Rethinking Graph Backdoor Attacks: A Distribution-Preserving Perspective

Zhiwei Zhang
The Pennsylvania State University
State College, USA
zbz5349@psu.edu

Minhua Lin
The Pennsylvania State University
State College, USA
mfl5681@psu.edu

Enyan Dai
The Pennsylvania State University
State College, USA
emd5759@psu.edu

Suhang Wang
The Pennsylvania State University
State College, USA
szw494@psu.edu

Abstract

Graph Neural Networks (GNNs) have shown remarkable performance in various tasks. However, recent works reveal that GNNs are vulnerable to backdoor attacks. Generally, backdoor attack poisons the graph by attaching backdoor triggers and the target class label to a set of nodes in the training graph. A GNN trained on the poisoned graph will then be misled to predict test nodes attached with trigger to the target class. Despite their effectiveness, our empirical analysis shows that triggers generated by existing methods tend to be out-of-distribution (OOD), which significantly differ from the clean data. Hence, these injected triggers can be easily detected and pruned with widely used outlier detection methods in real-world applications. Therefore, in this paper, we study a novel problem of unnoticeable graph backdoor attacks with in-distribution (ID) triggers. To generate ID triggers, we introduce an OOD detector in conjunction with an adversarial learning strategy to generate the attributes of the triggers within distribution. To ensure a high attack success rate with ID triggers, we introduce novel modules designed to enhance trigger memorization by the victim model trained on poisoned graph. Extensive experiments on real-world datasets demonstrate the effectiveness of the proposed method in generating in distribution triggers that can bypass various defense strategies while maintaining a high attack success rate. Our code is available at: https://github.com/zzwjames/DPGBA.

CCS Concepts

• **Computing methodologies → Machine learning**.

Keywords

Backdoor Attack; Graph Neural Networks

1 Introduction

Graph-structured data is pervasive in real world, such as social networks [13], molecular structures [31], and knowledge graphs [28]. With the growing interest in learning from graphs, Graph Neural Networks (GNNs), which have shown great ability in node representation learning on graphs, have become increasingly prominent. Generally, GNNs adopt the message-passing mechanism, which update a node's representation by recursive propagation and aggregation of information from a node's neighbors. The learned node representations preserve both node attributes and local graph structure information, which can benefit a range of downstream tasks, such as node classification [16, 21, 36], graph classification [40], and link prediction [45].

Though GNNs have achieved remarkable performance across various applications, recent studies [8, 37, 46] have shown that they are vulnerable to backdoor attacks. Generally, backdoor attacks generate and attach backdoor triggers to a selected group of nodes, known as target nodes, and assign target nodes a target class. Triggers are typically a node or a subgraph and can either be predefined or generated by a trigger generator. When a GNN model is trained on a dataset poisoned with these triggers, it learns to associate the presence of the trigger with the target class. Consequently, during inference, the backdoored model will misclassify test nodes attached with the trigger to the target class, while maintain high prediction accuracy on clean nodes, i.e., nodes without triggers attached. Backdoor attacks on graphs pose a significant threat to the adoption of GNNs in real-world, especially on high-stake scenarios such as banking systems and cybersecurity. For example, an adversary could inject backdoor triggers to the training data for fraud detection in transaction networks, and bypass the detection of a model trained on such poisoned graph by disguising illegal behaviors with backdoor triggers.

Hence, graph backdoor attack is attracting increasing attention and several initial efforts have been taken [8, 37, 46]. For example, SBA [46] conducts pioneering research on graph backdoor attacks.

It adopts randomly generated graphs as triggers. Building on this work, GTA [37] adopts a backdoor trigger generator to generate more powerful sample-specific triggers to improve the attack success rate. Dai et al. [8] shows that the generated triggers in previous work can be easily broken by pruning edges linking nodes with low cosine feature similarity. To alleviate this issue, they propose UGBA, which adopts an unnoticeable constraint to make the triggers and the target nodes to have large cosine similarity of features.

Despite their superior attack performance and the initial efforts to make backdoor attack unnoticeable, our preliminary analysis in Sec. 3.3 shows that the triggers generated by existing generator-based backdoor attack methods are typically *out-of-distribution* samples compared to the clean data, i.e., the feature vectors of the triggers are easily distinguishable from those of clean nodes. This is because the victim model, trained on a poisoned dataset, tends to memorize outlier triggers or associate outlier triggers with the target class more easily than in-distribution triggers. Consequently, when the trigger generator is trained without any constraints, it naturally exploits this shortcut to achieve higher attack success rate. Though UGBA aims to learn triggers that have large cosine feature similarity with target nodes, it does not take the magnitude of triggers into consideration, resulting in triggers having large features for higher attack success rate. This "out-of-distribution" property can be leveraged by outlier detection methods to identify and remove those triggers, thus significantly degrading the attack performance. As shown in our preliminary analysis in Sec. 3.3, with an unsupervised outlier detection, we can successfully remove/break the triggers in a poisoned graph, degrading the attack success rate from over 90% to 0% on Pubmed dataset. As outlier detection is widely deployed in real-world applications such as financial networks [18] and cybersecurity [35], the out-of-distribution issue undermines the real-world adoption value of these backdoor attack methods. For instance, in financial networks, outlier detection methods play a pivotal role in unveiling unusual transaction patterns that may indicate money laundering activities [1].

Developing in-distribution triggers that mimic legitimate patterns within these networks is promising to fool existing outlier detection methods. For instance, in a social network, an ID trigger could replicate the typical behavior patterns of genuine user accounts, making it more difficult for outlier detection methods to distinguish between legitimate activities and those designed to compromise the network's integrity. Hence, the development of an effective graph backdoor attack, using in-distribution (ID) triggers capable of bypassing widely deployed outlier detection methods while maintaining a high attack success rate, holds significant importance. However, there is no existing work on this.

Therefore, in this paper, we study a novel and important problem of developing an effective distribution-preserving graph backdoor attack. In essence, we confront two key challenges: (i) how to generate in-distribution triggers that are resistant to commonly employed outlier detection methods in real-world applications; and (ii) making triggers in-distribution might degrade the attack performance as it breaks the shortcut for the victim model to associate the trigger and the target label. How to achieve a high attack success rate with these ID triggers? In an attempt to address these challenges, we proposed a novel framework Distribution Preserving Graph Backdoor Attack (DPGBA). To generate ID triggers, we introduce an OOD

detector and adopt an adversarial learning strategy to constrain the attributes of the generated triggers. In order to enhance the attack success rate utilizing ID triggers, we introduce innovative modules aimed at promoting the memorization of generated triggers by the victim model and encouraging the learned embeddings of the poisoned nodes to resemble those belonging to the target class. In summary, our main contributions are:

- We empirically show that existing backdoor attacks suffer from either low attack success rate or outlier issues that allow outlier detection methods to significantly degrade their performance;
- We design a novel graph backdoor attack framework, which can generate in-distribution triggers that can bypass outlier detection and achieve high attack success rate;
- Extensive experiments on large-scale dataset demonstrate the effectiveness of our framework in backdooring different GNN models using ID triggers under different defense settings.

2 Related Works

2.1 Graph Neural Networks

With the increasing need for learning on graph structured data, Graph Neural Networks (GNNs), which have shown great power in modeling graphs, are developing rapidly in recent years [7, 48, 49]. Most GNN variants operate under the message-passing framework, which integrates pattern extraction and interaction modeling across each layer [16, 21, 45]. Essentially, GNNs handle messages derived from node representations, propagating these messages through various message-passing mechanisms to enhance node representations. These refined representations are subsequently applied to downstream tasks With the evolution of GNN technology, numerous advancements have been made to augment their performance and application scope. Innovations in self-supervised learning techniques for GNNs aim to lessen the dependency on annotated data [25, 29, 38, 43, 51]. Additionally, significant strides have been made in enhancing the fairness [9, 50], robustness [7] and interpretability of GNN frameworks [47]. Furthermore, specialized GNN architectures have been developed to address the unique challenges presented by heterophilic graphs [39], broadening the potential use cases of GNNs in complex networked systems.

2.2 Backdoor Attacks on Graph

Backdoor attacks have been widely studied in image domain [4, 15, 23]. Initial work directly poison training samples [4, 27]. Others have explored the invisibility of triggers [11, 24]. Besides, the hidden backdoor could also be embedded through transfer learning [22], modifying model parameters [3], and adding extra malicious modules [34]. Recent studies have begun to delve into backdoor attacks on GNNs, focusing on a strategy distinct from the more prevalent poisoning and evasion attacks. Backdoor attacks involve injecting malicious triggers in the training data, which cause the model to make incorrect predictions when these triggers are presented in test samples. This form of attack subtly manipulates the training phase of a model, ensuring it performs as expected under regular conditions but fails in the presence of trigger-embedded inputs. Among the pioneering efforts, SBA [46] introduced a method for injecting universal triggers into training samples through a subgraph-based approach. However, the attack success rate is poor.

