SPEAR: A Structure-Preserving Manipulation Method for Graph Backdoor Attacks

Yuanhao Ding[†‡]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
dingyuanhao19@mails.ucas.ac.cn

Yang Liu[*†]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
liuyang2023@ict.ac.cn

Yugang Ji
Individual Researcher
Hangzhou, China
jiyugangcs@gmail.com

Weigao Wen
Individual Researcher
Hangzhou, China
weigaowen@gmail.com

Qing He[†‡]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
heqing@ict.ac.cn

Xiang Ao[*†‡]
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
aoxiang@ict.ac.cn

Abstract

Graph Neural Networks (GNNs) are vulnerable to backdoor attacks, where adversaries implant malicious triggers to manipulate model predictions. Existing graph backdoor attacks are susceptible to defense mechanisms or robust classifiers because they rely on subgraph injection or structural perturbations, e.g., creating additional edges to attach backdoor triggers to the original graph. To enhance the stealthiness of graph backdoors, we propose SPEAR, a novel structure-preserving graph backdoor attack that avoids modifying the graph's topology. SPEAR operates within a limited attack budget by selectively perturbing node attributes while ensuring the triggers exert significant influence through a global importance-driven feature selection strategy. Additionally, a neighborhood-aware trigger generator is employed to underpin a high attack success rate by utilizing semantic information from the neighborhood. SPEAR amplifies effectiveness and stealthiness by combining subtle yet impactful attribute manipulation with a refined trigger generation mechanism. Extensive experiments demonstrate that SPEAR achieves state-of-the-art effectiveness in bypassing defenses on real-world datasets, establishing it as a potent and stealthy backdoor attack for graph-based tasks. Code is available at https://github.com/yhDing/SPEAR.

CCS Concepts

• **Computing methodologies** → **Machine learning**; • **Mathematics of computing** → *Graph algorithms*; • **Security and privacy**;

Keywords

Graph Neural Network, Adversarial Attack, Backdoor Attack

ACM Reference Format:
Yuanhao Ding, Yang Liu, Yugang Ji, Weigao Wen, Qing He, and Xiang Ao. 2025. SPEAR: A Structure-Preserving Manipulation Method for Graph Backdoor Attacks. In *Proceedings of the ACM Web Conference 2025 (WWW '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3696410.3714665

1 Introduction

Graph Neural Networks (GNNs) have become an indispensable tool for learning and extracting insights from graph-structured data [21, 28, 34], which is prevalent in many real-world domains such as social networks, molecular biology, and financial systems [2, 9, 14, 44]. The core mechanism that drives the success of GNNs is message-passing, where the model iteratively aggregates information from each node's neighbors, producing node representations that encode both local structure and feature information. This capability has led GNNs to excel in various tasks, including node classification, graph classification, and link prediction [15, 22, 37, 38].

Recent research [6, 35, 40] has shown that GNNs are vulnerable to backdoor attacks, which pose significant threats in sensitive applications such as fraud detection, cybersecurity, and healthcare. In backdoor attacks, adversaries implant backdoors by injecting malicious triggers into the target nodes and manipulating the labels of these nodes to a target class. When a GNN is trained on the dataset poisoned with these triggers, it easily learns the association between the trigger and the manipulated class and is named a backdoored GNN. When test samples containing the trigger are presented to the backdoored GNN, the backdoor attack is deemed successful if the backdoored GNN misclassifies the test samples associated with the trigger into the target class.

In backdoor attacks, triggers are typically constructed with the basic elements of data samples. Analogously to how pixel-level visual patterns serve as triggers in computer vision [3, 11] and word-level token-based triggers in NLP [25, 26], subgraphs are natural triggers in graph data, where nodes and edges form the fundamental building blocks [6, 35, 40]. For instance, SBA [40] is a pioneering work that uses random or sampled subgraphs as triggers, though its attack effectiveness is limited. GTA [35] introduces a

[*]Corresponding authors.
[†]Key Laboratory of AI Safety, Chinese Academy of Sciences. Xiang Ao is also at Institute of Intelligent Computing Technology, CAS, Suzhou, China.
[‡]The authors are also with the University of Chinese Academy of Sciences, CAS.

WWW '25, April 28-May 2, 2025, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1274-6/25/04
https://doi.org/10.1145/3696410.3714665

learnable trigger generator, which adapts subgraphs to specific samples, significantly improving attack performance. To evade detection, other approaches such as UGBA [6] and DPGBA [41] integrate regularization terms into their loss functions, allowing them to bypass certain defense mechanisms. Beyond subgraph-based triggers, other methods such as NFTA [4], perturb both node features and graph structures to implant triggers. The implantation of these triggers is typically categorized as local neighborhood manipulation.

While existing graph backdoor attacks have demonstrated strong performance, implanting triggers via local neighborhood manipulation may hinder their stealthiness, i.e., the ability to remain undetected and bypass defense mechanisms. The effectiveness of the local neighborhood manipulation relies on the malicious edges that link the triggers to the clean graph. Unlike the continuous and subtle perturbations typical of attribute manipulation, these edges are discrete and structurally conspicuous, making it considerably more challenging to maintain a high level of stealthiness for the attack. These anomalous edges, akin to a *crack in the backdoor*, provide clear entry points for defense mechanisms to exploit. The results of our empirical study in Table 1 indicate that neighborhood perturbation-based attacks are vulnerable to at least one defense strategy, whether anomalous edge detection or robust downstream classifiers.

To overcome these limitations, designing a backdoor attack that implants triggers without altering the graph's topology presents a promising solution to circumvent the creation of anomalous edges. The challenges of developing such an attack lie in two aspects. First, the budget for node attribute manipulation is limited. Although routine in many attacks, performing all-dimensional feature perturbation or generalization is costly and often infeasible in real-world scenarios where security and privacy are major concerns. For example, in financial systems, where nodes represent participants and edges represent transactions, modifying sensitive attributes such as credit records is impractical and likely to trigger alarms within regulatory systems. Thus, practical constraints demand a smaller attack budget, allowing only perturbations in limited feature dimensions. Second, the manipulated node attributes should be reasonable. With a limited attack budget, minimizing modifications is imperative to maintain the trigger's stealthiness and restraining significant semantic changes.

To tackle these issues, we propose an approach named **SPEAR** (Structure-PresErving grAph backdooR attack). First, to maximize the efficiency of the attack budget, we focus on effectively selecting poisoned nodes by examining the variety and prediction uncertainty of candidate nodes. Second, to ensure that the triggers exert significant influence over the downstream classifier and generalize well to unseen nodes, we implement a global importance-driven feature selection strategy to identify the most impactful feature for trigger implantation. The structure-preserving manipulation is achieved via a trigger generator that leverages neighborhood-aware semantic enrichment. In addition, we integrate a self-similarity regularization term into the loss function to promote minimal modifications. In summary, our main contributions can be summarized as follows:

- We empirically demonstrate the vulnerability of existing backdoor attacks to robust test models and anomalous edge detection, leading to our proposal of a structure-preserving attack named SPEAR.
- In SPEAR, we introduce a novel node selection method to mitigate antagonistic effects, a global importance-driven feature selection method to enhance the effectiveness of trigger implanting, along with a refined trigger generator to leverage neighborhood information.
- Extensive experiments on real-world datasets with various test and defense models demonstrate that SPEAR outperforms state-of-the-art backdoor attacks.

Table 1: Attack success rate (%) of backdoor attacks under different defenses on the dataset OGB-arxiv. The underlined results indicate unsuccessful attacks compared with performance without defense.

