48], RGAT [35]). The number of layers of GNNs is set to 2, the hidden layer dimension is set to 64, and RELU is used as the activation function. To ensure the fairness and validity of the comparison, we fixed the maximum number of training epochs as 500 and the learning rate is 0.005. In all experi-

Table 2 Statistics of three datasets

Datasets	# Nodes	# Edges	# Features	# d_{avg}	# x_{avg}	# Class	Binary
Cora	2485	5069	1433	4.18	18.30	7	Y
Citeseer	3327	4732	3703	2.84	20.34	6	Y
Cora-ML	2810	7981	2879	5.68	50.64	7	N

ments. the datasets are split with a training/validation/testing ratio of 0.1:0.1:0.8. In all experiments, the weight parameter λ is set to 1 by default. The number of fake nodes m is set to $m = \varepsilon$n, and ε is set to 0.01. We use Δ_S and Δ_F to limit the number of modified links and features, respectively. Specifically, $\Delta_S = md_{avg}$, where d_{avg} is the average degree of original nodes. The feature budget can be expressed as $\Delta_F = mx_{avg}, x_{avg} = \|X\|/n$ is the average number of original node features for discrete datasets, $x_{avg} = (X! = 0)/n$ is the average of the original node features that are not zero for continuous datasets. The d_{avg} and x_{avg} information of the datasets are summarized in Table 2.

Evaluation metric We use the classification rate as a measure of IMGIA performance. Specifically, the classification accuracy of GNNs on perturbation graphs, i.e., the number of correctly classified nodes/total number of classified nodes. The node classification accuracy metric reflects the effectiveness of IMGIA. Lower numbers indicate better results, and bolded numbers indicate optimal results. We also use the homophily distribution to measure the invisibility of the attack. A more similar homophily distribution before and after the attack indicates a good invisibility of the attack.

Experiment results

In "Attack performance under normal GNNs" section, we evaluate the attack performance of IMGIA. In "Attack performance under robust GNNs"–"Homophily distribution" sections, we demonstrate the imperceptible of IMGIA. In "Number of fake nodes"–"AMGIA on targeted attack" sections, we investigate the performance of IMGIA under different parameters. Note that in "Attack performance under normal GNNs"–"Homophily parameter" sections, the attack is set to untargeted attack, and "AMGIA on targeted attack" section to targeted attack.

Attack performance under normal GNNs

To evaluate the effectiveness and transferability of IMGIA, we perform attack experiments on three benchmark datasets under normal GNNs, and the results are shown in Table 3. For each dataset, we bold the best attack results. The specific experimental results are as follows.

Attack effectiveness Table 3 shows that the IMGIA and IMGIA-E2 achieve the lowest classification accuracy at all results. In our experiment using GCN as the victim model in Citeseer dataset, NIPA achieves the highest performance among baseline models, i.e., the classification accuracy of NIPA is 67.81%. In our models, the classification accuracy of IMGIA and IMGIA-E2 are 65.40%, 66.55% respectively. This indicates that our models achieve superior performance over the state-of-the-art models.

Comparing IMGIA with the two extension models reveals that in most cases IMGIA-E2 achieves the best attack. For example, the classification accuracies of IMGIA, IMGIA-E1 and IMGIA-E2 are 84.64%, 78.45%, 78.64% for Cora-ML, respectively. IMGIA-E2 uses a mask learning mechanism when generating features, and this approach focuses more on the harmfulness of the feature than on its invisibility (this observation will be verified later). The node link has a greater effect on the performance of GNNs than node features with the same budget, which shows that over-optimizing features does not significantly impact the model. Thus, IMGIA-E2 demonstrates marginally superior performance compared to IMGIA. IMGIA-E1 generates features in the same way as IMGIA-E2, but the random link generation is inefficient, so IMGIA-E1 can reduce the accuracy of GNNs but not significantly.

Attack transferability In addition to effectiveness, transferability is an important metric to judge the performance of a model. Table 3 shows the attack performance of our models in SGC and GraphSAGE. The results show that IMGIA-E2 achieves the best attack performance in most cases. For SGC, IMGIA-E2 shows significant improvement in attack performance under three datasets. For SCG, GNIA, NIPA, TDGIA, AFGSM, and GAFNC reduce the accuracy by 4.25%, 4.31%, 3.15%, 2.81%, and 5.02% in the case of Cora, respectively. IMGIA-E2 and IMGIA reduce the accuracy by 7.40% and 6.47%, respectively. In other cases, the results are the same. Overall, our attacks can be effectively transferred to other normal GNNs.

