GDetox: Purifying Backdoor Encoder in Graph Self-supervised Learning via Knowledge Distillation

Hao Sui, Jiale Zhang, *Member, IEEE,* Bing Chen, *Member, IEEE,* Chengcheng Zhu, Chunpeng Ge, *Member, IEEE,* Weizhi Meng, *Senior Member, IEEE,* and Willy Susilo, *Fellow, IEEE*

***Abstract*—Graph Neural Networks (GNNs) have powerful representation capabilities for graph data, achieving excellent performance across various fields. Considering the scarcity of labels in real-world scenarios, graph self-supervised learning (GSSL) has gained increasing attention due to its ability to train without relying on labels. However, recent studies have revealed that GNNs are vulnerable to stealthy backdoor attacks in GSSL scenarios, enabling the encoder to learn backdoor features simply by injecting triggers. Existing graph backdoor defense methods mainly focus on supervised settings and cannot be directly transferred to self-supervised scenarios due to the lack of label guidance. To bridge this gap, we propose *GDetox*, the first backdoor defense approach against backdoored encoders in GSSL. *GDetox* aims to eliminate backdoor logic in encoders while maintaining the encoder's original performance. Specifically, *GDetox* can purify the graph backdoor encoder based on the self-supervised distillation approach without relying on label information. Further, we introduce an adversarial contrastive learning that augments node representations without relying on labels to enhance teacher model performance, thereby improving distilled encoder performance. We evaluate the defense performance of *GDetox* on four node classifications and four graph classification datasets by comparing with four state-of-the-art (SOTA) defense methods against seven latest backdoor attack methods on GSSL. Extensive experiments demonstrate that *GDetox* far outperforms the SOTA defense methods, reducing the attack success rate to 4% with negligible degradation in encoder performance (within 2%) in both node-level and graph-level tasks.**

***Index Terms*—Backdoor defense, Graph neural networks, Adversarial contrastive learning, Knowledge distillation.**

I. Introduction

GRAPH-STRUCTURED data play a pivotal role in a wide range of real-world applications, including financial networks [1], social networks [2], and molecular structure graphs [3]. Graph neural networks (GNNs) [4]–[6] are becoming increasingly popular as they can effectively process graph-structured data. Building upon the success of GNNs, graph self-supervised learning (GSSL) [7]–[9], as a novel graph learning paradigm, enables encoders to learn high-quality feature representations from unlabeled data. However, the process of obtaining well-trained encoders via GSSL is both time-consuming and computationally expensive, leading many practitioners to rely on third-party pre-trained encoders [10]–[12] available on public platforms, such as HuggingFace[1] and ModelZoo[2]. Regrettably, these third-party pre-trained encoders are highly likely to be backdoor encoders that have been injected with triggers by malicious attackers, posing a significant threat to the security and integrity of GNN-based systems.

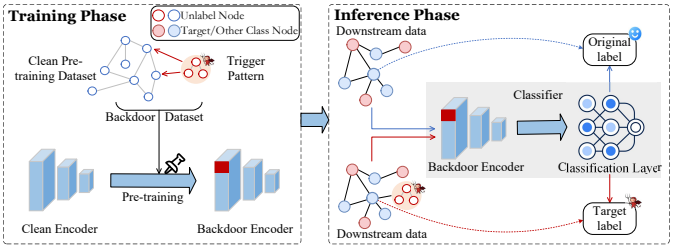


Fig. 1: The pipeline of the backdoor attack in GSSL.

As a typical type of attack, backdoor attacks pose a serious threat to the security of GNN models. Graph backdoor attacks are attracting increasing attention and have been extensively studied by several researchers, including supervised [13]–[20] and self-supervised [21], [22] scenarios. In supervised settings, the attacker initially selects backdoored samples from the original training dataset, subsequently injecting triggers into these samples and altering their labels to a predetermined target class. The graph backdoor model is then trained from the backdoored data along with other normal data. In GSSL, as shown in Fig. 1, attackers only inject triggers into backdoored samples and do not rely on label information to train the backdoor encoder in self-supervised scenarios. During inference, the backdoor encoder misclassifies downstream test samples injected with triggers into the target class while correctly identifying normal samples. Consequently, the inherent need for label modification may raise suspicion in supervised attacks, but is entirely absent in the self-supervised context, rendering graph self-supervised backdoor attacks markedly stealthier.

Currently, most graph backdoor defense mechanisms are focused on handling attacks in supervised settings and can be divided into detection-based and purification-based approaches. ❶ Detection-based methods involve using explainable techniques to identify triggers for detecting backdoors [23], [24]. However, these methods fail to further eliminate

Hao Sui and Bing Chen are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, 211106 (e-mail: {suihao36, cb_china}@nuaa.edu.cn).

Jiale Zhang and Chengcheng Zhu are with the School of Information Engineering, Yangzhou University, Yangzhou, China, 225127 (e-mail: jialezhang@yzu.edu.cn; chengchengzhu2022@126.com).

Chunpeng Ge is with the Joint SDU-NTU Centre for Artificial Intelligence Research (C-FAIR) & Software School, Shandong University, and the Quan Cheng Laboratory, Jinan, China, 250000 (e-mail: gechunpeng2022@126.com).

Weizhi Meng is with the School of Computing and Communications, Lancaster University, LA1 4YW Lancaster, U.K. (e-mail: weizhi.meng@ieee.org).

Willy Susilo is with the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, NSW 2522, Australia. (e-mail: wsusilo@uow.edu.au).

Jiale Zhang and Bing Chen are the corresponding authors.

[1]https://huggingface.co/models

[2]https://modelzoo.co/

the dangers of backdoors by identifying them. For example, Jiang et al. [25] and Yuan et al. [26] applied explainability evaluation metrics for pruning samples with large differences to eliminate backdoors. Yang et al. [27] employed Gaussian Mixture Models and filtering techniques to identify and eliminate backdoors. While effective, these methods may remove some normal samples, resulting in model performance degradation. ❷ Purification-based methods aim to purify the backdoor model into a clean model by removing backdoor information in the original data or eliminating backdoor neurons in this model. For instance, Zhang et al. [28] and Liu et al. [29] respectively leveraged robust training strategies and contrastive learning-based trigger detection methods to effectively neutralize backdoors. In addition, Yu et al. [30] employed discrepancy learning to eliminate the impact of label modification in backdoor attacks and leveraged a backdoor-free model to remove backdoors. Nevertheless, these methods rely on label information in supervised training settings and cannot precisely defend against self-supervised backdoor attacks in the absence of label information guidance. *Therefore, there is a critical need to develop a method for effectively defending against graph self-supervised backdoor attacks.*

To achieve the above objective, we propose *GDetox*, the first defense framework against graph self-supervised backdoor encoders. Essentially, we are faced with two main challenges: ❶ How to effectively purify backdoor encoders without relying on label information in graph self-supervised scenarios? ❷ How to maintain graph encoders' feature extraction capability while purifying backdoors in encoders?

Firstly, in real-world scenarios, defenders obtain potentially graph backdoor encoders from third parties, which means that only a small amount of the downstream dataset can be used for defense via fine-tuning technology. Moreover, the graph self-supervised backdoor attacks do not modify labels of backdoored samples, since self-supervised training is not reliant on label information. This mechanism increases the stealthiness of backdoor attacks and makes it impossible for defenders to purify backdoors with the aid of label information. *Thus, developing an effective approach to remove backdoors from graph self-supervised encoders under these constraints is a critical challenge*. To address this, we draw on knowledge distillation to purify backdoor information from the backdoored encoder, but this method is always deployed on logits. When using the labeled downstream dataset for distillation, it will close the distance between the target label and the backdoor features, making the deployment of distillation defense on logits ineffective (as shown in the experiment results in Fig. 7). Nevertheless, *GDetox* can overcome this by leveraging a combined loss that deploys attention alignment in hidden layers and BT loss in final embeddings, thus effectively purifying backdoor encoders in GSSL.

Secondly, the performance of the teacher model is crucial for successful knowledge distillation. Considering that if we use a backdoor encoder as the teacher model for attention alignment, it will weaken the encoder's original feature extraction capability. The traditional strategy for improving encoder performance is simple fine-tuning, but this method relies on label information and makes it difficult to generate robust encoders. *Thus, another key challenge is to maintain the encoder's original performance while purifying backdoors in encoders*. To tackle this, we incorporate graph contrastive learning to boost the teacher encoder's robustness before distillation. Yet standard contrastive learning [31] over multiple iterations can reduce the cosine similarity between original and perturbed graph views [32], inadvertently degrading performance. *GDetox* employs adversarial contrastive learning that maintains high similarity between clean and perturbed graph embeddings, thereby enhancing the teacher encoder and ensuring that the distilled model maintains high performance.

In summary, the contributions of our paper are:

- To our best knowledge, we devise the first backdoor defense scheme, named *GDetox*, capable of defending against graph backdoor attacks in self-supervised scenarios. *GDetox* can effectively defend against backdoors without relying on label information.
- We propose a self-supervised distillation approach that deploys attention alignment on hidden layers and BT loss on embeddings to purify graph backdoor encoders. On this basis, we further design an adversarial contrastive learning strategy to ensure the feature extraction ability of encoders by enhancing the teacher model's performance.
- We conduct extensive experiments on eight different real-world graph datasets to demonstrate the effectiveness of our defense method against the latest backdoor attacks in GSSL. Experimental results show that *GDetox* respectively outperforms all state-of-the-art (SOTA) defense methods on graph and node classification tasks.

II. Related Work

A. Backdoor Attack on GNNs

Backdoor attacks are adversarial attacks that pose a serious threat to model security. Existing research on graph backdoor attacks includes supervised and self-supervised scenarios.

In supervised settings, attackers train the backdoor model by injecting triggers into clean samples and modifying their labels to target labels. During the testing phase, samples with injected triggers will be predicted as target labels by the backdoor model, while clean samples will be predicted as original labels. We categorize the attacks into two types: ❶ **Both injecting triggers and modifying label information**. Zhang et al. [16] first proposed the graph backdoor attack by replacing the original subgraphs with designed fixed triggers and modifying their labels to the target label. Xu et al. [33], [34] used explainable techniques to inject fixed triggers at important locations. Based on this, studies [14], [15], [35]–[38] have designed adaptive triggers that can generate corresponding backdoor features based on different samples to enhance attack effectiveness. Considering that the trigger features generated by the above attacks are easily distinguishable from normal features, Zhang et al. [39] further employed adversarial generation strategies to create imperceptible in-distribution (ID) triggers to further improve the attack success rate. In addition, Xu et al. [40] and Wang et al. [41] proposed the multi-target backdoor attacks on GNNs, which design different triggers for each class. It is worth noting that our work focuses on backdoor defense in

self-supervised settings and does not address label-dependent multi-target backdoor attacks, which will be investigated in future research. ❷ **Only injecting triggers**. Xu et al. [18] were the first to propose a clean-label graph backdoor attack, injecting a fixed trigger without modifying the labels of backdoored samples. Xing et al. [19] and Fan et al. [20] improved attack effectiveness by maintaining feature similarity between target nodes and neighboring nodes and generating adaptive triggers, respectively. Although these clean-label attacks were developed in supervised settings, they do not alter sample labels, making them compatible with a self-supervised setting. In our experiments, we adapt these attacks to the self-supervised scenario to validate the effectiveness of *GDetox*.

In self-supervised scenarios, attackers can conduct backdoor attacks by training backdoored samples that only inject triggers without relying on label information, which increases the stealthiness of graph backdoor attacks. Zhang et al. [21] introduced graph backdoor attacks in self-supervised scenarios. It can leverage a small amount of downstream data to transform the pre-trained encoder into a backdoored encoder. Feng et al. [22] designed adaptive triggers and injected them into the original data for training to obtain graph backdoor encoders.

B. Backdoor Defense on GNNs

Several strategies have been proposed to defend against graph backdoor attacks, which can be mainly categorized into two types: ❶ *Backdoor detection*: Existing detection methods [23], [24] utilize graph explainable techniques to distinguish potential backdoor samples. ❷ *Backdoor purification*: Guan et al. [25] and Yuan et al. [26] distinguish between malicious edges and benign edges based on explainable evaluation scores and use iterative pruning to remove suspicious edges. Yang et al. [27] used a Gaussian model to filter test samples, which can purify backdoor models without accessing the training dataset. Furthermore, Sui et al. [42] and Zhang et al. [28] respectively employed counterfactual explanation and random edge removal strategies to defend against more stealthy ID backdoor attacks. Liu et al. [29] combined similarity metrics with contrastive learning to train a backdoor detector. Additionally, Yu et al. [30] used discrepancy learning to break the connection between backdoor features and the target label, and utilized the backdoor-free training to eliminate backdoors. Unfortunately, there is currently no defense research specifically targeting backdoor attacks in graph self-supervised scenarios.

