Clean-Label Graph Backdoor Attack in the Node Classification Task

Hui Xia, Xiangwei Zhao, Rui Zhang, Shuo Xu, Luming Wang

Ocean University of China
{xiahui@,zhaoxiangwei@stu.,zhangrui0504@stu.,xushuo@stu.,wangluming@stu.}ouc.edu.cn

Abstract

Graph neural networks (GNNs) have achieved impressive results in various graph learning tasks. Backdoor attacks pose a significant threat to GNNs, with a focus on dirty-label attacks. However, these attacks often necessitate the inclusion of blatantly incorrect inputs into the training set, rendering them easily detectable through simple filtering. In response to this challenge, we introduce Clean-Label Graph Backdoor Attack (CGBA). The majority of features in the generated poisoned nodes align with their true labels, significantly enhancing the difficulty of detecting the attack. Firstly, leveraging the uncertainty inherent in the GNNs, we develop a low-budget strategy for selecting poisoned nodes. This approach focuses on nodes in the target class with uncertain and low-degree classifications, allowing for efficient attacks within a limited budget while mitigating the impact on other clean nodes. Secondly, we present an innovative strategy for generating feature triggers. By boosting the confidence of poisoned samples in the target class, this tactic establishes a robust association between the trigger and the target class, even without modifying the labels of poisoned nodes. Additionally, we incorporate two constraints to reduce disruption to the graph structure. In conclusion, comprehensive experimental results unequivocally showcase CGBA's exceptional attack performance across three benchmark datasets and four GNNs models. Notably, the attack targeting the GraphSAGE model attains a 100% success rate, accompanied by a marginal benign accuracy drop of no more than 0.5%.

Introduction

Graph Neural Networks (GNNs) have achieved remarkable success in processing graph-structured data (Kipf and Welling 2016a; Velickovic et al. 2017), demonstrating widespread applications across various domains, e.g., drug design (Wu et al. 2022), recommendation systems (Wu et al. 2019), and the financial sector (Wang et al. 2019). GNNs, specifically, aggregate information from neighbors based on graph features and topology to update node representations, enabling predictions and classifications for nodes or entire graphs (Kipf and Welling 2016a). However, akin to Convolutional Neural Networks (CNNs) (Cheng et al. 2021; Zhang et al. 2022), GNNs are vulnerable to backdoor attacks (Xi et al. 2021; Zhang et al. 2021; Xu and Picek 2022; Dai et al. 2023). The exploration of GNNs backdoor attacks is essential for uncovering potential vulnerabilities in GNNs, providing crucial insights and guidance for constructing more secure GNNs models in the future.

The standard backdoor attack (termed dirty-label backdoor attacks hereinafter) (Chen et al. 2017) involves attaching triggers to poisoned samples during the training phase and assigning a target label to them, thereby associating the trigger with the target class. However, this method has a significant weakness: the poisoned samples with attached triggers are conspicuously mislabeled, making it relatively easy to classify these samples as outliers through simple filtering during the model inference phase, prompting further investigation of the attack (Xu and Picek 2022). To overcome this limitation, (Turner, Tsipras, and Madry 2018) introduced the concept of a clean-label backdoor attack. In this form of attack, only a small number of inputs from the target class are poisoned, without any modification to their labels, making it considerably more challenging to detect the presence of poisoned samples. Despite the effectiveness of clean-label backdoor attacks, there is currently no such attack specifically designed for GNNs node classification tasks. To address this gap, we propose a solution in this study.

The implementation of clean-label backdoor attacks is more challenging compared to dirty-label backdoor attacks. Firstly, existing GNNs dirty-label backdoor attacks often involve randomly poisoning a larger number of non-target class nodes, requiring a larger attack budget (Xu, Xue, and Picek 2021; Chen et al. 2023). In contrast, clean-label backdoor attacks can only select a small number of nodes within the target class for the attack. Secondly, existing GNNs dirty-label backdoor attacks mostly use random triggers (Zhang et al. 2021; Xu, Xue, and Picek 2021) and rely on the modification of labels of the poisoned samples to establish a strong association between the triggers and the target label. However, clean-label attacks cannot modify the labels of the poisoned samples. In summary, we face the following challenges: (1) Choosing which nodes to designate as poisoned nodes under a lower budget. (2) The need to establish a firm connection between triggers and the target label under clean-label conditions.

To address the challenges mentioned earlier, we present the Clean-Label Graph Backdoor Attack (CGBA). In response to challenge (1), we've developed a tailored strategy

for selecting poisoned nodes in clean-label attacks to optimize the use of node resources. By leveraging the principle of uncertainty in GNNs, we pinpoint nodes with ambiguous classification outcomes for the attack. Additionally, we introduced a constraint to bias the selection toward nodes with lower degrees, minimizing the impact on other clean nodes. For the poisoned features, we drew inspiration from (Xu, Xue, and Picek 2021), utilizing graph interpretability methods to select the most important features within the poisoned nodes for attaching triggers. To tackle challenge (2), we opt for node feature triggers to poison the data and introduce a novel feature trigger generation strategy. This strategy enhances the connection between triggers and the target label by boosting the confidence of the poisoned samples within the target class. Additionally, to minimize disruption to the graph structure, we impose constraints on the similarity between poisoned nodes and their neighboring nodes, as well as the similarity between the poisoned nodes before and after the poisoning process. These measures decrease the likelihood of detecting the attack. It's important to note that our entire attack process operates within a black-box scenario, devoid of any knowledge about the model parameters. In summary, our contributions can be succinctly summarized as follows:

