Black-box Graph Backdoor Defense

Xiao Yang[1], Gaolei Li[1], Xiaoyi Tao[2], Chaofeng Zhang[3], and Jianhua Li[1]

[1] Shanghai Jiao Tong University, Shanghai 200240, China
[2] Dalian Maritime University, Dalian 116026, China
[3] Advanced Institute of Industrial Technology, Tokyo 140-0011, Japan

Abstract. Recently, graph neural networks (GNNs) have been proven to be vulnerable to backdoor attacks, wherein the test prediction of the model is manipulated by poisoning the training dataset with trigger-embedded malicious samples during learning. Current defense methods against GNN backdoor are not practical due to their requirement for access to the GNN parameters and training samples. To address this issue, we present a **Bl**ack-b**o**x **G**NN **Ba**ckdoor **D**efense strategy, **BloGBaD**, that eliminates the backdoor without model parameter information and training dataset. Specifically, **BloGBaD** involves two primary phases: 1) test sample filtration, which identifies toxic graph nodes via the Gaussian mixture model and purifies their trigger features through clustering and filtration; and 2) model fine-tuning, which fine-tunes the model to a backdoor-free state by a loss function with a penalty regularization for poisoned features. We demonstrate the effectiveness of our method through extensive experiments on various datasets and attack algorithms under the assumption of black-box conditions.

Keywords: Graph neural networks · Backdoor attacks · Backdoor defense · Black-box defense.

1 Introduction

Graph neural networks (GNNs) are designed to handle graph-structured data that can be represented by nodes and edges, such as social media, biological molecules, transportation networks, and knowledge graphs. GNN propagates information between nodes in the graph, and subsequently makes predictions based on the data contained within the graph. It has been successfully applied in a variety of domains, including recommendation systems, community detection, transportation demand forecasting, and biological analysis [18].

Despite the remarkable success of GNNs, they are still susceptible to one potential threat: backdoor attacks. This attack is commonly implemented by poisoning the training set through the insertion of trigger nodes into the target graphs and modifying their labels as the premeditated class. Thereafter, the poisoned graphs will be employed by the model for learning along with the benign ones, and once the training is complete, trigger-embedded test graphs will be classified as premeditated class, since the model has learned the feature relation

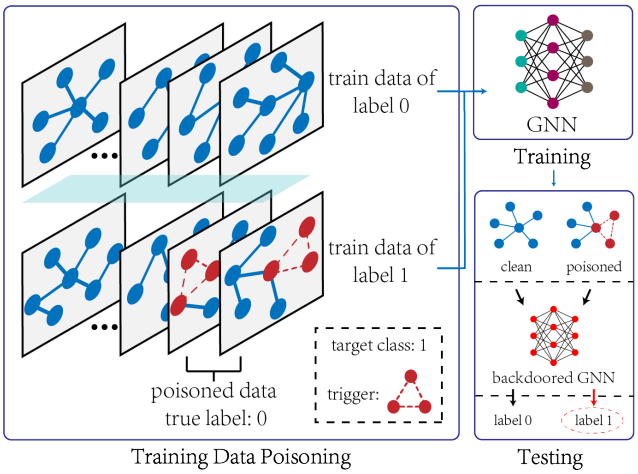


Fig. 1. Illustration for general GNN backdoor framework. In the training dataset, several graphs were poisoned by adding trigger nodes and modifying their true labels to the target class. This led to the model being backdoored during learning, causing it to output the target class when a trigger-embedded graph is inputted when testing.

between the trigger and the premeditated class [24]. The general framework of the backdoor attack is shown in Fig. 1.

The defense method against the GNN backdoor has not been extensively investigated. Several current studies primarily rely on either model parameters or explainability tools to identify suspect models or input samples, which is not applicable in real-world scenarios due to restrictions imposed by commercial property protection and user privacy policies that typically prohibit third-party access to the model parameters [11].

Our motivation to address this limitation derives from a statistical observation that for the trigger nodes in a poisoned graph, their feature distributions differ significantly from those of clean nodes, and if we could leverage the feature distribution to identify the poisoned nodes and analyze the trigger features, we could retrain the model to make it insensitive to these features, thereby mitigating the backdoor.

Based on the analysis above, in this paper, we present a novel **Bl**ack-b**o**x **G**NN **Ba**ckdoor **D**efense strategy: **BloGBaD**. It involves two main phases: 1) test sample filtration and 2) model fine-tuning. In phase 1), we first apply Gaussian mixture to model the graph data and detect potential suspect trigger nodes with anomalous distribution. Then, trigger features in detected nodes are identified through clustering algorithm, and filtered to clean state. While in phase 2), we fine-tune (retrain) the model by incorporating the trigger features detected in phase 1), and the loss function utilized for tuning will have a regularization penalty term associated with these features. Once the tuning is complete, the model will no longer be sensitive to the trigger, rendering the implanted backdoor

ineffective. The entire defense process does not leverage any model parameter information, making it a black-box defense approach.