III. Problem Definition and Threat Model

A. Problem Definition

In this section, we introduce the relevant definitions for graphs, including node-level representation learning and graph-level representation learning.

Node-level representation learning. In self-supervised scenarios, graph learning tasks include node-level representation learning and graph-level representation learning. For node-level representation learning, we assume obtaining an whole graph $G = \{V, E, X, A\}$, where V represents the set of all nodes in graph G, such as $V = \{v_1, ..., v_n\}$, and E represents the set of edges. X represents the initial features of each node, and A is the adjacency matrix representing the connection information of the edges in graph G, where $A = \{0, 1\}^{n \times n}$. Specifically, for two nodes v_i and v_j, $A_{ij} = 1$ means that these two nodes are connected; otherwise, they fail to form an edge. Node representation learning is for constructing an encoder $\mathcal{G}_0$ that represents the encoded features of each node. The encoded node features are represented as $\mathcal{G}_0(V, E, X, A) = \{h_1, h_2, ..., h_n\}$, h_n indicates the encoded feature information of the n-th node. We employ DGI [43], a typical method for learning node feature representations.

Graph-level representation learning. For graph-level representation learning, the graph dataset can be defined as $G = \{g_1, g_2, ..., g_n\}$, where g_n indicates the n-th graph in the graph data. This means that G is a series of graph data consisting of n graphs, each of which is composed of specific nodes and edges. In this case, we utilize the GCC method [44] to learn the specific feature representation of each graph via maximizing mutual information. Graph representation learning trains a high-quality encoder to represent the features of different graph data in the downstream dataset.

B. Threat Model

1) Attacker's Goal and Capability: These adversaries aim to embed specific triggers into the original data during the data processing phase and obtain a backdoor encoder through self-supervised training. Once this trigger is injected into the user's downstream dataset during the testing phase, a successful graph backdoor attack is achieved.

For backdoor attacks in GSSL, we evaluate them based on two aspects: attack effectiveness and attack evasion. **Effectiveness**: Given an initial GNN encoder $\mathcal{G}_o$, the attacker successfully embeds a trigger t into a original sample g_o without modifying its label information to construct the backdoor sample g_d. The trained encoder becomes the backdoor encoder $\mathcal{G}_d$. During the testing phase, the encoder is subjected to backdoor deception by misclassifying test samples with the original label y as the target label y_t. Specifically, the target label is designated by the attacker and is distinct from the original labels of backdoored samples. **Evasion**: Ensuring that $\mathcal{G}_o$ and $\mathcal{G}_d$ perform consistently on benign graphs. In summary, the attacker's objectives during the testing phase can be encompassed as follows:

$$\begin{cases} \mathcal{G}_d(g_d) = y_t \\ \mathcal{G}_d(g_o) = \mathcal{G}_o(g_o) = y \end{cases}, \tag{1}$$

Under supervised conditions, the adversary has direct access to the original graph dataset trained by users. However, in self-supervised scenarios, we define the adversary's capabilities into two cases based on the adversary's different backgrounds and knowledge: ❶ The adversary has access to the self-supervised pre-training dataset $G = (A, X)$ and injects triggers into a portion of this dataset without modifying the label information. The backdoored pre-training dataset is then published on a third-party data platform, enabling the adversary to launch an attack on users who download and utilize this backdoored dataset. ❷ The adversary does not have direct access to the pre-training dataset and can only access the

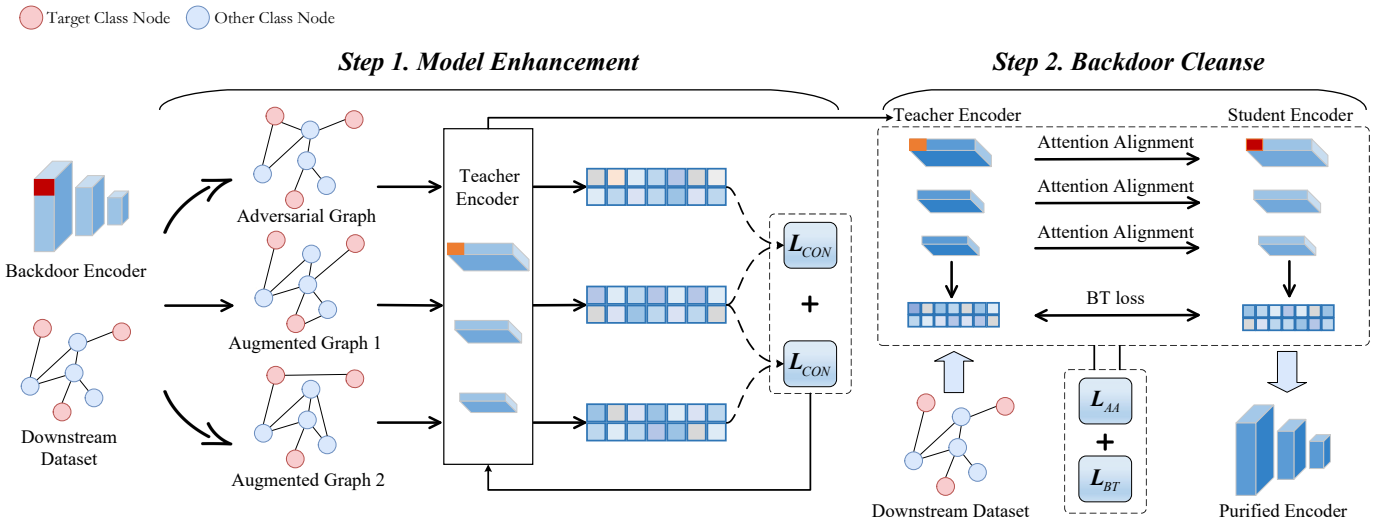


Fig. 2: The overview of *GDetox*. In *Stage 1*, to enhance teacher model performance by adversarial contrastive learning on a small number of downstream datasets. In *Stage 2*, to cleanse the backdoor in the encoder by deploying attention alignment in hidden layers and BT loss in final embeddings via the enhanced teacher model.

already pre-trained encoder. They modify the original encoder into a backdoor encoder with the help of downstream datasets. Regardless of the adversary's capabilities, the adversary's goal is to launch a successful backdoor attack. Therefore, in the *GDetox* setup, the adversary can fully manipulate data processing and training processes to execute effective graph self-supervised backdoor attacks.

2) Defense Goal and Capability: The defender's goal is to purify the impact of the backdoor on the backdoor encoder while ensuring that the purified encoder's performance on clean data is not compromised in self-supervised scenarios.

In terms of the defender's capabilities, we assume that the defender cannot access the pre-trained dataset injected with the backdoor, but can only access the backdoor encoder. The defender can use a small amount of downstream task datasets to assist in purifying graph backdoor encoders [27], [45]. This increases the challenge for defenders to defend against backdoor attacks in graph self-supervised scenarios.

IV. Proposed Defense Method

A. Defense Intuition and Challenges

Our objective is to cleanse the impact of the backdoor in the graph backdoor encoder. As illustrated in Fig. 1, the core idea of backdoor attacks in graph self-supervised learning is that the attacker injects triggers into a portion of the target class of data to closely associate the features of the trigger with the target class during the model training phase. Self-supervised backdoor attacks are even more stealthy because they do not require modifying the labels of backdoored samples, which greatly increases the difficulty of defense. Inspired by advancements in backdoor defense within the CV domain, a straightforward and intuitive approach to backdoor encoder purification is knowledge distillation. This method leverages a portion of the clean data fine-tuned model as a teacher model to guide the backdoor model, thus accomplishing backdoor purification. However, applying the aforementioned knowledge distillation to defend against backdoor attacks in GSSL still poses the following challenges:

Challenge 1: how to effectively purify graph backdoor encoders in self-supervised scenarios?

Although knowledge distillation can be feasible in purifying backdoors in computer vision models, it cannot be directly transferred to the graph domain due to ignoring the inherent topology structure in graphs. Moreover, distillation is always deployed on the logits of the model, which relies on label information. When using labeled downstream datasets to distill the backdoor encoder, it brings the target label closer to the backdoor features, rendering the distillation defense measures deployed on the logits ineffective (as evidenced by the experiment results in Fig. 7). Thus, there is an urgent need for an effective purify backdoor encoder method tailored to graph self-supervised scenarios that can capture rich graph structural information without relying on labels.

Challenge 2: how to obtain robust teacher models to maintain graph encoders' feature extraction capability?

An effective teacher model is critical for successful knowledge distillation: if its performance is poor, the backdoor encoder cannot be effectively cleansed. Traditional approaches rely on simple fine-tuning to boost encoder performance, but this requires labeled data and still fails to produce a robust teacher model. This intricate requirement increases the challenge of distilling the backdoor while preserving the encoder's original performance. Consequently, there is an urgent demand for a more practical approach to obtaining robust teacher models to address this challenge.

B. Overview

To tackle the above challenges, a novel overview of *GDetox* is illustrated in Fig. 2. *GDetox* aims for backdoor purification against self-supervised backdoor encoders and consists of two primary modules: ❶ In the *Model Enhancement* module, *GDetox* leverages adversarial contrastive learning to explicitly expose and amplify trigger-sensitive representations in the

Algorithm 1: Algorithm of Model Enhancement

Input: Downstream Graph dataset $G = (A, X)$, Backdoor encoder $f_b(\cdot)$, Adversarial contrastive learning epochs E_A
Output: Enhanced encoder $f_e(\cdot)$
for *each epoch* $k \in [1, 2, \cdots, E_A]$ **do**
  $G_S \leftarrow$ Randomly sample a subgraph for each node in graph G;
  $G_1, G_2 \leftarrow$ Generate augmented graphs of each node's subgraph G_S;
  $G_{adv} \leftarrow$ Generate the adversarial of each node's subgraph G_S via Equations (7) and (8);
  $f_e(\cdot) \leftarrow$ Update model parameters to minimize the loss L_{ACL} via Equation (11);
end
Return $f_e(\cdot)$;

teacher encoder, thereby isolating trigger-sensitive neurons and attention channels. ❷ In the *Backdoor Cleanse* module, *GDetox* performs attention-aligned self-supervised distillation, which selectively transfers only benign semantic structures from the adversarial enhanced teacher model to the student model. Furthermore, we deploy the BT loss that removes shortcut correlations between benign features and trigger activations in the embedding space, thus suppressing backdoor neurons.

C. Model Enhancement

Consider that fine-tuning the model using the downstream dataset will bring the trigger closer to the target class, thus increasing the success rate of backdoor attacks. To avoid this problem, we employ a graph contrastive learning approach with unlabeled data to enhance teacher encoder performance. In graph contrastive learning, we first construct two data augmented graphs G_1 and G_2 on the original graph G by modifying the topology structure of the graph [7]. Then, the feature embedding of each node is computed via the encoder $f(\cdot)$. Finally, we compute the contrastive loss in the feature space, resulting in bringing positive samples closer together and pushing negative samples further apart, which refers to GCA [31] and can be defined as:

$$L_{CON}(G_1, G_2) = \frac{1}{2n}\sum_{i=1}^{n}(l(\mathbf{u}_i, \mathbf{v}_i) + l(\mathbf{v}_i, \mathbf{u}_i)), \quad (2)$$

$$l(\mathbf{u}_i, \mathbf{v}_i) = \log \frac{e^{\theta(\mathbf{u}_i, \mathbf{v}_i)/\tau}}{e^{\theta(\mathbf{u}_i, \mathbf{v}_i)/\tau} + \sum\limits_{j \neq i} e^{\theta(\mathbf{u}_i, \mathbf{v}_j)/\tau} + \sum\limits_{j \neq i} e^{\theta(\mathbf{u}_i, \mathbf{u}_j)/\tau}}, \quad (3)$$

where $l(\mathbf{u}_i, \mathbf{v}_i)$ indicates the loss for each contrastive instance $(\mathbf{u}_i, \mathbf{v}_i)$. $\theta(\mathbf{u}_i, \mathbf{v}_i)$ is denoted as the cosine similarity of embeddings $\mathbf{u}_i$ and $\mathbf{v}_i$. τ is a temperature parameter.

Nevertheless, the cosine similarity between the original and perturbed graphs is reduced in multiple iterations of contrastive learning [32], leading to a degradation in the performance of the teacher encoder. Thus, *GDetox* leverages an adversarial contrastive learning to enhance the performance of the teacher encoder, which lays the foundation for enhancing the performance of the post-distillation backdoor encoder.