- We introduce the inaugural clean-label backdoor attack explicitly crafted for graph neural network node classification tasks. Unlike previous approaches, this attack refrains from altering the labels of poisoned nodes, simplifying implementation and reducing the likelihood of detection.
- We craft strategies for poisoning node selection and trigger generation specifically for clean-label backdoor attacks. These strategies establish a robust association between triggers and the target class, even when the labels of poisoned nodes remain unchanged, all within a low budget. To further minimize disruption to the graph structure, we introduced multiple constraints.
- We perform attacks on four types of GNNs using three widely utilized graph datasets. The experimental results reveal exceptional attack performance, especially in scenarios with constrained budgets. For example, on the Pubmed dataset, achieving a modest poisoning rate of only 0.5% led to an impressive 96.94% attack success rate.

Related Works

Backdoor Attacks

Based on whether the injected samples possess consistent characteristics and labels, existing backdoor attacks can be categorized into dirty-label attacks and clean-label attacks.

Dirty-label attacks. Most existing backdoor attacks (Xi et al. 2021; Zhang et al. 2021; Chen et al. 2023; Yang et al. 2022) fall into this category. These attacks typically start by selecting a set of clean examples from non-target classes, apply the backdoor trigger to these examples, and reset their labels to the target class. Training on such a poisoned dataset leads the model to memorize the association between the trigger and the target label. Since the remaining features in the poisoned inputs do not represent the target class, the model is easily able to learn the trigger pattern. Because triggers are typically small patterns, the poisoned inputs still appear to come from non-target classes, making them detectable even through relatively basic data filtering (Turner, Tsipras, and Madry 2018).

Clean-label attacks. To enhance the stealthiness of backdoor attacks, clean-label attacks have emerged. In this type of attack, the poisoned inputs have labels that are consistent with their true labels, making them appear benign even under human inspection. Turner et al. (Turner, Tsipras, and Madry 2018) initially introduced clean label backdoor attacks and implemented them in the field of images. Subsequently, various clean label backdoor attacks targeting CNNs have emerged, such as LC (Turner, Tsipras, and Madry 2019), HTBA (Saha, Subramanya, and Pirsiavash 2020), SAA (Souri et al. 2022), but research on clean label backdoor attacks for GNNs is still limited.

Graph Neural Network Backdoor Attacks

Recent research has shown that GNNs are also susceptible to backdoor attacks. (Xi et al. 2021) and (Zhang et al. 2021) were the first to demonstrate the feasibility of graph backdoor attacks using subgraphs as triggers. They use subgraph triggers to replace the original subgraph in the graph and modify the labels of poisoned graphs to deceive the model into classifying graphs with triggers as the target class. However, their attacks did not extensively explore attack locations. (Xu, Xue, and Picek 2021) and (Wang et al. 2024) guided by graph explanation models, explored the optimal locations for attacks. To reduce the structural disruption caused by the attached trigger, (Chen et al. 2023) improved the similarity of adjacent nodes by pruning low-similarity links, and (Dai et al. 2023) designed imperceptible backdoor attacks by constraining the similarity between attached nodes and poisoned nodes. Xu et al. (Xu and Picek 2022) first demonstrated the feasibility of implementing clean-label backdoor attacks on graph neural network graph classification tasks. However, they did not verify whether it is feasible for node-level classification tasks, and our work fills this gap.

Preliminary Analysis

Notations

We use $G = (V, E, X)$ to denote a graph, where $V = \{v_1, ..., v_N\}$ represents the set of nodes in the graph. E is the set of edges in the graph, and $X = \{x(v_1), ..., x(v_N)\}$ represents the set of node features, where $x(v_i)$ represents the features of a node. In this paper, we focus on node classification tasks, where during training, a set of nodes $V_L \subseteq V$ is associated with labels $Y_L = \{y_1, ..., y_{N_L}\}$. Here, θ represents the trained GNN model, and $f_\theta(x(v_t); G)$ represents the output for a node v_t in the graph G. We used g to represent the trigger, and $v_t : g$ represents a node v with an attached trigger. Nodes with attached triggers in the training set are referred to as poisoned nodes, while nodes with attached triggers in the test set are referred to as victim nodes.

We use G' to represent a graph that contains poisoned nodes or victim nodes.

Threat Model

Attacker's Knowledge and Capability. Similar to most current backdoor attacks, the training data of the target model is accessible to the attacker. The information about the target GNN model, including its architecture, is unknown to the attacker. The attacker can attach triggers to poisoned nodes before the target model's training. In the inference phase, the attacker can attach triggers to the victim nodes.

Attacker's Goal. Successful attacks require the model to have high accuracy on clean data and high attack accuracy on victim data. We can formalize our attack objective as follows:

$$\begin{cases} f_{\theta^*}\left(x\left(v:g\right);G'\right)=y_t \\ f_{\theta^*}\left(x(v);G'\right)=f_{\theta_0}\left(x\left(v\right);G'\right) \end{cases} \tag{1}$$

Where θ^* represents the poisoned model trained with a mixed dataset containing poisoned data, θ_0 represents the clean model, and y_t represents the target class set by the attacker. The first objective indicates that poisoned victim nodes will be misclassified as the target class, and the second objective indicates that for clean data, the output of the poisoned model and the clean model should be close.

Methodology

In this section, we will provide a detailed introduction to the CGBA model. Firstly, we will provide a brief overview of the attack and outline the attack process. Next, we will explain our strategies for selecting the nodes and feature positions for the attack. Finally, we will describe the trigger generation process in detail.