Attack performance under robust GNNs

In this section, we conduct the experiment to further validate the effectiveness and robustness of our models under two classes of robust GNNs, and the results are shown in Table 4. The performance of all attack models is reduced under RobustGCN and RGAT. Comparing our models with other benchmark models reveals that IMGIA and IMGIA-E2

Table 3 Classification accuracy (%) of GIA models under normal GNNs

	GCN			SGC			GraphSAGE		
	Cora	Citeseer	Cora-ML	Cora	Citeseer	Cora-ML	Cora	Citeseer	Cora-ML
Clean	85.44	71.96	85.34	85.79	70.12	84.17	83.08	71.64	84.85
GNIA	82.19	68.37	82.45	81.54	68.16	82.10	79.44	67.88	84.45
NIPA	81.48	67.81	81.05	81.48	67.94	80.75	80.15	68.12	81.88
TDGIA	82.86	67.14	80.11	82.64	68.36	79.15	81.11	67.11	79.87
AFGSM	83.60	69.72	81.67	82.98	69.84	80.75	81.48	70.56	81.15
GAFNC	81.14	67.89	79.15	80.77	67.34	78.21	80.16	66.78	78.51
IMGIA-E1	84.01	71.16	84.64	84.13	70.44	82.41	82.45	71.10	83.11
IMGIA-E2	79.63	**65.40**	**78.45**	**78.39**	**65.70**	**76.52**	**77.97**	**66.02**	77.48
IMGIA	**79.20**	66.55	78.64	79.32	66.23	77.97	78.53	66.78	**77.10**

Lower numbers indicate better results, and bolded numbers indicate optimal results. The results are the average of 10 runs

still manage to maintain the lowest classification accuracy. For example, the classification accuracy of GNIA, NIPA, TDGIA, AFGSM, and GAFNC are 84.62%, 83.41%, 84.25%, 85.21%, and 83.13% in RGAT with Cora, respectively. The classification accuracy of IMGIA and IMGIA-E2 are 81.67% and 81.59%, respectively. In summary, our models achieve the best attacks under different robust GNNs. In other words, the robustness of our model is better than other attack models.

Attack performance under defense mothods

In this section, we investigate the attack performance of IMGIA under the defense model. Note that the default setting of the victim model is GCN in this and subsequent sections.

Since randomly sampled links can reduce the propagation of malicious information than randomly sampled features, so here we use a Randomly Droped Link Fusion (RDLF) defense model. Specifically, we first generate multiple sampled perturbation graphs by deleting links (the number of sampled graphs is set to 5, and the number of deleted links is set to $0.01 \left\| E' \right\|$), then we use GCN to train the perturbation graph to obtain the node representations, and finally the node representations are summed and fused to obtain the prediction label of the nodes. Although RDLF is a simpler defense model, which can reduce the attack performance, so we use it here to verify the effectiveness and robustness of IMGIA.

Figure 3 shows GIA models classification accuracy results with and without defense. It is observed that RDLF effectively reduces the attack performance in all three datasets. Comparing the baseline models, IMGIA and IMGIA-E2 with defense obtained satisfactory results in the defense case. Fortunately, the difference between IMGIA and IMGIA-E2 with defense accuracy is small under Cora and Cora-ML datasets. All in all, IMGIA and its variants remain efficiently attacking under the defense model.

Perturbation graph visualization

To visualize the invisibility of IMGIA, we use T-SNE visualization to investigate the fake node feature in this section.

In Fig. 4, we find that GNIA and NIPA generate fake node features with poor invisibility, i.e., the fake node feature distribution is different from the original node feature distribution. AFGSM is extended from a targeted attack to an untargeted attack, and the distribution of fake node features is usually concentrated around the target node, which is easily detected by the defense model. Figure 4(f, g) show the distribution of IMGIA-E1 and IMGIA-E2 fake node features. Unfortunately, there are many outlier nodes, so we think that the mask learning focuses on attack efficiency rather than invisibility.

Since TDGIA and GAFNC use smoothed feature optimization and GAN to obtain the features of the fake nodes, respectively. Their fake node features are more hidden as shown in Fig. 4(c, e). Figure 4(h) shows that the fake node features of IMGIA can be well distributed and not concentrated. Table 5 shows the time complexity of generating fake node features for TDGIA, GAFNC and IMGIA. The fake nodes generated by TDGIA and GAFNC are well invisible, but they are time costly. Normal distribution sampling can generate invisible fake node features with low time complexity. Comparing the time complexity of TDGIA, GNFNC and IMGIA, we observe that IMGIA: $\mathcal{O}(md)$ < TDGIA: $\mathcal{O}(\|X\| md)$ < GAFNC: $\mathcal{O}(Tn_{epo} \|X\| md)$. In short, the above results demonstrate IMGIA can generate node features with good camouflage at the cost of low complexity.