GTA [37] furthered this by developing a technique for generating adaptive triggers, customizing perturbations for individual samples to enhance attack effectiveness. In UGBA [8], an algorithm for selecting poisoned nodes is introduced to optimize the utilization of the attack budget. Additionally, an adaptive trigger generator is employed to create triggers that demonstrate a high cosine similarity to the target node. While GTA and UGBA achieve a high attack success rate, the generated triggers tend to be outliers. This is because it is more efficient for a victim model to associate outlier triggers with the target class, leading the unconstrained trigger generator to exploit this shortcut for a higher attack success rate.

The aforementioned graph backdoor methods either have low attack success rate or outlier issues that makes them ineffective in presence of outlier detection. A detailed review of existing outlier detection on graph is given in Appendix D. Our proposed method is inherently different from these methods as (i) we aim to generate unnoticeable in-distribution triggers capable of bypassing the commonly used outlier detection methods in real-world applications. (ii) we focus on guaranteeing a high attack success rate with these in-distribution triggers.

3 Preliminaries Analysis

In this section, we give preliminaries of backdoor attacks on graphs and show out-of-distribution issues of existing backdoor attacks.

3.1 Notations

We denote an attributed graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ represents the set of N nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges, and $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denotes the set of node attributes, with $\mathbf{x}_i$ being the attribute of node v_i. The adjacency matrix of the graph $\mathcal{G}$ is denoted as $\mathbf{A} \in \mathbb{R}^{N \times N}$, where $A_{ij} = 1$ if nodes v_i and v_j are connected; otherwise, $A_{ij} = 0$. In this paper, we concentrate on backdoor attack for semi-supervised node classification task within the inductive setting. Specifically, the training graph $\mathcal{G}$ includes a small subset of labeled nodes $\mathcal{V}_L \subseteq \mathcal{V}$ with labels as $\mathcal{Y}_L = \{y_1, \ldots, y_{N_L}\}$. The remaining nodes of $\mathcal{G}$ are unlabeled, denoted as $\mathcal{V}_U$. We denote $\mathcal{V}_{Tr} = \mathcal{V}_L \cup \mathcal{V}_U$ as the node set for the training graph. The test nodes, denoted as $\mathcal{V}_T$, are not part of the training graph $\mathcal{G}$, i.e., $\mathcal{V}_T \cap \mathcal{V}_{Tr} = \emptyset$. We aim to add backdoor triggers within budget to the training graph such that a GNN model trained on the backdoored graph will be fooled to give targeted label for test nodes attached with triggers.

3.2 Preliminaries of Graph Backdoor Attacks

Next, we elaborate on the attacker's objectives, knowledge, and capabilities, followed by the details of the inductive setting employed for evaluating the attack.

Attacker's Goal: The attacker aims to add backdoor triggers, i.e., nodes or subgraphs, to a small set of target nodes in the training graph and label them as a target class, such that a GNN model trained on the poisoned graph will memorize the backdoor trigger and be misguided to classify target nodes attached with triggers as the target class. Meanwhile, the attacked GNN model should behave normally for clean nodes without triggers attached.

Attacker's Knowledge and Capability: In the context of most poisoning attacks [33], attackers have access to the training data of

Table 1: Results of backdoor defense (Attack Success Rate (%) | Clean Accuracy (%)) on PubMed dataset.

Defense	Clean	SBA-Samp	SBA-Gen	GTA	UGBA
None	84.9	30.4 | 84.7	32.0 | 84.6	86.6 | 84.9	92.3 | 84.9
OD	84.8	29.6 | 84.9	31.7 | 84.6	0.0 | 85.0	0.0 | 84.7

the target model. However, they lack information about the specifics of the target GNN models, including their architecture. Attackers have the capability to attach triggers and labels to nodes within a predefined budget prior to the training of the target models in order to poison the graphs. In the inference phase, attackers retain the ability to attach triggers to the target test nodes.

Evaluation Setting: Given $\mathcal{V}_P \subseteq \mathcal{V}_U$ as a set of poisoned node, we attach the generated trigger $g_i = (\mathbf{X}_i^g, \mathbf{A}_i^g)$ to the node $v_i \in \mathcal{V}_P$ and assign $\mathcal{V}_P$ with target class y_t to form the backdoored dataset. The victim model is then trained on this dataset. During inference, triggers generated by trigger generator f_g are attached to test nodes $v_i \in \mathcal{V}_T$ to evaluate the attack performance.

3.3 Outlier Issues of Graph Backdoor Attacks

An implicit requirement for backdoor attacks is that the generated triggers should be indistinguishable from clean inputs. This condition is commonly satisfied in the image domain [23] by constraining backdoor triggers to the input, such as using small patch patterns or imperceptible perturbations. However, in the context of backdoor attacks on graphs, where new samples are generated, without specific design to constrain in-distribution trigger generation, the trigger generator may take a shortcut and produce outlier triggers which can be easily memorized by the victim model. Though such triggers have high attack success rate, they are outliers and can be easily removed by simple outlier detection algorithms, making them ineffective in practice.

To show that the triggers generated by existing graph backdoor attack methods are outliers, we conduct analysis on Pubmed dataset [32]. We first adopt existing backdoor attack algorithms to add backdoors to the graph under the semi-supervised setting. We then apply Principal Component Analysis (PCA) to reduce the dimensionality of node attributes for both clean nodes and triggers' nodes, and visualize them in a 2-dimensional space as shown in Fig. 1, where the blue and the red dots denote the clean node and the generated triggers, respectively. From the figure, it is obvious that the generated triggers of GTA [37] and UGBA [8] are very different and far from the clean data distribution, showing that the triggers generated by many existing algorithms are outliers.

To show that such triggers are ineffective in practice, i.e., can be easily detected, we employ outlier detection (**OD**) to defend against existing backdoor attacks. Specifically, we adopt DOMINANT [10], a popular unsupervised outlier detection method based on autoencoder for graph-structured data, which utilize the reconstruction error on both graph structural and node attribute for outlier detection. The intuition is that the autoencoder will be better at reconstructing instances that are similar to the majority of the data it was trained on (presumably normal data) and worse at reconstructing outliers [26]. Given a backdoored dataset, we train DOMINANT on it and then discard those samples with high reconstruction loss. Experiment results on Pubmed [32] with $|\mathcal{V}_P|$

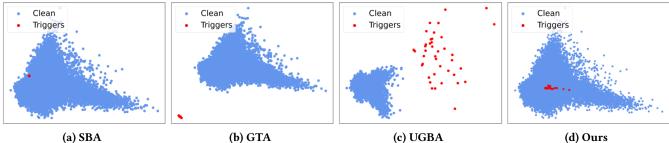**(a) SBA**   **(b) GTA**   **(c) UGBA**   **(d) Ours**

Figure 1: PCA visualization of features of clean and generated triggers by different attacks. Red dots are overlapped in (a) as SBA generate the same trigger for all target nodes.

set as 40 are presented in Table 1. The architectures of the target model is GCN [21] and the size of triggers is limited to contain three nodes. We filter out the top 3% of samples with the highest reconstruction losses. More results on other datasets can be found in Table 3. The accuracy of the backdoored GNN on clean test set is also reported in Table 1 to show how the defense strategy affect the prediction performance. Accuracy on a clean graph without any attacks is reported as reference. All the results are averaged scores of 5 runs. The details of evaluation protocol is in Sec 5.1.3. From the table, we can observe: (i) Both GTA and UGBA exhibit a high attack success rate without a defense method. However, employing a straightforward outlier detection strategy effectively eliminates all of their triggers, degrading the attack success rate to 0; (ii) For SBA, which generates triggers based on the mean and standard deviation of the clean input, it achieves a low attack success rate despite its triggers not being classified as outliers. Evidently, *existing backdoor attacks methods on graph suffer from either a low attack success rate or outlier issues.* Thus, it is important to design a framework capable of generating ID triggers that can achieve a high attack success rate and bypass outlier detection.

3.4 Problem Definition

Our preliminary analysis shows that existing backdoor attacks either have a low attack success rate or encounter outlier issues. To address these problems, we propose to develop a novel and effective distribution preserving graph backdoor attack that can generate in-distribution triggers capable of bypassing commonly employed outlier detection techniques, while maintaining a high attack success rate. As we aim to bypass outlier detection techniques, we define the distribution preserving as follows.

In-Distribution Constraint on Triggers. Let $\mathcal{G}_B = (\mathcal{V} \cup \mathcal{T}_P, \mathcal{E} \cup \mathcal{E}_P, \mathbf{X} \cup \mathbf{X}_P)$ be the backdoored graph, where $\mathcal{T}_P$ represents the set of generated triggers, $\mathcal{E}_P$ denotes the edge set containing edges within the triggers $g_i \in \mathcal{T}_P$ and edges attaching these triggers to nodes $v_i \in \mathcal{V}$, and $\mathbf{X}_P$ represents the node attributes of the generated triggers. Let f_o be an outlier detection model trained on $\mathcal{G}_B$. Then, our in-distribution constraint on trigger g_i is defined as:

$$f_o(g_i) < \tau \tag{1}$$

where $f_o(g_i)$ is the anomaly score of g_i and τ is a threshold which can be tuned based on datasets.