Defense	SBA-Gen	GTA	UGBA	DPGBA
None	47.52	74.99	97.39	94.65
OD	12.91	0.00	10.29	92.40
RIGBD	0.00	53.56	0.00	0.00
GNNGuard	40.03	0.94	97.29	91.22

2 Preliminaries

2.1 Notions

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{X})$ represent an attributed graph with N nodes, where $\mathcal{V}$ is the node set, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set, and $\mathcal{X} = \{x_1, \ldots, x_N\}$ is the original feature matrix of nodes, where $x_i \in \mathbb{R}^d$ is the node feature of v_i. The adjacency matrix of the graph $\mathcal{G}$ is denoted by $\mathbf{A} \in \{0,1\}^{N \times N}$, with $\mathbf{A}_{ij} = 1$ indicating an edge between nodes v_i and v_j. The neighborhood of node v_i, including the node itself, is denoted by $\mathcal{N}_i$. In this paper, we focus on an inductive semi-supervised node classification task, where a small set of nodes $\mathcal{V}_L \subseteq \mathcal{V}$ in the training graph $\mathcal{G}$ are provided with labels from $\mathcal{L} = \{1, \ldots, C\}$, and the test graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T, \mathcal{X}_T)$ is not available during the training stage. Let $\mathcal{Y} = \{y_1, \ldots, y_N\}$ denote the ground-truth labels of nodes in the training graph, with $\mathcal{Y}_L$ and $\mathcal{Y}_U$ denoting the ground-truth labels of labeled and unlabeled nodes, respectively.

2.2 Threat Model

In this paper, following prior studies [35, 40, 41], we focus on gray-box backdoor attacks on node classification tasks. In a gray-box scenario, attackers have access to the training data, including node attributes, graph structure, and label information, but lack knowledge of the specific architecture or parameters of the target model. The objective of the backdoor attack is to manipulate the downstream GNN classifier into producing malicious outputs on poisoned samples, while behaving normally on clean ones. To achieve this, attackers poison the training set by implanting triggers into a set of poisoned nodes, then labeling them with the target class. Ideally, this approach ensures that the backdoored GNN associates the trigger with the target label, causing any node containing the trigger to be misclassified as the target class. However, the effectiveness of typical graph backdoor attack lies on introducing malicious

edges which can be detected then eliminated by defenses, making it significantly harder to achieve a high level of stealthiness for the attack.

2.3 Problem Formulation

We consider a standard semi-supervised inductive node classification task in which the goal is to learn a mapping function $f : \mathcal{V} \to \mathcal{L}$, where $\mathcal{V}$ denote nodes in the training graph comprising of labelled nodes $\mathcal{V}_L$ and unlabelled ones $\mathcal{V}_U$, $\mathcal{L}$ denotes the ground-truth labels, and f is a GNN classifier. Following the standard training scheme of graph backdoor attack, we first poison the training set by implanting triggers in target nodes before classifier is trained. We denote $\mathcal{V}_P \subset \mathcal{V}_U$ as the poisoned samples selected from unlabelled nodes in the training graph. For each node $v_i^P \in \mathcal{V}_P$, we transform it into a poisoned sample by implanting trigger in attribute space and alter its ground-truth label as shown in Eq. (1):

$$x_i^P = \text{IMPLANT}(\mathcal{T}_\phi(v_i^P), x_i), \quad y_i^P = y_t, \tag{1}$$

where x_i is the original feature, which is transformed into x_i^P after trigger implanting, $\mathcal{T}_\phi(\cdot)$ denotes the trigger generator that takes v_i as input, and $y_t \in \mathcal{L}$ is the target label assigned by the attacker. Taking the attack budget into account, the transformation also need to satisfy the following inequality:

$$\sum_{k=1}^{d} \mathbb{I}(x_i[k] \neq x_i^P[k]) \leq \Delta_D, \tag{2}$$

where Δ_D is the budget for attribute manipulation, $\mathbb{I}(\cdot)$ is an indicator function, which equals 1 when $x_i[k] \neq x_i^P[k]$. We denote the modified features as $\mathcal{D}$.

Given a GNN classifier f trained on the poisoned training set, ideally, its behavior is manipulated so that:

$$f(x_j, \mathcal{N}_j) = y_j, \quad f(x_j^P, \mathcal{N}_j) = y_t, \tag{3}$$

for node v_j from an unseen test graph $\mathcal{G}_T$ that is sampled from the same data distribution as the training graph. Following the setting of gray-box attack, the architecture and parameters of f are unknown to the attacker. Therefore, we adopt surrogate model f_ω to simulate the downstream classifier. In the empirical risk minimization setting, the objective is to minimize the loss function in Eq. (4) on the training graph:

$$\mathcal{L}_{sur}(\omega, \phi) = \sum_{v_i \in \mathcal{V}_L} l(f_\omega(x_i, \mathcal{N}_i), y_i) + \sum_{v_i \in \mathcal{V}_P} l(f_\omega(x_i^P, \mathcal{N}_i), y_t), \tag{4}$$

where ω denotes the learnable parameter of f_ω, and $l(\cdot)$ is the cross entropy loss. Our goal is to learn the selection of poisoned nodes $\mathcal{V}_P$, the selection of feature dimensions $\mathcal{D}$ for trigger implanting, and the trigger generator $\mathcal{T}_\phi$ by solving a bi-level optimization problem:

$$\begin{aligned} &\min_{\mathcal{V}_P, \mathcal{D}, \mathcal{T}_\phi} \sum_{v_i \in V} l(f_{\omega^*}(x_i^P, \mathcal{N}_i), y_t), \\ &\text{s.t.}(i) \quad \omega^* = \arg\min_{\omega} \mathcal{L}_{sur}(\omega, \phi), \\ &\quad (ii) \quad |\mathcal{V}_P| \leq \Delta_P, \ \mathcal{T}_\phi(v_i) \in \mathcal{U}, \end{aligned} \tag{5}$$

where Δ_P is the budget of poisoned nodes in training set, and $\mathcal{U}$ denotes all triggers that meet the stealthiness requirement. Considering that jointly optimizing $\mathcal{V}_P$, $\mathcal{D}$, and $\mathcal{T}_\phi$ is computationally prohibitive, we break the optimization into two steps. First, we heuristically select poisoned nodes and trigger implanting dimensions as a preprocessing step to approximate the optimal $\mathcal{V}_P$ and $\mathcal{D}$. Then, we fix these selections to optimize $\mathcal{T}_\phi$.

3 Methodology

3.1 Overall Architecture of SPEAR

The overall architecture of our framework revolves around three key components, namely poisoned node selection, feature selection, and trigger generator. Effective poisoned node selection is achieved by assessing both the class variety and prediction uncertainty among unlabelled nodes, focusing on those that meet criteria for both effectiveness and stealthiness. To ensure that the implanted triggers exert an impactful influence on the downstream classifier while maintaining generalizability to unseen data, we employ a global importance-driven feature selection strategy. This method identifies the most critical feature dimensions for trigger implantation by assessing their overall contribution to the model's prediction. Lastly, the trigger generator is designed to perform structure-preserving manipulations, utilizing semantic information from the neighborhood of poisoned nodes. The generator also incorporates a self-similarity normalization term into the loss function, promoting minimal perturbations to enhance stealthiness. The overall framework is illustrated in Fig. 1.

3.2 Effective Poisoned Nodes Selection

Given the extensive and diverse nature of graph data, the selection of poisoned nodes is critical for ensuring efficient use of the attack budget. Accordingly, the selection of poisoned nodes should adhere to two principles: (i) the chosen nodes should effectively deceive the downstream classifier into associating the trigger with the poisoned label; and (ii) manipulations of these nodes should not have adverse effects on the classifier's performance on clean data.

Therefore, we propose selecting samples with high classification uncertainty within each category as poisoned nodes, which provides several notable advantages. First, sampling from different categories provides class variety which is crucial for successfully implanting targeted backdoor triggers across different class distributions. Second, this approach helps to avoid the creation of strong outliers, thereby mitigating the negative impact on the classifier's generalization, ensuring that the clean accuracy is intact. Besides, robust samples with high classification confidence demonstrate a strong semantic correlation between their attributes and true labels, leading to antagonistic effects between the original attributes and implanted triggers [1, 10], making it more complex for the classifier to learn the mapping between triggers and the target label.