To enhance the performance of graph encoders suffering from backdoor attacks, we assimilate graph contrastive learning and adversarial training. Adversarial training can improve the accuracy and robustness of the encoder obtained with contrastive training only by introducing samples generated with adversarial attacks for training. In the self-supervised scenario, the aim of the adversarial attacks generate the most challenging adversarial samples by maximizing the contrastive loss on both an adversarial view that adds imperceptible perturbations to the original graph and a data-augmented view of the same graph, which can be expressed as follows:

$$G_{\text{adv}} = \arg\max_{G^{'}} L_{CON}(G_1, G^{'}), \quad (4)$$

where G_1 denotes the augmented view graph derived from adding or removing edges of the original graph G. G_{adv} is generated by the original graph G. Notably, $G^{'}$ is constrained by Δ_{X} and Δ_{A}:

$$\sum_{i,j} |\mathrm{X}^{'}[i,j] - \mathrm{X}[i,j]| \leq \Delta_{\text{X}}, \quad (5)$$

$$\sum_{i,j} |\mathrm{A}^{'}[i,j] - \mathrm{A}[i,j]| \leq \Delta_{\text{A}}, \quad (6)$$

where $\mathrm{X}^{'}$ and $\mathrm{A}^{'}$ respectively denote the node features and the adjacency matrix of the graph $G^{'}$, $[i,j]$ denotes the node pair (v_i, v_j). Specifically, we define the structural budget as $\Delta_{\text{A}} = \delta_{\text{A}} \sum \mathrm{A}$, where δ_{A} is the edge modification ratio. For feature perturbations, we primarily enforce an element-wise c-norm constraint δ_{X}, such that the total change Δ_{X} is bounded by $n \times d \times \delta_{\text{X}}$, where n and d are the number of nodes and feature dimensions, respectively.

Taking into account the performance and time complexity of the algorithm, our approach respectively refers to the PGD method [46] and the PGD_GNN method [47] to adversarial attacks on node features and graph topology structure. Specifically, we define the node feature and graph adjacency matrix by adding perturbations as follows:

$$\mathrm{X}_{\text{adv}} = \mathrm{X} + \mathrm{C}_{\text{X}}, \quad (7)$$

$$\mathrm{A}_{\text{adv}} = \mathrm{A} + \mathrm{D} \circ \mathrm{C}_{\text{A}}, \quad (8)$$

where $\mathrm{C}_{\text{X}} \in \mathbb{R}^{n \times d}$ indicates the perturbation to be added to the node feature matrix. $\mathrm{C}_{\text{A}} \in \{0,1\}^{n \times n}$ is the symmetric matrix used to modify the edges in the graph, with element values of 1 indicating a modification and 0 being unchanged. $\mathrm{D} = \bar{\mathrm{A}} - \mathrm{A}$ is a discrepancy matrix used to distinguish the positions of added or deleted edges, and $\bar{\mathrm{A}} = 1_{n \times n} - \mathrm{I}_n - \mathrm{A}$ is used to construct the negative pair by fetching the supplement of the original graph. Ultimately, $\circ$ executes an element-wise product to determine whether edges are added or removed.

On this basis, we further optimize constraints C_{X} and C_{A} to control the added adversarial noise during the training process. For the perturbed node feature, we set the constraint C_{X} to satisfy $S_{\text{X}} = \{\mathrm{C}_{\text{X}} | \|\mathrm{C}_{\text{X}}\|_\infty \leq \delta_{\text{X}}, \mathrm{C}_{\text{X}} \in \mathbb{R}^{n \times d}\}$. Also, for the perturbed adjacency matrix, we set the constraint C_{X} to satisfy $S_{\text{A}} = \{\mathrm{C}_{\text{A}} | \sum_{i,j} \mathrm{C}_{\text{A}} \leq \Delta_{\text{A}}, \mathrm{C}_{\text{A}} \in [0,1]^{n \times n}\}$. Besides,

Algorithm 2: Algorithm of Backdoor Cleanse

Input: Downstream Graph dataset $G = (A, X)$, Backdoor encoder $f_b(\cdot)$, Enhanced encoder $f_e(\cdot)$, training learning epochs E_S
Output: Purified encoder $f_p(\cdot)$
for *each epoch* $k \in [1, 2, \cdots, E_S]$ **do**
  $G_S \leftarrow$ Compute the attention map on each layer of the model via Equation (13);
  $L_{AA} \leftarrow$ Compute the attention alignment loss based on attention maps of each layer via Equation (14);
  $L_{BT} \leftarrow$ Compute the BT loss based on the teacher model and student model embedding via Equation (15);
  $f_e(\cdot) \leftarrow$ Update model parameters to minimize the total loss L combining the losses L_{AA} and L_{BT} via Equation (16);
end
Return $f_e(\cdot)$;

we will update two constraints in each iteration of the training:

$$\mathrm{C}_{\mathrm{X}}^{(t)} = \Pi_{S_{\mathrm{X}}}\left[\mathrm{C}_{\mathrm{X}}^{(t-1)} + \eta_X \cdot \mathrm{sgn}(\nabla_{\mathrm{C_X}} L_{CON}(G_1, G_{\mathrm{adv}}^{(t-1)}))\right], \quad (9)$$

$$\mathrm{C}_{\mathrm{A}}^{(t)} = \Pi_{S_{\mathrm{A}}}\left[\mathrm{C}_{\mathrm{A}}^{(t-1)} + \eta_A \cdot \nabla_{\mathrm{C_A}} L_{CON}(G_1, G_{\mathrm{adv}}^{(t-1)})\right], \quad (10)$$

where $sgn(\cdot)$ is the element-wise signum function. η_X and η_A are hyperparameters affecting the generation of adversarial samples in the t-th round. $G_{\mathrm{adv}}^{(t-1)}$ is represented as $\{\mathrm{A} + \mathrm{D} \circ \mathrm{C}_{\mathrm{A}}^{(t-1)}, \mathrm{X} + \mathrm{C}_{\mathrm{X}}^{(t-1)}\}$. $\Pi_{S_{\mathrm{X}}}$ and $\Pi_{S_{\mathrm{A}}}$ respectively perform the projection operation for $\mathrm{C_X}$ and $\mathrm{C_A}$.

The adversarial contrastive loss can be defined as:

$$L_{ACL} = L_{CON}(G_1, G_2) + \lambda L_{CON}(G_1, G_{\mathrm{adv}}), \quad (11)$$

where λ is a weight coefficient to balance the two loss terms. L_{ACL} as a semantic filter to enhance teacher encoder performance. The enhancement process of the graph adversarial contrastive learning is shown in Algorithm 1.

D. Backdoor Cleanse

Recall that a challenge of defending backdoors in GSSL is that it is impossible to precisely remove the backdoor from the encoder. To overcome this challenge, *GDetox* employs the deployment of attention alignment in the hidden layer to accurately purify the backdoor by correcting the focus region of the backdoor encoder to the normal region instead of the anomalous region. It hires the adversarial enhanced encoder as the teacher model to distill the backdoor encoder treated as the student model.

For a graph self-supervised encoder f, we represent the activation map of the l-th layer as $f \in \mathbb{R}^{N \times C_l}$, where N denotes the number of nodes and C_l denotes the dimension of the l-th layer output channel. To enable an accurate representation of the contribution of each element to the classification result, we define the function $\mathcal{F} : \mathbb{R}^{C_l \times N} \rightarrow \mathbb{R}^N$ to be used to map the activation map to the attention representation. On this basis, we fully consider the topological structure information inherent in the graph data by adding the degree property $D = deg(V)$ of each node. Combining these two aspects, we set the attention operator $\mathcal{F}$ of the l-th layer as:

$$\mathcal{F}^p(f^l) = \frac{D}{\max(D)} \sum_{i=1}^{C_l} \left| f^{l(i)} \right|^p, \quad (12)$$

where $\max(D)$ is the maximum of all nodes D, $\frac{D}{\max(D)}$ is used to consider the degree property information of each node. The activation map of the i-th channel is denoted as $f^{l(i)}$. $|\cdot|$ is the absolute value function to ensure that $f^{l(i)}$ is nonnegative. p is used to amplify the distance between benign and malicious neurons, which we experimentally set to $p > 0$ and further validate the effect of p on our method in our experiments.

We need to distill each layer of neurons of the encoder using the attention operator. Moreover, we perform the normalization function on attention representations to ensure that the attention representations are consistent across layers, as follows:

$$\psi(\mathcal{F}^l) = \frac{\mathcal{F}^l}{\|\mathcal{F}^l\|_2}, \quad (13)$$

where $\psi(\cdot)$ is a normalization function. We fix the teacher model and align the backdoor neurons in the student model with the clean neurons in the teacher model to achieve backdoor purification, which can be indicated as:

$$L_{AA}(f_T^l, f_S^l) = \left\| \psi(\mathcal{F}_T^l) - \psi(\mathcal{F}_S^l) \right\|_2, \quad (14)$$

where $\psi(\mathcal{F}_T^l)$ and $\psi(\mathcal{F}_S^l)$ respectively denote the teacher-model and student-model attention maps of the l-th layer, and $\left\| \psi(\mathcal{F}_T^l) - \psi(\mathcal{F}_S^l) \right\|_2$ is the difference in distribution between them computed by the L_2 paradigm.

Based on this, we further introduce the BT loss [48] to enhance the performance of the purified encoder by computing the cross-correlation matrix of a graph between the teacher model and the student model representations. It ensures the consistency of the feature diagonal elements and the independence of the feature off-diagonal elements to learn more useful information under different encoders. Formally,

$$L_{BT} = \sum_i (1 - C_{ii}) + \mu \left(\sum_i \sum_{j \neq i} C_{ij} \right), \quad (15)$$

where C denotes the cross-correlation matrix of teacher and student model outputs for the identical graph across different dimensions. μ controls the weights between consistency of diagonal elements and independence of non-diagonal elements.

Ultimately, the total loss of the backdoor cleanse algorithm can be defined as:

$$L = \alpha \sum_{l=1}^{k} L_{AA}\left(f_T^l, f_S^l\right) + \beta L_{BT}, \quad (16)$$

where α and β are two hyperparameters to balance the performance of the purified backdoor encoder, k denotes the number of layers of the encoder. The backdoor cleansing process of our method is shown in Algorithm 2. It is worth noting that we adopt a two-stage schedule: first, fully train the

teacher encoder with L_{ACL}, then fix the teacher and train the student encoder with L.

V. Performance Evaluation

This section will comprehensively evaluate the performance of *GDetox* on different datasets for node and graph classification tasks. To demonstrate the superiority and robustness of *GDetox*, we will answer four key research questions:

- **RQ1** (Effectiveness Analysis): Can it be demonstrated that the superiority of *GDetox* compared to SOTA defense methods against five backdoor attacks in the node classification task, as well as the robustness of *GDetox* under different configurations, and the time complexity of *GDetox* on different datasets?
- **RQ2** (Ablation Analysis): How significant is the impact of different components in our method on purifying attacks and ensuring model performance?
- **RQ3** (Sensitivity Analysis): What is the effect of *GDetox*'s defensive performance in various hyperparametric settings, such as λ, α, etc.?
- **RQ4** (Extensibility Analysis): How effective is our method in defending against different backdoor attacks in graph classification tasks?

TABLE I: **Analysis of the node classification datasets**

Datasets	Cora	Citeseer	PubMed	Flickr
#Nodes	2,708	3,327	19,717	89,250
#Edges	5,429	4,732	44,338	899,756
#Feature	1,433	3,703	500	500
#Classes	7	6	3	7
#Target label	1	3	1	0

A. Experimental Settings

1) Dataset: We leverage four real-world datasets to evaluate our defense method in the node classification task, including Cora, CiteSeer, PubMed, and Flickr [18], [49]. The specific information of these datasets is as follows and shown in Table I. In addition, to demonstrate the extensibility of our method, we selected eight graph datasets to validate the defensive performance of *GDetox* in graph classification tasks. Among them, we follow the settings in these relevant research [22], [44] to select the DBLP, IMDB, Academia, and Facebook datasets as self-supervised pre-training datasets [50], and the IMDB-B, IMDB-M, COLLAB, and RDT-M datasets [51] as downstream task datasets, which are specifically detailed in Table II and Table III.

TABLE II: **Analysis of pre-training dataset for the graph classification task.**

Datasets	Academia	DBLP	IMDB	Facebook
#Nodes	137,969	317,080	896,305	3,097,165
#Edges	739,384	2,099,732	7,564,894	47,334,788

TABLE III: **Analysis of downstream datasets for the graph classification task.**

Datasets	IMDB-B	IMDB-M	COLLAB	RDT-M
#Graphs	1,000	1,500	5,000	5,000
Avg. #Nodes	429.6	13	74.5	508.5
Avg. #Edges	96.53	65.94	2457.78	594.87
#Classes	2	3	3	5
#Target label	1	1	1	1

2) Attack Methods: In our experiment, we select seven latest graph backdoor attack methods, including GRBA [22], GCBA [21], CGBR [19], ECGBA [20], CLBA [18], GTA [15], and BadGraph [16], to verify the superiority of the proposed approach. Given that only GRBA and GCBA were originally designed for self-supervised settings, we also introduced three clean-label attacks (CGBR, ECGBA, and CLBA) and two label-modified attacks (GTA and BadGraph) from the supervised domain, transferring them to the self-supervised scenario by following the settings in [22]. For the node classification task, we choose GRBA, GCBA, CGBR, ECGBA, and CLBA as backdoor attack methods; for the graph classification task, we apply GRBA, CLBA, GTA, and BadGraph as backdoor attack methods. These methods are specifically described as follows: (*i*) GRBA was a graph backdoor attack method in unsupervised learning, which first designed adaptive triggers and then injected triggers to train for obtaining an effective attack encoder; (*ii*) GCBA enabled graph backdoor attacks in the encoder training phase, which utilized a small number of downstream datasets to pollute clean pre-trained encoders as backdoor encoders; (*iii*) CGBR only maintained the feature similarity between the target node and neighbor nodes, thus achieving the stealthy graph backdoor attack; (*iv*) ECGBA injected generative adaptive triggers in target samples without modifying the label information to achieve clean-label graph backdoor attack; (*v*) CLBA was the first Clean-label graph backdoor attack that uses the Erdős-Rényi (ER) model to generate trigger replacements for the original subgraphs in the graph without modifying the original labels of the backdoored nodes; (*vi*) GTA used encoders to generate adaptive triggers to achieve attack effectiveness; (*vii*) BadGraph utilized the injection of fixed triggers to pollute the original graphs.