Following [8], the clean prediction for a node v_i can be denoted as $f_\theta(v_i) = f_\theta(\mathcal{G}_C^i)$, where $\mathcal{G}_C^i$ is the K-hop subgraph centered at

v_i. For a node v_i attached with the trigger g_i, the predicted label is denoted as $f_\theta(\tilde{v}_i)$, where $\tilde{v}_i = a\left(\mathcal{G}_C^i, g_i\right)$ and $a(\cdot)$ being the operation of trigger attachment. With the above descriptions and notations, the effective distribution preserving graph backdoor attack is formally defined as:

PROBLEM 1 (DISTRIBUTION PRESERVING GRAPH BACKDOOR ATTACK). *Given a clean attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ with a set of nodes $\mathcal{V}_L$ provided with labels $\mathcal{Y}_L$, we aim to learn an adaptive trigger generator $f_g(v_i) \rightarrow g_i$. This generator will produce triggers that bypass outlier detection while ensuring that a GNN f, trained on the poisoned graph will classify the test node attached with the trigger to the target class y_t. This objective is achieved by solving:*

$$\min_{\theta_g} \sum_{v_i \in \mathcal{V}_U} l\left(f_{\theta_s^*}(\tilde{v}_i), y_t\right)$$

$$s.t. \ \theta_s^* = \arg\min_{\theta_s} \sum_{v_i \in \mathcal{V}_L} l\left(f_s(v_i), y_i\right) + \sum_{v_i \in \mathcal{V}_P} l\left(f_s(\tilde{v}_i), y_t\right), \tag{2}$$

$$\forall v_i \in \mathcal{V}_P \cup \mathcal{V}_U, g_i \ meets \ Eq. \ (1) \ and \ |g_i| < \Delta_g$$

$$|\mathcal{V}_P| \le \Delta_P$$

where $l(\cdot)$ is the cross entropy loss, y_t is the target class label and θ_g denotes the parameters of the adaptive trigger generator f_g. In the constraints, the node size of trigger $|g_i|$ is limited by Δ_g, and the size of poisoned nodes is limited by Δ_P. The architecture of the target GNN f is unknown. Hence, a surrogate GNN classifier f_s with parameters θ_s is used.

4 Methodology

In this section, we present the details of the proposed framework, which aims to optimize Eq. (2) to conduct effective distribution preserving graph backdoor attacks. Two challenges remain to be addressed: (i) how to generate ID triggers that have the capability to bypass outlier detection defense methods; (ii) how to learn the trigger generator to obtain triggers that meet ID constraint while maintaining a high attack success rate. To address these challenges, a novel framework DPGBA is proposed, which is illustrated in Fig. 2. DPGBA is composed of an OOD detector f_d, a trigger generator f_s and a surrogate node classifier f_s. Specifically, to address the first challenge, an adversarial training strategy involving an OOD detector f_d and a trigger generator f_g is introduced. The OOD detector is trained to differentiate between clean data from a graph $\mathcal{G}$ and triggers generated by f_g. In turn, the trigger generator enhances its capability to create triggers that closely mimic the clean data. To
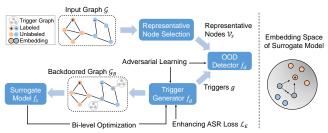
Figure 2: Framework of DPGBA

address the second challenge, we propose novel objective functions that promote generated triggers to exert a dominant influence on target nodes. This encourages the victim model to memorize these triggers, leading to a high attack success rate. Next, we give the details of each component.

4.1 In-Distribution Triggers Generation

In this subsection, we detail the design of the trigger generator f_g and the adversarial learning strategy proposed to ensure the generator produces triggers that are in-distribution.

To make the trigger generator more effective and flexible in generating triggers, instead of using predefined triggers, following [8], we use an MLP as the adaptive trigger generator f_g, which generate triggers based on the target node's node attributes. Specifically, given a node v_i, f_g generates the node features and graph structure of a trigger to be attached to v_i as:

$$\mathbf{h}_i^m = \text{MLP}(\mathbf{x}_i), \quad \mathbf{X}_i^g = \mathbf{W}_f \mathbf{h}_i^m, \quad \mathbf{A}_i^g = \mathbf{W}_a \mathbf{h}_i^m, \quad (3)$$

where $\mathbf{x}_i$ is the node attributes of v_i, and $\mathbf{W}_f, \mathbf{W}_a$ are the learnable parameters for feature and structure generation. We generate the synthetic features $\mathbf{X}_i^g \in \mathbb{R}^{s \times d}$ and adjacency matrix $\mathbf{A}_i^g \in \mathbb{R}^{s \times s}$ for trigger nodes corresponding to node v_i. In accordance with the discrete nature of real-world graphs, we binarize $\mathbf{A}_i^g$ for the forward computation to align with the binary structure of the graph, while the continuous adjacency matrix is utilized during the gradient computation in backpropagation following [8].

To make sure that f_g can generate distribution preserving triggers, we adopt the adversarial learning strategy. Specifically, we introduce an OOD detector f_d which aims to differentiate if an input is from the original graph $\{v_i \in \mathcal{V}, Y = 1\}$ or from generated triggers $\{g_i \in \mathcal{T}_P, Y = 0\}$. Following the Generative Adversarial Network (GAN) framework in [14], f_d refines its ability to discern between clean inputs and generated triggers by minimizing the binary classification loss. Concurrently, the generator f_g craft triggers to deceive f_d by maximizing the binary classification loss. This min-max game equips the generator with the ability to produce triggers that are indistinguishable from in-distribution data. The min-max process is mathematically described as:

$$\min_{\theta_g} \max_{\theta_d} \mathcal{L}_D = \sum_{v_i \sim \mathcal{V}_s} \log(f_d(v_i)) + \sum_{g \in \mathcal{T}_P} \log(1 - f_d(g)), \quad (4)$$

where $\mathcal{V}_s \in \mathcal{V}$ is a selected set of representative nodes. The reason why we select $\mathcal{V}_s$ instead of using $\mathcal{V}$ is because benchmark datasets inherently contain outlier samples at the edges of the feature distribution [26]. These outliers, when taken as inputs by the detector f_d,

simplify the task for the trigger generator f_g to deceive f_d by producing triggers similar to these outliers, thereby undermining the objective of generating in-distribution triggers. Thus, the careful choice of representative samples for training the OOD detector is critical. To obtain $\mathcal{V}_s$, we pretrain an auto-encoder on the original graph $\mathcal{G}$ and select samples whose reconstruction losses are close to the mean loss, e.g., within one standard deviation. The details are described in Appendix B. The idea behind using an autoencoder for this purpose is that it learns to identify and capture the most crucial and frequently occurring features and patterns within the data. These chosen samples can be regarded as "typical" based on the criteria of reconstruction loss.

4.2 Enhancing ID Trigger Effectiveness

Though using an OOD detector and adversarial strategy restricts f_g to generating in-distribution triggers, our empirical results indicate that the attack success rate is not comparable to that achieved by GTA [37] and UGBA [8]. This is because the generated triggers by GTA and UGBA are outliers that deviate a lot from the original nodes, which makes it easier for the victim model to create a shortcut to associate the backdoor trigger with the target class. In contrast, when using ID triggers, it becomes challenging for the victim model trained on our poisoned dataset to discern a specific trigger pattern, resulting in non-activation when we attach the trigger to the target node. To ensure the attack performance with ID triggers, we design novel modules to enhance trigger memorization by the victim model and improve attack adaptability against unseen, new targets. Next, we give the details of each module.

4.2.1 Enhancing Memorization of Triggers. A key to successful graph backdoor attacks is the ability of having the victim model to correlate attached triggers with the target class. Thanks to the message-passing mechanism, trigger attributes can directly influence the attributes of the target nodes. Considering the diverse attributes across target nodes, we propose to encourage the generator to produce triggers that, once attached to different target nodes, can prompt the victim model to learn similar embeddings for the poisoned target nodes. Specifically, for each pair of nodes $v_i \in \mathcal{V}_P$ and $, v_j \in \mathcal{V}_P$, when attaching triggers to them, our objective is to ensure that these triggers can guide the surrogate classifier f_s to learn a high cosine similarity between $z_s(\tilde{v}_i)$ and $z_s(\tilde{v}_j)$, where $\tilde{v}_i = a\left(\mathcal{G}_C^i, g_i\right)$ denotes v_i attached with backdoor trigger g_i and $z_s(\tilde{v}_i)$ represents the learned embedding of poisoned target node $\tilde{v}_i$ by the surrogate classifier, i.e., the last layer embedding of f_s before feeding to Softmax function. This approach ensures that the trigger attributes significantly impact the target node attributes, making them the dominant features within the embeddings of poisoned target nodes learned by the victim model. This dominance ensures that the victim model memorizes these triggers more effectively, resulting in a higher likelihood of a successful attack.