Recall that poisoned nodes are selected from the unlabeled training set $\mathcal{V}_U$, in order to obtain the uncertainty and pseudo labels, we train a GNN classifier $\hat{f}_p$ on clean data based on Eq. (6):

$$\mathcal{L}_{pre} = \sum_{v_i \in \mathcal{V}_L} l(\hat{f}_p(x_i, \mathcal{N}_i), y_i). \tag{6}$$

For each node v_i in the training set, its classification uncertainty and pseudo label are calculated as follows:

$$H(v_i) = -\sum_{c \in \mathcal{L}} P_{v_i}(c|\hat{f}_p) \log P_{v_i}(c|\hat{f}_p), \quad \bar{Y}(v_i) = \hat{f}_p(x_i, \mathcal{N}_i), \tag{7}$$

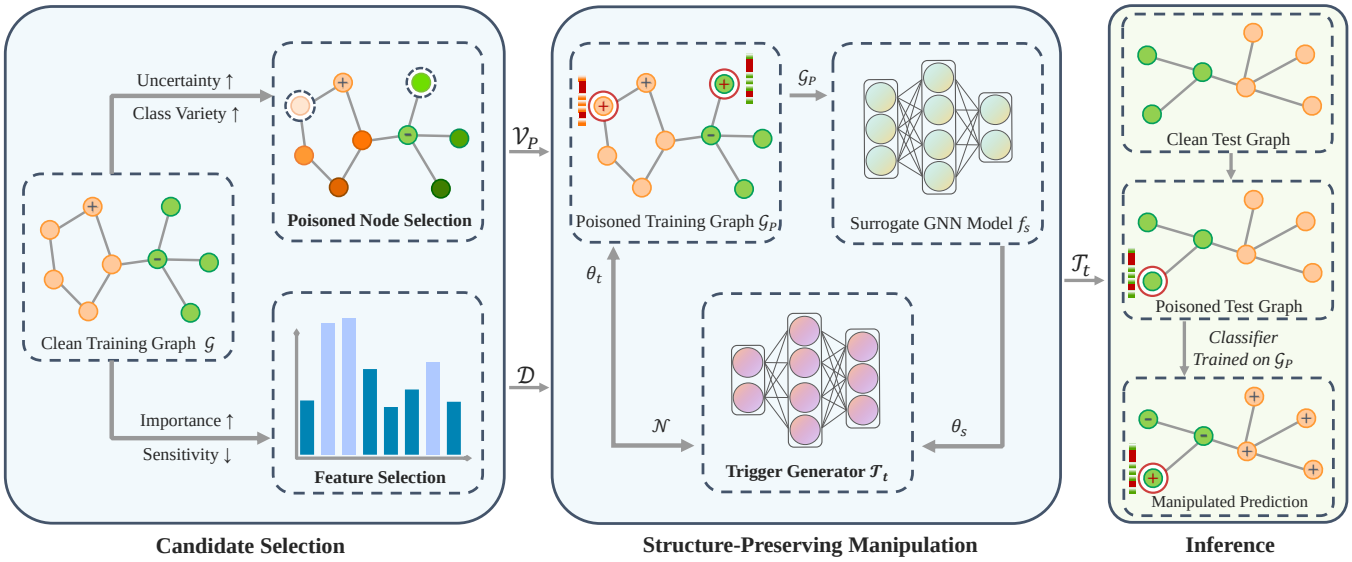


Figure 1: Overall framework of SPEAR. We use '+' and '-' to denote the annotated labels, with shades of orange and green indicating the corresponding ground-truth labels. During the training phase (blue background), SPEAR selects target nodes $\mathcal{V}_P$, identifies high global-importance features $\mathcal{D}$, and optimizes the trigger generator $\mathcal{T}_t$ to produce the poisoned graph $\mathcal{G}_P$. The downstream classifier is trained on $\mathcal{G}_P$ and is embedded with backdoor consequently. In the inference phase (green background), SPEAR implants triggers into the test graph to manipulate the classifier's predictions.

where $\mathcal{L}$ denotes label set, and $P_{v_i}(c|\hat{f}_p)$ indicates the confidence of $\hat{f}_p$ that node v_i belongs to class c. For nodes assigned with pseudo-label c, we rank them in descending order by their classification uncertainty, as defined in Eq. (7), and denote the resulting sequence as T_c. Given the size of poisoned nodes Δ_P, the final selected node set $\mathcal{V}_P$ can be written as:

$$\mathcal{V}_P = \bigcup_{c\in\mathcal{L}} T_c\left[:\left\lfloor\frac{\Delta_P}{C}\right\rfloor\right]. \tag{8}$$

3.3 Global Importance-Driven Feature Selection

As mentioned in Sec. 1, performing all-dimension manipulation is impractical in real-world scenarios, so we set a budget Δ_D to restrict the number of poisoned features, and introduce $\mathcal{F}$ to denote sensitive features that are not allowed to be modified. Considering that the dimension of node features is typically large for graph data, as shown in Table 2, randomly selecting poisoned features exhibits high instability and may waste the attack budget. Therefore, we need to determine which dimensions the triggers should be implanted in. To enhance the influence of the trigger on the prediction result, we propose selecting the most important features, that is, the features that contribute the most to the classifier's prediction.

The challenge lies in selecting an appropriate criterion to measure the importance of features. One straightforward approach is to compute Shapley values for feature combinations that satisfy the budget for each node individually to decide the candidate feature set. However, it is computationally expensive and fails to generalize to test nodes that are unseen during the training phase. Additionally, inconsistent trigger placement across different nodes can confuse the classifier, weakening its ability to associate the trigger with the target label. Hence, inspired by [5], we aggregate information from all labeled nodes to compute the global importance of each feature, and greedily select the top-Δ_D most important features.

Given the complete feature set $\mathcal{D}_a$ and a GNN classifier $\hat{f}_p$ trained on clean graph, we define the global predictive power of a subset $\mathcal{S}$ as $r(\mathcal{S})$:

$$r(\mathcal{S}) = T - \sum_{v_i\in\mathcal{V}_L}\left[l(\hat{f}_p(x_i^{\mathcal{S}}, \mathcal{N}_i^{\mathcal{S}}), y_i)\right], \tag{9}$$

where $\mathcal{S}\subseteq\mathcal{D}_a$, T represents a constant value for the mean prediction loss of $\hat{f}_c$, $x_i^{\mathcal{S}}$ denotes the partial feature of node v_i according to $\mathcal{S}$, and $l(\cdot)$ is the cross entropy loss. For each feature m, we compute its global importance in Eq. (10) by calculating a weighted average of the incremental changes when adding m to subsets $\mathcal{S}$:

$$\phi(m) = \frac{1}{|\mathcal{D}_a|}\sum_{S\subseteq\mathcal{D}_a\setminus\{m\}}\binom{|\mathcal{D}_a|-1}{|\mathcal{S}|}^{-1}\left(r(\mathcal{S}\cup\{m\}) - r(\mathcal{S})\right). \tag{10}$$

After obtaining the global importance of the features, we sort them in descending order and ensure the feature budget and sensitivity constraints are met to obtain the selected feature set $\mathcal{D}$:

$$\mathcal{D} = (\arg\operatorname*{sort}_m \phi(m)\setminus\mathcal{F})[:\Delta_d]. \tag{11}$$

3.4 Neighborhood-Aware Trigger Generator

After deciding on the poisoned nodes $\mathcal{V}_P$ and feature $\mathcal{D}$, we proceed to implant the triggers into the training set. A key challenge in backdoor attacks is generating effective triggers. We employ sample-specific triggers generated by a learnable generator $\mathcal{T}_t$, which is

parameterized by θ_t. The optimization objective of the trigger generator is equivalent to solving a simplified optimization problem in Eq. (5), formulated as:

$$\begin{aligned} &\min_{\theta_t} \sum_{v_i \in \mathcal{V}_L} l(f_{s^*}(\tau_i, \mathcal{N}_i), y_t) \\ &\text{s.t.} \quad \theta_s^* = \arg\min_{\theta_s} \mathcal{L}_{sur}(\theta_s, \theta_t),\ \mathcal{T}_t(v_i) \in \mathcal{U}, \end{aligned} \tag{12}$$

where θ_s is the learnable parameter of surrogate model f_s and τ_i represents the node feature after trigger implanting. For simplification, we denote the loss function in the upper-level objective as $\mathcal{L}_{atk}$, which effectively guides $\mathcal{T}_t$ to produce triggers that can generalize to various nodes in $\mathcal{V}$.