3) Defense Baselines: To verify the superiority of GDetox, we compare four typical SOTA defense methods in node and graph classification tasks, including BloGBaD [27], E-SAGE [26], SimGuard [29], and DShield [30]. Notably, since there are currently no defense methods specifically designed for graph backdoor attacks in self-supervised scenarios, we select these graph backdoor defense techniques that could be transferred from supervised scenarios to self-supervised scenarios and evaluate them. These defense methods are specifically described as follows: (*i*) BloGBaD proposed a black-box GNN backdoor defense strategy to defend against the backdoor attack without using training data; (*ii*) E-SAGE defended against the impact of backdoors by an iterative edge pruning method during the testing phase; (*iii*) SimGuard designed contrastive learning-based trigger detectors to defend against backdoors;

TABLE IV: **Hyperparameter Lookup Table.**

Parameter	Module	Default	Auto-tuning guidance
η_X	Adversarial	0.01	Set η_X and η_A decrease
η_A	Contrastive Learning	0.01	if the contrastive loss oscillates
δ_{X}	Adversarial	0.5	Select δ_X and δ_A based on
δ_{A}	Perturbation Budget	0.1	graph scale and adversarial strength
λ	Adversarial Contrastive Loss Balancing	1	Increase λ if the ACC remains below the target threshold.
E_A	ACL Training epoch	500	Determine E_A by identifying the plateau where the ACL loss curve converges.
E_S	SSD Training epoch	300	Select based on the trade-off between purification efficiency and overfitting.
p	Attention Alignment	2	Adjust p to maximize the distance between benign and malicious neurons
μ	BT loss	0.01/0.005	Choose μ by matching diagonal correlation to target
α	Backdoor Cleanse	1	Optimize α and β by balancing the
β	Loss Balancing	0.5	performance metrics of the purification encoder

(*iv*) DShield employed the idea of discrepancy learning to defend against graph backdoor attacks. To align with the self-supervised setting, we employ these SOTA methods during defense without relying on the label information of samples.

4) Evaluation Metrics: To evaluate the performance of *GDetox*, we use two metrics: attack success rate (ASR) and accuracy (ACC). ASR is an attack evaluation metric used to assess the proportion of backdoored samples that an attack method successfully misclassifies as target labels. In our experiments, we use this metric to measure the effectiveness of *GDetox*'s defense against attacks. ACC is the metric used to evaluate the model's performance. Since the backdoor attack or defense can affect the model's performance. Therefore, we use ACC to measure the model's performance after defense.

5) Implementation Details: GDetox is implemented by the PyTorch framework in Python. The environment used in our experiments is Intel(R) Xeon(R) Silver 4310 CPU 377G (CPU), NVIDIA RTX A6000 (GPU), 75 GiB memory, and Ubuntu 20.04 (OS). Our experiments verify the defense performance of *GDetox* on the node and graph classification tasks. For the node classification task, we employ DGI as a pre-trained model. In the training phase, we randomly select 20% of the data as the training set and inject 5% of the data into the trigger as backdoor samples for training. The other 80% data is used as the test set. Among them, we select 5% of the data for defense. For the graph classification task, we employ GCC as a pre-trained model. In contrast to the node classification task, the pre-training dataset for the graph classification task is inconsistent with the downstream dataset. We choose the dataset in Table II to train the pre-trainer encoder. In the downstream dataset, we choose 5% of the data injected into the trigger to perturb the pre-trained encoder to become a backdoor encoder. On this basis, our experiment chooses 5% of the clean data to purify the backdoor encoder. For all experiments, we set the learning rate to 0.005 and the training epochs for adversarial contrastive learning (E_A) and self-supervised distillation (E_S) to 500 and 300, respectively. For our proposed method, we set the hyperparameters δ_{X}, δ_{A}, λ, η_X, η_A for the model enhancement to 0.5, 0.1, 1, 0.01, and 0.01, and the hyperparameters p, α, and β for the backdoor cleanse to 2, 1, and 0.5, respectively. It is important to emphasize that for the hyperparameter μ, we set it to 0.01 in Cora and Citeseer and to 0.005 in PubMed and Flickr. The impact of these hyperparameters on *GDetox* will be specifically verified in RQ4. In particular, for the fairness of our experimental results, we conduct five experiments and average the results to avoid the instability of a single result. Finally, we provide a streamlined hyperparameter lookup table (Table IV) to enhance transparency and reproducibility.

B. RQ1: Effectiveness Analysis

To answer RQ1, we will compare the defense performance of *GDetox* and four SOTA methods against the latest graph backdoor attack methods across different graph datasets in a self-supervised scenario. The four SOTA defense methods and latest attack methods have been given a description in Section V-A3 and Section V-A2, respectively. We further conduct a comprehensive analysis of *GDetox* under various configurations, including poisoning ratio, trigger influence scope, and proportion of downstream data used. Additionally, we analyze the time complexity of *GDetox*.

1) Comparing with SOTA methods: Table V presents the defense performance of *GDetox* and SOTA defense methods against SOTA self-supervised graph backdoor attacks across different datasets. It is worth noting that Table V highlights the results of our method in Grey and bolds the best ASR and ACC results among all defense methods. To ensure a rigorous evaluation of statistical significance and reproducibility, we report the mean values along with 95% confidence intervals (CIs) over 5 independent runs. The experimental results show that *GDetox* achieves a lower ASR than all SOTA defenses, indicating its superiority in mitigating various backdoor attacks in graph self-supervised scenarios. At the same time, *GDetox*'s model accuracy also exceeds that of all SOTA defenses, demonstrating that our method maintains strong model performance while providing effective defense. ACC of our method is within 2% of the difference between the pre-defense (Before) model, with a few post-defense models actually performing better than the attacked models.

It can be observed that BloGBaD's defense performance is generally inferior to ours. On the Citeseer dataset, *GDetox*'s ASR is approximately 20% lower than that of BloGBaD. This is mainly because BloGBaD relies on clustering to filter backdoor samples, which struggles to identify highly stealthy backdoor features in an unlabeled setting, resulting in degraded performance. Compared to BloGBaD, E-SAGE exhibits even lower defense performance. This is because E-SAGE uses an iterative pruning method based on explainability scores to remove backdoor edges; however, for CGBR, a backdoor attack method based on feature modification, it cannot directly eliminate backdoor features in node attributes. SimGuard shows ACC and ASR significantly better than BloGBaD and E-SAGE defenses, but still falls short of *GDetox*. Although SimGuard employs DBSCAN to detect triggers, attackers do not alter backdoored-sample labels in a self-supervised scenario, so a density-based clustering algorithm cannot effectively detect triggers. DShield's defense performance is close to *GDetox*; however, for GCBA and GRBA attacks, it is clearly weaker than *GDetox*. For example, on Cora under GCBA attacks, *GDetox*'s ASR is about 7% lower than that of DShield. This is because DShield uses a labeled graph and discrepancy matrix for clustering to distinguish potential backdoor samples, but

TABLE V: **The defense performance of *GDetox* compared with the SOTA methods (%).**

Dataset	Attack method	Before		BloGBaD(2023)		E-SAGE(2024)		SimGuard(2025)		DShield(2025)		*GDetox*	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
Cora	CLBA(2022)	77.32±0.54	59.72±0.79	69.73±0.69	15.77±1.21	66.83±0.47	15.63±0.35	70.11±0.71	9.35±0.31	74.37±0.52	2.01±0.44	**76.99±0.31**	**0.47±0.37**
	ECGBA(2024)	71.49±0.90	71.76±0.51	68.35±1.18	20.36±0.75	64.58±0.75	21.45±0.66	68.13±0.98	11.36±0.75	70.88±0.80	3.45±0.55	**72.03±0.41**	**1.14±0.29**
	CGBR(2024)	70.67±0.69	78.32±0.85	62.76±0.84	17.68±0.22	59.89±0.62	17.34±0.21	63.82±0.85	12.47±0.17	67.42±0.67	4.83±0.06	**70.14±0.33**	**1.95±0.18**
	GCBA(2023)	82.59±1.58	88.19±2.04	75.52±1.03	23.64±1.31	69.43±1.49	25.88±1.52	75.38±1.13	15.91±1.19	81.09±0.77	7.13±0.89	**82.96±0.53**	**0.37±0.31**
	GRBA(2024)	83.33±1.08	72.32±2.83	74.46±0.52	20.68±0.96	71.52±1.30	19.65±1.34	73.86±0.92	12.75±0.86	78.45±1.75	6.57±1.08	**81.85±1.31**	**2.21±1.12**
Citeseer	CLBA(2022)	75.27±0.59	71.31±0.80	66.63±0.77	15.43±1.19	61.12±0.43	17.56±0.91	67.15±1.47	12.62±0.85	71.65±1.57	3.52±0.65	**75.01±1.08**	**0.27±0.51**
	ECGBA(2024)	66.53±1.37	72.35±0.82	60.42±0.82	19.75±0.64	45.36±0.67	20.47±0.75	60.27±0.81	14.37±0.47	64.97±1.35	4.11±1.06	**66.87±0.66**	**0.87±0.41**
	CGBR(2024)	68.31±0.88	81.87±0.89	58.93±0.81	20.35±1.25	55.79±1.39	22.31±1.67	61.83±2.43	13.02±1.06	64.48±1.45	4.76±0.86	**67.68±1.07**	**1.46±0.82**
	GCBA(2023)	72.89±2.01	88.29±0.65	65.47±0.76	23.33±0.50	59.21±1.41	29.92±2.08	66.53±1.07	14.64±0.98	70.11±1.09	8.79±1.11	**72.29±0.48**	**3.65±0.74**
	GRBA(2024)	74.70±0.52	83.37±0.31	67.32±0.47	23.02±0.79	60.75±0.93	26.43±0.33	67.99±0.48	13.79±0.67	71.88±0.90	6.17±1.08	**74.17±0.59**	**2.46±1.03**
PubMed	CLBA(2022)	79.02±1.85	69.56±1.18	70.12±0.49	20.64±0.26	61.77±0.43	19.85±0.25	71.35±0.50	10.37±0.13	74.38±0.52	3.11±0.04	**78.27±0.61**	**0.35±0.21**
	ECGBA(2024)	77.58±0.94	65.46±0.65	71.03±0.58	15.47±0.19	62.51±0.94	22.58±1.28	73.35±0.81	12.59±2.16	76.01±1.23	5.09±1.06	**77.25±1.31**	**1.78±0.72**
	CGBR(2024)	73.48±1.31	68.95±0.47	68.22±0.58	14.87±1.19	60.92±0.43	28.52±1.36	68.99±0.88	14.37±0.98	70.22±0.49	4.78±0.66	**73.88±0.39**	**0.89±0.51**
	GCBA(2023)	83.79±0.59	81.64±0.89	76.31±0.53	16.83±0.21	65.36±0.46	23.51±0.29	77.25±0.54	15.48±0.19	81.78±0.57	6.21±0.08	**82.19±0.32**	**2.25±0.12**
	GRBA(2024)	84.88±2.19	77.99±1.45	78.62±0.25	14.72±0.78	67.82±0.37	20.92±0.46	79.62±1.36	11.68±0.84	82.33±0.58	4.94±0.76	**83.21±0.63**	**0.61±0.46**
Flickr	CLBA(2022)	47.28±1.33	32.98±0.41	42.85±1.00	9.98±0.92	39.42±0.98	16.23±0.20	43.29±1.03	9.03±0.11	46.15±2.22	3.67±0.55	**46.78±0.69**	**0.31±0.22**
	ECGBA(2024)	40.12±0.38	30.62±0.81	40.72±0.59	12.23±1.15	36.61±1.26	17.83±0.32	41.02±0.99	11.42±1.14	41.34±0.48	4.79±0.86	**41.93±0.47**	**0.93±0.51**
	CGBR(2024)	41.58±0.29	39.84±0.49	38.62±0.27	16.83±0.21	35.68±0.25	21.75±0.27	38.76±0.27	14.39±0.18	39.88±0.28	4.52±0.16	**41.18±0.16**	**0.62±0.22**
	GCBA(2023)	51.11±0.86	59.33±1.72	46.04±0.62	13.01±2.16	41.78±0.69	18.95±0.74	47.89±0.94	12.76±1.16	49.74±1.35	6.88±0.59	**50.73±0.29**	**1.12±0.73**
	GRBA(2024)	50.89±0.36	38.89±0.49	45.78±0.92	13.65±0.77	40.65±0.88	16.84±0.41	47.37±0.73	12.21±0.35	48.27±0.54	7.71±0.93	**50.36±0.42**	**0.76±0.25**

TABLE VI: **Comparison of lightweight baseline method.**

Dataset	LSSG		*GDetox*	
	ACC	ASR	ACC	ASR
Cora	0.7315	0.0742	0.8185	0.0221
Citeseer	0.6702	0.0811	0.7417	0.0246
Pubmed	0.7934	0.0573	0.8321	0.0061
Flickr	0.4692	0.0655	0.5036	0.0076

in the self-supervised setting, attackers inject triggers without modifying the label of backdoored samples, making the attack performance weak when using discrepancy learning to purify the attack. Overall, our method outperforms all SOTA defenses against the latest self-supervised graph backdoor attacks across different datasets, demonstrating the superiority of *GDetox*.