The contribution of this work can be summarized as follows:

- In this study, a novel two-phase GNN backdoor defense strategy, **BloGBaD**, is proposed. The proposed method consists of two phases: 1) test sample filtration and 2) model fine-tuning. In the first phase, we identify and remove trigger nodes and features that are responsible for activating the backdoor. In the second phase, the victim model is fine-tuned using trigger features to eliminate the backdoor. To the best of our knowledge, this is the first work that introduces black-box backdoor defense for GNN.
- We present a new method for identifying trigger nodes and features in poisoned graph that activate backdoors. It employs Gaussian mixture models to identify the trigger nodes and a clustering algorithm to compute the trigger features on these nodes. By clearing trigger features, the poisoned graph could be normally classified.
- Detailed experiments based on different real-world datasets and backdoor methods are conducted to evaluate the defense performances of **BloGBaD**, and it performs high detection success rate and significant attack success rate drop on malicious data without distinct model accuracy degradation.

The rest of this paper is organized as follows: Section II summarizes the related works; Section III provides the details of the proposed defense strategy, **BloGBaD**; Section IV introduces the experimental settings and results; Section V concludes this paper and discusses the future trend.

2 Related Work

2.1 Graph Neural Network Backdoor

The vulnerability of GNNs to backdoors was initially exposed in [24]. The method utilizes a specific subgraph nodes as trigger to insert into graph training data and modifies its true label as target. When the model completes learning, it will predict the test graph sample with trigger as the target category. In a similar vein, [19] assesses backdoor feasibility through topology and employs specific sub-topology structures as trigger patterns to implant backdoors. While conventional GNN backdoors are implemented in a transductive training environment, [22] examines the transferability of GNNs to introduce a specific graph structure into the training set for inductive attacks. Federated learning has recently gained attention, and [20] investigates the susceptibility of federated GNNs to backdoor attacks: malicious attack that intends to alter the center's parameter update by introducing a specific trigger pattern into client training data. The current attacks' trigger design is too apparent, so [6] proposes an adaptable trigger design to generate imperceptible poisoning data to implant a backdoor. Since GNN node data is aggregated from neighboring nodes, [4] introduces a neighbor-based attack technique to propagate the trigger's influence to nearby nodes. Additionally, there are some enhanced approaches based on triggers that have demonstrated increased success rates in attacks [21, 3, 5, 25].

2.2 Black Box Backdoor Defenses

Study [2] first introduce a backdoor defense method for deep neural networks (DNNs) that operates under black box settings, meaning that it does not require access to the DNN's architecture or parameters for backdoor identification and mitigation. This approach utilizes the model parameters to recover the original training data and subsequently assesses the normality of the model. To make the defense available in a more strong assumption (without access to model parameters), [7] proposes a more detailed method based on analyzing the response of the network to recover a small set of synthetic inputs that are then employed to reveal the presence of a backdoor via inverse trigger construction. Guo et al. (2021) designed a backdoor defense method called Aeva, which works by analyzing the extreme values of the DNN's output when presented with different inputs and comparing them to a threshold that is learned during training. This approach can detect backdoors even under black-box conditions [8]. Aiming to make defense method applicable in the context of machine learning as a service, SCALE-UP method is presented. This approach could identify and filter out malicious testing samples by analyzing their prediction consistency during the pixel-wise amplification process [9]. Zhang et al. (2022) employ an auxiliary network to monitor the output of the target network and detect anomalies that may indicate the presence of a backdoor. This adaptive detection mechanism of the method allows it to adjust to new backdoor attacks and maintain high detection accuracy [23].

Most of the existing research on black-box defense methods focuses on computer vision domains, and to the best of our knowledge, there is currently no black-box defense method for GNN backdoor attacks.

3 Methodology

To defend against previous research on GNN backdoors, in this section, our defense strategy, **BloGBaD**, will be presented. **BloGBaD** comprises two phases. In the first phase, trigger nodes and features are identified through Gaussian mixture modeling and clustering algorithms. The second phase involves fine-tuning the model using a loss function that incorporates a penalty term related to the trigger features, aiming to eliminate backdoors. The general framework of **BloGBaD** is shown in Fig. 2.

We first outline our assumptions, objectives, and key intuition, and then provide a detailed description of **BloGBaD**.

3.1 Defense Assumptions and Goals

We make some definitions of the capabilities and available resources of the defenders. It is first assumed that we have access to the output label and structure of the model, but not to the parameters. Additionally, we possess a set of samples (not the original training samples) that may potentially include poisoned samples.

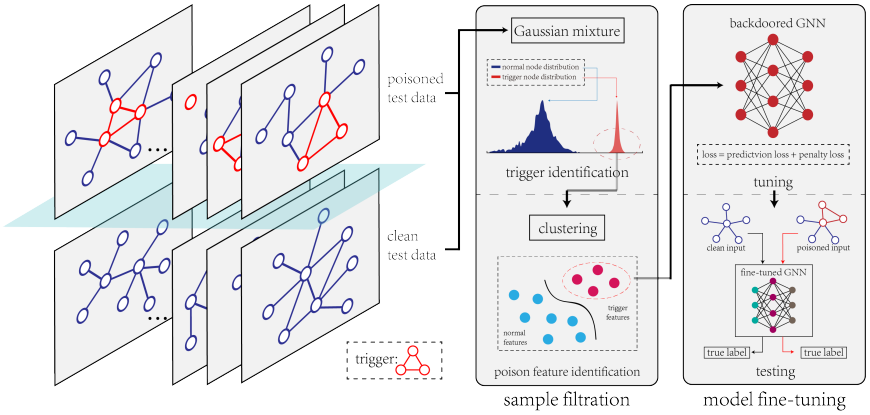


Fig. 2. Illustration for the proposed **BloGBaD** defense scheme. The test samples will undergo initial examination using a Gaussian mixture model to assess the presence of trigger nodes. In case trigger nodes are detected, clustering will be employed to extract the corresponding trigger features. Subsequently, a loss function incorporating penalty terms related to these features will be utilized for fine-tuning the original model, aiming to render it insensitive to triggers.