Homophily distribution

In the existing GIA models, many researchers have noticed the distribution of the fake node in T-SNE visualization. Intuitively, this is a preliminary exploration of the attack's

Table 4 Classification accuracy (%) of GIA models under normal GNNs

	RobustGCN			RGAT		
	Cora	Citeseer	Cora-ML	Cora	Citeseer	Cora-ML
Clean	86.78	73.47	87.23	87.03	73.92	87.80
GNIA	84.44	70.64	85.13	84.62	70.49	84.67
NIPA	84.61	71.03	84.37	83.41	69.16	83.16
TDGIA	83.75	70.34	84.05	84.25	70.17	84.54
AFGSM	84.55	71.87	85.69	85.21	70.87	85.05
GAFNC	83.89	69.31	83.77	83.13	69.04	82.21
IMGIA-E1	85.46	72.17	86.42	86.22	72.13	86.10
IMGIA-E2	**81.46**	**67.97**	82.57	81.67	**67.34**	**81.76**
IMGIA	82.02	68.31	**82.50**	**81.59**	68.41	82.01

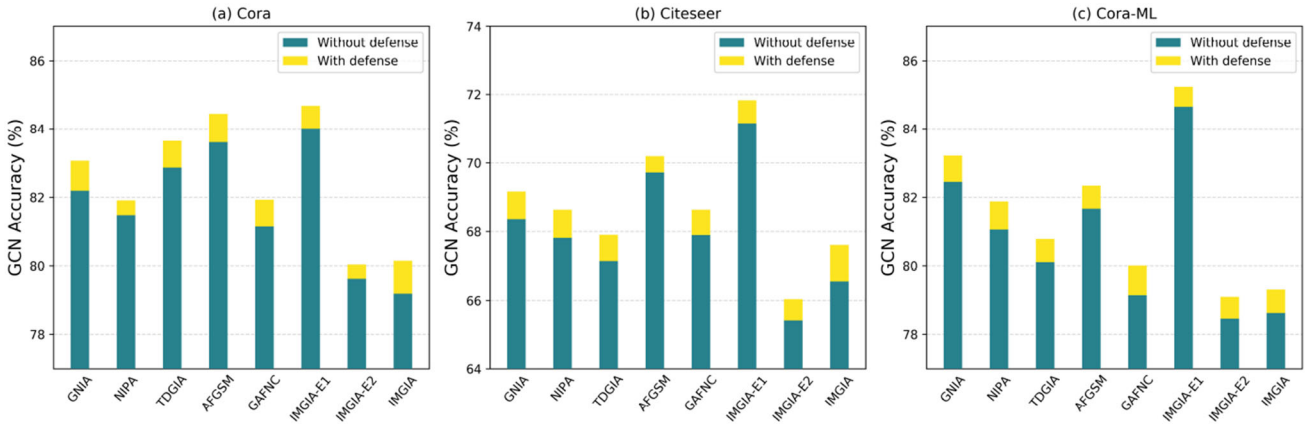

Fig. 3 Accuracy of different GIA models with and without defense

Table 5 The time complexity of generating nodes for TDGIA, GAFNC and IMGIA

Model	Time Complexity
TDGIA	$\|X\| md$
GAFNC	$Tn_{epo} \left\|X'\right\| md$
IMGIA	md

In GAFNC, the GCN is chosen to implement the discriminator and T represents the number of GAN network optimizations

invisibility. We further investigate the impact of our models on homophily. Figure 5 visualizes the homophily distribution of the GIA models before and after the attack. The blue and orange colors indicate the homophily distributions of the original and perturbed graphs, respectively.

Figure 5(a–e) show that both baseline models damage the homophily of the original graph, making a large difference in the homophily distribution before and after the attack. From Fig. 5(f–h), we find that our models are able to recover the damage of GIA on graph homogeneity, especially in IMGIA and IMGIA-E2. Homophily distribution is an important characteristic to ensure the invisibility of attacks. The experiment shows that IMGIA solves the GIA vulnerability problem in homophily detection.

Number of fake nodes

This section investigates the impact of the number of fake nodes on the performance of GIA models, and the results are shown in Fig. 6. The number of fake nodes is positively related to the performance of the GIA models, i.e., the more fake nodes, the lower the accuracy rate. Figure 6 shows that the performance of IMGIA and IMGIA-E2 is always. Specifically, taking Cora as an example, the classification accuracy of IMGIA and IMGIA-E2 are {82.99%, 79.14%, 77.78%}, {83.12%, 79.08%, 77.10%} when the number of fake nodes are {10, 30, 50}, respectively.