To investigate the efficacy of lightweight learned filters under unlabeled constraints, we introduce a Lightweight Self-Supervised Gating (LSSG) method. Consisting of a single-layer MLP, the LSSG is trained by maximizing the mutual information between the original and pruned graph representations (contrastive signal) without any label supervision. This method scores and prunes nodes/edges to probe whether a simple learned filter can approximate the effects of complex attention alignment at a linear computational cost. As shown in Table VI, LSSG exhibits lower ASR than *GDetox* against the GRBA attack on four datasets due to the absence of deep attention distillation. Additionally, LSSG's ACC is significantly lower than SimGuard and DShield because providing pruned nodes and edges tends to ignore normal information, thereby impacting the model's overall performance.

2) Impacts of poisoning ratio: In our experiments, we set the poisoning ratio to 5% of the training data for all attack methods. To further assess the impact of different poisoning ratios on GDetox, we conducted experiments with ratios of $\{1\%, 3\%, 5\%, 7\%, 9\%\}$. Fig. 3 presents the results for our method and SOTA defenses under these poisoning ratios against the latest attack methods on the Cora dataset.

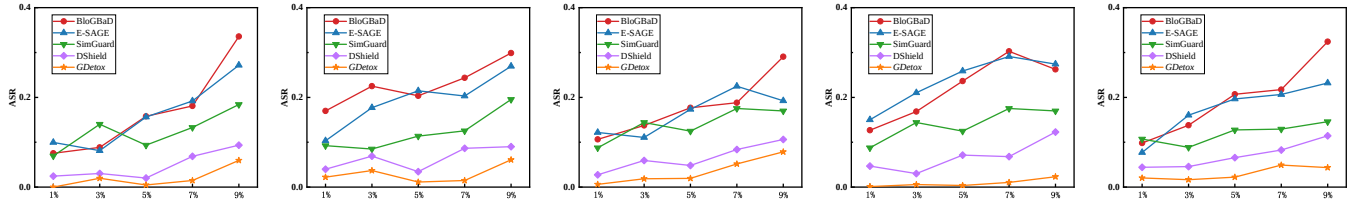
The experimental results align with our expectations: as the poisoning ratio increases, the ASR steadily rises while the ACC correspondingly declines. When the poisoning ratio is 1%, *GDetox*'s ASR tends to 0, indicating its ability to effectively purify backdoors under a low poisoning ratio. At a 9% poisoning ratio, the ASR of the four other SOTA defenses increases sharply, indicating that their performance needs to be improved when a large number of backdoored samples are injected. Compared to ASR, ACC metric exhibits smaller fluctuations, *GDetox*'s accuracy variation remains within 2% across different poisoning ratios against the GRBA attack. These experiments demonstrate that *GDetox* consistently provides robust and superior backdoor defense across varying poisoning ratios.

In addition, to further evaluate the defense capability of *GDetox* on larger datasets under different poisoning ratios, we conduct experiments on a large sparse graph, PubMed, and a large dense graph, Flickr, to analyze the impact of varying poisoning ratios on our method. The experiment results, reported in Table VIII, show that on the PubMed dataset, even when the poisoning ratio reaches 9%, ASR remains below 4% while the model accuracy decreases by only 3.21%. These results show the robustness of *GDetox* on larger-scale graphs.

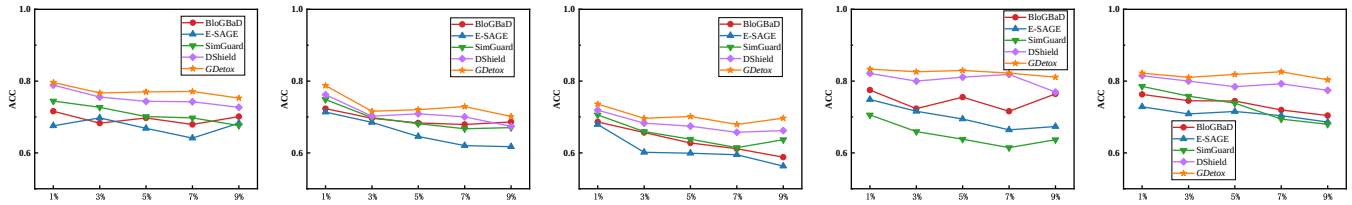
TABLE VII: **Counterfactual Edge Editing and K-hop ASR Analysis under GRBA (%).**

Setting	Cora		Citeseer		Pubmed		Flickr	
	Before	*GDetox*	Before	*GDetox*	Before	*GDetox*	Before	*GDetox*
$k=1$	72.32	2.21	83.37	2.46	77.99	0.61	38.89	0.76
$k=2$	35.62	0.75	36.48	1.07	25.83	0.24	16.42	0.18
$k=3$	22.14	0	19.67	0.28	17.51	0	9.14	0
CEE	8.14	2.06	10.42	2.29	6.38	0.58	7.12	0.71
Remove Random	69.27	2.13	80.15	2.37	74.81	0.60	36.47	0.74
Remove								

3) Impacts of trigger score: In node classification tasks, training in GNNs propagates a node's influence to its surrounding neighbors. Therefore, we explore the scope of trigger influence before defense (Before) and after defense (*GDetox*) across four datasets, and investigate the impact on K-hop nodes for $K \in \{1, 2, 3\}$, as shown in Table XIV. We observe

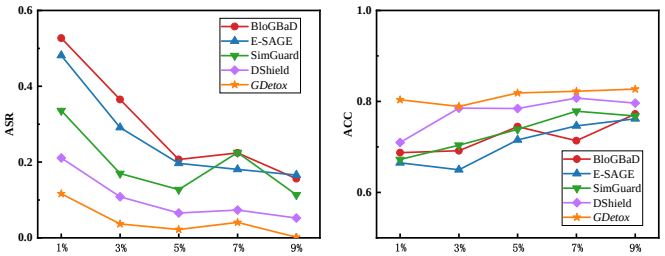


(a) ASR under CLBA attack (b) ASR under ECGBA attack (c) ASR under CGBR attack (d) ASR under GCBA attack (e) ASR under GRBA attack



(f) ACC under CLBA attack (g) ACC under ECGBA attack (h) ACC under CGBR attack (i) ACC under GCBA attack (j) ACC under GRBA attack

Fig. 3: Impacts of the different proportions of backdoored samples against five attacks on the Cora dataset.



(a) ASR (b) ACC

Fig. 4: Impacts of the different proportions of downstream data against the GRBA attack on the Cora dataset.

that as K increases, the ASR steadily decreases, and *GDetox*'s defense effectiveness correspondingly improves. When $K = 2$ and 3, the ASR before defense is significantly lower than at $K = 1$, indicating that triggers' influence on higher-hop nodes is weak. At $K = 3$, *GDetox* achieves an ASR of 0 on Cora, PubMed, and Flickr datasets, demonstrating that *GDetox* can mitigate triggers' effect on surrounding K-hop nodes.

To verify that the reduction of ASR is caused by suppressing trigger propagation channels rather than broadly smoothing, we conduct counterfactual edge editing experiments at $K = 1$. Specifically, we remove edges with the highest attribution scores identified by GNNExplainer [52] (named CEE Remove), as well as an equal number of randomly selected edges (named Random Remove). As shown in Table XIV, removing important edges leads to a dramatic ASR drop in backdoored model, while random edge removal has little effect. In contrast, both edits barely affect *GDetox*, indicating that trigger propagation channels have already been eliminated. This confirms that *GDetox* precisely suppresses backdoor-specific pathways instead of broadly smoothing representations.

4) Impacts of downstream data ratio: Our defense scenario assumes that the defender only has access to the attacked encoder and a downstream dataset. A small portion of the downstream data must be used to purify the backdoored encoder. Therefore, we analyze the impact of different downstream data ratios on *GDetox* and SOTA defenses. We set the downstream data proportions to {1%, 3%, 5%, 7%, 9%}, and the experimental results are shown in Fig. 4. It can be observed that the proportion of downstream data has a significant effect on the ASR of these defenses. When using 1% of the downstream data for defense, ASR remains high (e.g., BloGBaD ASR = 52.7%, *GDetox* ASR = 11.65%). It indicates that the defense is weak with a low data ratio, yet our method still outperforms other SOTA defense methods under this condition. When using 9% of downstream data, *GDetox*'s ASR is only 0.16%, demonstrating a very strong defense effect. In addition, the model accuracy of these defenses increases as the downstream data proportion grows (as shown in Fig. 4(b)). It highlights that the amount of downstream data is crucial for defending against self-supervised graph backdoor attacks and validates the effectiveness of *GDetox*.

To further assess the performance of *GDetox* under different downstream data availability on larger datasets, we evaluate our method on PubMed and Flickr with varying proportions of downstream data. As shown in Table VIII, on the large dense Flickr graph, even when only 1% of downstream data is used, *GDetox* achieves an ASR of 10.72% with an accuracy drop of only 2.53%. This further highlights the necessity of downstream data for effective purification in *GDetox*.

5) Applicability and overhead: **Applicability Analysis.** To evaluate the scalability of *GDetox*, we further conduct experiments on two larger datasets: the node classification dataset OGB-arxiv[3] and the graph classification dataset DD[4], against the CLBA and GRBA attack methods applicable to both node and graph classification. The results are shown in Table IX. It can be observed that on the OGB-arxiv dataset, the ASR after applying *GDetox* against GRBA attack is only 2.87%, and against CLBA attack it is just 1.61%, with accuracy maintained within 0.35% on average. Therefore, *GDetox* can still achieve a strong defense capability and maintain high model performance on large-scale datasets.

[3]https://ogb.stanford.edu/docs/nodeprop/#ogbn-arxiv
[4]https://networkrepository.com/DD.php

TABLE VIII: **The impact of different poisoning ratios and downstream data proportions in large datasets PubMed and Flickr against GRBA attack**

Task	Dataset	1%		3%		5%		7%		9%	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
Poisoning ratios	PubMed	0.8406	0.0057	0.8387	0.0033	0.8321	0.0061	0.8221	0.0138	0.8085	0.0396
	Flickr	0.5057	0.0051	0.4968	0.0093	0.5036	0.0076	0.5072	0.0267	0.4623	0.0212
Downstream data proportions	PubMed	0.8163	0.0931	0.8018	0.0564	0.8321	0.0061	0.8419	0.0049	0.8476	0.0061
	Flickr	0.4783	0.1072	0.4924	0.0351	0.5036	0.0076	0.5036	0.0068	0.5077	0.0051

TABLE IX: **Applicability analysis of *GDetox* on large datasets against different attack methods**

Method	Dataset	Before		*GDetox*	
		ACC	ASR	ACC	ASR
CLBA	OGB-arxiv	0.6437	0.6323	0.6402	0.0161
	DD	0.7752	0.5287	0.7623	0.0203
GRBA	OGB-arxiv	0.6812	0.7197	0.6785	0.0287
	DD	0.7952	0.5534	0.7768	0.0369

TABLE X: **Training time and memory overhead of *GDetox* on different datasets.**

Method	Overhead	Cora	Citeseer	PubMed	Flickr	OGB-arxiv
ACL	Train. Time(s)	2.47	2.94	4.23	8.41	23.18
	Train. Mem.(GiB)	0.020	0.053	0.075	0.478	1.159
	Fwd. Time(ms)	2.340	2.365	1.730	2.400	5.458
	Fwd. Mem.(GiB)	0.0199	0.0526	0.0749	0.4778	1.1585
SSD	Train. Time(s)	3.97	4.39	16.18	65.02	119.42
	Train. Mem.(GiB)	0.038	0.114	0.085	0.399	0.318
	Fwd. Time(ms)	2.602	2.607	2.657	2.703	2.694
	Fwd. Mem.(GiB)	0.0378	0.1139	0.0846	0.3986	0.3176

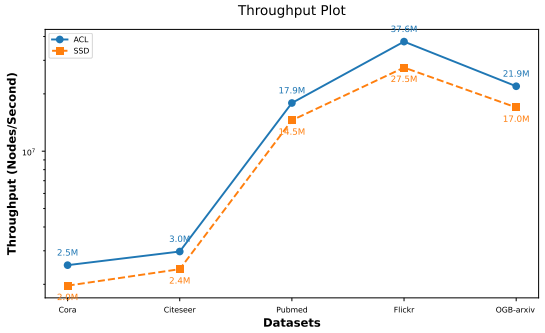
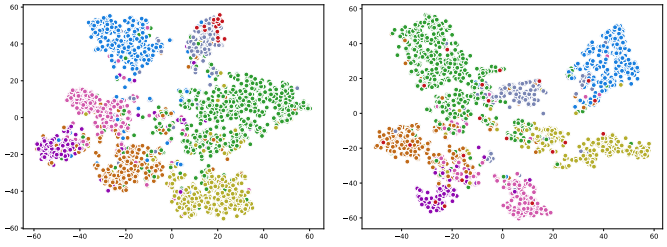


Fig. 5: Throughput Plot of *GDetox* on different datasets.