Our goals include three specific ones:

- **Keep Model Performance**: For clean samples, it is essential to maximize the model's accuracy in classifying them.
- **Identify Malicious Samples**: Regarding poisoned graphs, we should accurately detect the infection status and perform purification procedures to enable correct classifications by the model. Meanwhile, the misdetection of clean samples as poisoned ones should be prevented.
- **Mitigate Model Backdoor**: When we detect poisoned graphs, we want to make the model immune to other trigger-embedded samples.

3.2 Key Intuition

Before **BloGBaD** is specified, we will explain what the intuition of the design is, and to simplify the narrative, we use the graph classification task of GNN as an example.

GNN operates by aggregating information from neighboring nodes within the graph and subsequently conducting a readout function, for instance, sumup function, on the entire graph. To ensure that the model effectively associates the features on the trigger nodes with the target class during the learning process, the adversary should keep the vector values obtained through the sumup on the trigger features sufficiently discernible (i.e., enhance the learnability of trigger features by the model). Hence, the feature value distribution on trigger nodes will significantly differ from that on normal nodes.

Motivated by this inspiration, we could identify and purify these trigger nodes based on their feature distribution, and then retrain the model to mitigate the attack.

3.3 BloGBaD

BloGBaD involves two phases: 1) test sample filtration and 2) model fine-tuning. We will sequentially introduce them.

1) Test Sample Filtration Given a test graph sample $G = (V, E)$, our primary objective is to identify whether it is poisoned. If an infection is detected, our subsequent task is to ascertain the presence of trigger nodes and features within the graph. Based on the motivation discussed in Section 3.2, we employ Gaussian mixture to model all nodes V within a graph structure and estimate the distribution of trigger nodes [13]. By treating the feature vector of each node as individual data point, the formula for the posterior probability of one node belonging to specific distribution is:

$$\gamma_{ik} = \frac{\pi_k \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K=2} \pi_j \cdot \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}, \tag{1}$$

where γ_{ik} represents the posterior probability of node i belonging to the kth distribution (we set K as 2 in accordance with our assumption of two node distributions, namely, a normal distribution $\mathcal{N}_{nor}$ and an infected node distribution $\mathcal{N}_{poi}$, and then our objective is to determine the trigger node distribution $\mathcal{N}_{poi}$), π_k is the mixing coefficient of the kth distribution, $\mathbf{x}_i$ is the observed node vector, $\boldsymbol{\mu}_k$ is the mean of the kth distribution, $\boldsymbol{\Sigma}_k$ is the covariance matrix of the kth distribution, and $\mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ denotes the probability density function of the multivariate normal distribution. π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ are iteratively computed using the expectation maximization (EM) method [14], the update equations for the model parameters are shown below:

$$\pi_i = \frac{N_i}{N}, \tag{2}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} \mathbf{x}_i, \tag{3}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{N} \gamma_{ik} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top, \tag{4}$$

where N_k is the node number in the kth distribution, while N is the total node number in the graph.

Through Gaussian mixture modeling, we obtain two distributions of a graph, and the smaller distribution is regarded as $\mathcal{N}_{poi}$ since the number of clean nodes generally exceeds that of the poisoned.

However, in the case of a clean graph, the node distribution is exclusively normal, without a poisoned node distribution, thus easily leading to misdetection. Consequently, to solve this problem, we employ the following formula to calculate a score value for graph and ascertain the presence of poisoning if it exceeds the valve:

$$S = \sum_{i=1}^{n} (\boldsymbol{\mu}_1^i - \boldsymbol{\mu}_2^i) \tag{5}$$

where $\boldsymbol{\mu}_k^i$ represents the ith value in the vector mean of the kth distributions obtained from Gaussian mixture. The idea of the equation design originates from a simple observation, where two different distributions exhibit significant disparities in their means, while two similar distributions demonstrate minimal differences in their means, which could be leveraged for distribution distinction.

Following the aforementioned procedure, poisoned nodes (trigger nodes, $\mathbf{x}_i \in \mathcal{N}_{poi}$) could be identified. Next, our aim is to detect the trigger features in these nodes.

In the current GNN backdoor attack settings, the trigger configuration typically assigns higher values to certain features, which are considerably larger in magnitude compared to the regular values. Consequently, we leverage a clustering algorithm for their identification and the cluster with a higher mean value will be recognized as trigger features.