Attack Overview

Figure 1 displays the framework of our clean-label node backdoor attack. Our attack occurs in two phases: the training phase and the inference phase. In the training phase, we first train a surrogate model using a clean dataset, which will guide the selection of poisoned nodes and trigger generation. Next, under the guidance of the surrogate model, we select a few nodes with label y_t and attach triggers to a small number of their features. This dataset containing nodes with attached triggers is referred to as the backdoor training dataset, and throughout this process, the labels of all nodes remain unchanged. Finally, the poisoned model is trained using the backdoor training dataset to create a GNN model known as the backdoor GNN. Since the nodes with attached triggers all have the label y_t, the backdoor GNN directly associates the triggers with the label y_t. In the inference phase, for clean data, the backdoor GNN will produce correct outputs. However, when the attacker attaches triggers to the nodes, the backdoor GNN will predict them as the target labely_t.

Poisoned Node Selection

In this subsection, we provide the details of the poisoned node selection algorithm. Unlike dirty-label attacks, for clean label attacks, the poisoned nodes are chosen from the target class nodes. Intuitively, choosing to attach triggers to nodes in the target class that are more important could be less effective, as these nodes already have high confidence, and slight feature modifications might not have a significant impact on them. On the other hand, selecting nodes with uncertain classification results can confuse the backdoor GNN into believing that the poisoned nodes are classified as the target class due to the attachment of triggers, thereby strengthening the association between triggers and the target class. Therefore, we recommend selecting nodes in the target class with more uncertain classification results as poisoned nodes.

First, we train a simple surrogate model using the training data. Based on the output of the surrogate model, we select nodes with the lowest confidence in the target class for poisoning. In this case, we use a two-layer GCN model as the surrogate model. The loss function for the surrogate model is as follows:

$$\theta=\arg\min_{\theta}\sum_{v_i\in V_L} l\left(f_\theta\left(x(v_i);G\right),y_i\right) \tag{2}$$

Where y_j represents the true label of node v_i, and $l\left(\,,\right)$ represents the cross-entropy loss. Furthermore, we also take into account that if the degree of poisoned nodes is high, then modifying them will affect more nodes, which would have a more significant impact on the classification of other nodes, thereby reducing the accuracy of clean data. Therefore, we introduce a constraint to make the selected nodes more inclined to choose nodes with lower degrees. Finally, we compute an overall score for each node with the target class label, which is formulated as follows:

$$Score\left(v_i\right)=-\left(m\left(f_\theta\left(v_i,y_t;G'\right)\right)+\beta\cdot deg\left(v_i\right)\right) \tag{3}$$

Where $m\left(f_\theta\left(v_i,y_t;G'\right)\right)$ represents the confidence of node v_i on target class y_t in the proxy model θ, and $deg\left(v_i\right)$ denotes the degree of node v_i, and β is an empirical parameter used to control the contribution of node degree in the node selection process. After obtaining the score for each node, we select the top n nodes with the highest scores from all the target class nodes to form the poisoned node set V_p.

Poisoned Feature Selection

Inspired by (Xu, Xue, and Picek 2021), we use the graph explanation network $Graphlime$ (Huang et al. 2022) to select the most important features for the poisoned nodes. $Graphlime$ takes as input a $1\times N$ node feature and the parameters of the graph neural network, and it outputs a $1\times N$ tensor where each value represents the importance of the corresponding feature. A higher value indicates higher importance. In CGBA, we calculate the average importance factor for all the features of the poisoned nodes:

$$AIF=\frac{1}{len\left(V_p\right)}\sum_{x_i\in V_p} Graphlime\left(x_i\right) \tag{4}$$

Where V_p represents the set of poisoned nodes, $Graphlime\left(x_i\right)$ represents the output of the node x in the graph explanation network $Graphlime$. Finally, we select

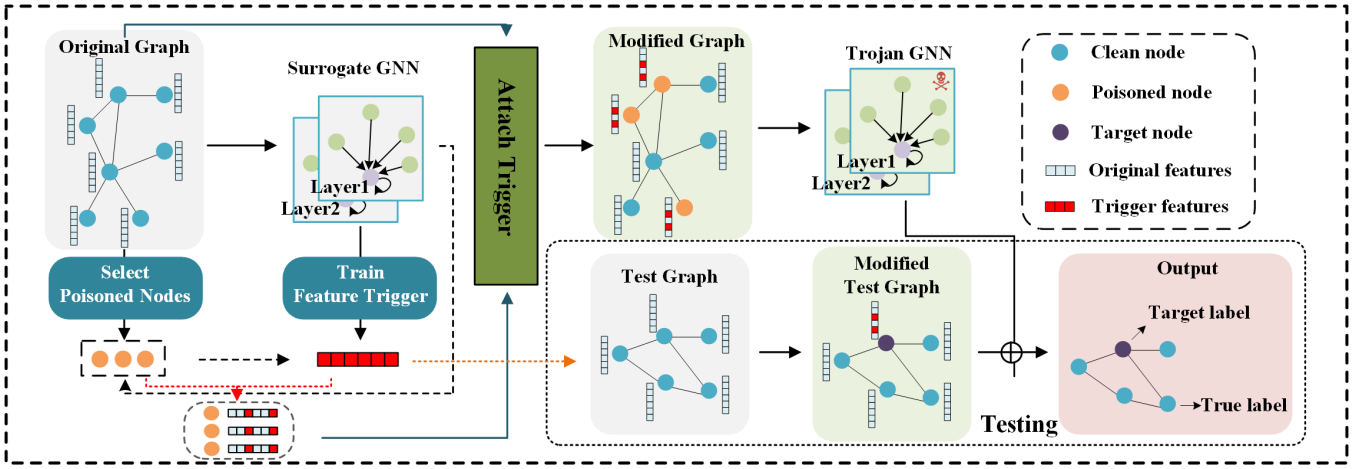


Figure 1: The framework of CGBA. In the lower right corner is the testing phase, while outside the box is the training phase.

the positions of the top m values in AIF as the feature attack locations. Our feature selection method has shown significant improvements in attack effectiveness, as demonstrated in the results in section 5.2.4.