(a) GCBA without defense (b) GCBA with *GDetox*

Fig. 6: The t-SNE of the feature representations against the GCBA attack on Cora with/without defense method.

Overhead Analysis. To assess the efficiency of our algorithm, we analyze the training time (denoted as Train. Time, in seconds), training memory overhead (denoted as Train. Time, in GiB), forward-pass time per epoch (denoted as Fwd. Time, in milliseconds), and forward-pass memory per epoch (denoted as Fwd. Mem., in GiB) of the adversarial contrastive learning (ACL) and self-supervised distillation (SSD) components in *GDetox* against the GCBA graph backdoor attack on five datasets. As shown in Table X, ACL exhibits relatively low training time on all five datasets verified in our experiments. The time complexity of SSD is relatively high due to the time-consuming distillation and BT loss algorithms. Regarding training memory consumption, larger datasets require greater memory allocation. A granular analysis reveals that the per-epoch forward-pass overhead for both algorithms exhibits minimal, with memory footprints closely aligned with total training overhead. These results demonstrate the practicality and deployment feasibility of *GDetox* in real-world scenarios. Furthermore, we report a throughput analysis, measured as the number of processed nodes per second in a full-graph forward pass, as shown in Fig. 5. The throughput plot shows near-linear scaling with increasing graph sizes, indicating that *GDetox* maintains efficient inference performance even on large-scale graphs such as OGB-arxiv. In summary, *GDetox*'s superior defense performance in Table V and Table IX, we conclude that its overall time, memory, and throughput overhead is reasonable.

Complexity Analysis. The dominant operation of *GDetox* is graph message passing with per-forward cost $\mathcal{O}(L|E|d)$, where L is the number of GNN layers and d is the feature dimension. ACL multiplies this cost by the number of views and any adversarial iterations (yielding $\mathcal{O}((n_{\text{views}} + I_{\text{adv}})L|E|d)$ per ACL iteration), while SSD multiplies by the number of teacher/student forwards $\mathcal{O}((n_{\text{teacher}} + n_{\text{student}})L|E|d)$ per distillation step). Memory is dominated by node embeddings and activation buffers $\mathcal{O}(|V|d+|E|)$, with additional training-time overhead for gradients and optimizer states.

6) Visual analysis: To visually evaluate whether *GDetox* effectively eliminates backdoor features from the latent space, we employ t-SNE to project the node representations of the backdoored and purified encoders into a 2D space. As illustrated in Fig. 6(a), nodes embedded with triggers (marked in red) are forcibly aggregated into a distinct and tight anomalous cluster, regardless of their original semantic categories. This indicates that the backdoored encoder is dominated by trigger-specific patterns. In contrast, as shown in Fig. 6(b), this anomalous cluster disappears after the *GDetox* defense. The poisoned nodes are redistributed into their corresponding se-

TABLE XI: **Perturbation and Transferability Analysis.**

Task	Setting	Teacher model		Purified model	
		ACC	ASR	ACC	ASR
Perturbation Method	RS	0.8115	0.7012	0.7932	0.2542
	ED	0.8068	0.658	0.7855	0.0832
	SP	0.775	0.6815	0.7521	0.1464
	GDetox	0.8444	0.5974	0.8296	0.0037
Cross-attack Analysis	GCBA ->CLBA	0.7254	0.6841	0.7166	0.0178
	GCBA ->GRBA	0.8289	0.6425	0.8143	0.0462
Cross-dataset Analysis	Citeseer ->Cora	0.7895	0.6214	0.7825	0.0221
	Citeseer ->PubMed	0.8008	0.5892	0.7817	0.0546

TABLE XII: **Probe-based Semantic Preservation Results.**

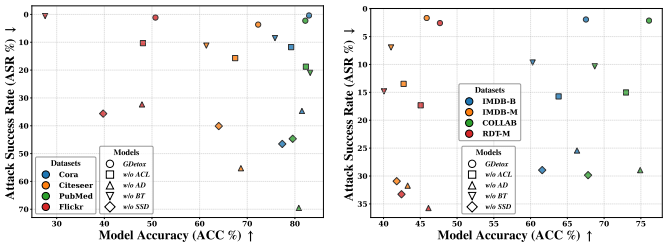
Dataset	Before		*GDetox*	
	Precision	Recall	Precision	Recall
Cora	0.8375	0.8123	0.9155	0.8947
Citeseer	0.8256	0.7842	0.8882	0.8615
Pubmed	0.9021	0.8763	0.9346	0.9131
Flickr	0.8143	0.8092	0.8562	0.8318

mantic clusters and overlap with clean nodes of the same class. These visualization results prove that *GDetox* successfully purifies the latent space by decoupling trigger features from normal semantic representations.

7) Perturbation and transferability analysis: To evaluate the impact of different perturbation strategies on the teacher model, we further analyze the performance of randomized smoothing (RS), edge dropout (ED), and spectral perturbation (SP) methods against GCBA attack on Cora dataset. As shown in Table XI, PGD substantially outperforms these alternatives, as it explicitly approximates the worst-case trigger-sensitive directions under a strict perturbation budget. This targeted adversarial perspective is crucial for exposing and suppressing structured backdoor features, especially in label-free self-supervised learning scenarios.

To evaluate whether *GDetox* learns attack-agnostic invariants, we conduct comprehensive cross-attack and cross-dataset generalization analyses. We train the backdoor model using GCBA method and evaluate the defense on CLBA and GRBA under Cora dataset. As shown in Table XI, *GDetox* consistently suppresses the ASR to below 0.0046, demonstrating its ability to generalize across different backdoor patterns. We train the model on Citeseer dataset and test its performance on Cora and PubMed dataset under the GRBA attack. The results indicate that *GDetox* successfully transfers its defensive capability (e.g., achieving 0.0221 ASR on Cora).

8) Probe-based semantic preservation: To verify that *GDetox* does not over-suppress task-relevant semantics, we construct a tiny probe set consisting of 1% clean labeled nodes, which is not used for training. We apply GNNExplainer [52] to identify salient neighbors for each probe node before and after defense, and report the precision and recall of preserved salient structures. It is worth noting that the precision and recall here represent the task-related semantics preserved by the Before and *GDetox* models compared to the clean model, rather than classification performance. As shown in Table XII,



(a) Node Datasets (b) Graph Datasets

Fig. 7: Ablation analysis against GCBA attack.

the backdoor model exhibits relatively low precision and recall against GCBA attacks across four datasets. This indicates that its predictions are dominated by the poisoned decision logic, causing the identified salient structures to deviate significantly from those of the clean model. In contrast, *GDetox* consistently achieves high precision and recall across all datasets, demonstrating that it effectively suppresses backdoor propagation while preserving core semantic structures.

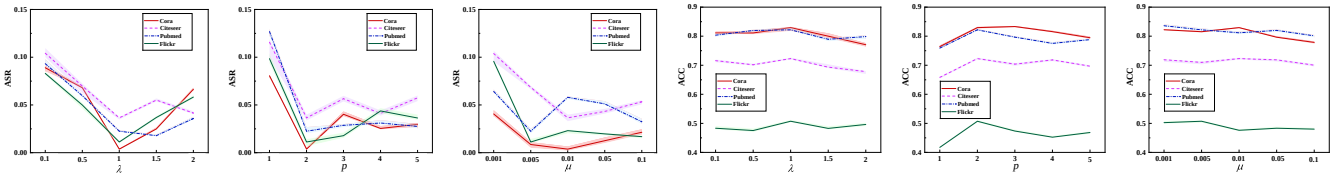
C. RQ2: Ablation Analysis

In this section, we conduct an ablation analysis of the GCBA attack across four datasets to thoroughly assess the contribution of each component in *GDetox* for defending against self-supervised graph backdoor attacks. The experiment removes the adversarial contrastive learning module (*w/o ACL*), the attention alignment module (*w/o AA*), and the BT loss module (*w/o BT*). Since removing the adversarial contrastive learning module means no useful teacher model is available, we choose the fine-tuned backdoor encoder as the teacher model via fine-tuning. To assess the impact of distillation on logits for purifying self-supervised backdoor encoders, the experiment will be conducted on logits rather than using the self-supervised distillation module *w/o SSD*. Thus, we quantify the contribution of each module by reporting the performance deltas with 95% confidence intervals (CIs) over five independent runs, as summarized in Table XIII.

TABLE XIII: Ablation analysis deltas with CIs (%).

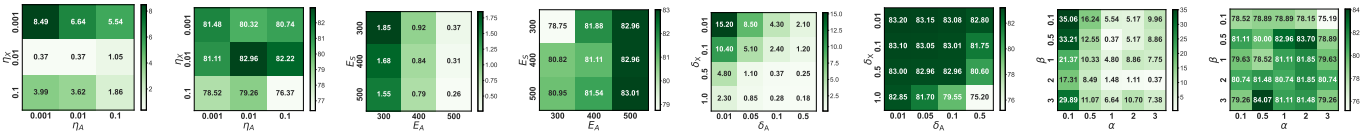
Dataset	*w/o ACL*		*w/o AD*		*w/o BT*		*w/o SSD*	
	ΔACC	ΔASR	ΔACC	ΔASR	ΔACC	ΔASR	ΔACC	ΔASR
Cora	-3.75±0.42	+11.37±1.24	-1.48±0.31	+34.32±1.87	-7.14±0.65	+8.12±0.92	-5.61±0.58	+46.16±2.15
Citeseer	-4.83±0.51	+12.04±1.18	-3.62±0.44	+51.61±2.53	-10.88±0.82	+7.54±0.81	-8.27±0.74	+36.46±1.92
PubMed	+0.13±0.05	+16.53±1.47	-1.37±0.22	+67.27±3.12	+1.02±0.18	+18.74±1.65	-2.65±0.39	+41.43±2.24
Flickr	-2.62±0.33	+9.19±1.02	-2.83±0.38	+31.21±1.68	-23.16±1.54	-0.55±0.15	-10.97±0.95	+34.52±1.84

In addition, we observe that the full *GDetox* model consistently achieves the optimal Pareto analysis between model accuracy (ACC) and purification effectiveness (measured by ASR), as illustrated in Fig. 7(a). Specifically, removing the distillation module (*w/o SSD*) leads to a dramatic surge in ASR (e.g., +46.16% on Cora), confirming that logit-based distillation is insufficient for purifying backdoor encoders in GSSL. The *w/o AA* variant shows a similar ACC but significantly higher ASR, indicating that attention alignment is the primary driver for feature purification. Notably, on the Flickr dataset, *w/o BT* fails to maintain model utility, with ACC plummeting by 23.16%, which underscores the critical role of BT loss in preserving semantic integrity while defending against attacks.



(a) λ (ASR) (b) p (ASR) (c) μ (ASR) (d) λ (ACC) (e) p (ACC) (f) μ (ACC)

Fig. 8: Hyperparameter sensitivity analysis against GCBA attack on four real-world datasets.



(a) ASR (b) ACC (c) ASR (d) ACC (e) ASR (f) ACC (g) ASR (h) ACC

Fig. 9: Confusion matrix of hyperparameter analysis against GCBA attack on Cora dataset (%).

These findings demonstrate that each module in *GDetox* is essential for striking an effective balance between utility and robustness under a fixed computational budget.