Moreover, once triggers have exerted a strong influence on the target node embeddings, enhancing the feature-level similarity between the poisoned target nodes and the nodes of the target class can further mislead the victim model into misclassifying the poisoned target nodes as belonging to the specific target class. Specifically, for a pair of nodes $v_i \in \mathcal{V}_P$ and $v_j \in \mathcal{V}_t$, where $\mathcal{V}_t \in \mathcal{V}_L$

denotes nodes from target class, our goal is to generate triggers that guide the surrogate model to learn a high cosine similarity between embeddings $z_s(\tilde{v}_i)$ and $z_s(v_j)$, while ensuring the similarity between $z_s(\tilde{v}_i)$ and $z_s(v_k)$ is lower, where $v_k \in \mathcal{V}_L \backslash \mathcal{V}_t$ belongs to non-target class. Combining the aforementioned two goals, we propose the following loss function for the trigger generator f_g:

$$
\mathcal{L}_E = - \sum_{v_i \in \mathcal{V}_U} \sum_{v_j \in \mathcal{V}_U} S(z_s(\tilde{v}_i), z_s(\tilde{v}_j)) +
$$
$$
\sum_{v_i \in \mathcal{V}_U} \sum_{v_j \in \mathcal{V}_t} - \log \left(\frac{S(z_s(\tilde{v}_i), z_s(v_j))}{S(z_s(\tilde{v}_i), z_s(v_j)) + \sum_{v_k \in \mathcal{V}_L \backslash \mathcal{V}_t} S(z_s(\tilde{v}_i), z_s(v_k))} \right),
\tag{5}
$$

where S measures the cosine similarity of the embeddings. By minimizing $\mathcal{L}_E$, the victim model trained on the poisoned dataset can better correlate the presence of triggers with the target class, ultimately leading to a higher attack success rate.

4.2.2 Enhancing Attack Effectiveness against Unseen Targets. To fully harness the attack budget, we propose implementing a strategy that involves assigning varying weights to accessible nodes. The idea is to enhance the adaptability and effectiveness of the trigger generator against new and unseen targets by prioritizing nodes that have proven to be particularly challenging to attack. The core of our challenge lies in measuring the difficulty level of attacking each node. To measure this, we employ the predicted probability distribution provided by the surrogate model for poisoned nodes. Specifically, for a given node $v_i \in \mathcal{V}_P$, $p_t^i = f_s(\tilde{v}_i)_t$ gives the probability that poisoned node $\tilde{v}_i$ is classified to the target class by surrogate model. A large p_t^i indicates a successful attack, suggesting that the trigger generator has effectively learned to attack this target, and therefore, we assign it a smaller weight. Conversely, a target with a small p_t^i is considered more challenging and is assigned a larger weight, directing the trigger generator to focus more on this target. Then, we integrate this strategy into the outer loss in Eq. (2) and obtain:

$$
\mathcal{L}_T = \sum_{v_i \in \mathcal{V}_U} w_i \cdot l(f_s(\tilde{v}_i), y_t),
\tag{6}
$$

where $w_i = \exp(-p_t^i)$.

4.3 Final Objective Function of DPGBA

To ensure the effectiveness of the generated triggers, we optimize the adaptive trigger generator to successfully attack the surrogate classifier f_s, which is trained on the backdoored dataset. The training of the surrogate classifier is formulated as:

$$
\min_{\theta_s} \mathcal{L}_s(\theta_s, \theta_g) = \sum_{v_i \in \mathcal{V}_L} l(f_s(v_i), y_i) + \sum_{v_i \in \mathcal{V}_P} l(f_s(\tilde{v}_i), y_t),
\tag{7}
$$

where θ_s represents the parameters of the surrogate model f_s, y_i is the label of labeled node $v_i \in \mathcal{V}_L$ and y_t is the target class label.

Then, with $\mathcal{L}_T$ in Eq. (6) aimed at misleading the surrogate model f_s to predict various nodes from $\mathcal{V}$ to be y_t once attached with generated triggers, $\mathcal{L}_D$ in Eq. (4) constraining the in-distribution property of generated triggers, and $\mathcal{L}_E$ in Eq. (5) enhancing the

attack performance for these in-distribution triggers, the final objective function of DPGBA is given as:

$$
\min_{\theta_g} \max_{\theta_d} \mathcal{L} = \mathcal{L}_T(\theta_s^*, \theta_g) + \alpha \mathcal{L}_D(\theta_d, \theta_g) + \beta \mathcal{L}_E(\theta_s^*, \theta_g)
$$
$$
s.t. \theta_s^* = \arg \min_{\theta_s} \mathcal{L}_s(\theta_s, \theta_g)
\tag{8}
$$

where α and β are scalars to control the contributions of $\mathcal{L}_D$ and $\mathcal{L}_E$, θ_g, θ_s and θ_d represent the parameters for trigger generator f_g, surrogate model f_s and OOD detector f_d, respectively. We adopt bi-level optimization to optimize Eq. (8). Next, we give details of each optimization process.

Lower level Optimization In lower-level optimization, the surrogate model f_s will be trained on the backdoored dataset. We update θ_s for N inner iterations with fixed θ_g to approximate θ_s^* as:

$$
\theta_s^{t+1} = \theta_s^t - \alpha_s \nabla_{\theta_s} \mathcal{L}_s(\theta_s, \theta_g),
\tag{9}
$$

where θ_s^t denotes model parameters after t iterations, α_s is the learning rate for training the surrogate model.

The OOD detector f_d is optimized to enhance its capability to distinguish between clean inputs and generated triggers by maximizing $\mathcal{L}_D$. Similarly, we update θ_d with K inner iterations with fixed θ_g to approximate θ_d^* as:

$$
\theta_d^{k+1} = \theta_d^k + \alpha_d \nabla_{\theta_d} \mathcal{L}_D(\theta_d, \theta_g),
\tag{10}
$$

where θ_d^k denotes model parameters after k iterations, α_d is the learning rate for training the surrogate model.

Upper level optimization In the upper level optimization, the updated surrogate model parameters θ_s^T and OOD detector parameters θ_d^K are used to approximate θ_s^* and θ_d^*, respectively. We then apply first-order approximation to compute gradients of θ_g by:

$$
\theta_g^{m+1} = \theta_g^m - \alpha_g \nabla_{\theta_g} \left(\mathcal{L}_T(\bar{\theta}_s, \theta_g) + \alpha \mathcal{L}_D(\bar{\theta}_d, \theta_g) + \beta \mathcal{L}_E(\bar{\theta}_s, \theta_g) \right),
\tag{11}
$$

where $\bar{\theta}_s$ and $\bar{\theta}_d$ indicate gradient propagation stopping, θ_g^m denotes model parameters after m iterations. The training algorithm of DPGBA is given in Algorithm 1. Time complexity analysis can be found in Appendix J.

5 Experiments

In this section, we will evaluate the proposed DPGBA on various datasets to answer the following research questions:

- **RQ1:** Can our framework conduct effective backdoor attacks on GNNs and simultaneously ensure in-distribution property?
- **RQ2:** How do the number of poisoned nodes affect the performance of backdoor attacks?
- **RQ3:** How do the in-distribution constraint and the enhancing attack performance module influence attack efficacy in scenarios both with and without defense mechanisms?

5.1 Experimental settings

5.1.1 Datasets. To demonstrate the effectiveness of our DPGBA, we conduct experiments on four public real-world datasets, i.e., Cora, Pubmed [32], Flickr [44], and OGB-arxiv [17], which are widely used for inductive semi-supervised node classification. Cora and Pubmed are small citation networks. Flickr is a large-scale

Table 2: Dataset Statistics

Datasets	#Nodes	#Edges	#Features	#Classes
Cora	2,708	5,429	1,443	7
Pubmed	19,717	44,338	500	3
Flickr	89,250	899,756	500	7
OGB-arxiv	169,343	1,166,243	128	40

graph that links image captions sharing the same properties. OGB-arixv is a large-scale citation network. The statistics of the datasets are summarized in Table 2. More details of the dataset can be found in Appendix C.

5.1.2 Compared Methods. We compare DPGBA with representative and state-of-the-art graph backdoor attack methods, including UGBA [8], GTA [37], SBA-Samp [46] and its variant SBA-Gen. More details of these compared methods can be found in Appendix H. For a fair comparison, hyperparameters of all the attack methods are tuned based on the performance of the validation set.

5.1.3 Evaluation Protocol. Following the evaluation protocol in UGBA [8], we conduct experiments on the inductive node classification task. In this setup, attackers do not have access to test node during trigger generator training. We randomly exclude 20% of nodes from the original dataset, denoted as $\mathcal{V}_T$, using half as targets for assessing attack effectiveness and the other half as clean test nodes for evaluating the accuracy of models under attack on normal samples. The training graph $\mathcal{G}$ consists of the remaining 80% of nodes, with the labeled node set and validation set each containing 10% of nodes. We measure backdoor attack performance using the average success rate (ASR) on target nodes and clean accuracy on clean test nodes. A two-layer GCN acts as the surrogate model for all attack strategies. To evaluate the transferability of backdoor attacks, we target GNNs with different architectures—GCN, Graph-Sage, and GAT. We conduct experiments on each GNN architecture five times and report the average ASR and clean accuracy from the total of 15 experiments. The attack budget Δ_P on size of poisoned nodes $\mathcal{V}_P$ is set as 10, 40, 160, and 565 for Cora, Pubmed, Flickr, and OGB-arxiv, respectively. The number of nodes in the trigger size is limited to 3 for all experiments. Our DPGBA deploys a 2-layer GCN as the surrogate model. A 2-layer MLP is used as the trigger generator. More details of the hyperparameter setting can be found in Appendix I.