Feature Trigger Generation

Our attack utilizes fixed triggers, and next, we will describe the generation process of the fixed trigger g.

Design of Feature Trigger. In order to achieve our goal, we aim for the victim nodes to produce a classification result that strongly favors the target class y_t after attaching the trigger. For other non-poisoned nodes, we want to maintain clean outputs. First, we attach the trigger to the poisoned nodes in a way that maximizes the confidence for class y_t while minimizing confidence for other classes. This is done to confuse the backdoor GNN into thinking that the poisoned nodes' classification results are strongly associated with the target class, thanks to the trigger. Next, for nodes that haven't been attached with the trigger, we increase their confidence in their true labels. We can formulate our trigger loss as follows:

$$L_c = \min_g \sum_{v_i \in V_p} l\left(f_\theta\left(v_i : g, y_t; G'\right)\right) + \sum_{v_j \in V_L / V_p} l\left(f_\theta\left(v_j, y_j; G'\right)\right) \tag{5}$$

Where V/V_p represents nodes that are not in the poisoned dataset V_p. θ is the GCN proxy model used in Section 4.2.

Design of Concealment. In order to ensure that nodes with attached triggers remain sufficiently concealed, we introduce two constraints. First, considering the homogeneity of neighboring nodes in the graph structure, if a node has a very low similarity with its neighboring nodes, it is likely to be considered an outlier. Therefore, we aim to make nodes with attached feature triggers as similar as possible to their neighboring nodes, to avoid being detected as outliers. We introduce the following loss to guide the generation of more concealed triggers:

$$L_n = \min_g \sum_{v_i \in V_p} \left(\sum_{v_j \in V_i} T - sim\left(x\left(v_i : g\right) - x\left(v_j\right)\right) \right) \tag{6}$$

Where $x\left(v_i : g\right)$ represents the features of node v_i after attaching trigger g, V_i represents the set of all neighboring nodes of node v_i, sim denotes the similarity between the features of two nodes, and we use cosine similarity, and T represents the threshold for neighbor node similarity.

Algorithm 1: CGBA

Input: θ, D_{train} , y_t,n, m,$epoch$.
Output: g.

get the node set V_{y_t} with label y_t from D_{train}.
//get the poisoned node set V_p.
for $v_i \in V_{y_t}$ **do**
  get $Score(v_i)$ by using Eq.(3).
end for
$V_p \leftarrow topn(Score(v_i))$.
//get the poisoned features positions p.
for $v_j \in V_p$ **do**
  get AIF by using Eq.(4).
end for
$p \leftarrow topm(AIF)$.
//train feature trigger.
for $k = 1 \rightarrow epoch$ **do**
  $g^k \leftarrow MLP(g^{k-1})$.
  $V_p[p] \leftarrow g^k[p]$; //Replace the features at position p with g^k for all nodes in V_p.
  calculate the loss L_{all} by using Eq.(8) and optimize MLP.
end for
if D_{train} is discrete dataset **then**
  update g^k by using Eq.(10).
end if
return g^k

Second, we noticed that using Eq.(5) to train the generated triggers could result in some abrupt values. To constrain the magnitude of modifications and make the nodes with attached triggers look more like "normal nodes," we applied a second constraint on the extent of modifications to the training set. This ensures that the nodes with attached triggers are as similar as possible to their original state. We formulate this constraint as follows:

$$L_s = \min_g \sum_{v_i \in V_p} \left(K - sim\left(x\left(v_i : g\right) - x\left(v_i\right)\right)\right) \quad (7)$$

Where K represents the threshold for the extent of modifications, and the similarity function used is still the cosine similarity. Our final trigger generation loss consists of three parts from Eq.(5), Eq.(6)and Eq.(7):

$$L_{all} = L_c + \lambda_1 L_s + \lambda_2 L_n \quad (8)$$

Here λ_1 and λ_2 are weight parameters. Our optimization model uses a two-layer MLP. We optimize the MLP model using Eq.(8) to obtain the final feature trigger, resulting in the final feature trigger g, where x^0 is the randomly generated feature.

$$x^{(k)} = \text{MLP}\left(x^{k-1}\right) \qquad g = x^{(k)} \quad (9)$$

In particular, when the dataset is a discrete dataset, we adopt a truncation method to obtain the final result. For example, when the features consist of only 0 and 1, if the final generated feature value is greater than 0.5, it is set to 1, otherwise, it is set to 0:

$$\begin{cases} g\left[i\right] = 1, \ g\left[i\right] \geq 0.5 \\ g\left[i\right] = 0, \ g\left[i\right] < 0.5 \end{cases} \quad (10)$$

Algorithm 1 describes the details of the CGBA attack. The input includes the proxy model θ, the training dataset D_{train}, the target class y_t, the number of poisoned nodes n, and the number of poisoned features m. The output is the trained feature trigger g.