For identified trigger node $\mathbf{x}_i = [f_1, f_2, ..., f_n] \in \mathcal{N}_{poi}$, we regard the values of all the features as independent data points (f_i) and then classify them in two categories (i.e., normal and trigger features) using a clustering algorithm. In this study, we employ the K-means algorithm for clustering [1]. The algorithm works by iteratively assigning data points to clusters and optimizing cluster centroids to minimize the overall intra-cluster variance, and the process is given by:

1. **Preparation**: Extract feature values as individual data points ($\{f_1, f_2, ..., f_n\}$).
2. **Setting**: Set the optimal number of clusters as two (i.e., normal and trigger features), and randomly initialize two cluster centroids (λ_1, λ_2) for cluster one C_1 and two C_2 respectively.
3. **Assignment** For each data point, calculate its distance to each cluster centroid and assign it to the cluster with the nearest centroid by

$$c_i = \arg\min_m \|f_i - \lambda_m\|^2, \tag{6}$$

   where λ_m is the centroid.
4. **Update**: Recalculate the centroids of each cluster by taking the mean of the feature values of all data points assigned to that cluster.

$$\lambda_m = \frac{1}{|C_m|} \sum_{f_i \in C_m} f_i, \tag{7}$$

   where C_m is the cluster.

5. **Iteration**: Iterate the assignment and update steps until convergence, and the objective function is given by

$$J = \sum_{m=1}^{2} \sum_{i=1}^{n} \psi_{ij} \left\| f_i - \lambda_m \right\|^2, \tag{8}$$

where ψ_{ij} is an indicator function that takes the value of 1 if data point f_i belongs to cluster center $\lambda_m j$, and 0 otherwise.

Through the above process, two clusters will be found (C_1 and C_2), and we identify the smaller one as trigger feature cluster C_{poi}. By eliminating them, the nodes can revert to their clean state and thus be correctly classified by the victim model.

2) Model Fine-tuning By employing phase one, we can already achieve an input sample filtration-based backdoor protection in testing. But, to completely eliminate backdoors, the model should be fine-tuned so as to mitigate backdoor.

Utilizing the trigger features ($f_i \in C_{poi}$) obtained in the previous phase, we fine-tune (retrain) the model to keep it insensitive to malicious features, while preserving accuracy on normal sample testing. This is achieved by adding the loss function with a regularized penalty term and fine-tune the backdoored model [17]. The loss is provided as follows:

$$\begin{aligned} L =& \frac{1}{p} \sum_{i=1}^{p} (y_i - \hat{y}_i)^2 + R(\omega), \\ & R(\omega) = \alpha \sum_{i=1}^{l} \|\omega_i^1\|^2, \end{aligned} \tag{9}$$

where the first part of L is the normal loss calculation method (e.g., mean square error or cross entropy), $R(\omega)$ is the penalty regularization and ω_i^1 is the weight parameter related to trigger feature in the first layer of the model (we just need to specify the parameter position without knowing its specific value). Adding penalty item will minimize the attention given to these weights during parameter updates, and the final model activation will be less influenced by trigger features, thus invalidating the trigger and mitigating the backdoor impact on overall model performance.

In fact, directly setting ω_i^1 to 0 (also known as neuron pruning) could also reduce sensitivity to trigger features, but it may also lead to a loss of accuracy on clean samples. To circumvent this issue, incorporating a penalty term and subsequently fine-tuning the model enables the preservation of the model's normal performance while effectively reducing sensitivity to trigger features.

4 Experiment

In this section, we aim to evaluate the performance of our proposed algorithm: **BloGBaD**. We will first commence by introducing the experiment settings, followed by a presentation of the evaluation metrics employed in the experiment. Subsequently, experimental defense results will be analyzed. As **BloGBaD** is the first black-box backdoor defense in GNN, we mainly conduct ablation experiments in our study. To obtain a reliable estimate of the model's performance, all experiments will be repeated ten times, and the average values are computed.

4.1 Experiment Settings

To investigate the effectiveness of **BloGBaD**, we employ two classic backdoor attacks [19] (GTA attack) and [24] (GNN backdoor, we abbreviate it to GBA) will be utilized to test our defense approach, while GCN (which leverages convolution in GNN [12]), GAT (which employs attention mechanism in GNN [15]), GraphSAGE (which utilizes neighboring sampling and aggregating in GNN [10]), and GIN [16] will be set as victim models for experiments. As for datasets, 1) Bitcoin, 2) Twitter, 3) COLLAB, 4) AIDS, 5) ChEMBL, 6) Toxicant, and 7) MUTAG will be selected to analyze the defense performances under different tasks. Dataset Details are described in Table 1.

Table 1. Dataset information

Dataset	Graphs	Classes	Average Nodes	Task Type
Bitcoin	658	2	11.53	finance fraud detection
COLLAB	5,000	3	73.49	academic community detection
AIDS	2,000	2	15.69	molecule category prediction
Toxicant	10,315	2	18.67	chemical material classification
MUTAG	188	2	17.93	chemical molecule classification

GTA Attack Configuration We trained both the corresponding topological and feature trigger generators and the backdoored models according to the method proposed in [19]. To maintain backdoor effectiveness and model performance, we adjust the poison rate (limited to 15%) to achieve an attack success rate exceeding 96% while ensuring an accuracy level above 93%. After the training is completed, the trigger generator is used to generate some poisoned graphs to test **BloGBaD**.

GBA Attack Configuration We also poison the training dataset according to the method proposed in [24] and then leverage them model learning, thus

acquiring a backdoored model. For the trigger pattern, we use Erdos Renyl and keep the poison rate within 20%. The backdoor attack achieves a success rate surpassing 75%, while the accuracy for normal samples remains above 74%. Subsequently, the backdoored model will be utilized for the generation of poisoned graphs to evaluate **BloGBaD**.