For the defense strategy **OD**, in line with the in-distribution constraint outlined in Sec. 3.4, we use DOMINANT [10] as f_o and train it on the poisoned graph $\mathcal{G}_B$ with triggers attached to nodes in $\mathcal{V}_P$. The threshold τ, as specified in Eq. (1), is set such that data points with a reconstruction loss greater than τ comprise 3% of the dataset. The remaining 97% of the data points have a reconstruction loss at or below τ. Before training the surrogate model f_s on the poisoned graph $\mathcal{G}_B$, we prune those nodes with reconstruction loss above τ. Once f_s is trained, f_a and τ are fixed for the testing phase to perform inference on test nodes in $\mathcal{V}_T$ and the associated generated triggers. Nodes with a reconstruction loss above τ are pruned.

5.2 Backdoor Attack Performance

To answer **RQ1**, we evaluate DPGBA against baseline methods on four real-world graphs, considering scenarios with and without the
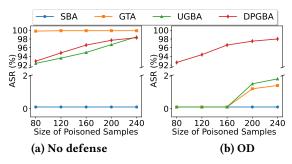

(a) No defense   **(b) OD**

Figure 3: Impacts of sizes of poisoned nodes on Flicker.

OD defense strategy as outlined in Sec. 5.1.3. We report the average results in backdooring three different GNN architectures in Tab. 3. Detailed results for each architecture are provided in Tab. 4 – 6 in Appendix E. From the table, we make the following observations:

- When no backdoor defense strategy is applied, DPGBA shows a comparable or slightly better ASR than leading baselines such as GTA and UGBA, while SBA-Samp and its variant, SBA-Gen, consistently achieve lower ASRs. This indicates the effectiveness of our modules in enhancing the influence of triggers on the target nodes. Regarding clean accuracy, our framework consistently demonstrates comparable results with all the baselines.
- When applying a simple outlier detection defense, triggers generated by GTA and UGBA are removed, but DPGBA still achieves over 90% ASR. This demonstrates that our DPGBA effectively generates imperceptible ID triggers that can successfully bypass commonly used outlier detection methods in real applications.
- Though we employ GCN as the surrogate model during training, the generated triggers consistently achieve high ASR across three different GNN architectures, as shown in Tab. 4 – 6. This indicates the transferability of the trigger generator within our framework.

5.3 Impact of the Size of Poisoned Nodes

To answer **RQ2**, we conduct experiemnts to explore the attack performance of DPGBA given different budgets in the size of poisoned nodes. Specifically, we vary the size of poisoned samples as {40, 80, 120, 160, 200}. The other settings are the same as the evaluation protocol in Sec. 5.1.3. Hyperparameters are selected with the same process as described in Appendix I. Fig. 3 shows the results on Flicker dataset. We have similar observations on other datasets. We only report the attack success rate as we did not observe any significant change in clean accuracy for all the baselines and our DPGBA. From Fig. 3, we can observe that:

- The attack success rate of UGBA [8] and DPGBA consistently rises as the number of poisoned samples increases in (a), which aligns with our expectation. Our method maintains a comparable ASR when no defense is applied, highlighting the effectiveness of our attack performance enhancement module.
- When OD defense is applied on the backdoor attacks in (b), our DPGBA still achieve promising performances. In contrast, all the baseline methods achieve an almost 0% ASR, as anticipated. That is because our method can generate trigger nodes with in-distribution property.

Table 3: Backdoor attack results (ASR (%) | Clean Accuracy (%)). Only clean accuracy is reported for clean graphs. The best results are marked with boldface.

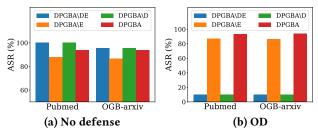| Datasets	Defense	Clean Graph	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA
Cora	None	83.9	35.2 | 83.5	45.3 | 83.2	91.6 | **83.6**	95.1 | 83.5	**96.7** | 83.6
	OD	83.8	34.7 | 83.0	44.1 | **83.6**	0.00 | 83.4	0.00 | **83.6**	**93.9** | 83.5
Pubmed	None	85.1	33.8 | 84.7	34.4 | 84.6	88.1 | 84.9	92.5 | **85.2**	**92.6** | 85.1
	OD	85.0	33.3 | 84.9	33.5 | 84.6	0.00 | 84.7	0.00 | 85.0	**91.8** | **85.1**
Flickr	None	46.0	0.00 | **46.2**	0.00 | 45.8	88.6 | 45.0	94.9 | 45.4	**96.4** | 45.9
	OD	46.2	0.00 | **45.9**	0.00 | 45.6	0.00 | 45.1	0.00 | 45.4	**94.8** | 45.8
OGB-arxiv	None	65.9	36.0 | 65.8	43.4 | **65.9**	92.5 | 65.8	**98.2** | 65.3	95.1 | 65.6
	OD	65.8	35.0 | **65.6**	42.3 | 65.5	0.00 | 64.9	0.00 | 64.5	**92.4** | 65.4

(a) Flicker

(b) OGB-arxiv

Figure 4: Reconstruction loss distributions on Flicker and OGB-arxiv

(a) No defense

(b) OD

Figure 5: Ablation studies on Pubmed and OGB-arxiv

5.4 In-distribution Property Analysis

In this subsection, to further demonstrate the in-distribution property of triggers generated by our framework, we first conduct backdoor attacks on Flicker and OGB-arxiv datasets, then apply the outlier detection method on the poisoned graph, and finally show the reconstruction loss for both clean data and generated triggers. The histograms of the reconstruction loss are plotted in Fig. 4. From the figure, we observe that the reconstruction loss of the generated triggers closely aligns with the mean of the distribution of reconstruction losses for clean inputs. This alignment can be attributed to the selection of representative samples $\mathcal{V}_S$ for the OOD detector f_d and adversarial learning to make the trigger in-distribution. Additional experiments in Appendix F demonstrate the efficacy of our generated triggers in bypassing various advanced graph outlier detection methods.

5.5 Ablation Studies

To answer **RQ3**, we conduct ablation studies to explore the effects of the ID constraint and the enhancing triggers attack performance module. To demonstrate the effectiveness of the ID constraint module, we set $\alpha = 0$ and obtain a variant named as DPGBA\D. To show the benefits brought by our enhancing attack performance module, we train a variant DPGBA\E which set the β as 0. We also implement a variant of our model by removing both ID constraint and enhancing attack performance module, which is named as DPGBA\DE. The average results and standard deviations on Pubmed and OGB-arxiv are shown in Fig. 5. All the settings of evaluation follow the description in Sec. 5.1.3. And the hyperparameters of the variants are also tuned based on the validation set for fair comparison. From Fig. 5, we observe that: (**i**) When no defense method is applied, DPGBA demonstrates a comparable attack performance, despite

DPGBA\DE and DPGBA\D taking shortcuts to generate outlier triggers. However, when the OD defense method is employed, DPGBA still exhibits a high ASR, while triggers generated by DPGBA\DE and DPGBA\D are almost all eliminated. This observation indicates the effectiveness of the proposed ID constraint module in generating ID triggers; and (**ii**) Compared to DPGBA\E, DPGBA achieves superior attack performance under both defense settings, which shows the effectiveness of our enhancing attack performance module.

5.6 Hyper-parameter Sensitivity Analysis

In this subsection, we further investigate how the hyperparameter α and β affect the performance of DPGBA, where α and β control the weight of ID constraint and enhancing attack performance module, respectively. To explore the effects of α and β, we vary the values of α and β as $\{0.01, 0.1, 1, 10, 100\}$ for Flicker dataset. We report the attack success rate (ASR) of attacking in both no defense and OD defense settings in Fig. 6. The test model is fixed as GCN. We observe that (**i**) In the absence of defense strategies, increasing β improves attack effectiveness, while higher α values lead to reduced attack performance. (**ii**) With outlier detection method deployed, to preserve the in-distribution characteristic of generated triggers and ensure a high attack success rate, it is recommended to set $\alpha \geq 1$ and adjust β accordingly as α increases, ensuring β remains closely aligned with α. This observation eases hyperparameter tuning.

6 Conclusion

In this paper, we empirically verify that existing backdoor attack methods on graph suffer from either a low attack success rate or outlier issues, which can be leveraged by outlier detection methods to identify and remove those triggers, thus significantly degrading their attack performance. To address these problems, we study
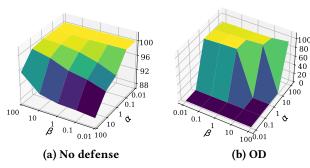
(a) No defense   **(b) OD**

Figure 6: Hyperparameter Sensitivity Analysis

a novel problem of conducting effective distribution-preserving graph backdoor attacks. Specifically, an Out-Of-Distribution (OOD) detector, in conjunction with an adversarial learning strategy, is implemented to constrain the in-distribution property of generated triggers. Additionally, a novel module is proposed to guide the victim model trained on the poisoned dataset to better correlate the presence of triggers with the target class. Extensive experiments on large-scale datasets demonstrate that our proposed method can effectively bypass commonly used outlier detection methods in real-world applications while achieving a high attack success rate in backdooring various target GNN models.