D. RQ3: Sensitivity Analysis

To analyze the impact of hyperparameters in *GDetox* on defense performance, we selected seven critical hyperparameters for evaluation against the GCBA attack on the Cora dataset. Specifically, we choose to analyze the hyperparameters η_X and η_A that control the perturbation ratio in adversarial contrastive learning, as well as the hyperparameters δ_{X} and δ_{A} that are constraints on adversarial example features and edges. The hyperparameters E_A and E_S represent the ACL and SSD training rounds, and the hyperparameter λ controls the adversarial contrastive loss. In addition, we consider analyzing the hyperparameter p, which amplifies feature differences in attention alignment, the hyperparameter μ that regulates the independence of non-diagonal elements in the BT loss, and the hyperparameters α and β that control the backdoor removal algorithm. The experimental results are presented in Figs. 8 and 9.

1) Effect of η_X and η_A: The hyperparameters η_X and η_A, which control the perturbation ratio in adversarial contrastive learning, influence the performance of the enhanced encoder. We set η_X and η_A to $\{0.001, 0.01, 0.1\}$, and the experimental results are presented in the confusion matrix in Fig. 9(a) and 9(b). It is observed that *GDetox* achieves optimal ASR and ACC when both η_X and η_A equal 0.01. When η_X and η_A are set to 0.001, GDetox yields an ASR of 0.0849. This is because the perturbation is too small, and the generated adversarial samples are not sufficiently effective, resulting in inadequate improvement of the teacher model's performance.

2) Effect of δ_{X} and δ_{A}: The hyperparameters δ_{X} and δ_{A} are perturbation constraints to control adversarial example. We conducted a grid search over the edge modification ratio $\delta_{\mathrm{A}} \in \{0.01, 0.05, 0.1, 0.5\}$ and feature amplitude $\delta_{\mathrm{X}} \in \{0.01, 0.1, 0.5, 1.0\}$ on the Cora dataset under GCBA attack. The results in Fig. 9(c) and 9(d) demonstrate that smaller perturbation constraints lead to higher ASR, while larger perturbations result in reduced ACC. We identify $\delta_{\mathrm{A}} = 0.1$ and $\delta_{\mathrm{X}} = 0.5$ as the optimal default configuration, as it achieves a superior trade-off between benign task utility (82.96%) and robust defense (0.37% ASR).

3) Effect of E_A and E_S: We investigated the impact of training epochs by conducting a grid search over ACL epochs $E_A \in \{300, 400, 500\}$ and SSD epochs $E_S \in \{300, 400, 500\}$. As shown in Fig. 9(e) and 9(f), a low number of E_A training rounds reduces ACC, while a low E_S negatively impacts ASR. Considering both ACC and ASR simultaneously, we set $E_A = 500$ and $E_S = 300$.

4) Effect of λ: The parameter λ serves as a tunable factor in adversarial contrastive learning, controlling the adversarial contrastive loss and thereby influencing model training. In our experiments, we select $\lambda = 1$ and set different values $\{0.1, 0.5, 1, 1.5, 2\}$ to explore this hyperparameter. As shown in Fig. 8(a) and 8(d), *GDetox*'s ASR and ACC fluctuate under different λ settings. When $\lambda = 0.1$, *GDetox* exhibits high ASR across all four datasets. This is because a too-small λ reduces the cosine similarity of node representations, leading to degraded encoder performance. When $\lambda = 2$, *GDetox*'s ACC decreases, indicating that an overly large λ harms the model's inherent performance. Thus, we choose a value of λ of 1 in *GDetox*.

5) Effect of p: The hyperparameter p controls the amplification of feature differences in the attention alignment mechanism. In our experiments, we explore $p \in \{1, 2, 3, 4, 5\}$. As shown in Fig. 8(b) and 8(e), *GDetox*'s ASR and ACC vary markedly when p = 1. When p = 1, both *GDetox*'s ASR and ACC are noticeably worse than at p = 2. This is because with p = 1, the feature difference amplification is too small, making it difficult for *GDetox* to effectively remove backdoor features from the encoder using attention alignment. When $p \geq 2$, defense performance remains relatively stable. Considering that p = 2 yields the best ASR and ACC, we choose p = 2 for our experiments.

6) Effect of μ: Fig. 8(c) and 8(f) show performance under different values of the hyperparameter $\mu \in \{0.001, 0.005, 0.01, 0.05, 1\}$. The hyperparameter μ controls the independence of non-diagonal elements in the BT loss operation. It can be observed that when μ = 0.001, GDetox's ASR is relatively high and GDetox's ACC remains stable, indicating weak defense performance. When μ = 0.001, *GDetox*'s ASR and ACC

TABLE XIV: **The defense performance of *GDetox* compared with SOTA methods on the graph classification task (%).**

Dataset	Attack method	Before		BloGBaD(2023)		E-SAGE(2024)		SimGuard(2025)		DShield(2025)		*GDetox*	
		ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR	ACC	ASR
IMDB-B	BadGraph	65.73±1.05	38.97±0.26	60.11±1.76	9.03±1.24	58.74±0.55	10.34±1.76	61.21±0.76	7.02±1.41	63.02±1.55	4.86±0.74	**65.14±0.74**	**1.46±0.35**
	GTA	66.12±1.66	40.91±2.12	60.73±1.14	8.92±0.54	59.02±1.42	11.02±1.29	61.54±1.19	6.98±1.16	63.62±0.73	4.33±0.49	**65.78±0.58**	**1.18±0.54**
	CLBA	66.58±1.72	41.47±1.20	61.02±1.71	8.42±1.58	59.93±1.25	9.78±1.53	61.88±0.92	7.85±1.50	64.89±0.77	5.12±0.81	**66.06±0.87**	**1.54±0.43**
	GRBA	67.93±0.99	43.76±0.83	62.71±0.70	9.35±0.77	61.11±1.03	10.87±1.11	62.17±0.36	8.42±0.91	65.28±0.57	5.98±0.61	**67.53±0.36**	**1.94±1.07**
IMDB-M	BadGraph	44.21±1.17	48.72±1.38	38.03±1.37	11.64±1.44	36.23±0.71	12.34±0.71	38.75±1.09	8.12±1.47	42.02±1.28	5.76±1.38	**43.68±1.10**	**0.86±0.62**
	GTA	44.98±0.73	47.21±0.53	38.56±0.93	12.31±1.08	37.54±0.42	11.65±1.35	39.14±1.67	8.76±1.16	42.87±0.92	7.02±0.60	**44.18±0.29**	**1.24±0.54**
	CLBA	45.82±1.38	51.47±1.44	39.42±1.70	13.58±1.51	37.95±1.22	14.03±1.12	40.32±0.97	9.02±0.62	42.76±0.78	6.24±0.89	**45.02±0.36**	**1.03±0.39**
	GRBA	46.69±0.98	54.02±2.02	40.36±0.67	15.73±1.46	38.15±1.00	14.89±1.11	41.24±1.21	9.67±1.58	43.99±1.37	7.84±0.71	**45.87±0.44**	**1.67±0.95**
COLLAB	BadGraph	73.64±1.26	46.84±1.30	67.42±1.41	10.63±1.67	65.21±0.96	11.35±1.27	68.25±1.08	9.75±0.43	71.83±0.78	6.18±0.66	**72.84±0.57**	**0.62±0.38**
	GTA	74.38±1.66	48.21±0.39	68.96±0.94	13.67±1.20	65.82±1.40	14.05±1.35	68.53±1.62	10.73±1.28	71.24±1.29	6.44±0.33	**73.32±1.04**	**1.45±0.55**
	CLBA	74.41±0.93	50.65±0.92	69.46±0.73	14.58±1.01	68.42±1.05	14.86±0.59	70.64±0.94	10.45±1.69	72.37±0.50	6.75±0.81	**73.64±0.26**	**1.92±0.41**
	GRBA	76.76±0.61	52.83±1.96	71.51±0.78	14.37±0.54	69.62±1.23	15.67±1.34	71.08±2.13	11.23±1.14	73.46±1.04	7.12±1.39	**76.11±0.39**	**2.14±0.84**
RDT-M	BadGraph	45.26±1.06	56.57±1.38	42.72±1.07	15.94±1.70	40.32±1.05	16.38±1.10	42.36±1.39	10.76±0.24	42.24±0.76	5.01±0.75	**44.62±0.43**	**0.94±0.47**
	GTA	45.97±1.76	57.01±2.17	41.65±1.19	15.32±1.52	39.83±1.26	15.97±0.47	41.96±1.53	11.86±1.47	43.61±1.40	5.64±0.35	**44.47±0.78**	**1.15±0.25**
	CLBA	47.35±0.67	59.13±1.26	43.07±0.98	17.63±1.02	41.03±0.89	16.85±1.38	42.87±0.92	11.04±1.21	43.52±0.64	7.98±0.65	**46.72±0.29**	**1.86±0.50**
	GRBA	48.27±1.12	60.35±0.34	43.39±1.08	17.89±0.41	41.62±1.09	17.34±1.05	43.27±1.23	12.53±1.88	45.85±0.58	8.46±0.83	**47.67±0.33**	**2.58±1.36**

both decrease overall, suggesting that the independence of non-diagonal elements should not be too large. Notably, for the Cora and Citeseer datasets, *GDetox*'s ASR and ACC are optimal at μ = 0.01; for the PubMed and Flickr datasets, *GDetox*'s ASR and ACC are optimal at μ = 0.005. Therefore, we choose μ = 0.01 for the Cora and Citeseer datasets, and μ = 0.005 for the PubMed and Flickr datasets.

7) Effect of α and β: The hyperparameters α and β affect the loss balancing of attention alignment and the BT loss, determining the effectiveness of backdoor removal. As shown in Fig. 9, we set α and β to $\{0.1, 0.5, 1, 2, 3\}$. In the confusion matrices of ASR (Fig. 9(g)) and model accuracy (Fig. 9(h)), *GDetox* achieves the best defense and model performance at α = 1 and β = 0.5. When α = 0.1, *GDetox*'s ASR is high, indicating poor defense performance. When β = 0.1, *GDetox*'s ACC is low (below 0.8), indicating poor model performance. This demonstrates that the attention alignment and BT loss modules are critical for *GDetox*'s defense effectiveness and overall performance. Therefore, we choose α = 1 and β = 0.5 in our experiments.

E. RQ4: Extensibility Analysis

To validate *GDetox*'s extensibility, we transfer it to graph classification tasks. As shown in Table XIV, we compare our method with SOTA defense methods against the four latest backdoor attack methods on four real-world graph classification datasets. For intuitive understanding, we highlight our method's results with Grey color and bold the best ASR and ACC values among all defense methods.

From the results in this table, *GDetox* achieves a lower ASR than all compared defenses and attains higher ACC across all four SOTA methods, indicating superior defense performance against various self-supervised graph backdoor attacks on these graph classification datasets. BadGraph and GTA are originally designed for graph-level backdoor attacks in supervised settings, and we adapt them to the self-supervised scenario. As shown in the "Before" results in the table, BadGraph and GTA perform worse in self-supervised settings than in supervised settings. This is because self-supervised settings do not rely on sample label information. Among the defense methods, E-SAGE exhibits a relatively high ASR (over 10% overall), because its iterative pruning cannot reliably distinguish normal edges from malicious ones, leading to high attack success. In addition, the ACC of the BloGBaD and SimGuard defense methods is relatively low overall due to the lack of guidance from label information. Although DShield outperforms other SOTA methods, it still underperforms *GDetox*. This is attributed to the fact that DShield needs to combine label information during the clustering process to effectively distinguish potential backdoor features. Thus, these experiments demonstrate that *GDetox* is generalizable and maintains its superiority in graph classification tasks.

Furthermore, we conduct ablation studies against the GRBA attack on four graph classification datasets, as shown in Fig. 7(b). The *w/o ACL* variant yields lower ACC than *GDetox*, indicating that adversarial contrastive learning is essential for maintaining model utility. In addition, *w/o AA* and *w/o BT* exhibit noticeably higher ASR compared to *GDetox*, demonstrating that both components play a critical role in improving defense success rate. These results validate the design rationale of incorporating each module into *GDetox*.

VI. Discussion of Limitations & Ethical Considerations

The attention alignment mechanism in *GDetox* is specifically designed for white-box scenarios, as it relies on accessing the encoder's internal parameters to extract layer-wise attention maps. Such internal access is crucial for imposing structural constraints between the teacher and student models. In a strictly black-box setting, where the defender can only access final embeddings or logits, attention alignment cannot be effectively implemented, and cannot expect to achieve the same purification performance. We intend to further explore effective graph backdoor defense strategies in black-box scenarios in future work.

We also acknowledge a dual-use risk inherent to attention-based defense. Since *GDetox* operates by aligning internal attention patterns, a sophisticated adaptive adversary might intentionally craft triggers that mimic the attention statistics of benign samples, thereby reducing the effectiveness of attention alignment. While this risk does not invalidate our defense itself, it suggests that a more diversified defense strategy is required under stronger adaptive threat models. To mitigate such risks, future research should focus on designing in-depth defense strategies to safeguard against these adaptive adversaries.