Existing backdoor attacks usually take attributes as input, yet we argue that, in order to boost the trigger's efficacy, the core idea is to leverage neighborhood information from the target nodes, ensuring the trigger is well-adapted to graph-structured data. For each nodes v_i, its neighborhood-aware encoding can be obtained through an aggregation function:

$$h_i = \text{AGGREGATE}^{(l)}(\{x_u : v_u \in \mathcal{N}_i^{(l)}\}), \tag{13}$$

where $\mathcal{N}_i^{(l)}$ denotes the neighbors of node v_i that are at most l hops away. Using h_i as input, the trigger for the node v_i can be defined as $\mathcal{T}_t(h_i)$. Then we can implant the trigger and obtain the poisoned node attribute τ_i:

$$\tau_i = x_i + \mathcal{T}_t(h_i), \tag{14}$$

where $\mathcal{T}_t(h_i)$ is constrained to be zero outside the features defined by $\mathcal{D}$. In our experiments, we observe that the choice of aggregation function has a vital influence on the quality of triggers. Optimal performance is achieved when the aggregation function and the surrogate model share the same architecture and parameters. We believe this effectiveness stems from the surrogate model closely simulating the behavior of a backdoored model, thus ensuring that the generated triggers are well-aligned with a manipulated model's decision-making process.

Even though limiting the number of perturbed features enhances the trigger's stealthiness, it still remains essential to further constrain the magnitude of these perturbations to prevent substantial semantic shifts. Thus, we propose a self-similarity loss function, defined as:

$$\mathcal{L}_{sim}(\theta_t) = \sum_{v_i \in \mathcal{V}_P} -\log\left(1 + \alpha\left(1 - \frac{x_i \cdot \tau_i}{\|x_i\|_2 \|\tau_i\|_2}\right)\right), \tag{15}$$

where α controls the contribution of the self-similarity loss.

Combining Eq. (12) and Eq. (15), the final objective can be formulated as follows:

$$\begin{aligned} &\min_{\theta_t} \mathcal{L}_{atk}(\theta_s^*, \theta_t) + \mathcal{L}_{sim}(\theta_t) \\ &\text{s.t.} \quad \theta_s^* = \arg\min_{\theta_s} \mathcal{L}_{sur}(\theta_s, \theta_t). \end{aligned} \tag{16}$$

Optimizing Eq. (16) directly can be computationally expensive, so we solve it with an alternative optimization schema as [46] does. In inner-level optimization, we update θ_s for E iterations on $\mathcal{G}_P$ to accelerate the training process:

$$\theta_s^{(e+1)} = \theta_s^{(e)} - \gamma_s \nabla_{\theta_s} \mathcal{L}_{sur}(\theta_s^{(e)}, \theta_t), \tag{17}$$

Table 2: Dataset statistics

Datasets	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,443	7
Pubmed	19,717	44,338	500	3
OGB-arxiv	169,343	1,166,243	128	40

where $\theta_s^{(e)}$ is the learnable parameter of surrogate model f_s in the e-th iteration, and γ_s is the corresponding learning rate. In upper-level optimization, we fix θ_s and update θ_t using a first-order approximation:

$$\theta_t^{(k+1)} = \theta_t^{(k)} - \gamma_t \nabla_{\theta_t}\left(\mathcal{L}_{atk}(\theta_s^*, \theta_t^{(k)}) + \mathcal{L}_{sim}(\theta_t^{(k)})\right), \tag{18}$$

where $\theta_t^{(k)}$ is the learnable parameter of the trigger generator $\mathcal{T}_t$ in the k-th iteration, γ_t is the learning rate of $\mathcal{T}_t$, and θ_s^* is he latest value of θ_s obtained from the lower-level optimization. Once θ_t is obtained, we use $\mathcal{T}_t$ to update the poisoned graph $\mathcal{G}_P$. The detailed training algorithm can be found in Algorithm 1.

4 Experiments

In this section, we empirically analyze the effectiveness and stealthiness of SPEAR on various datasets. Specifically, we aim to answer the following research questions:

- **RQ1:** Does SPEAR outperform the state-of-the-art backdoor models under various defenses?
- **RQ2:** What is the impact of different attack budgets on SPEAR's performance?
- **RQ3:** How do the key components contribute to the attack performance?
- **RQ4:** How does SPEAR balance training time and performance?

4.1 Experimental Settings

4.1.1 Datasets. To evaluate the effectiveness of our proposed method, we conduct experiments on three widely used datasets, i.e., Cora, Pubmed [27], and OGB-arxiv [13], which correspond to small, medium, and large graphs, respectively. The detailed statistics of these datasets are presented in Table 2.

4.1.2 Baselines. We compare SPEAR against five representative and state-of-the-art graph backdoor attack methods, namely **SBA-Samp**, **SBA-Gen** [40], **GTA** [35], **UGBA** [6], and **DPGBA** [41]. To assess the stealthiness of **SPEAR**, we implement three graph backdoor defenses: **Prune** [6], **OD** [41], and **RIGBD** [42]. Additionally, to validate its stealthiness and transferability, we conduct tests across various models on OGB-arxiv, including prominent GNN architectures such as **GCN** [15], **GraphSAGE** [12], and **GAT** [31], as well as robust GNNs like **GNNGuard** [39] and **RobustGCN** [43]. Comprehensive details regarding these methods can be found in Appendix B.

4.1.3 Evaluation. Following [6, 42], we perform experiments on the inductive node classification task, where the test graph is unseen by both the attacker and the victim model before inference. To evaluate effectiveness and evasiveness, we use two metrics: (i)

Table 3: Backdoor attack results (ASR (%) | CA (%)) under different defenses. For clean graphs (Clean), only clean accuracy is reported. The top two performances in terms of ASR are highlighted in bold and underline.

Datasets	Defense	Clean	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA	SPEAR
Cora	None	83.49	29.52 | 82.49	44.65 | 82.96	91.79 | 84.07	96.05 | 83.70	95.67 | 82.96	**98.85 | 83.96**
	Prune	81.48	16.70 | 82.98	19.56 | 83.19	0.06 | 84.04	97.41 | 82.82	16.97 | 78.52	**97.78 | 82.66**
	OD	83.82	34.70 | 83.00	44.10 | 83.60	44.00 | 84.81	0.00 | 83.60	94.78 | 84.08	**96.31 | 84.07**
	RIGBD	83.41	11.56 | 83.04	0.03 | 82.75	0.00 | 83.70	29.33 | 83.33	0.01 | 85.19	**91.56 | 83.33**
Pubmed	None	84.93	26.47 | 85.13	29.16 | 85.03	92.49 | 85.11	93.17 | 85.44	**95.64 | 83.97**	95.03 | 85.08
	Prune	84.31	22.26 | 85.08	20.99 | 85.08	23.36 | 85.08	91.01 | 84.35	64.25 | 85.29	**96.70 | 85.24**
	OD	85.27	21.05 | 85.29	28.45 | 85.44	85.39 | 85.34	20.84 | 84.53	91.99 | 84.37	**92.04 | 85.64**
	RIGBD	84.86	0.00 | 84.35	0.00 | 84.71	0.01 | 84.32	0.01 | 85.13	0.01 | 84.32	**95.33 | 84.78**
OGB-arxiv	None	65.60	17.70 | 65.19	47.52 | 65.03	74.99 | 63.22	**97.39 | 65.65**	94.65 | 64.56	96.95 | 66.91
	Prune	63.22	0.04 | 63.64	0.02 | 63.53	0.00 | 63.32	94.81 | 63.49	0.00 | 63.27	**98.66 | 64.83**
	OD	65.71	12.91 | 64.18	40.03 | 64.22	0.00 | 64.73	10.29 | 65.03	**92.84 | 65.12**	88.92 | 66.17
	RIGBD	65.53	0.00 | 64.08	0.00 | 63.97	53.56 | 64.03	0.00 | 65.21	0.00 | 65.24	**96.20 | 65.96**

Table 4: Backdoor attack results (ASR (%) | CA (%)) on different test models using the OGB-arxiv dataset. The top two performances in terms of ASR are highlighted in bold and underline.