ACKNOWLEDGMENTS

References

[1] Mohiuddin Ahmed, Abdun Naser Mahmood, and Md Rafiqul Islam. 2016. A survey of anomaly detection techniques in financial domain. *Future Generation Computer Systems* 55 (2016), 278–288.

[2] Sambaran Bandyopadhyay, N Lokesh, and M Narasimha Murty. 2019. Outlier aware network embedding for attributed networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 12–19.

[3] Huili Chen, Cheng Fu, Jishen Zhao, and Farinaz Koushanfar. 2021. Proflip: Targeted trojan attack with progressive bit flips. In *ICCV*. 7718–7727.

[4] 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).

[5] Zhenxing Chen, Bo Liu, Meiqing Wang, Peng Dai, Jun Lv, and Liefeng Bo. 2020. Generative adversarial attributed network anomaly detection. In *CIKM*. 1989–1992.

[6] Enyan Dai and Jie Chen. 2022. Graph-Augmented Normalizing Flows for Anomaly Detection of Multiple Time Series. In *International Conference on Learning Representations*. https://openreview.net/forum?id=45L_dgP48Vd

[7] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. 2022. Towards robust graph neural networks for noisy graphs with sparse labels. In *WSDM*. 181–191.

[8] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. 2023. Unnoticeable Backdoor Attacks on Graph Neural Networks *(WWW '23)*. https://doi.org/10.1145/3543507.3583392

[9] Enyan Dai and Suhang Wang. 2021. Say No to the Discrimination: Learning Fair Graph Neural Networks with Limited Sensitive Attribute Information *(WSDM '21)*. 680–688. https://doi.org/10.1145/3437963.3441752

[10] Kaize Ding, Jundong Li, Rohit Bhanushali, and Huan Liu. 2019. Deep Anomaly Detection on Attributed Networks. In *SIAM International Conference on Data Mining (SDM)*.

[11] Khoa Doan, Yingjie Lao, and Ping Li. 2021. Backdoor attack with imperceptible input and latent modification. *Advances in Neural Information Processing Systems* 34 (2021), 18944–18957.

[12] Haoyi Fan, Fengbin Zhang, and Zuoyong Li. 2020. Anomalydae: Dual autoencoder for anomaly detection on attributed networks. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5685–5689.

[13] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph Neural Networks for Social Recommendation *(WWW '19)*. https://doi.org/10.1145/3308558.3313488

[14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[15] 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. https://doi.org/10.1109/ACCESS.2019.2909068

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

[17] 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. *NIPs* (2020), 22118–22133.

[18] Dongxu Huang, Dejun Mu, Libin Yang, and Xiaoyan Cai. 2018. CoDetect: Financial fraud detection with anomaly feature detection. *IEEE Access* 6 (2018), 19161–19174.

[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[20] Thomas N Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. *NIPS Workshop on Bayesian Deep Learning* (2016).

[21] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*.

[22] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight poisoning attacks on pre-trained models. *arXiv preprint arXiv:2004.06660* (2020).

[23] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[24] Yuezun Li, Yiming Li, Baoyuan Wu, Longkang Li, Ran He, and Siwei Lyu. 2021. Invisible backdoor attack with sample-specific triggers. In *Proceedings of the IEEE/CVF international conference on computer vision*. 16463–16472.

[25] Minhua Lin, Teng Xiao, Enyan Dai, Xiang Zhang, and Suhang Wang. 2023. Certifiably Robust Graph Contrastive Learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[26] Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, Lichao Sun, Jundong Li, George H Chen, Zhihao Jia, and Philip S Yu. 2022. BOND: Benchmarking Unsupervised Outlier Node Detection on Static Attributed Graphs. In *Advances in Neural Information Processing Systems*, Vol. 35. 27021–27035.

[27] Yunfei Liu, Xingjun Ma, James Bailey, and Feng Lu. 2020. Reflection backdoor: A natural backdoor attack on deep neural networks. In *ECCV*. Springer, 182–199.

[28] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S. Yu. 2022. Federated Social Recommendation with Graph Neural Network. *TIST* (Aug. 2022), 1–24. https://doi.org/10.1145/3501815

[29] Qian Ma, Hongliang Chi, Hengrui Zhang, Kay Liu, Zhiwei Zhang, Lu Cheng, Suhang Wang, Philip S Yu, and Yao Ma. 2024. Overcoming Pitfalls in Graph Contrastive Learning Evaluation: Toward Comprehensive Benchmarks. *arXiv preprint arXiv:2402.15680* (2024).

[30] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z Sheng, Hui Xiong, and Leman Akoglu. 2021. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering* 35, 12 (2021), 12012–12038.

[31] Elman Mansimov, Omar Mahmood, Seokho Kang, and Kyunghyun Cho. 2019. Molecular Geometry Prediction using a Deep Generative Graph Neural Network. *Scientific Reports* 9, 1 (Dec. 2019). https://doi.org/10.1038/s41598-019-56773-5

[32] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (Sep. 2008), 93. https://doi.org/10.1609/aimag.v29i3.2157

[33] Yiwei Sun, Suhang Wang, Xianfeng Tang, Tsung-Yu Hsieh, and Vasant Honavar. 2020. Adversarial attacks on graph neural networks via node injections: A hierarchical reinforcement learning approach. In *Proceedings of the Web Conference 2020*. 673–683.

[34] Ruixiang Tang, Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. 2020. An Embarrassingly Simple Approach for Trojan Attack in Deep Neural Networks *(KDD '20)*. 218–228. https://doi.org/10.1145/3394486.3403064

[35] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. 2017. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

[36] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJXMpikCZ

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

[38] Teng Xiao, Huaisheng Zhu, Zhiwei Zhang, Zhimeng Guo, Charu C. Aggarwal, Suhang Wang, and Vasant G Honavar. 2024. Efficient Contrastive Learning for Fast and Accurate Inference on Graphs. In *Forty-first International Conference on Machine Learning*. https://openreview.net/forum?id=vsy21Xodrt

[39] Junjie Xu, Enyan Dai, Xiang Zhang, and Suhang Wang. 2022. HP-GMN: Graph Memory Networks for Heterophilous Graphs. In *ICDM*. 1263–1268. https://doi.org/10.1109/ICDM54844.2022.00165

[40] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ryGs6iA5Km

[41] Zhiming Xu, Xiao Huang, Yue Zhao, Yushun Dong, and Jundong Li. 2022. Contrastive attributed network anomaly detection with data augmentation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 444–457.

[42] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *ICML*. PMLR, 40–48.

[43] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. In *Advances in Neural Information Processing Systems*, Vol. 33. 5812–5823. https://proceedings.neurips.cc/paper/2020/file/3fe230348e9a12c13120749e3f9fa4cd-Paper.pdf

[44] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2020. GraphSAINT: Graph Sampling Based Inductive Learning Method. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJe8pkHFwS

[45] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks *(NIPS'18)*.

[46] 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.

[47] Zaixi Zhang, Qi Liu, Hao Wang, Chengqiang Lu, and Cheekong Lee. 2022. ProtGNN: Towards Self-Explaining Graph Neural Networks. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022*. AAAI Press, 9127–9135. https://doi.org/10.1609/AAAI.V36I8.20898

[48] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. GraphSMOTE: Imbalanced Node Classification on Graphs with Graph Neural Networks. In *WSDM (WSDM '21)*. https://doi.org/10.1145/3437963.3441720

[49] Yu Zhou, Haixia Zheng, Xin Huang, Shufeng Hao, Dengao Li, and Jumin Zhao. 2022. Graph Neural Networks: Taxonomy, Advances, and Trends. *ACM Transactions on Intelligent Systems and Technology* 13, 1 (Jan. 2022), 1–54. https://doi.org/10.1145/3495161

[50] Huaisheng Zhu, Guoji Fu, Zhimeng Guo, Zhiwei Zhang, Teng Xiao, and Suhang Wang. 2023. Fairness-aware Message Passing for Graph Neural Networks. arXiv:2306.11132 [cs.LG]

[51] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2020. Deep Graph Contrastive Representation Learning. In *ICML Workshop on Graph Representation Learning and Beyond*. http://arxiv.org/abs/2006.04131

A TRAINING ALGORITHM

The DPGBA algorithm is detailed in Algorithm 1. Initially, we identify the poisoned nodes $\mathcal{V}_P$ and label them with the target class y_t (lines 3-4). From lines 5-13, the trigger generator f_g is trained to both attack the surrogate model f_s and deceive the OOD detector f_d, utilizing a bi-level optimization approach. Specifically, in the lower level, we update the surrogate model (lines 6-8) and the OOD detector (lines 9-11) through gradient descent on θ_s and θ_d, respectively, guided by Eq. (9) for f_s and Eq. (10) for f_d. In the upper level, the generator f_g is updated (line 12) by applying gradient descent on θ_g, as outlined in Eq. (11). After that, from line 14 to 17, we use the well-trained f_g to generate a trigger g_i for each poisoned node $v_i \in \mathcal{V}_P$ and attach g_i with v_i to obtain the poisoned graph $\mathcal{G}_B$.