Experiments

In this section, we will evaluate the proposed method on different datasets and GNN models to answer the following research questions:

- RQ1: Can our proposed method effectively perform backdoor attacks on GNNs while remaining unnoticed?
- RQ2: How do the number of poisoned nodes and the number of poisoned features affect our attack?
- RQ3: Do our node selection and feature selection strategies work effectively?

Experimental Settings

5.1.1 Datasets To demonstrate the effectiveness of our two methods, we conducted experiments on three publicly available real-world datasets: Cora, Citeseer, and Pubmed. Cora and Pubmed are continuous datasets, while Citeseer is a discrete dataset (Sen et al. 2008). These datasets are widely used for inductive semi-supervised node classification. Dataset summaries are provided in Table 1. For each dataset, we split the nodes into a training dataset (70%),a validation dataset (10%), a clean test dataset (10%), and a test dataset with embedded triggers (10%). For the target class of the attack, we choose the class located at class_count / 2 to ensure randomness.

Datasets	Nodes	Edges	Classes	Target class
Cora	2708	5429	7	3
Citeseer	3327	4608	6	3
Pubmed	19717	44338	3	1

Table 1: Dataset Statistics

5.1.2 Evaluation In this paper, we evaluate the effectiveness and evasion capability of our model using two metrics: clean accuracy drop and attack success rate.

(1) Benign Accuracy (BA): BA refers to the accuracy of the GNN model's output when tested on a clean test dataset.

(2) Attack Success Rate (ASR): ASR is the rate at which the targeted nodes in the test dataset, which have been attacked, are successfully classified as the target class.

$$\text{Attack Success Rate}\left(ASR\right) = \frac{\#successful\ trials}{\#total\ trials} \quad (11)$$

5.1.3Baseline We compared our approach with the state-of-the-art subgraph trigger backdoor attack method GTA (Xi et al. 2021), and EGNN (Wang et al. 2024), the node injection backdoor attack UGBA (Dai et al. 2023), the feature trigger backdoor attack methods MIA (Xu, Xue, and Picek 2021), and NFTA (Chen et al. 2023). In addition, we transferred CBA (Xu and Picek 2022) from graph classification task attacks to node classification task attacks (using randomly generated feature triggers injected into random nodes for the attack). MIA, NFTA, and CBA maintain the same experimental settings as CGBA, with the same number of poisoned nodes and attacked features. GTA and EGNN attacked the same number of subgraphs as CBAN. UGBA attached the same number of nodes as CBAN had poisoned nodes. In particular, since NFTA is primarily designed for discrete data sets and flips attacked positions, in our settings, when attacking continuous data sets, its way of generating features is the same as MIA, i.e., randomly generated.

5.1.4 Parameter settings The poisoning ratio in the training set for Cora and Citeseer is 2%, and for Pubmed, it is 1%. The feature modification ratio for CGBA and the other two feature modification attacks is 5%. The value of parameter β is 0.01. We deployed a 2-layer GCN as the proxy model, and CGBA used a 2-layer MLP to train the feature triggers.

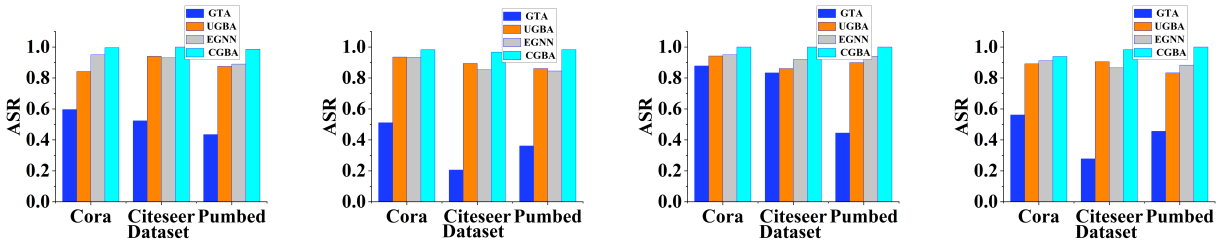
Attack Results

5.2.1Comparisons with baseline To answer RQ1, we will compare CGBA with baselines on three real-world graph datasets to understand the attack performance and stealthiness.

In Table 2, we report the results of CGBA and three feature trigger attack methods on four different GNN mod-

Datasets	GNN Type	clean	MIA	NFTA	CBA	CGBA
Cora	GCN	86.44	87.22|85.30	90.56|85.56	61.67|85.24	**98.89**|**86.17**
	GAT	86.42	90.55|85.80	95.56|85.67	73.33|85.42	**98.33**|**86.54**
	GraphSAGE	85.56	93.89|85.18	99.44|85.18	83.22|85.56	**100.0**|**85.62**
	GAE	85.30	86.67|81.60	86.67|81.17	73.17|82.04	**93.89**|**84.94**
Citeseer	GCN	75.78	90.56|**75.80**	80.00|74.19	40.56|74.38	**98.89**|75.37
	GAT	74.52	70.56|72.08	80.37|74.55	43.89|72.03	**96.67**|**74.60**
	GraphSAGE	74.46	99.44|74.14	78.89|73.04	74.33|73.43	**100.0**|**74.40**
	GAE	72.95	87.78|73.89	79.63|72.24	41.15|73.80	**98.33**|**74.19**
Pubmed	GCN	86.98	38.89|84.75	55.01|85.77	47.22|86.67	**96.67**|**87.13**
	GAT	86.25	41.11|85.74	55.56|85.77	56.11|85.83	**98.33**|**86.18**
	GraphSAGE	88.57	67.78|**88.69**	45.56|88.47	62.78|88.46	**100.0**|88.52
	GAE	88.12	54.44|88.17	41.67|88.11	61.674|87.95	**100.0**|**88.20**

Table 2: Backdoor attack results (ASR (%) |BA (%)). Only clean accuracy is reported for clean graphs. The optimal effect is achieved by using bold font to highlight the representation.