Test Models	Clean	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA	SPEAR
GCN	65.60	17.70 | 65.19	47.52 | 65.03	74.99 | 63.22	**97.39 | 65.65**	94.65 | 64.56	96.95 | 66.91
GAT	66.25	49.75 | 65.06	94.86 | 65.01	1.72 | 63.00	96.53 | 65.08	95.88 | 64.79	**96.86 | 64.56**
GraphSAGE	65.86	21.64 | 65.47	40.49 | 65.44	96.67 | 65.20	96.61 | 65.20	78.30 | 65.33	**97.93 | 65.50**
GNNGuard	66.03	22.59 | 64.66	39.76 | 64.60	0.94 | 65.14	97.29 | 65.64	91.22 | 63.94	**97.64 | 65.65**
RobustGCN	61.36	56.50 | 62.03	55.03 | 64.02	86.86 | 61.01	94.93 | 61.19	89.90 | 61.08	**95.99 | 61.37**

attack success rate (ASR), which measures the likelihood that the backdoored GNN classifies trigger-implanted nodes into the target class y_t; and (ii) clean accuracy (CA), which measures the classification accuracy of the backdoored GNN on clean nodes. We randomly mask out 20% of the nodes in each dataset, with half designated as target nodes for assessing attack performance and the other half as clean test nodes to evaluate performance on clean data. The remaining 80% of nodes form the training graph, where 10% of the nodes are used for labeled training and another 10% for validation.

4.1.4 Implementation. For SPEAR, the rate of poisoned nodes in training data is set to less than 1% for each dataset, and the feature budget $\Delta_D = \max(0.02d, 5)$, where d is the dimension of node attribute. For baseline attack methods, we adopt the same rate of poisoned nodes, and set the trigger size to 3 with all-dimensional manipulable generated nodes. A two-layer MLP is deployed as trigger generator. As for the surrogate model, a two-layer GCN is used for all attack methods. Each experiment is run five times per architecture, and the average ASR and CA are reported. More details can be found in Appendix C.

4.2 Attack Performance

To answer RQ1, we evaluate SPEAR against baseline backdoor attacks across three datasets with or without defenses. To further assess the transferability and stealthiness, we vary the test models from prominent GNN architectures to robust GNNs.

4.2.1 Comparison with Baseline Attacks. Table 3 shows the performance on three datasets against baseline attacks. The top two performances are highlighted in bold and underline. From this experiment, we have the following observations:

- All baseline attacks fail to bypass at least one defense across all datasets, revealing their vulnerabilities. Specifically, SBA-Samp and its variant consistently exhibit lower ASRs, while GTA fails under Prune and OD defenses in most cases. UGBA and DPGBA benefit from specific countermeasures, showing resistance to Prune and OD, respectively, but remain ineffective against other defenses. In contrast, SPEAR shows significantly smaller ASR drops across all defenses, highlighting its superior stealthiness.
- When no defense is applied, SPEAR consistently achieves comparable or better ASR compared with baselines across all datasets, demonstrating its effectiveness. This consistent performance highlights SPEAR's ability to implant effective and impactful triggers without producing anomalous edges.
- SPEAR maintains CA levels similar to the baselines, and often higher than those on the clean graph, both with and without defense methods. This shows that the SPEAR-backed downstream classifier preserves its classification ability in clean samples, further highlighting the stealthiness of SPEAR.

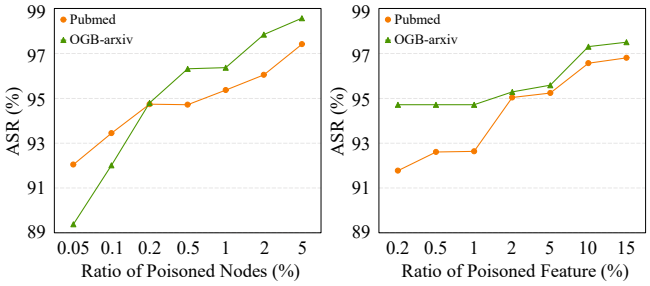


Figure 2: Impact of size of attack budget.

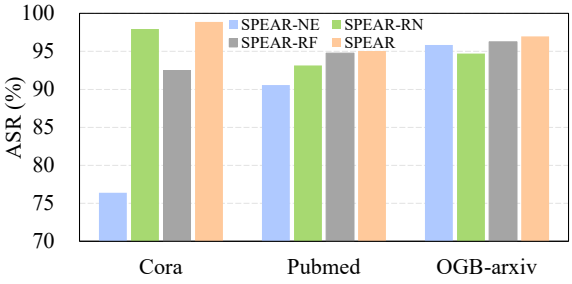


Figure 3: Comparisons between SPEAR and its variants.

- We note that SPEAR's ASR on OGB-arxiv against OD was slightly lower than that of DPGBA by about 4%. This discrepancy can be attributed to DPGBA's distribution-preserving module, which is tailored for out-of-distribution detectors, enhancing its ability to bypass OD. However, this advantage is less effective against other defense mechanisms. In contrast, SPEAR focuses on overall stealthiness, enabling it to demonstrate resilience against a broader range of defenses.

4.2.2 Performance with Different Test Models. Table 4 shows the performance of SPEAR and baseline models with different test models on OGB-arxiv, including three mainstream GNN models (GCN, GAT and GraphSAGE) and two robust GNN models (GNNGuard and RobustGCN), to demonstrate its transferability and stealthiness. From the table, we make the following observations:

- For the two robust GNN models, GNNGuard calculates edge pruning probabilities through a non-linear transformation, while RobustGCN uses Gaussian distributions to represent nodes and employs a variance-based attention mechanism. Results demonstrate that SPEAR is resistant to both GNNGuard and RobustGCN, showcasing its ability to bypass robust GNNs that focus on either graph structure or node attributes.
- SPEAR maintains leading ASR across all test models compared to baseline attacks. Given that we fix the surrogate model to be a 2-layer GCN while varying the test models, such performance demonstrates SPEAR's ability to generate triggers that effectively adapt to different GNN architectures.

4.3 Impact of Attack Budget

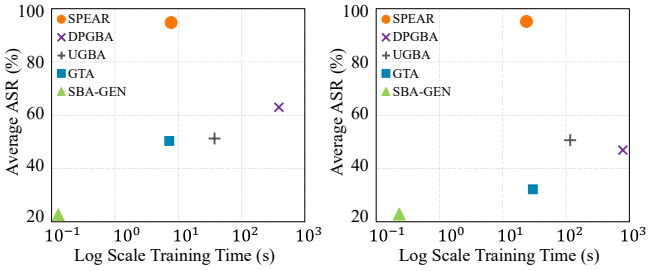
To answer RQ2, we conduct experiments to evaluate the sensitivity of SPEAR under different attack budgets. Specifically, for $\mathcal{V}_P$, we vary the ratio of poisoned nodes in the training graph across $\{0.05, 0.1, 0.2, 0.5, 1, 2, 5\}$%, and for $\mathcal{D}$, we adjust the ratio of manipulated features to $\{0.2, 0.5, 1, 2, 5, 10, 15\}$%, with at least one feature manipulated. Fig. 2 shows the results on Pubmed and OGB-arxiv. We only report attack success rate, as no notable changes in clean accuracy were detected across all baselines and SPEAR. From Fig. 2 we can observe that:

- As the ratio of poisoned nodes grows, SPEAR's ASR shows a consistent increase on both datasets, highlighting that it can effectively leverage a larger attack budget to achieve better results. Meanwhile, SPEAR maintains an ASR above 89% even with a poisoned node ratio as low as 0.05%, showcasing its efficient use of the attack budget.
- When varying the ratio of features budget $\mathcal{D}$, similar trends are observed. The ASR increases steadily as the proportion of perturbed features grows, indicating that SPEAR benefits from manipulating more features. Even with a feature budget of just 1%, meaning that only 1 out of 128 features is manipulable in OGB-arxiv, the attack remains highly effective. This confirms that SPEAR can still be applied in real-world environments where feature sensitivity is a critical concern.