B REPRESENTATIVE NODES SELECTION

For selecting representative nodes $\mathcal{V}_s$ from a clean graph $\mathcal{G}$, we use the outlier detection method outlined in DOMINANT [10]. This approach is first applied to $\mathcal{G}$ to determine the mean μ and standard deviation δ of the reconstruction losses. Representative nodes are then selected based on their reconstruction loss d, ensuring that $d < \mu + \gamma\delta$. The parameter γ is set to 1 for the Cora dataset and adjusted to 0.01 for the Pubmed, Flicker, and OGB-arxiv datasets.

Algorithm 1 Algorithm of DPGBA

Input: Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}), \mathcal{Y}_L, \beta, T$
Output: Backdoored dataset $\mathcal{G}_B$, trigger generator f_g
1: Initialize $\mathcal{G}_B = \mathcal{G}$;
2: Randomly initialize θ_s, θ_d and θ_g for f_s, f_d and f_g;
3: Randomly select poisoned nodes $\mathcal{V}_P$;
4: Assign class t as labels of $\mathcal{V}_P$;
5: **while** not converged yet **do**
6:    **for** $t = 1, 2, \ldots, N$ **do**
7:       Update θ_s by descent on $\nabla_{\theta_s}\mathcal{L}_s$ based on Eq. (9);
8:    **end for**
9:    **for** $t = 1, 2, \ldots, K$ **do**
10:       Update θ_d by descent on $\nabla_{\theta_d}\mathcal{L}_D$ based on Eq. (10);
11:    **end for**
12:    Update θ_g by descent on $\nabla_{\theta_g}\mathcal{L}_T + \alpha\mathcal{L}_D + \beta\mathcal{L}_E$ based on Eq. (11);
13: **end while**
14: **for** each $v_i \in \mathcal{V}_P$ **do**
15:    Generate the trigger g_i for v_i by using f_g;
16:    Update $\mathcal{G}_B$ based on $a(\mathcal{G}_B^i, g_i)$;
17: **end for**
18: **return** $\mathcal{G}_B$ and f_g;

C DATASETS DETAILS

- **Cora** and **PubMed** [42]: They are citation networks where nodes denote papers, and edges depict citation relationships. In Cora and CiteSeer, each node is described using a binary word vector, indicating the presence or absence of a corresponding word from a predefined dictionary. In contrast, PubMed employs a TF/IDF weighted word vector for each node. For all three datasets, nodes are categorized based on their respective research areas.
- **Flicker** [44]: In this graph, each node symbolizes an individual image uploaded to Flickr. An edge is established between the nodes of two images if they share certain attributes, such as geographic location, gallery, or user comments. The node features are represented by a 500-dimensional bag-of-word model provided by NUS-wide. Regarding labels, we examined the 81 tags assigned to each image and manually consolidated them into 7 distinct classes, with each image falling into one of these categories.
- **OGB-arxiv** [17]: It is a citation network encompassing all Computer Science arXiv papers cataloged in the Microsoft Academic Graph. Each node is characterized by a 128-dimensional feature vector, which is derived by averaging the skipgram word embeddings present in its title and abstract. Additionally, the nodes are categorized based on their respective research areas.

D ADDITIONAL RELATED WORKS

Graph outlier detection is a critical task in machine learning, involving the identification of anomalous nodes within a graph. Unlike traditional outlier detection on tabular or time-series data, graph outlier detection presents unique challenges due to the rich information inherent in graph structures and the computational complexity of training with complex machine learning models. The emergence of deep learning techniques has revolutionized outlier detection,

Table 4: Results of backdooring GCN (ASR (%) | Clean Accuracy (%)). Only clean accuracy is reported for clean graph.

Datasets	Defense	Clean Graph	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA
Cora	None	83.6	33.8±5.2 | 83.4±0.8	48.3±8.2 | 83.3±0.5	94.6±1.6 | 83.6±1.0	98.3±0.1 | 83.5±0.8	97.7±1.4 | 83.3±0.8
	OD	83.4	33.4±4.9 | 82.9±0.6	47.8±9.3 | 83.7±0.6	0±0.0 | 83.4±0.7	0±0.0 | 83.7±0.5	94.4±1.1 | 83.5±1.0
Pubmed	None	85.1	36.5±11.4 | 85.1±0.2	36.1±3.7 | 85.0±0.1	88.8±1.7 | 85.1±0.2	93.1±1.3 | 85.1±0.2	92.3±1.8 | 85.0±0.2
	OD	85.1	35.8±12.1 | 85.2±0.1	35.5±3.2 | 85.2±0.1	0±0.0 | 85.3±0.3	0±0.0 | 85.2±0.1	91.2±1.2 | 85.1±0.2
Flickr	None	46.2	0±0.0 | 45.5±0.2	0±0.0 | 45.5±0.1	99.9±0.1 | 45.0±0.3	96.9±2.3 | 44.8±0.4	98.8±1.6 | 46.4±0.4
	OD	46.0	0±0.0 | 45.8±0.4	0±0.0 | 45.3±0.2	0±0.0 | 45.3±0.4	0±0.0 | 44.4±0.3	96.0±1.7 | 45.9±0.2
OGB-arxiv	None	66.2	35.2±5.7 | 65.9±0.1	48.5±4.2 | 65.9±0.1	83.6±2.8 | 65.3±0.3	99.4±0.1 | 65.3±0.5	95.6±0.8 | 65.8±0.4
	OD	65.9	34.6±6.6 | 65.6±0.2	47.2±4.5 | 65.4±0.3	0±0.0 | 64.7±0.3	0±0.0 | 65.4±0.5	93.2±1.0 | 65.5±0.3

Table 5: Results of backdooring GraphSage (ASR (%) | Clean Accuracy (%)). Only clean accuracy is reported for clean graph.

Datasets	Defense	Clean Graph	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA
Cora	None	83.8	34.2±4.0 | 83.0±1.5	40.4±5.6 | 82.7±1.2	96.0±3.3 | 83.3±1.2	92.7±2.1 | 83.6±1.4	95.3±1.5 | 83.7±1.2
	OD	83.6	33.9±3.3 | 82.5±1.1	38.3±5.3 | 83.1±1.3	0±0.0 | 82.8±1.1	0±0.0 | 83.3±1.7	91.2±1.1 | 83.5±0.8
Pubmed	None	84.9	38.0±3.8 | 84.8±0.3	40.0±4.2 | 84.9±0.2	89.0±6.4 | 84.9±0.2	90.2±1.0 | 85.1±0.1	91.8±1.3 | 85.1±0.4
	OD	85.0	37.8±3.3 | 85.4±0.2	39.2±5.3 | 84.8±0.2	0±0.0 | 85.1±0.5	0±0.0 | 85.0±0.1	91.0±1.2 | 85.2±0.3
Flickr	None	46.0	0±0.0 | 45.5±0.1	0±0.0 | 45.4±0.1	99.7±0.2 | 46.0±0.3	91.5±2.1 | 45.7±0.3	94.8±1.8 | 45.6±0.2
	OD	46.3	0±0.0 | 45.0±0.3	0±0.0 | 45.1±0.1	0±0.0 | 46.3±0.1	0±0.0 | 46.0±0.2	93.3±2.4 | 45.7±0.3
OGB-arxiv	None	65.8	33.0±5.6 | 66.1±0.4	38.7±1.9 | 66.1±0.3	99.6±0.3 | 64.4±0.4	97.7±0.1 | 65.5±0.1	94.2±0.8 | 65.8±0.6
	OD	66.1	32.2±5.9 | 65.6±0.6	37.6±2.3 | 65.8±0.2	0±0.0 | 64.8±0.3	0±0.0 | 65.6±0.2	91.8±0.6 | 65.4±0.4

Table 6: Results of backdooring GAT (ASR (%) | Clean Accuracy (%)). Only clean accuracy is reported for clean graph.