Homophily parameter

In this section, we performed an ablation study to analyze the effect of homophily parameters on our models, and the results are shown in Fig. 7. We find that the homophily parameter does not significantly affect the IMGIA performance. For example, in Cora, the classification accuracy of IMGIA-

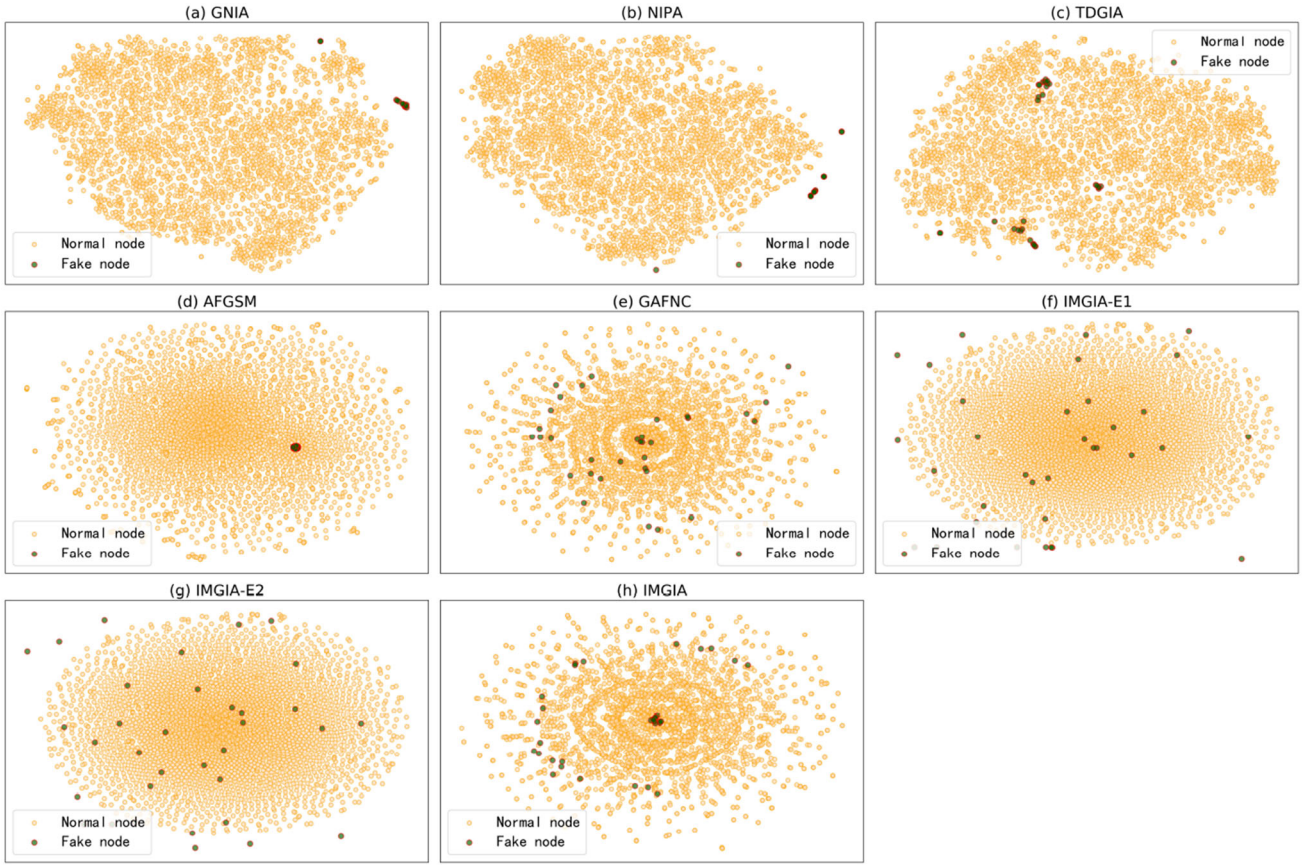

Fig. 4 T-SNE visualization of GIA models in Citeseer

E1, IMGIA-E2 and IMGIA are {85.41%, 85.14%, 84.12%}, {80.64%, 79.63%, 79.11%} and {80.20%, 79.14%, 79.05%} when λ are {0.2, 0.6, 1.2}, respectively. Also, similar results are obtained on other datasets. In short, the homophily parameter does not affect the overall performance of our models.

AMGIA on targeted attack

In this section, we extend IMGIA and its variants from the untargeted to targeted attack to demonstrate the attack's generalizability. The untargeted attack aims at misclassifying nodes without other label requirements. The targeted attack aims to misclassify specified nodes as a pre-specified label. Table 6 investigates the performance of our model under the targeted attack.

Overall, IMGIA and IMGIA-E2 have superior attack effectiveness under targeted attack. Specifically, when the pre-specified label is 1, the attack success rates of IMGIA-E1, IMGIA-E2 and IMGIA are 12.02%, 73.13% and 70.63% in Citeseer, respectively. Moreover, we observe that our models have the lowest attack performance in Cora-ML. For example, the average classification accuracies of IMGIA are 70.73%, 71.41%, and 67.04% in Cora, Citeseer, and Cora-ML, respectively. Intuitively, Cora-ML is a continuous feature, IMGIA is more difficult to generate continuous features than discrete features, so the number of victim nodes misclassified to the target label is lower.