4.4 Ablation Study

To answer RQ3, we conduct an ablation study to examine the contributions of the key components in SPEAR: the node selection module, the feature selection module, and the neighborhood-aware mechanism in trigger generation. To evaluate the effectiveness of node selection, we design a variant of SPEAR named SPEAR-RN, which randomly selects poisoned nodes in the training graph. To assess the contribution of the global importance-driven feature selection strategy, we replace the original feature selection module with random selection, naming it SPEAR-RF. We also introduce a variant called SPEAR-NE, where an MLP is used to substitute the aggregation function in Eq. (13) to evaluate the role of the neighborhood-aware mechanism in trigger generation. Hyperparameters for each variant are tuned based on validation set performance. Fig. 3 illustrates the results, showing that:

- We observe a notable drop in ASR for SPEAR-NE compared to SPEAR, with reductions of 22.47% on Cora, 4.48% on Pubmed, and 1.14% on OGB-arxiv. This result highlights the crucial role of the neighborhood-aware mechanism in SPEAR's effectiveness. We infer that the larger benefit observed on Cora arises from its simple local structure and high homophily, which allow the aggregation function to extract more relevant information and thus enhancing trigger generation.
- SPEAR achieves higher ASR than its variants, SPEAR-RN and SPEAR-RF, affirming the contributions of both the node selection and feature selection modules. We notice that the advantage of the global importance-driven feature selection method becomes more pronounced as the feature size increases. This suggests that the method is highly effective at identifying influential features from large candidate sets, a critical capability for datasets with a large number of features.



(a) Pubmed (b) OGB-arxiv

Figure 4: Training time of trigger generator vs. performance

4.5 Trade-off between Time and Performance

To answer RQ4, we conduct experiments examining the trade-off between the training time of the trigger generator and the average ASR on Pubmed and OGB-arxiv. We compute the average ASR both with and without defenses, following the setup in Sec. 4.2.1. For the learnable trigger generators in SPEAR, DPGBA, UGBA, and GTA, the same number of training epochs is used. On OGB-arxiv, we adopt the simplified version of DPGBA, which is designed to handle large-scale graphs. The experiments are performed on an NVIDIA GeForce RTX 4090 with 24GB of memory, and the results are shown in Fig. 4. Compared to other attacks with similar or shorter training time, SPEAR significantly outperforms SBA-Gen and GTA in terms of average ASR. In contrast, DPGBA and UGBA require considerably more time for training the trigger generator, e.g., DPGBA takes 391.98 seconds on Pubmed, making them 50.51× and 4.84× slower than SPEAR, respectively. Overall, SPEAR strikes a superior balance between attack success rate and training efficiency, making it a more practical choice for real-world applications.

5 Related Works

5.1 Adversarial Attacks against GNNs

GNNs are indispensable tools for complex graph-structured data analysis [19, 23, 31], excelling in tasks like node classification and link prediction [34, 37]. Foundational models such as Graph Convolutional Networks (GCN) [15] and GraphSAGE [12] offer scalable solutions, driving the adoption of GNNs across fields, including fraud detection, biological systems, and recommendation systems [2, 8, 14, 33]. However, GNNs remain vulnerable to adversarial attacks [7, 16, 17, 45].

Adversarial attacks exploit inherent vulnerabilities of GNN models, leading to either reduced accuracy or manipulated predictions [18, 29, 46]. Broadly classified into evasion and poisoning attacks, these methods undermine GNNs at different stages. Evasion attacks occur during the testing phase, where adversaries perturb the graph structure or node attributes of a trained model [30, 32, 36]. Poisoning attacks, on the other hand, target the training phase, where attackers tamper with training data to mislead the model [20, 24, 46].

As a specific form of poisoning attack, backdoor attacks implant triggers that activate malicious behavior only under certain conditions, making them particularly hard to detect and defend against [35, 40]. This growing threat underscores the urgent need to address security concerns in GNNs.

5.2 Backdoor Attacks and Defenses on GNNs

Backdoor attacks implant malicious triggers within training data, causing the model to function normally under standard conditions but misbehave with trigger-implanted samples. Early methods like [40] used subgraph-based backdoors with universal triggers. More recent techniques, such as GTA [35], generate adaptive triggers tailored to individual samples. To improve stealthiness, methods like UGBA [6] and DPGBA [41] use regularization terms to evade anomalous edge detection. NFTA [4] manipulates both node features and graph structure, but its reliance on binary features limits adaptability. Our method differs by: (i) manipulating the attribute space to avoid anomalous edges, and (ii) using a novel candidate selection method that adapts to real-world constraints.

To counteract backdoor attacks, several defenses have been proposed. Among these, anomalous edge detection stands out as a promising approach. Prune [6] aims to mitigate the effect of triggers by identifying and removing edges that deviate from the homophily assumption. Another approach, OD [41], identifies out-of-distribution edges by filtering those with the highest reconstruction errors. By analyzing prediction variance after removing anomalous edges, RIGBD [42] was developed to identify compromised target nodes and mitigate their impact on downstream tasks. Furthermore, robust GNN models such as GNNGuard [39] and RobustGCN [43] have demonstrated resistance to backdoor attacks. Our proposed method is capable of bypassing such defenses as we avoid introducing anomalous edges through structure-preserving manipulation and constrain the magnitude of manipulation to prevent substantial semantic shifts.

6 Conclusion

We propose SPEAR, a structure-preserving backdoor attack that exploits GNN vulnerabilities while minimizing detection risks. Unlike traditional attacks that alter graph structure, SPEAR preserves the graph's topology and focuses on perturbing node attributes. By employing novel selection methods and neighborhood-aware trigger generation, SPEAR achieves a balance between attack success rate and stealthiness, making it ideal for sensitive graph-based applications. Extensive experiments on real-world datasets show that SPEAR outperforms existing methods in both effectiveness and stealth, even against robust defenses. This work underscores the growing threat of stealthy, structure-preserving backdoor attacks as GNNs are deployed in security-sensitive domains. Future research may explore optimizations in trigger generation and defenses specifically targeting attribute-based backdoor attacks.

Acknowledgments

The research work is supported by National Key R&D Plan No.2022YFC3303302, the National Natural Science Foundation of China under Grant (No. U2436209, 62476263, 62406307). Xiang Ao is also supported by the Beijing Nova Program 20230484430, the Innovation Funding of ICT, CAS under Grant No. E461060. Yang Liu is also supported by the Postdoctoral Fellowship Program of CPSF under Grant Number GZB20240761.

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. 2022. Feature purification: How adversarial training performs robust deep learning. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 977–988.

[2] Pietro Bongini, Monica Bianchini, and Franco Scarselli. 2021. Molecular generative graph neural networks for drug discovery. *Neurocomputing* 450 (2021), 242–252.

[3] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[4] Yang Chen, Zhonglin Ye, Haixing Zhao, and Ying Wang. 2023. Feature-Based Graph Backdoor Attack in the Node Classification Task. *International Journal of Intelligent Systems* 2023, 1 (2023), 5418398.

[5] Ian Covert, Scott M Lundberg, and Su-In Lee. 2020. Understanding global feature contributions with additive importance measures. *Advances in Neural Information Processing Systems* 33 (2020), 17212–17223.

[6] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. 2023. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of the ACM Web Conference 2023*. 2263–2273.