4.2 Evaluation Metrics

Several evaluation metrics are employed to assess the effectiveness of **BloGBaD**, including 1) attack success rate (ASR, successful attack trials to total trials), 2) model accuracy (Acc, the accuracy for clean samples), 3) detection rate (DR, detected poisoned samples to total attack trials), and 4) false detection rate (FDR, misdetected clean samples to all clean samples).

4.3 GTA Defense

We first present the experimental results of **BloGBaD** based on GTA, including results for various datasets, observations of decision boundary changes, results for different poison rates, and results for different trigger sizes.

Test Results Based on Different Datasets We selected GCN, GAT, and GraphSAGE as victim models to conduct GTA attacks on the Bitcoin, COLLAB, AIDS, and Toxicant datasets, and utilized **BloGBaD** for defense. The defense performance is given in Table 2.

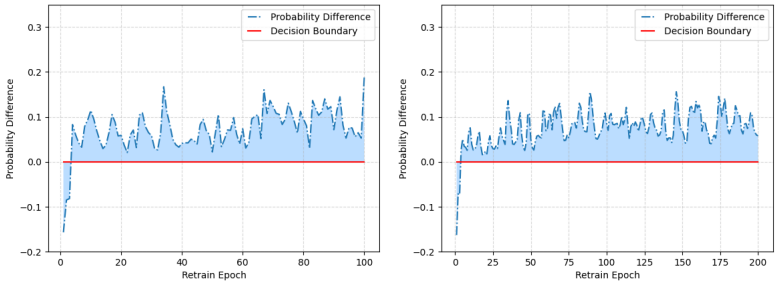
Table 2. Defense results against GTA under GCN, GAT, and GraphSAGE. The best performances are highlighted in this table.

Model	Dataset	ASR (backdoored)	ASR (purified)	Acc	DR	FDR
GCN	Bitcoin	91.4%	11.6%	**93.2%**	100%	0
	COLLAB	88.9%	**9.4%**	84.8%	100%	0
	AIDS	**96.9%**	19.7%	90.1%	99.7%	0.3%
	Toxicant	89.5%	17.6%	90.4%	99.9%	0.1%
GAT	Bitcoin	93.5%	14.4%	92.0%	100%	0
	COLLAB	89.5%	17.2%	87.56	100%	0
	AIDS	95.2%	13.7%	91.0%	99.9%%	0.1%
	Toxicant	93.9%	20.3%	89.8%	99.7%	0.3%
GraphSAGE	Bitcoin	92.6%	9.8%	91.48%	100%	0
	COLLAB	86.1%	18.3%	88.6%	99.8%	0.2%
	AIDS	92.5%	20.8%	91.1%	99.9%	0.1%
	Toxicant	88.3%	14.7%	87.8%	99.8%	0.2%

For GCN, in the Bitcoin dataset, GTA achieves an ASR of 91.4% for the backdoored model, which significantly decreases to 11.6% after employing **BloGBaD** defense. The overall accuracy of the purified model reaches 93.2%, indicating its ability to maintain normal performance. **BloGBaD** exhibits a remarkable detection rate of 100% and no false discoveries. In the COLLAB dataset, the backdoored model has an ASR of 88.9%, reducing to 9.4% after purification. The accuracy of the purified model is 84.8%, and **BloGBaD** continues to demonstrate effective detection without any misdetection. In the AIDS dataset, the ASR for the backdoored model is 96.9%, which decreases to 19.7% after defense. The accuracy reaches 90.1%, and the defense showcases a high detection rate and a low false detection rate. In the Toxicant dataset, the ASR is 89.5%, which drops to 17.6% after employing **BloGBaD** defense. The accuracy remains relatively high at 90.4

Regarding GAT and GraphSAGE, the experimental results exhibited a consistent trend, albeit with slight differences in values.

Decison Boundary Changes In order to further illustrate the change in the model fine-tuning on the decision boundary, we randomly select a poisoned graph sample and investigate the probability difference associated with its classification into the normal class versus the target class throughout the fine-tuning process. The GCN model and AIDS dataset will be selected as the target model, and target dataset for the analysis under 100 and 200 fine-tuning epochs. The results are shown in Fig. 3.



(a) Fine-tuing for 100 epochs (b) Fine-tuing for 200 epochs

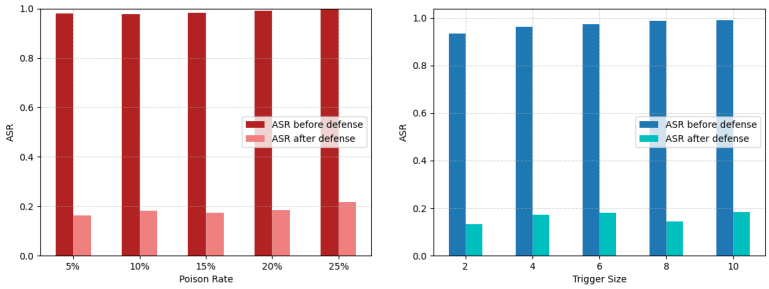
Fig. 3. Illustration of the decision boundary changing process.