Conclusion

In previous graph injection attacks, many attacks focus only on the attack's effectiveness but neglect the attack's invisibility, which makes the attack easily vulnerable. Therefore, in this paper, we design an effective and imperceptible graph injection attack model, namely IMGIA. IMGIA considers the imperceptibility of attacks in terms of features and structure, which has been rarely discussed in previous work. The experiments demonstrate that IMGIA achieves the lowest GNNs classification accuracy compared to some of the advanced GIA models. Besides, IMGIA shows good invisibility in various defense experiments.

In future work, we plan to explore efficient and stealthy attacks in directed graphs or hypergraphs. In addition, our work reflects the problem of GNNs' vulnerability, which will inspire us to design more robust GNNs.

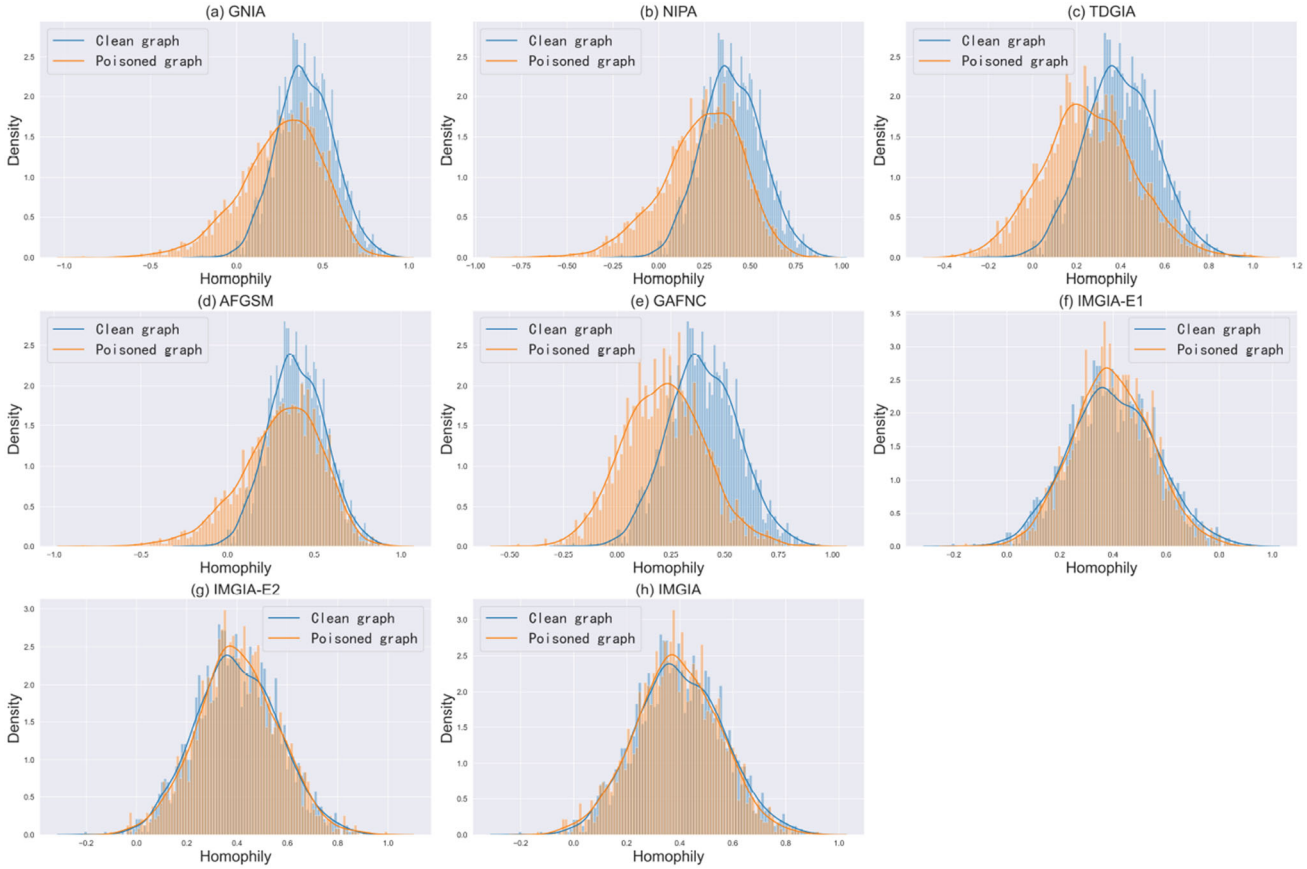

Fig. 5 The distribution of homology before and after the attack in Citeseer

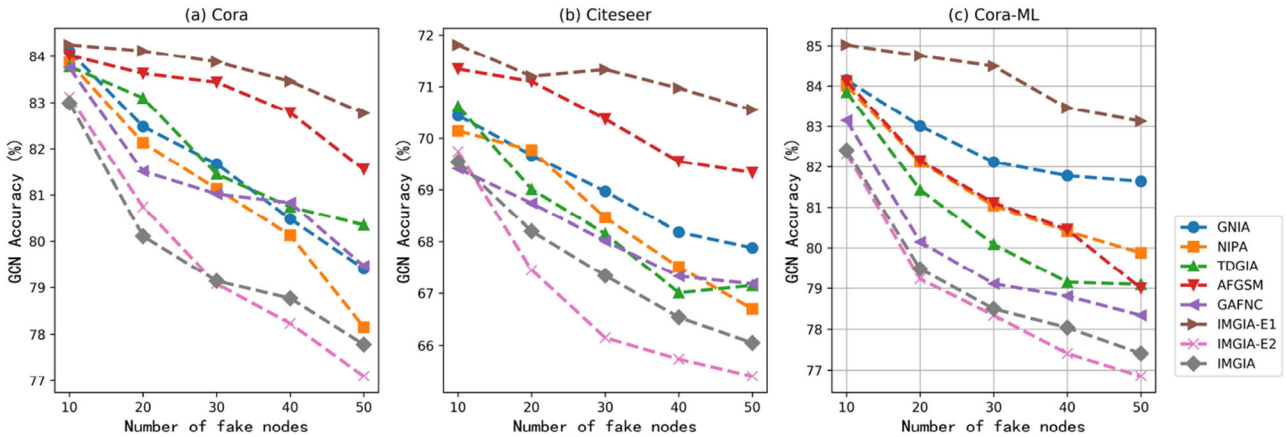

Fig. 6 The classification accuracy under the different number of fake nodes

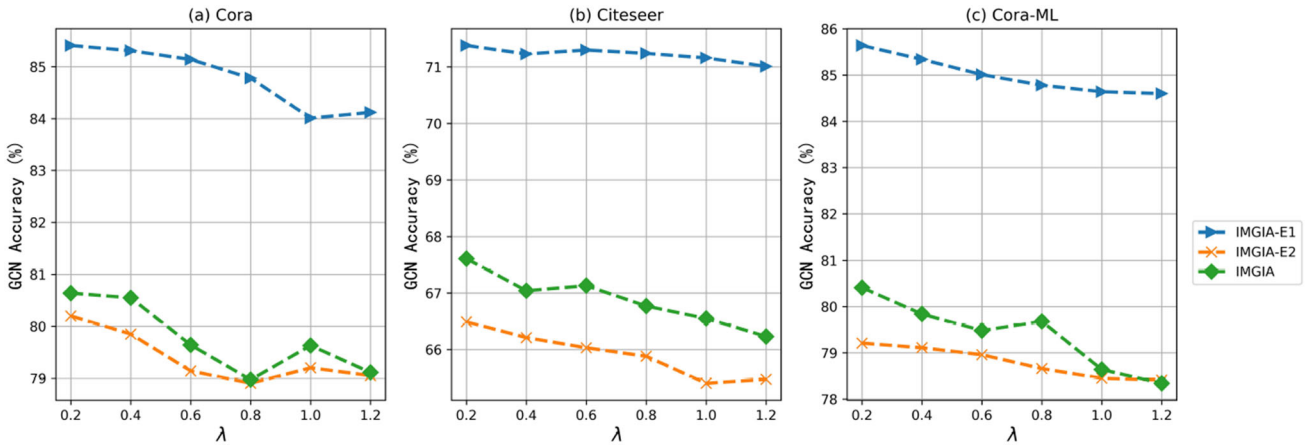