(a) GCN (b) GTA (c) GraphSage (d) GAE

Figure 2: Comparison Results with Other Attack Methods(ASR), (a), (b), (c), and (d) correspond to different GNN models being attacked.

els (GCN (Kipf and Welling 2016a), GraphSAGE (Hamilton, Ying, and Leskovec 2017),GAT (Velickovic et al. 2017),GAE (Kipf and Welling 2016b)). Overall, our proposed backdoor attack exhibits excellent attack effectiveness and stealthiness on all three datasets. Specifically, our attack method achieves an ASR of over 96% on all three datasets, indicating that our attack can successfully change the labels of victim nodes in the test set to the target label. Moreover, CGBA's BA is advantageous compared to the baseline model, outperforming the baseline model in most cases. This suggests that our attack can maintain the accuracy of predictions for clean nodes, achieving a stealthy attack. Finally, the consistent success across different GNN models demonstrates the robust transferability of our model.

In addition, we compare CGBA with the subgraph trigger attack method GTA and EGNN and the node injection backdoor attack UGBA in terms of ASR in the Figure 2. From the figure, we can see that CGBA has a significant advantage in attack success rate compared to GTA and UGBA. In the Table 3, we list the comparison of CGBA with GTA ,EGNN and UGBA in terms of BA on different neural network models. From the table, it can be seen that CGBA's performance is generally better than the baseline models. Overall, CGBA shows significant improvements in both attack success rate and stealthiness compared to the baseline models.

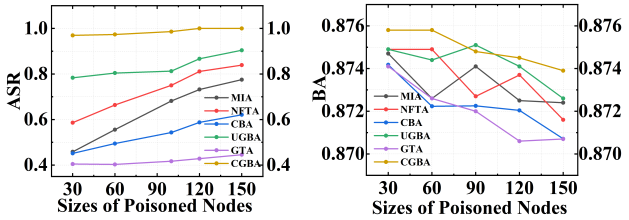
In order to further validate the stealthiness of our attack, we also tested the performance of our attack against potential defenses. Two defense methods are based on measuring the similarity of neighboring nodes. When the similarity between adjacent nodes falls below a fixed threshold, these methods consider the edge to be anomalous. The Prune defense method removes these anomalous edges, while the Isolate method not only removes the anomalous edges but also isolates anomalous nodes connected to the anomalous edges. The results are averaged over four different GNN models. The results are presented in Table4. When facing two defense schemes, the success rates of other attack methods all decrease to varying degrees. However, CGBA shows almost no decrease in the face of defense, and even exhibits an increase in ASR on the Cora and PubMed datasets. This may be because our triggers are deliberately designed to maintain a high level of similarity between poisoned nodes and their neighboring nodes. When the defense is applied, the poisoned nodes are not affected, and clean nodes may be removed, leading to an increase in ASR.

Impact of the Sizes of Poisoned Nodes and Features

To answer RQ2, we conducted experiments to explore the impact of different numbers of poisoned nodes and modified features during the training phase on our attack.

Datasets	GNN Type	clean	UGBA	GTA	EGNN	CGBA
Cora	GCN	86.44	85.86	86.10	85.55	**86.17**
	GAT	86.42	86.10	85.68	86.00	**86.54**
	GraphSAGE	85.56	85.56	84.07	85.43	**85.62**
	GAE	85.30	83.27	80.18	82.98	**84.94**
Citeseer	GCN	**75.78**	75.05	75.00	75.22	75.37
	GAT	74.52	74.50	74.40	74.11	**74.60**
	GraphSAGE	74.46	73.79	73.67	73.56	**74.40**
	GAE	72.95	72.14	72.20	72.81	**74.19**
Pubmed	GCN	86.98	86.94	86.76	86.74	**87.13**
	GAT	86.25	85.89	85.52	85.69	**86.18**
	GraphSAGE	88.57	**88.73**	88.35	87.53	88.52
	GAE	88.12	88.15	**88.20**	88.13	**88.20**

Table 3: Comparison Results with Other Attack Methods (BA (%)). Only clean accuracy is reported for clean graphs. The optimal effect is achieved by using bold font to highlight the representation.

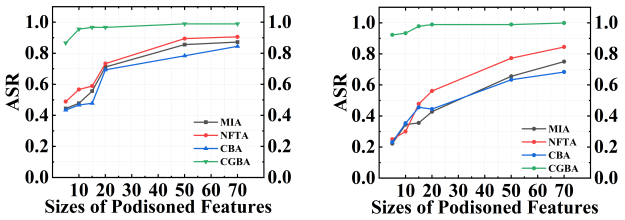


(a) ASR (b) BA

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

Impacts of sizes of poisoned nodes. We conducted attacks by poisoning the training set with {30, 60, 90, 120, 150} poisoned nodes, while keeping the other settings consistent with those mentioned in section 5.1.4. The results, presented in Figure 3, show the average ASR and BA on the Pubmed dataset for four different graph neural network models. Similar results were obtained for the other datasets. As expected, the ASR increases and BA decreases as the number of poisoned nodes increases. ASR remains at a high level when the number of poisoned nodes reaches 90. Moreover, our results consistently outperform the baseline across different numbers of poisoned nodes, providing strong evidence for the effectiveness of our attack method. Additionally, even at lower poisoning rates, our method maintains a high level of attack performance, demonstrating that our approach can achieve successful attacks on a lower budget.