VII. SUMMARY AND FUTURE WORK

In this paper, we presented *GDetox*, the effective defense method against backdoor attacks in GSSL. By leveraging the self-supervised distillation approach that deploys attention alignment and the BT loss, *GDetox* can purify graph backdoor encoders. Considering that training the encoder with the fine-tuning method will bring the backdoor features closer to the target labels, we proposed the adversarial contrastive learning method to boost the performance of the teacher model for distillation without relying on labels. *GDetox* can effectively purify self-supervised backdoor encoders and ensure the encoder's feature representation performance for downstream data. On eight real-world datasets, we validated extensive experimental demonstrations that *GDetox* outperforms all SOTA defense methods against the latest backdoor attacks in GSSL of node and graph classification tasks. In forthcoming endeavors, we will further explore how to mitigate the latest graph backdoor attacks in distributed scenarios.

REFERENCES

[1] J. Zhang, H. Sui, X. Sun, C. Ge, L. Zhou, and W. Susilo, "Grabphisher: Phishing scams detection in ethereum via temporally evolving gnns," *IEEE Transactions on Services Computing*, 2024.

[2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, 2019, pp. 417–426.

[3] E. Mansimov, O. Mahmood, S. Kang, and K. Cho, "Molecular geometry prediction using a deep generative graph neural network," *Scientific reports*, vol. 9, no. 1, p. 20381, 2019.

[4] S. Cao, X. Sun, X. Wu, D. Lo, L. Bo, B. Li, and W. Liu, "Coca: improving and explaining graph neural network-based vulnerability detection systems," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, 2024, pp. 1–13.

[5] Y. Xie, J. Jia, C. Wen, D. Li, and M. Li, "Multi-topology contrastive graph representation learning," *Science China Information Sciences*, vol. 69, no. 2, p. 122102, 2026.

[6] G. Yang, M. Li, H. Feng, and X. Zhuang, "Deeper insights into deep graph convolutional networks: Stability and generalization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 48, no. 2, pp. 1707–1719, 2026.

[7] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in neural information processing systems*, vol. 33, pp. 5812–5823, 2020.

[8] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "Infogcl: Information-aware graph contrastive learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 30 414–30 425, 2021.

[9] S. Zhang, L. Chen, C. Wang, S. Li, and H. Xiong, "Temporal graph contrastive learning for sequential recommendation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 8, 2024, pp. 9359–9367.

[10] Y. Cao, J. Xu, C. Yang, J. Wang, Y. Zhang, C. Wang, L. Chen, and Y. Yang, "When to pre-train graph neural networks? from data generation perspective!" in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 142–153.

[11] Y. Lu, X. Jiang, Y. Fang, and C. Shi, "Learning to pre-train graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4276–4284.

[12] X. Yu, Z. Liu, Y. Fang, Z. Liu, S. Chen, and X. Zhang, "Generalized graph prompt: Toward a unification of pre-training and downstream tasks on graphs," *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[13] J. Dai and Z. Xiong, "A semantic backdoor attack against graph convolutional networks," *arXiv preprint arXiv:2302.14353*, 2023.

[14] L. Chen, N. Yan, B. Zhang, Z. Wang, Y. Wen, and Y. Hu, "A general backdoor attack to graph neural networks based on explanation method," in *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2022, pp. 759–768.

[15] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 1523–1540.

[16] Z. Zhang, J. Jia, B. Wang, and N. Z. Gong, "Backdoor attacks to graph neural networks," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, 2021, pp. 15–26.

[17] L. Chen, Q. Peng, J. Li, Y. Liu, J. Chen, Y. Li, and Z. Zheng, "Neighboring backdoor attacks on graph convolutional network," *arXiv preprint arXiv:2201.06202*, 2022.

[18] J. Xu and S. Picek, "Poster: Clean-label backdoor attack on graph neural networks," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3491–3493.

[19] X. Xing, M. Xu, Y. Bai, and D. Yang, "A clean-label graph backdoor attack method in node classification task," *Knowledge-Based Systems*, vol. 304, p. 112433, 2024.

[20] X. Fan and E. Dai, "Effective clean-label backdoor attacks on graph neural networks," in *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, 2024, pp. 3752–3756.

[21] H. Zhang, J. Chen, L. Lin, J. Jia, and D. Wu, "Graph contrastive backdoor attacks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 40 888–40 910.

[22] B. Feng, D. Jin, X. Wang, F. Cheng, and S. Guo, "Backdoor attacks on unsupervised graph representation learning," *Neural Networks*, vol. 180, p. 106668, 2024.

[23] Z. Guan, M. Du, and N. Liu, "Xgbd: Explanation-guided graph backdoor detection," *arXiv preprint arXiv:2308.04406*, 2023.

[24] J. Downer, R. Wang, and B. Wang, "Securing gnns: Explanation-based identification of backdoored training graphs," *arXiv e-prints*, pp. arXiv–2403, 2024.

[25] B. Jiang and Z. Li, "Defending against backdoor attack on graph nerual network by explainability," *arXiv preprint arXiv:2209.02902*, 2022.

[26] D. Yuan, X. Xu, L. Yu, T. Han, R. Li, and M. Han, "E-sage: Explainability-based defense against backdoor attacks on graph neural networks," in *International Conference on Wireless Artificial Intelligent Computing Systems and Applications*. Springer, 2024, pp. 402–414.

[27] X. Yang, G. Li, X. Tao, C. Zhang, and J. Li, "Black-box graph backdoor defense," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2023, pp. 163–180.

[28] Z. Zhang, M. Lin, J. Xu, Z. Wu, E. Dai, and S. Wang, "Robustness-inspired defense against backdoor attacks on graph neural networks," *arXiv preprint arXiv:2406.09836*, 2024.

[29] C. Liu, H. Huang, Y. Xing, and X. Zuo, "Boosting graph robustness against backdoor attacks: An over-similarity perspective," *arXiv preprint arXiv:2502.01272*, 2025.

[30] H. Yu, C. Ma, X. Wan, J. Wang, T. Xiang, M. Shen, and X. Liu, "Dshield: Defending against backdoor attacks on graph neural networks via discrepancy learning," in *Network and Distributed System Security Symposium, NDSS*, 2025.

[31] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang, "Graph contrastive learning with adaptive augmentation," in *Proceedings of the web conference 2021*, 2021, pp. 2069–2080.

[32] S. Feng, B. Jing, Y. Zhu, and H. Tong, "Adversarial graph contrastive learning with information regularization," in *Proceedings of the ACM web conference 2022*, 2022, pp. 1362–1371.

[33] J. Xu, G. Abad, and S. Picek, "Rethinking the trigger-injecting position in graph backdoor attack," *arXiv preprint arXiv:2304.02277*, 2023.

[34] J. Xu, M. Xue, and S. Picek, "Explainability-based backdoor attacks against graph neural networks," in *Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning*, 2021, pp. 31–36.

[35] H. Zheng, H. Xiong, J. Chen, H. Ma, and G. Huang, "Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs," *IEEE Transactions on Computational Social Systems*, 2023.

[36] S. Yang, B. G. Doan, P. Montague, O. De Vel, T. Abraham, S. Camtepe, D. C. Ranasinghe, and S. S. Kanhere, "Transferable graph backdoor attack," in *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, 2022, pp. 321–332.

[37] E. Dai, M. Lin, X. Zhang, and S. Wang, "Unnoticeable backdoor attacks on graph neural networks," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2263–2273.

[38] H. Wang, T. Liu, Z. Sheng, and H. Li, "Explanatory subgraph attacks against graph neural networks," *Neural Networks*, vol. 172, p. 106097, 2024.

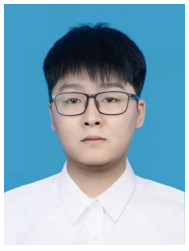
[39] Z. Zhang, M. Lin, E. Dai, and S. Wang, "Rethinking graph backdoor attacks: A distribution-preserving perspective," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 4386–4397.

[40] J. Xu and S. Picek, "Poster: Multi-target & multi-trigger backdoor attacks on graph neural networks," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, 2023, pp. 3570–3572.
[41] K. Wang, H. Deng, Y. Xu, Z. Liu, and Y. Fang, "Multi-target label backdoor attacks on graph neural networks," *Pattern Recognition*, vol. 152, p. 110449, 2024.
[42] H. Sui, B. Chen, J. Zhang, C. Zhu, D. Wu, Q. Lu, and G. Long, "Dmgnn: Detecting and mitigating backdoor attacks in graph neural networks," *arXiv preprint arXiv:2410.14105*, 2024.
[43] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax." *ICLR (poster)*, vol. 2, no. 3, p. 4, 2019.
[44] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, and J. Tang, "Gcc: Graph contrastive coding for graph neural network pre-training," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 1150–1160.
[45] T. Han, S. Huang, Z. Ding, W. Sun, Y. Feng, C. Fang, J. Li, H. Qian, C. Wu, Q. Zhang *et al.*, "On the effectiveness of distillation in mitigating backdoors in pre-trained encoder," *IEEE Transactions on Dependable and Secure Computing*, 2024.
[46] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[47] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," *arXiv preprint arXiv:1906.04214*, 2019.
[48] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," in *International conference on machine learning*. PMLR, 2021, pp. 12 310–12 320.
[49] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *Proceedings of the twelfth ACM international conference on web search and data mining*, 2019, pp. 393–401.
[50] S. C. Ritchie, S. Watts, L. G. Fearnley, K. E. Holt, G. Abraham, and M. Inouye, "A scalable permutation approach reveals replication and preservation patterns of network modules in large datasets," *Cell systems*, vol. 3, no. 1, pp. 71–82, 2016.
[51] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 2015, pp. 1365–1374.
[52] Z. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "Gnnexplainer: Generating explanations for graph neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

Hao Sui received the M.S. degree from Yangzhou University, Yangzhou, China, in 2024. He is currently working toward the PH.D. degree in Cyberspace Security with Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include AI security and graph neural networks.

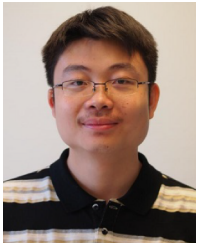
Jiale Zhang (Member, IEEE) received the Ph.D. degree in computer science and technology the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2021. He is currently an Associate Professor with the School of Information Engineering, Yangzhou University, Yangzhou, China. He has published over 60 research papers in refereed international conferences and journals, such as IEEE TDSC, IEEE TIFS, IEEE TSC, CVPR, IJCAI, and WWW. His research interests are mainly AI security, federated learning, and software security.

Bing Chen (Member, IEEE) is currently a full professor at the College of Compute Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), China. He received his B.S. and M.S. in Computer Engineering from NUAA in 1992 and 1995, and the Ph.D. degrees from the College of Compute Science and Technology, NUAA, in 2008. His research interests include cloud/edge computing, security and privacy, FL, wireless communications.



Chengcheng Zhu received the M.S. degree from Yangzhou University, Yangzhou, China, in 2025. He is currently working toward the PH.D. degree in Software Engineering with Nanjing University, Nanjing, China. His research interests include backdoor attacks and adversarial learning.

Chunpeng Ge (Member, IEEE) received the Ph.D. degree in Computer Science from Nanjing University of Aeronautics and Astronautics in 2016. He was research fellow at University of Wollongong and Singapore University of Technology and Design. His current research interests include information security and privacy-preserving for cloud computing, blockchain, security and privacy of AI systems. His recent work has focused on the topics of cryptographic tools for federated learning, data aggregation and collaborated learning. He has published more than 60 papers in prestigious journal and conferences. He served as the editor of journal of CSI and program committee for more than 30 international conferences.



Weizhi Meng (Senior Member, IEEE) is currently a Full Professor at the School of Computing and Communications, Lancaster University, U.K., and an Adjunct Faculty at the Department of Applied Mathematics and Computer Science, Technical University of Denmark, Denmark. His research interests include intersections among cyber security, artificial intelligence, and blockchain technology, such as intrusion detection, the loT security, biometric authentication, and blockchain. He received the lEEE ComSoc Best Young Researcher Award for Europe, Middle East, and Africa Region (EMEA) in 2020. He serves as the Chair for various international conferences, such as ACM CCS'23 and ESORICS'22.

Willy Susilo (Fellow'2021) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is currently a Distinguished Professor and the Head of the School of Computing and Information Technology and the Director of the Institute of Cybersecurity and Cryptology, University of Wollongong. He has published over 400 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. He was a recipient of the Australian Research Council (ARC) Future Fellow by the ARC and the Researcher of the Year Award by the University of Wollongong in 2016. He is the Editor-in-Chief of the Information journal. He has served as a program committee member in dozens of international conferences. He is currently serving as an Associate Editor in several international journals, including the Computer Standards and Interface (Elsevier) and the International Journal of Information Security (Springer). His work has been cited over 16000 times in Google Scholar.