[7] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. 2018. Adversarial attack on graph structured data. In *International conference on machine learning*. PMLR, 1115–1124.

[8] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[9] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. 2020. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2033–2047.

[10] Yinghua Gao, Yiming Li, Linghui Zhu, Dongxian Wu, Yong Jiang, and Shu-Tao Xia. 2023. Not all samples are born equal: Towards effective clean-label backdoor attacks. *Pattern Recognition* 139 (2023), 109512.

[11] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. Badnets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244.

[12] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[13] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. 2020. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems* 33 (2020), 22118–22133.

[14] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM web conference 2022*. 1311–1321.

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[16] Kuan Li, YiWen Chen, Yang Liu, Jin Wang, Qing He, Minhao Cheng, and Xiang Ao. 2023. Boosting the Adversarial Robustness of Graph Neural Networks: An OOD Perspective. In *The Twelfth International Conference on Learning Representations*.

[17] Kuan Li, Yang Liu, Xiang Ao, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2022. Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 925–935.

[18] Kuan Li, Yang Liu, Xiang Ao, and Qing He. 2023. Revisiting graph adversarial attack and defense from a data distribution perspective. In *The Eleventh International Conference on Learning Representations*.

[19] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated Graph Sequence Neural Networks. In *4th International Conference on Learning Representations, ICLR 2016, May 2-4, 2016, Conference Track Proceedings*.

[20] Xixun Lin, Chuan Zhou, Jia Wu, Hong Yang, Haibo Wang, Yanan Cao, and Bin Wang. 2023. Exploratory adversarial attacks on graph neural networks for semi-supervised node classification. *Pattern Recognition* 133 (2023), 109042.

[21] Yang Liu, Xiang Ao, Linfeng Dong, Chao Zhang, Jin Wang, and Qing He. 2020. Spatiotemporal activity modeling via hierarchical cross-modal embedding. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (2020), 462–474.

[22] Yang Liu, Xiang Ao, Fuli Feng, and Qing He. 2022. Ud-gnn: Uncertainty-aware debiased training on semi-homophilous graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1131–1140.

[23] Yang Liu, Xiang Ao, Fuli Feng, Yunshan Ma, Kuan Li, Tat-Seng Chua, and Qing He. 2023. FLOOD: A flexible invariant learning framework for out-of-distribution generalization on graphs. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1548–1558.

[24] Zihan Liu, Yun Luo, Lirong Wu, Zicheng Liu, and Stan Z. Li. 2024. Towards reasonable budget allocation in untargeted graph structure attacks via gradient debias *(NIPS '22)*. Red Hook, NY, USA, Article 2028, 12 pages.

[25] Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2020. Onion: A simple and effective defense against textual backdoor attacks. *arXiv preprint arXiv:2011.10369* (2020).

[26] Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. *arXiv preprint arXiv:2105.12400* (2021).

[27] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[28] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30, 3 (2013), 83–98.

[29] Lichao Sun, Yingtong Dou, Carl Yang, Kai Zhang, Ji Wang, S Yu Philip, Lifang He, and Bo Li. 2022. Adversarial attack and defense on graph data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 35, 8 (2022), 7693–7711.

[30] Shuchang Tao, Qi Cao, Huawei Shen, Junjie Huang, Yunfan Wu, and Xueqi Cheng. 2021. Single node injection attack against graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 1794–1803.

[31] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[32] Binghui Wang, Minhua Lin, Tianxiang Zhou, Pan Zhou, Ang Li, Meng Pang, Hai Li, and Yiran Chen. 2024. Efficient, direct, and restricted black-box graph evasion attacks to any-layer graph neural networks via influence function. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 693–701.

[33] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[35] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*. 1523–1540.

[36] He Zhang, Bang Wu, Xiangwen Yang, Chuan Zhou, Shuo Wang, Xingliang Yuan, and Shirui Pan. 2021. Projective ranking: A transferable evasion attack method on graph neural networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3617–3621.

[37] Muhan Zhang and Yixin Chen. 2018. Link Prediction Based on Graph Neural Networks. In *Advances in Neural Information Processing Systems*, Vol. 31.

[38] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[39] Xiang Zhang and Marinka Zitnik. 2020. Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in neural information processing systems* 33 (2020), 9263–9275.

[40] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2021. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*. 15–26.

[41] Zhiwei Zhang, Minhua Lin, Enyan Dai, and Suhang Wang. 2024. Rethinking graph backdoor attacks: A distribution-preserving perspective. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4386–4397.

[42] Zhiwei Zhang, Minhua Lin, Junjie Xu, Zongyu Wu, Enyan Dai, and Suhang Wang. 2024. Robustness-Inspired Defense Against Backdoor Attacks on Graph Neural Networks. *arXiv preprint arXiv:2406.09836* (2024).

[43] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 1399–1407.

[44] Xiaoqian Zhu, Xiang Ao, Zidi Qin, Yanpeng Chang, Yang Liu, Qing He, and Jianping Li. 2021. Intelligent financial fraud detection practices in post-pandemic era. *The Innovation* 2, 4 (2021).

[45] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. 2018. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2847–2856.

[46] Daniel Zügner, Oliver Borchert, Amir Akbarnejad, and Stephan Günnemann. 2020. Adversarial attacks on graph neural networks: Perturbations and their patterns. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 14, 5 (2020), 1–31.

A Algorithm

The overall algorithm of SPEAR is outlined in Algorithm 1. The process begins with parameter initialization in line 1. From lines 2 to 7, we perform candidate selection. Specifically, in lines 2-3, we pretrain a clean GCN model, f_p, to identify the poisoned node set, $\mathcal{V}_p$, as specified by Eq. (8). Once the poisoned nodes are identified, we modify their labels to y_t. Line 7 determines which features will be triggered. To optimize the trigger generator and the poisoned graph, we employ bi-level optimization from lines 8 to 13, updating the surrogate model, f_s, the trigger generator, f_t, and the poisoned graph, $\mathcal{G}_P$.

Algorithm 1 Training algorithm of SPEAR

Input: $\mathcal{G}$ – training graph; y_t – target label; Δ_P, Δ_D – attack budget; α, E – hyperparameter.
Output: $\mathcal{G}_P$ – poisoned graph; θ_t – parameters of trigger generator.

1: Initialize θ_p, θ_t and θ_s; Set $\mathcal{G}_P = \mathcal{G}$;
2: **while** *not converged* **do**
3: Update θ_p according to Eq. (6);
4: **end while**
5: Select poisoned nodes $\mathcal{V}_P$ according to Eq. (8);
6: Alter labels of nodes in $\mathcal{V}_P$ to y_t;
7: Select implanting dimension $\mathcal{D}$ according to Eq. (11);
8: **while** *not converged* **do**
9: **for** $e = 1, 2, \ldots, E$ **do**
10: Update θ_s according to Eq. (17);
11: **end for**
12: Update θ_t according to Eq. (18);
13: Update $\mathcal{G}_P$ according to θ_t;
14: **end while**
15: **return** $\mathcal{G}_P$ and θ_t;

B Baselines Information

- **SBA-Samp & SBA-Gen** [40]: Two methods inject fixed or generated subgraph triggers. These methods face challenges in high ASR and are susceptible to defense mechanisms such as pruning.
- **GTA** [35]: GTA introduces a trigger generator that tailors triggers to specific samples, improving attack effectiveness but vulnerable to several defense mechanisms.
- **UGBA** [6]: UGBA selects poisoned nodes based on representativeness and generates triggers that obey the homophily assumption, maintaining high stealth under a limited budget.
- **DPGBA** [41]: DPGBA improves by ensuring that triggers remain within distribution through adversarial learning, further reducing detectability.
- **Prune** [6]: Prune removes dissimilar edges to disrupt the effectiveness of backdoor by filtering out suspicious connections.
- **OD** [41]: OD leverages autoencoders for outlier detection, identifying and removing out-of-distribution edges to mitigate backdoor attacks.
- **RIGBD** [42]: RIGBD detects poisoned nodes by leveraging random edge dropping and the prediction variance, offering strong defense while maintaining high clean accuracy.