Impacts of sizes of poisoned features. Meanwhile, we conducted experiments with varying numbers of poisoned node features. In Figure 4, we report the variation in ASR with the increase in the number of poisoned features in the Cora and Pubmed datasets. The results indicate that the effect of modifying features is similar to that of the number of poisoned nodes, as ASR increases with the increase in the number of



(a) Cora (b) Pubmed

Figure 4: Impacts of sizes of poisoned features.

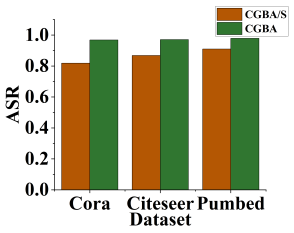


Figure 5: Impact of Different Node Selection Strategies

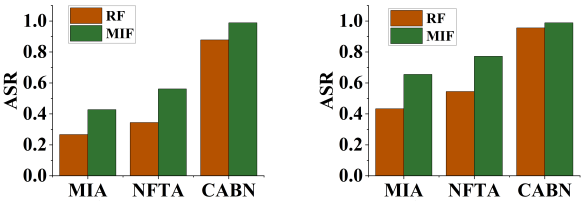
poisoned features. Most importantly, our approach significantly outperforms the baseline model with the same number of feature attacks. It is worth noting that even with a very small number of feature attacks, our attack still achieved significant success.

Ablation Experiments

To answer RQ3, we conducted ablation experiments to examine the impact of different node selection and feature selection strategies on our model.

Parameter Sensitivity Analysis

Impact of Different Node Selection Strategies. We conducted a variant experiment called CGBA/S, which randomly selects and poisons nodes. To highlight the effectiveness of our node selection strategy, we conducted attacks with fewer poisoned nodes, using one-third of the poisoning

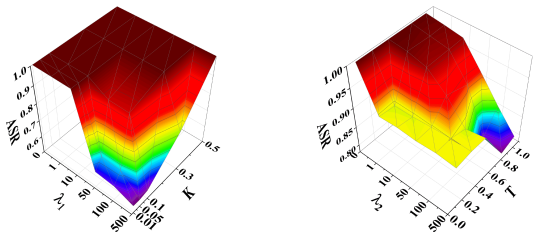


(a) Pubmed-20 (b) Pubmed-50

Figure 6: Impact of Different Feature Selection Strategies. (a) Present the results of poisoning 20 nodes on the Pubmed dataset, and (b) display the results of poisoning 50 nodes on the Pubmed dataset.

Datasets	Defense	MIA	NFTA	CBA	UGBA	GTA	EGNN	CGBA
Cora	None	90.83	90.56	70.12	92.56	44.43	91.36	**98.89**
	Prune	90.75	86.67	66.56	91.11	21.06	83.53	**97.26**
	Isolate	62.91	80.10	65.36	89.99	35.01	80.27	**98.63**
Citeseer	None	86.67	80.00	55.23	94.45	44.43	92.98	**99.16**
	Prune	85.61	76.33	41.52	85.00	35.05	89.36	**98.89**
	Isolate	69.95	74.11	43.29	91.11	6.67	84.56	**98.33**
Pubmed	None	49.56	55.01	60.25	89.70	54.43	86.55	**96.67**
	Prune	44.34	54.44	51.25	83.88	37.75	77.53	**96.59**
	Isolate	31.11	54.29	49.36	89.44	12.74	79.12	**95.55**

Table 4: Comparative ASR(%) Evaluation under Varied Defensive Strategies. None indicates that no defense was used.



(a) Kandλ_1 (b) Tandλ_2

Figure 7: The Impact of Parameters.

quantity from the experiments in Section 5.2.1, on all three datasets. All other settings remained the same for fairness. Figure 5 presents the results of our experiments on the three datasets. From the table, we can observe that our node selection strategy significantly outperforms the variant experiment. This demonstrates that selecting nodes with the lowest confidence in the target label from the training set can effectively enhance the attack performance.

Impact of Different Feature Selection Strategies. We explored the impact of different feature selection strategies by comparing Most Important Feature (MIF) selection to Random Feature (RF) selection. We conducted experiments with two different feature attack quantities, which were 20 and 50. The results, as shown in Figure 6, Selecting the most important features for poisoning had a significantly better effect compared to random feature selection, especially when attacking a smaller number of features. This clearly demonstrates the effectiveness of choosing features with the highest average importance for poisoning. Additionally, even when attacking the same features, our ASR outperforms the baseline model, indicating that our trained feature triggers can establish a stronger association with the target labels.

In this subsection, we further investigate the impact of hyperparameters K, T, λ_1, and λ_2 on the performance of CGBA. Here, $K$and T represent the thresholds for feature modification magnitude and the similarity threshold between a poisoned node and its neighboring nodes, while $\lambda_1$and λ_2 represent the weights for these two components.