Fig. 7 The influence of the homophily parameter on the performance of our models

Table 6 Classification accuracy (%) of IMGIA and extended models under the targeted attack

Datasets	Model	Target label						
		1	2	3	4	5	6	7
Cora	IMGIA-E1	10.19	11.51	10.01	11.02	10.54	13.05	9.23
	IMGIA-E2	**71.31**	**71.06**	70.18	**72.23**	**71.67**	70.32	**71.92**
	IMGIA	70.02	70.43	**70.94**	71.12	70.64	**71.67**	70.31
Citeseer	IMGIA-E1	12.02	11.03	11.45	11.57	10.66	12.40	–
	IMGIA-E2	**73.13**	**72.97**	**73.45**	71.66	**72.75**	**73.09**	–
	IMGIA	70.63	71.64	71.67	**72.16**	70.98	71.39	–
Cora-ML	IMGIA-E1	10.65	9.48	10.14	10.63	10.54	11.19	10.96
	IMGIA-E2	**68.64**	**67.45**	**67.43**	**68.31**	**67.61**	66.50	66.41
	IMGIA	66.33	67.06	67.10	67.41	66.97	**67.63**	**66.78**

Higher numbers indicate better results, and bolded numbers indicate optimal results in each attack setting. The results are the average of 10 runs

Acknowledgements This work was supported by the National Key R&D Program of China (No. 2020YFC1523300), and Construction Project for Innovation Platform of Qinghai Province (No. 2022ZJT02).