Fig. 3(a) depicts the changing of the decision boundary under 100 fine-tuning epochs. Initially, the model predicts a higher probability of the poisoned sample belonging to the target class. However, as the fine-tuning progresses, the probability of classifying the poisoned sample to the normal class asymptotically

surpasses the probability of classifying it to the target class. This suggests that the model is becoming more robust to triggers as it undergoes further fine-tuning.

Fig. 3(b) shows the fine-tuning process for 200 epochs, which exhibits a similar trend to that of 100 epochs.

Poison Rate and Trigger Size From empirical intuition, the increase in the poison rate and trigger size of a backdoor attack typically leads to a higher success rate. To assess the robustness of our defense method against such attack scenarios, we evaluate our defense scheme using different infection rates and trigger sizes. GCN and AIDS dataset are employed as the test model and data. The results are shown in Fig. 4.



(a) Poison rate test result (b) Trigger size test result

Fig. 4. Illustration of the test results for various poison rates and trigger sizes.

Fig. 4(a) illustrates the performance of **BloGBaD** against GTA attacks of varying poison rates. As the poison rate increases from 5% to 25%, the ASR gradually improves from 97% to 99%, while still maintaining a consistently high value (average of 98.4%). Moreover, purified by our defense, the ASRs remain stable at around 18%, with a minimum of 16.3% and a maximum of 21.8%.

Fig. 4(b) presents the performance of **BloGBaD** against backdoor attacks of varying trigger sizes. As the size of the trigger increases from 2 to 10, the ASR also improves from 93% to 99% (average 96.9%). Following purification by **BloGBaD**, the ASR drops to a range of 13% to 18% (average 16.2%).

As Fig. 4 demonstrates, **BloGBaD** is capable of achieving consistently good defense results across different infection rates and trigger sizes.

4.4 GBA Defense

We further evaluate the effectiveness of our defense against GBA. Following the open source code in [24], GIN is employed as the victim model and Bitcoin, COLLAB, AIDS, and MUTAG datasets are leveraged for conducting the attacks.

Then, **BloGBaD** is used for backdoor defense. The results under GBA attack are shown in Table 3.

Table 3. Defense results against GIN. The best performances are highlighted in this table.

Model	Dataset	ASR (backdoored)	ASR (purified)	Acc	DR	FDR
GIN	Bitcoin	**89.3%**	11.7%	**82.9%**	100%	0
	COLLAB	85.5%	9.9%	80.4%	100%	0
	AIDS	80.4%	13.1%	79.2%	99.8%	0.2%
	MUTAG	79.5%	**8.5%**	81.3%	99.7%	0.3%

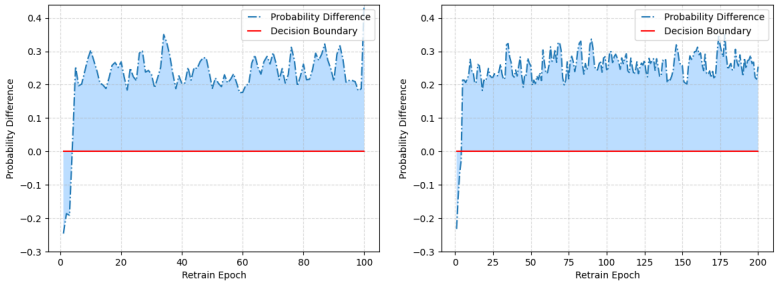
In the Bitcoin dataset, GBA achieved an ASR of 89.3% for the backdoored model, which decreased to 11.7% after purification. The fine-tuned model achieved an accuracy of 82.9%. **BloGBaD** exhibited a perfect detection rate of 100% and a false detection rate of 0%, indicating accurate identification of poisoned graphs. In the COLLAB dataset, the ASR for the backdoored model was 85.5%, reducing to 9.9% after purification. The fine-tuned model achieved an accuracy of 80.4%. **BloGBaD** also demonstrated a high detection rate and a low false detection rate. For the AIDS dataset, GBA resulted in an ASR of 80.4% for the backdoored model and 13.1% for the purified model. The defended model achieved an accuracy of 79.2%. In the MUTAG dataset, GBA showed an ASR of 79.5% for the backdoored model, decreasing to 8.5% after purification. The accuracy of the fine-tuned model remained at 81.3%.

Decision Boundary Changes In line with the GTA-based defense, we also witness the retraining procedure of a GBA attack defense. To better examine the change in decision boundaries, experimental evaluations will be performed at epochs 100 and 200. The test results are shown in Fig. 5.

As depicted in Fig 5(a), for the experiment under 100 epochs, during the initial epochs, the model exhibits a higher probability of classifying the contaminated samples into the target class rather than the normal class (peaking at approximately 24%). Subsequently, as the fine-tuning process advances, the probability of accurate classification by the model gradually increases. Eventually, the probability of correct classification surpasses that of misclassification by an average margin of 25%.

For for the experiment under 200 epochs, similar patterns are also observed in Fig. 5(b).