K and λ_1 Parameter Analysis. We set the value of K as $\{0.01, 0.005, 0.1, 0.3, 0.5\}$ and the value of λ_1 as $\{0, 1, 10, 50, 100, 500\}$. In Figure 7(a), we report the impact of K and λ_1 on the Attack Success Rate (ASR) of CGBA on the Pubmed dataset. We observe that as K decreases and λ_1 increases, the magnitude of feature modifications to nodes becomes smaller, leading to a reduction in ASR. This is as expected, because when the restriction on the magnitude of modifications to nodes is smaller, the estimated budget will increase, leading to a stronger attack effect. It is worth noting that when K exceeds 0.1, the ASR can be maintained at a high level, and when K reaches 0.5, the ASR can remain at 100%. This indicates that our attack can achieve a very strong effect on the Pubmed dataset with a maximum node modification magnitude of 0.5, which is within our expectations.

T and λ_2 Parameter Analysis. We set the value of T as $\{0.1, 0.3, 0.5, 0.8, 1.0\}$, and the value of λ_2 as $\{0, 1, 10, 50, 100, 500\}$. In Figure 7(b), we report the effects on the Pubmed dataset. We can observe that when λ_2 has a lower value, ASR increases with an increase in T. We hypothesize that this effect may be due to the homogeneity of the graph, where neighboring nodes tend to have similar classification results. Poisoned nodes are more likely to have neighbors with classifications biased towards the target class y_t. As we increase the similarity with neighboring nodes, it further boosts the confidence of poisoned nodes in the target class. When we increase the similarity with neighboring nodes, it further boosts the confidence of the poisoned nodes in the target class. Conversely, when λ_2 has a higher value, ASR decreases as T increases. This is possibly because excessive weight negatively impacts the optimization of L_c, leading to a reduction in the effectiveness of the attack.

Conclusion

In this paper, we propose CGBA, a groundbreaking clean-label backdoor attack approach tailored specifically for GNNs node classification tasks. Unlike traditional graph backdoor attacks, clean-label backdoor attacks achieve a backdoor attack by modifying only the output data without altering the labels. Our approach addresses the gap in clean-label node-level backdoor attacks on graph neural networks. Experiments have shown that CGBA can effectively achieve clean-label node-level backdoor attacks.

Acknowledgements

This research is supported by National Key Research and Development Program of China under grant number 2024YFB3311802, the National Natural Science Foundation of China (NSFC) under grant number 62172377, the Taishan Scholars Program of Shandong province under grant number tsqn202312102, and the Startup Research Foundation for Distinguished Scholars under grant number 202112016.

References

Chen, X.; Liu, C.; Li, B.; Lu, K.; and Song, D. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*.

Chen, Y.; Ye, Z.; Zhao, H.; Wang, Y.; et al. 2023. Feature-Based Graph Backdoor Attack in the Node Classification Task. *International Journal of Intelligent Systems*, 2023.

Cheng, S.; Liu, Y.; Ma, S.; and Zhang, X. 2021. Deep feature space trojan attack of neural networks by controlled detoxification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 1148–1156.

Dai, E.; Lin, M.; Zhang, X.; and Wang, S. 2023. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of the ACM Web Conference 2023*, 2263–2273.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Huang, Q.; Yamada, M.; Tian, Y.; Singh, D.; and Chang, Y. 2022. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*.

Kipf, T. N.; and Welling, M. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N.; and Welling, M. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Saha, A.; Subramanya, A.; and Pirsiavash, H. 2020. Hidden trigger backdoor attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 11957–11965.

Sen, P.; Namata, G.; Bilgic, M.; Getoor, L.; Galligher, B.; and Eliassi-Rad, T. 2008. Collective classification in network data. *AI magazine*, 29(3): 93–93.

Souri, H.; Fowl, L.; Chellappa, R.; Goldblum, M.; and Goldstein, T. 2022. Sleeper agent: Scalable hidden trigger backdoors for neural networks trained from scratch. *Advances in Neural Information Processing Systems*, 35: 19165–19178.

Turner, A.; Tsipras, D.; and Madry, A. 2018. Clean-label backdoor attacks.

Turner, A.; Tsipras, D.; and Madry, A. 2019. Label-consistent backdoor attacks. *arXiv preprint arXiv:1912.02771*.

Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y.; et al. 2017. Graph attention networks. *stat*, 1050(20): 10–48550.

Wang, D.; Lin, J.; Cui, P.; Jia, Q.; Wang, Z.; Fang, Y.; Yu, Q.; Zhou, J.; Yang, S.; and Qi, Y. 2019. A semi-supervised graph attentive network for financial fraud detection. In *2019 IEEE International Conference on Data Mining (ICDM)*, 598–607. IEEE.

Wang, H.; Liu, T.; Sheng, Z.; and Li, H. 2024. Explanatory subgraph attacks against Graph Neural Networks. *Neural Networks*, 172: 106097.

Wu, L.; Cui, P.; Pei, J.; Zhao, L.; and Guo, X. 2022. Graph neural networks: foundation, frontiers and applications. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4840–4841.

Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 346–353.

Xi, Z.; Pang, R.; Ji, S.; and Wang, T. 2021. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*, 1523–1540.

Xu, J.; and Picek, S. 2022. Poster: Clean-label Backdoor Attack on Graph Neural Networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 3491–3493.

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

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

Zhang, Q.; Ma, W.; Wang, Y.; Zhang, Y.; Shi, Z.; and Li, Y. 2022. Backdoor attacks on image classification models in deep neural networks. *Chinese Journal of Electronics*, 31(2): 199–212.

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