Table 5: Hyperarameter settings for different models.

Model	α	E	Δ_P
GCN	0.1	1	0.5
GAT	0.1	1	0.5
GraphSAGE	0.1	1	0.5
GNNGuard	1	2	1
RobustGCN	5	2	1
Prune	1/1/5	2	0.5
OD	5/1/15	3	1
RIGBA	1/1/5	3	1

Table 6: Hyperparameters settings for different datasets.

Dataset	N_h	γ
Cora	80	0.001
Pubmed	64	0.01
OGB-arxiv	32	0.01

- **GCN** [15]: A standard GNN widely used for node classification, serving as a benchmark for graph-based tasks.
- **GAT** [31]: GAT assigns different attention weights to neighboring nodes, enabling more expressive and flexibility without requiring prior knowledge of the global graph structure.
- **GraphSAGE** [12]: GraphSAGE generates inductive node embeddings by sampling and aggregating information from local node neighborhoods.
- **RobustGCN** [43]: RobustGCN models nodes with Gaussian distributions and employs a variance-based attention mechanism to reduce the spread of adversarial attacks through the network.
- **GNNGuard** [39]: GNNGuard dynamically reweights edges based on cosine similarity, pruning adversarial connections and improving the model's robustness against attacks.

C Detailed Implementation

C.1 Hyperparameters

All hyperparameters are tuned based on the loss and accuracy of the validation set. For SPEAR, the parameter α, which controls the contribution of the self-similarity loss, is tuned from {0.1, 1, 3, 5, …, 15}, and E, the number of repeated iterations for inner-optimization, is tuned from {1, 3, 5, 7, 9, 11}. A 2-layer GCN is deployed as $\hat{f}_p$, and a 2-layer MLP is used as the trigger generator, both with hidden dimensions set to 32. Another 2-layer GCN is deployed as the surrogate model, which also acts as the aggregator providing input for the trigger generator. Its hidden dimension N_h, which determines the level of information compression in the input to the generator, is tuned from {16, 32, 64, 80, 128}. The learning rate γ for the surrogate and trigger generator is tuned from {0.0001, 0.001, 0.005, 0.01}. The ratio of poisoned nodes Δ_P in training set is tuned from {0.5, 1}% for experiments in Sec. 4.2.1, with all baseline attack

Table 7: Performance (ASR (%) | CA (%)) comparison in incomplete attribute scenarios. Better performance is underlined.

Dataset	Attack	25%	50%	75%
Cora	SPEAR	95.20 | 83.33	100.0 | 82.22	100.0 | 61.85
	UGBA	99.33 | 81.85	99.63 | 81.48	100.0 | 60.37
Pubmed	SPEAR	100.0 | 84.12	100.0 | 84.56	100.0 | 81.13
	UGBA	97.57 | 84.42	97.41 | 83.97	98.07 | 81.94
OGB-arxiv	SPEAR	97.16 | 66.20	97.61 | 64.24	99.86 | 59.74
	UGBA	96.33 | 66.07	97.85 | 64.23	98.49 | 59.60

methods using the same poisoning rate as SPEAR to guarantee a fair comparison.

The specific choice of hyperparameters is listed in Table 5 and 6. In Table 5, the values separated by '/' correspond to the parameter settings for Cora, Pubmed, and OGB-arxiv, respectively.

C.2 Computing infrastructures

We implement the proposed methods with Numpy 1.26.0, PyTorch 2.2.1 and PyTorch Geometric 2.5.1. We conduct the experiments on a Linux server with an Intel Xeon E5-2680 v4 CPU and a NVIDIA GeForce RTX 4090 GPU with 24GB memory.

D Adaptability to Limited Attributes

D.1 Incomplete Attributes

In this scenario, we randomly mask out 25%, 50%, or 75% of the original attributes. The baseline attack method, UGBA, manipulates all unmasked attributes, while the proposed method, SPEAR, only perturbs a small fraction of the attributes, with a maximum limit of $max(0.02d, 5)$, where d is the total number of attributes. The results of these experiments are shown in Table 7.

In this setting, we observe that as the percentage of masked attributes increases, the ASR of SPEAR improves, while the Clean Accuracy (CA) declines. This is expected, as the higher the fraction of poisoned attributes, the more susceptible the predictions become to manipulation. Both SPEAR and UGBA show strong adaptability, but SPEAR generally achieves higher ASR in most cases.

D.2 Partially Unmodifiable Attributes

A portion of the attributes are designated as unmodifiable, and the attacker can only manipulate the remaining modifiable attributes. We experiment with 50%, 75%, and 95% of attributes being unmodifiable. The results of these experiments are summarized in Table 8.

The results show that when the number of unmodifiable attributes increases, SPEAR exhibits significantly stronger adaptability compared to UGBA. UGBA's performance deteriorates, especially on datasets with smaller feature sizes (e.g., OGB-arxiv), while SPEAR maintains consistently high ASR across different unmodifiable attribute rates. This demonstrates the superior efficiency of SPEAR in utilizing the available attack budget and its robustness under constrained attribute conditions.

Table 8: Performance (ASR (%) | CA (%)) comparison with partially unmodifiable attributes. Better performance is underlined.

Dataset	Attack	50%	75%	95%
Cora	SPEAR	98.52 | 83.70	98.54 | 83.52	95.28 | 83.92
	UGBA	96.22 | 83.33	96.42 | 83.96	91.51 | 84.07
Pubmed	SPEAR	96.54 | 85.02	96.14 | 84.98	95.56 | 84.79
	UGBA	93.48 | 84.93	90.06 | 84.56	72.46 | 84.98
OGB-arxiv	SPEAR	97.12 | 66.03	96.28 | 66.45	95.20 | 66.24
	UGBA	97.07 | 66.94	87.02 | 66.89	69.55 | 66.89

Table 9: Performance (ASR (%) | CA (%)) comparison with different surrogate models.

Dataset	Defense	Sur_GCN	Sur_GAT	Sur_SAGE
Cora	None	98.85 | 83.96	96.31 | 82.96	98.26 | 84.07
	Prune	97.78 | 82.66	97.42 | 74.07	95.57 | 78.15
	OD	96.31 | 84.07	95.20 | 79.63	98.15 | 82.96
	RIGBD	91.56 | 93.33	92.05 | 83.70	93.94 | 84.81
Pubmed	None	95.03 | 85.08	94.57 | 84.93	97.95 | 84.93
	Prune	96.70 | 85.24	98.02 | 85.34	98.80 | 84.87
	OD	92.04 | 85.64	92.34 | 85.29	91.23 | 85.29
	RIGBD	95.33 | 84.78	95.84 | 85.24	94.24 | 84.75
OGB-arxiv	None	96.95 | 66.91	OOM	92.99 | 66.81
	Prune	98.66 | 64.83	OOM	89.61 | 66.92
	OD	88.92 | 66.17	OOM	94.24 | 66.74
	RIGBD	96.20 | 65.96	OOM	96.51 | 66.81

E Evaluation with Different Surrogate Models

To further investigate the performance of SPEAR, we incorporated two alternative surrogate models, GraphSAGE and GAT within the SPEAR framework which are denoted as **Sur_SAGE** and **Sur_GAT**, respectively. The original framework with the GCN surrogate model is referred to as **Sur_GCN**. We evaluated the performance of SPEAR with these three surrogate models under different defense scenarios. The performance results, measured by ASR and CA, are summarized in Table 9.

The results show that SPEAR consistently achieves high attack success rates and clean accuracy across all three surrogate models, demonstrating the robustness of the method across different surrogate settings. Despite the variation in the surrogate models, the attack performance remains competitive, indicating that the proposed framework can effectively handle different types of surrogate models. Upon evaluating the computational complexity and performance stability across various defense methods, we find that the GCN surrogate model remains the most suitable choice for our framework. This conclusion is based on its balance between attack success and computational efficiency, especially when considering the challenges posed by different defense strategies.