Data availability The processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study. In the future, the processed data that support the findings of this study are available from the corresponding author upon reasonable request.

Declarations

Conflict of interest The authors declare no conflicts of interest.

References

1. Amara A, Hadj T, Ben AM (2022) Cross-network representation learning for anchor users on multiplex heterogeneous social network. Appl Soft Comput 118:108461
2. Yin X, Lin W, Sun K, Wei C, Chen Y (2023) A^2s^2-GNN: Rigging GNN-based social status by adversarial attacks in signed social networks. IEEE Trans Inf Forensics Secur 18:206–220
3. Pornprasit C, Liu X, Kiattipadungkul P, Kertkeidkachorn N (2022) Enhancing citation recommendation using citation network embedding. Scientometrics 127(1):233–264
4. Wu H, Ng MK (2022) Hypergraph convolution on nodes-hyperedges network for semi-supervised node classification. ACM Trans Knowl Discov Data 16(4):1–19
5. Pang Y, Huang T, Wang Z, Li J, Hosseini P (2022) Graph decipher: a transparent dual-attention graph neural network to understand the message-passing mechanism for the node classification. Int J Intell Syst 37(11):8747–8769
6. Xiao L, Xu P, Jing L, Akujuobi U, Zhang X (2022) Semantic guide for semi-supervised few-shot multi-label node classification. Inf Sci Int J 591:235–250