Datasets	Defense	Clean Graph	SBA-Samp	SBA-Gen	GTA	UGBA	DPGBA
Cora	None	84.3	37.5±8.7 | 84.0±1.3	47.1±18.0 | 83.7±1.1	84.1±3.8 | 83.9±0.9	94.3±1.4 | 83.3±0.7	97.1±1.7 | 83.7±1.0
	OD	84.4	36.8±7.7 | 83.6±2.2	46.3±17.6 | 83.9±1.3	0±0.0 | 84.1±0.6	0±0.0 | 83.7±0.9	96.0±2.0 | 83.6±0.8
Pubmed	None	85.2	26.9±4.5 | 84.1±0.3	27.1±3.8 | 83.9±0.2	86.4±2.6 | 83.8±0.2	94.2±1.5 | 85.4±0.1	93.8±2.6 | 85.1±0.1
	OD	84.9	26.3±3.7 | 84.2±0.4	25.8±4.4 | 83.6±0.2	0±0.0 | 83.6±0.5	0±0.0 | 84.8±0.3	93.3±2.1 | 85.0±0.2
Flickr	None	46.7	0±0.0 | 46.5±0.2	0±0.0 | 46.6±0.4	66.2±34.9 | 44.0±0.6	96.2±4.2 | 45.6±0.3	95.7±4.4 | 45.6±0.2
	OD	46.4	0±0.0 | 46.9±0.4	0±0.0 | 46.4±0.7	0±0.0 | 43.7±0.4	0±0.0 | 45.8±0.4	95.1±3.6 | 45.8±0.3
OGB-arxiv	None	65.6	39.7±7.2 | 65.3±0.3	43.0±10.4 | 65.6±0.4	94.3±2.5 | 64.8±0.1	97.6±0.1 | 65.1±0.2	95.4±1.3 | 65.2±0.2
	OD	65.3	38.3±6.1 | 65.5±0.5	42.1±11.3 | 65.3±0.3	0±0.0 | 65.1±0.5	0±0.0 | 65.5±0.4	92.1±0.9 | 65.4±0.3

shifting from traditional methods to neural network approaches [30]. One popular neural network architecture for this task is the autoencoder (AE) [19], which learns to reconstruct the original data and identifies outliers based on reconstruction errors. This unsupervised learning approach makes AEs effective for detecting outliers without the need for labeled data. Furthermore, graph neural networks (GNNs) have demonstrated superior performance in capturing complex patterns within graph data, considering both node attributes and graph structure. GNNs encode representations for each node, enabling effective outlier detection. Notably, GNNs can be combined with AEs [2, 6, 10, 12, 20, 41], leveraging the strengths of both approaches for more robust outlier detection in graph data.

E EXPERIMENTS ON ATTACK TRANSFERABILITY

To demonstrate the transferability of our trigger generator in attacking various GNN architectures, we employ GCN as the surrogate model and evaluate the ASR and clean accuracy when attacking GCN, GraphSage [16] and GAT [36], respectively. The results are presented in Tab. 4 – 6. From the tables, we observe that our DPGBA consistently achieves a high attack success rate while maintaining the clean accuracy across different target models and various

defense settings. This indicates the adaptability and transferability of our framework, enhancing its practical value in real-world applications.

F AGAINST VARIOUS OUTLIER DETECTION METHODS

To further demonstrate the in-distribution property of the triggers generated by our DPGBA, we adopt various state-of-the-art graph outlier detection methods, including DOMINANT [10], DONE [2] and its variant AdONE, AnomalyDAE [12], GAAN [5] and CONAD [41], as defense mechanisms and conduct backdoor attacks on four datasets. The other settings are the same as the evaluation protocol in Sec. 5.1.3. The results of ASR are reported in Tab. 7. From the table, we observe that DPGBA consistently exhibits its capability to evade various graph outlier detection methods and maintain a high attack success rate. This consistency underscores the practical application value of DPGBA in real-world scenarios.

G ADDITIONAL EXPERIMENTS

To demonstrate the robust adaptability of the trigger generator within DPGBA, we compare DPGBA with UGBA [8] using the defense strategy **Prune** proposed in [8], which involves removing

Table 7: Backdoor attack results against various graph outlier detection methods

Defense	Cora	Pubmed	Flicker	OGB-arxiv
None	97.7	92.3	98.8	95.6
DOMINANT	94.4	91.2	96.0	93.2
DONE	94.3	90.9	97.3	94.4
AdONE	95.7	92.0	98.0	93.4
AnomalyDAE	96.4	91.7	97.4	95.1
GAAN	96.8	91.8	98.6	94.9
CONAD	96.6	91.3	98.5	94.7

edges connecting nodes with low cosine similarity. All experimental configurations adhere to the evaluation protocol outlined in Section 5.1.3. Following [8], we incorporate the unnoticeable loss proposed in [8] to ensure that generated triggers exhibit high cosine similarity to target nodes. We set the pruning threshold to exclude approximately 10% of dissimilar edges. Table 8 presents the results of ASR and clean accuracy. From the table, we observe that DPGBA consistently exhibits comparable ASR and slightly higher clean accuracy compared to UGBA across four datasets. Notably, generated triggers in DPGBA maintain in-distribution property, whereas UGBA fails to evade detection by outlier detection methods. These findings indicate the superior performance and robustness of DPGBA in diverse settings.

Table 8: Backdoor attack results (ASR (%) | Clean Accuracy (%)). Only clean accuracy is reported for clean graphs.

Datasets	Defense	Clean Graph	UGBA	DPGBA
Cora	OD	83.4	0.0 | 83.7	94.4 | 83.5
	Prune	83.6	95.9 | 82.5	91.8 | 85.2
Pubmed	OD	85.1	0.0 | 85.2	91.2 | 85.1
	Prune	85.1	89.1 | 85.4	88.6 | 85.1
Flickr	OD	46.2	0.0 | 44.4	96.0 | 45.9
	Prune	45.3	99.7 | 41.7	94.7 | 45.9
OGB-arxiv	OD	65.8	0.0 | 65.4	93.2 | 65.5
	Prune	66.3	93.4 | 63.0	90.4 | 67.5

H DETAILS OF COMPARED METHODS

The details of compared methods are described following

- **SBA-Samp** [46]: This method introduces a static subgraph as a trigger into the training graph for each poisoned node. The subgraph's connections are formed based on the Erdos-Renyi (ER) model, while its node features are randomly selected from those in the training graph.
- **SBA-Gene**: An adaptation of SBA-Samp, SBA-Gen differentiates itself by employing synthetically generated features for the trigger nodes. These features are drawn from a Gaussian distribution, the parameters of which—mean and variance—are derived from the attributes of actual nodes.
- **GTA** [37]: GTA utilizes a trigger generator that crafts subgraphs as triggers tailored to individual samples. The optimization of the trigger generator focuses exclusively on the backdoor attack loss, disregarding any constraints related to trigger detectability.

Table 9: Training Time

Metrics	GTA	UGBA	DPGBA
ASR (None)	92.0	94.3	92.7
ASR (OD)	0.00	0.00	92.0
Time	37.7s	41.8s	60.6s

- **UGBA** [8]: UGBA select representative and diverse nodes as poisoned nodes to fully utilize the attack budget. An adaptive trigger generator is optimized with an constraint loss so that the generated triggers are ensured to be similar to the target nodes.

I IMPLEMENTATION DETAILS

A 2-layer GCN is utilized as the surrogate model, another 2-layer GCN is used for f_d, while a 2-layer MLP serves as the in-distribution trigger generator. We set all hidden dimensions to 256. The number of inner iteration steps, N and K, are consistently set to 1 and 20 across all experiments. The hyperparameters α and β are selected based on the grid search on the validation set. For the **OD** defense, the pruning threshold is set to exclude roughly 3% of the samples with the highest reconstruction loss.

J TIME COMPLEXITY ANALYSIS

During the bi-level optimization phase, the computation cost of each outer iteration consist of updating of surrogate GCN model and OOD detector in inner iterations and training adaptive trigger generator. Let h denote the embedding dimension. The cost for updating the surrogate model is approximately $O(Nhd|\mathcal{V}|)$, where d is the average degree of nodes and N is the number of inner iterations for the surrogate model, which is generally small. The cost for updating the OOD detector is approximately $O(Khd(|\mathcal{V}_s| + |\mathcal{T}_P|))$, where K is the number of inner iterations for the OOD detector. For trigger generator, the cost for optimizing $\mathcal{L}_T$ is $O(hd|\mathcal{V}_U|)$, for optimizing $\mathcal{L}_D$ is $O(hd(|\mathcal{V}_s| + |\mathcal{T}_P|))$, and for optimizing $\mathcal{L}_E$ is $O(hd|\mathcal{V}| + |\mathcal{V}_U|^2 h + |\mathcal{V}_U||\mathcal{V}_t||\mathcal{V}_L|h)$, where $|\mathcal{V}_L|$ and $|\mathcal{V}_t|$ are generally small compared to $|\mathcal{V}|$. In our empirical experiments conducted on large-scale datasets, such as Flickr and OGB-arxiv, which comprise 899,756 and 169,343 nodes respectively, we streamlined the training process by selecting a subset of $\mathcal{V}_U$ and setting $|\mathcal{V}_U| = 4096$ for each epoch. Despite this simplification, DPGBA still achieve a high attack success rate, as evidenced in Tab. 4 – 6. In Table 9, we report the overall training time and corresponding ASR of our DPGBA compared to GTA and UGBA on the OGB-arxiv dataset. All models were trained on a Nvidia A6000 GPU with 48GB of memory. The results indicate that DPGBA requires only approximately 20 seconds more training time compared to the baselines on the OGB-arxiv dataset. Given that our DPGBA achieves an ASR of over 90%, while the baseline methods achieve nearly 0% with OD defense adopted, this additional time is justified. This demonstrates that DPGBA effectively generates triggers that the victim model quickly memorizes, highlighting its potential for conducting scalable targeted attacks.