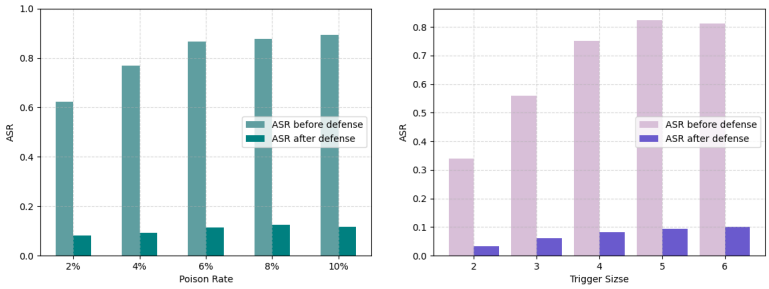
Poison Rate and Trigger Size To assess the effectiveness of our defense approach against GBA attacks with varying infection rates and trigger sizes, we conducted additional defense experiments by systematically varying the poison



(a) Fine-tuing for 100 epochs (b) Fine-tuing for 200 epochs

Fig. 5. Illustration of the decision boundary changing process under GBA defense.

rates and trigger sizes. The COLLAB dataset was selected as the test dataset for our experiments. The results are shown in Fig. 6.



(a) Poison rate test result (b) Trigger size test result

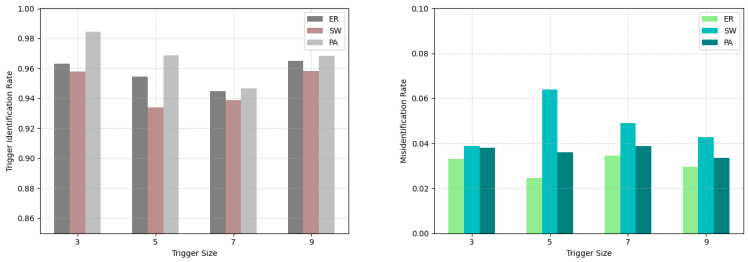
Fig. 6. Illustration of the test results for various poison rates and trigger sizes under GBA defense.

Fig. 6(a) gives the defense result against GBA under various poison rates. As the poison rate increased from 2% to 10%, the ASR increased to approximately 85%, compared to an initial success rate of around 62%. After applying our proposed defense strategy, the ASR success rate was effectively reduced to below 12% (reaching a minimum of 8% and an average of 10.66%).

Fig. 6(b) illustrates the defensive performance of our **BloGBaD** approach against GBA with different trigger sizes. It can be observed that as the trigger size increased from 2 to 6, ASR gradually rose from 34% to 81% (with an average of 65.6%). When **BloGBaD** is applied, all ASR reduced to below 11% (ranging from a minimum of 3.4% to a maximum of 10.2%, with an average of 7.5%).

Phase 1 Performances Based on Sections 4.3 and 4.4, we have demonstrated the overall effectiveness of BloGBaD. In this section, we will further examine the phase 1 efficacy of BloGBaD.

We generated 50 test graphs via GBA on the COLLAB dataset to observe the average trigger identification rate (TIR, the ratio of correctly identified poisoned nodes to the total trigger nodes) and misidentification rate (MIR, misidentified nodes to all detected nodes). Trigger patterns were chosen as Erdős-Rényi subgraph (ER), Small World subgraph (SW), and Preferential Attachment subgraph (PA), with sizes set at 3, 5, 7, and 9 respectively. The test results are shown in Fig. 7.



(a) Trigger Identification Rate (b) Misidentification Rate

Fig. 7. Test results of trigger identification rate and misidentification rate for phase 1.

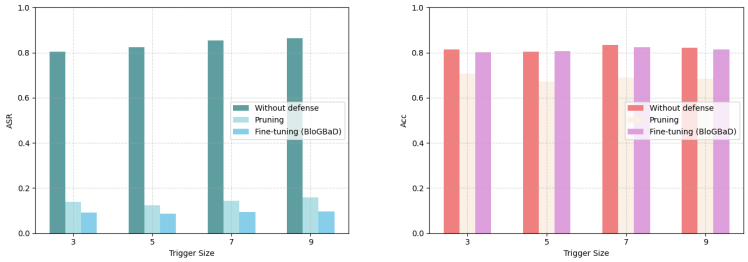
Fig. 7(a) gives the results about trigger identification rate. In the varying settings of different trigger modes and sizes, the TIR of **BloGBaD** in phase 1 consistently surpassed 90%, specifically, 95.69% for Erdős-Rényi subgraph (ER), 94.73% for Small World subgraph (SW), and 96.70% for Preferential Attachment subgraph (PA).

Fig. 7(b) shows misidentification rate results. Across different settings, **BloGBaD** maintained a misclassification rate of less than 7% in phase 1.

Phase 2 Performances We further investigate the performance of phase 2 in BloGBaD. We compare our method with neuron pruning to observe the decrease in ASR and the preservation of Acc. We employ GBA attack under default setting in the code of [24] based on the COLLAB dataset as the target of defense. The test results are shown in Fig. 8.

As can be seen from Fig. 8(a), both neuron pruning and phase 2 in BloGBaD significantly reduce the Attack Success Rate, with BloGBaD demonstrating slightly better performance compared to pruning.

Fig. 8(b) illustrates the accuracy performance of the model after processing. It can be observed that compared to the original backdoored model, the pruned model experiences a certain decrease in accuracy (approximately 11% reduction). However, after the processing with BloGBaD in phase 2, it does not exhibit significant accuracy degradation.



(a) Attack Success Rate (b) Accuracy

Fig. 8. Test results of ASR and Acc for phase 2.