7. Xie Y, Lv S, Qian Y, Wen C, Liang J (2022) Active and semi-supervised graph neural networks for graph classification. IEEE Trans Big Data 8(4):920–932
8. Doshi S, Chepuri SP (2022) Graph neural networks with parallel neighborhood aggregations for graph classification. IEEE Trans Signal Process 70:4883–4896
9. Wang Z, Liu M, Luo Y, Xu Z, Xie Y, Wang L, Cai L, Qi Q, Yuan Z, Yang T (2022) Advanced graph and sequence neural networks for molecular property prediction and drug discovery. Bioinformatics 38(9):2579–2586
10. Kipf TN, Welling M (2016) Semi-supervised classification with graph convolutional networks. arXiv:1609.02907
11. Veličković P, Cucurull G, Casanova A, Romero A, Lio P, Bengio Y (2017) Graph attention networks. arXiv:1710.10903
12. Hamilton W, Ying Z, Leskovec J (2017) Inductive representation learning on large graphs. Adv Neural Inf Process Syst 30
13. Yang S, Doan BG, Montague P, DeVel O (2022) Transferable graph backdoor attack. In: Proceedings of the 25th international symposium on research in attacks, intrusions and defenses. p 321–332
14. Liu Z, Luo Y, Wu L, Liu Z, Li SZ (2022) Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias. In: Oh AH, Agarwal A, Belgrave D, Cho K (eds) Advances in neural information processing systems
15. Chen Y, Ye Z, Zhao H, Meng L, Wang Z, Yang Y (2022) A practical adversarial attack on graph neural networks by attacking single node structure. In: 2022 IEEE 24th international conferences on high performance computing & communications. IEEE, p 143–152
16. Zhang Z, Song X, Sun X, Stojanovic V (2023) Hybrid-driven-based fuzzy secure filtering for nonlinear parabolic partial differential equation systems with cyber attacks. Int J Adapt Control Signal Process 37(2):380–398
17. Song X, Wu N, Shuai S, Stojanovic V (2023) Switching-like event-triggered state estimation for reaction-diffusion neural networks against dos attacks. Neural Process Lett
18. Chen J, Wu Y, Xu X, Chen Y, Zheng H, Xuan Q (2018) Fast gradient attack on network embedding. arXiv:1809.02797
19. Zügner D, Akbarnejad A, Günnemann S (2018) Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. p 2847–2856
20. Sharma K, Trivedi R, Sridhar R, Kumar S (2022) Imperceptible adversarial attacks on discrete-time dynamic graph models. In: NeurIPS 2022 temporal graph learning workshop
21. Ju M, Fan Y, Zhang C, Ye Y (2022) Let graph be the go board: gradient-free node injection attack for graph neural networks via reinforcement learning. arXiv:2211.10782
22. Chen J, Huang G, Zheng H, Yu S, Jiang W, Cui C (2021) Graph-fraudster: adversarial attacks on graph neural network based vertical federated learning. https://doi.org/10.48550/arXiv.2110.06468
23. Zügner D, Günnemann S (2019) Adversarial attacks on graph neural networks via meta learning. In: International conference on learning representations (ICLR)
24. Lin X, Zhou C, Wu J, Yang H, Wang H, Cao Y, Wang B (2023) Exploratory adversarial attacks on graph neural networks for semi-supervised node classification. Pattern Recogn 133:109042. https://doi.org/10.1016/j.patcog.2022.109042
25. Chen Y, Ye Z, Zhao H, Wang Y et al (2023) Feature-based graph backdoor attack in the node classification task. Int J Intell Syst 2023
26. Nguyen TT, Quach KND, Nguyen TT, Huynh TT, Vu VH, Le Nguyen P, Jo J, Nguyen QVH (2022) Poisoning GNN-based recommender systems with generative surrogate-based attacks. ACM Trans Inf Syst. https://doi.org/10.1145/3567420
27. Li Y, Liao J, Liu C, Wang Y, Li L (2022) Node similarity preserving graph convolutional network based on full-frequency information for node classification. Neural Process Lett. https://doi.org/10.1007/s11063-022-11094-z
28. Zhuang J, Hasan MA (2022) How does bayesian noisy self-supervision defend graph convolutional networks? Neural Process Lett 54(4):2997–3018
29. Wang X, Eaton J, Hsieh C-J, Wu SF (2018) Attack graph convolutional networks by adding fake nodes. arXiv:1810.10751
30. Wang J, Luo M, Suya F, Li J, Yang Z, Zheng Q (2020) Scalable attack on graph data by injecting vicious nodes. Data Min Knowl Discov 34(5):1363–1389. https://doi.org/10.1007/s10618-020-00696-7
31. Sun Y, Wang S, Tang X, Hsieh T-Y, Honavar V (2020) Adversarial attacks on graph neural networks via node injections: a hierarchical reinforcement learning approach. In: Proceedings of the web conference 2020. WWW '20. p 673–683. https://doi.org/10.1145/3366423.3380149
32. Sharma AK, Kukreja R, Kharbanda M, Chakraborty T (2023) Node injection for class-specific network poisoning. arXiv:2301.12277
33. Dai J, Zhu W, Luo X (2022) A targeted universal attack on graph convolutional network by using fake nodes. Neural Process Lett 54(4):3321–3337. https://doi.org/10.1007/s11063-022-10764-2
34. Boukerche A, Zheng L, Alfandi O (2020) Outlier detection: methods, models, and classification. ACM Comput Surv. https://doi.org/10.1145/3381028
35. Chen Y, Yang H, Zhang Y, KAILI M, Liu T, Han B, Cheng J (2022) Understanding and improving graph injection attack by promoting unnoticeability. In: International conference on learning representations
36. Liu Z, Wang G, Luo Y, Li SZ (2022) What does the gradient tell when attacking the graph structure. arXiv:2208.12815
37. Fang J, Wen H, Wu J, Xuan Q, Zheng Z, Tse CK (2022) Gani: global attacks on graph neural networks via imperceptible node injections. arXiv:2210.12598
38. Wu B, Yang X, Pan S, Yuan X (2021) Adapting membership inference attacks to GNN for graph classification: approaches and implications. arXiv:2110.08760
39. Lin L, Blaser E, Wang H (2022) Graph structural attack by perturbing spectral distance. In: Zhang A, Rangwala H (eds) KDD '22: the 28th ACM SIGKDD conference on knowledge discovery and data mining, Washington, DC, USA, August 14–18, 2022. p 989–998. https://doi.org/10.1145/3534678.3539435
40. Liu Z, Luo Y, Wu L, Li S, Liu Z, Li SZ (2022) Are gradients on graph structure reliable in gray-box attacks? In: Association for computing machinery, vol 9. p 1360–1368
41. Liu Z, Luo Y, Zang Z, Li SZ (2022) Surrogate representation learning with isometric mapping for gray-box graph adversarial attacks. p 591–598. https://doi.org/10.1145/3488560.3498481
42. Tao S, Cao Q, Shen H, Huang J, Wu Y, Cheng X (2021) Single node injection attack against graph neural networks. In: Proceedings of the 30th ACM international conference on information; knowledge management. p 1794–1803. https://doi.org/10.1145/3459637.3482393
43. Wang Z, Hao Z, Wang Z, Su H, Zhu J (2021) Cluster attack: query-based adversarial attacks on graphs with graph-dependent priors. arXiv:2109.13069
44. Zou X, Zheng Q, Dong Y, Guan X, Kharlamov E, Lu J, Tang J (2021) TDGIA: Effective Injection Attacks on Graph Neural Networks. arXiv:2106.06663
45. Tao S, Cao Q, Shen H, Wu Y, Hou L, Cheng X (2022) Adversarial camouflage for node injection attack on graphs. arXiv:2208.01819
46. Jiang C, He Y, Chapman R, Wu H (2022) Camouflaged poisoning attack on graph neural networks. In: Proceedings of the 2022 international conference on multimedia retrieval. p 451–461
47. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K (2019) Simplifying graph convolutional networks. In: Proceedings of the 36th international conference on machine learning. p 6861–6871

48. Zhu D, Zhang Z, Cui P, Zhu W (2019) Robust graph convolutional networks against adversarial attacks. In: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining. p 1399–1407