5 Conclusion

In this study, we first discuss the impracticality of current GNN backdoor defense methods: requiring access to model parameters, and then propose a black-box backdoor defense strategy in GNNs: **BloGBaD**. Our proposed approach utilizes a two-phase scheme to effectively eliminate backdoors. In the first phase, referred to as test sample filtration, Gaussian mixing and clustering algorithms are employed to detect and purify trigger nodes and features. The second phase involves model fine-tuning, wherein the trigger features obtained from the previous phase are utilized to retrain the model, ensuring that it becomes immune to the trigger and consequently eliminating the influence of backdoor. Experimental results across different datasets and utilizing two classical attack scenarios demostrates the effectiveness of **BloGBaD** in reducing the attack success rate, detecting triggers, and maintaining model accuracy.

In future work, we plan to improve our approach by designing defense schemes that do not require any poison test samples or prior knowledge about the model.

References

1. Ahmed, M., Seraj, R., Islam, S.M.S.: The k-means algorithm: A comprehensive survey and performance evaluation. Electronics **9**(8), 1295 (2020)
2. Chen, H., Fu, C., Zhao, J., Koushanfar, F.: Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In: IJCAI. vol. 2, p. 8 (2019)
3. Chen, J., Xiong, H., Zheng, H., Zhang, J., Jiang, G., Liu, Y.: Dyn-backdoor: Backdoor attack on dynamic link prediction. arXiv preprint arXiv:2110.03875 (2021)
4. Chen, L., Peng, Q., Li, J., Liu, Y., Chen, J., Li, Y., Zheng, Z.: Neighboring backdoor attacks on graph convolutional network. arXiv preprint arXiv:2201.06202 (2022)
5. Chen, Y., Ye, Z., Zhao, H., Wang, Y., et al.: Feature-based graph backdoor attack in the node classification task. International Journal of Intelligent Systems **2023** (2023)
6. Dai, E., Lin, M., Zhang, X., Wang, S.: Unnoticeable backdoor attacks on graph neural networks. arXiv preprint arXiv:2303.01263 (2023)

7. Dong, Y., Yang, X., Deng, Z., Pang, T., Xiao, Z., Su, H., Zhu, J.: Black-box detection of backdoor attacks with limited information and data. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16482–16491 (2021)
8. Guo, J., Li, A., Liu, C.: Aeva: Black-box backdoor detection using adversarial extreme value analysis. arXiv preprint arXiv:2110.14880 (2021)
9. Guo, J., Li, Y., Chen, X., Guo, H., Sun, L., Liu, C.: Scale-up: An efficient black-box input-level backdoor detection via analyzing scaled prediction consistency. arXiv preprint arXiv:2302.03251 (2023)
10. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
11. Jiang, B., Li, Z.: Defending against backdoor attack on graph nerual network by explainability. arXiv preprint arXiv:2209.02902 (2022)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. OpenReview.net (2017), https://openreview.net/forum?id=SJU4ayYgl
13. Reynolds, D.A., et al.: Gaussian mixture models. Encyclopedia of biometrics **741**(659-663) (2009)
14. Sammaknejad, N., Zhao, Y., Huang, B.: A review of the expectation maximization algorithm in data-driven process identification. Journal of process control **73**, 123–136 (2019)
15. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. stat **1050**(20), 10–48550 (2017)
16. Wang, X., Zhang, M.: How powerful are spectral graph neural networks. In: Chaudhuri, K., Jegelka, S., Song, L., Szepesvári, C., Niu, G., Sabato, S. (eds.) International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA. Proceedings of Machine Learning Research, vol. 162, pp. 23341–23362. PMLR (2022), https://proceedings.mlr.press/v162/wang22am.html
17. Weng, C.H., Lee, Y.T., Wu, S.H.B.: On the trade-off between adversarial and backdoor robustness. Advances in Neural Information Processing Systems **33**, 11973–11983 (2020)
18. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)
19. Xi, Z., Pang, R., Ji, S., Wang, T.: Graph backdoor. In: USENIX Security Symposium. pp. 1523–1540 (2021)
20. Xu, J., Wang, R., Liang, K., Picek, S.: More is better (mostly): On the backdoor attacks in federated graph neural networks. arXiv preprint arXiv:2202.03195 (2022)
21. Xu, J., Xue, M., Picek, S.: Explainability-based backdoor attacks against graph neural networks. In: Proceedings of the 3rd ACM Workshop on Wireless Security and Machine Learning. pp. 31–36 (2021)
22. Yang, S., Doan, B.G., Montague, P., De Vel, O., Abraham, T., Camtepe, S., Ranasinghe, D.C., Kanhere, S.S.: Transferable graph backdoor attack. In: Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses. pp. 321–332 (2022)
23. Zhang, X., Chen, H., Huang, K., Koushanfar, F.: An adaptive black-box backdoor detection method for deep neural networks (2022)
24. Zhang, Z., Jia, J., Wang, B., Gong, N.Z.: Backdoor attacks to graph neural networks. In: Proceedings of the 26th ACM Symposium on Access Control Mod-

els and Technologies. p. 15–26. SACMAT '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3450569.3463560, https://doi.org/10.1145/3450569.3463560
25. Zheng, H., Xiong, H., Chen, J., Ma, H., Huang, G.: Motif-backdoor: Rethinking the backdoor attack on graph neural networks via motifs. arXiv preprint arXiv:2210.13710 (2022)