Backdoor Attack of Graph Neural Networks Based on Subgraph Trigger

Yu Sheng, Rong Chen, Guanyu Cai, and Li Kuang(✉)

School of Computer Science and Engineering, Central South University, Changsha Hunan 410075, China
kuangli@csu.edu.com

Abstract. Graph Neural Networks (GNN) is a kind of deep learning model to process structural and semantic features of graph data. They are widely used in node classification, graph classification, and link prediction. However, deep learning models require a lot of training data and computational costs, and users usually choose the models provided by third-party platforms. Attackers make full use of their insecurity, subtly modify the training data, and affect the model accuracy. To ensure the service quality and the model robustness, researches on model attacks and defenses are launched. As a new type of attack, backdoor attacks have also been verified on the GNN model. However, existing research still has the following problems: 1) the design of triggers is single; 2) the selection of attack nodes is random; 3) the attack is only effective for some specific GNN models. To address these problems, we study the GNN backdoor attack based on the subgraph trigger. We design the trigger based on the features of the sample data and use the random graph generation algorithm to obtain the subgraph trigger. We propose to select the attack nodes by fusing the local and global structural features and fine-tuned edges when inserted into datasets. We apply it to multiple GNN models. Finally, we use fewer nodes, smaller densities and randomly fine-tune the trigger structure, the experimental results show that the attack we propose has a significant effect on the real datasets, in which clean accuracy drop is less than 0.07 and the attack success rate increases more than 75%.

Keywords: Backdoor attack · Graph classification · GNN

1 Introduction

In the real world, graph data has wide applications. Nodes and edges can represent individuals and relationships in different scenarios, respectively. For example, in social networks, a graph illustrates the attributes of each person and the relationships among them. In transportation areas, a graph can represent the changes in traffic flow between different regions. Due to the powerful expression of graphs, deep learning models which analyze and capture information in graphs have also attracted more attention. The graph neural network model learns the structural features of the graph by aggregating the information of nodes and their neighbors, which has achieved excellent performance in various graph analysis tasks such as node classification, graph classification and link prediction. With the promotion of deep learning, the computing costs

H. Gao and X. Wang (Eds.): CollaborateCom 2021, LNICST 407, pp. 276–296, 2021.
https://doi.org/10.1007/978-3-030-92638-0_17

for dealing with large amounts of data and huge models are increased. Service providers upload trained models to third-party platforms, and users use them to obtain services. Due to the incredible providers, malicious attackers have the opportunity to attack models. Previous studies have shown that deep learning models have some inherent weaknesses, there is uncertainty with some subtle disturbances. If a malicious attacker makes a simple modification to the training dataset, the accuracy will be affected. The attacker operates samples during the training and testing phases, reducing the model's accuracy to affect the quality of services, or to achieve some illegal intrusion purposes, such as releasing malicious advertisement.

Common model attacks include poisoning attacks [1–5], model inversion attacks [6–9], and model extraction attacks [10–13]. The poisoning attack occurs at the model training stage, aiming at reducing the model performance. The attacker mixes well-designed samples into the training dataset to mislead the model. For example, a panda picture is mixed with noise in the image data, and the picture is identified as Gibbon [14]. In the network graph, edges are added or deleted to modify the graph structure or to change the features of the nodes. Thereby the accuracy of the classification task is affected. Model inversion attack uses the memory information in the training process of the neural network to infer the statistical information of the dataset from the model. The attacker can use the inferred information to synthesize the datasets, and the users' private information will also be leaked during the iterative inference. The model extraction attack aims to infer the model's parameters, hyper-parameters, architectures, and functions. The attackers continuously submit input samples to query the corresponding prediction of the model. As a result, the confidential information of the model can be extracted from many inputs and outputs, and the entire model can even be inferred, which leads to an approximate replacement model constructed for further attacks.

The backdoor attack is a new type of poisoning attack. The attacker inserts well-designed triggers into datasets, the backdoor model normally shows in clean samples. When the samples contain triggers, they will be misclassified as the label which the attacker specifies. Deep neural network (DNN) models have been demonstrated that they were easily attacked, such as image recognition [15, 16] and text prediction [17, 18]. For example, an attacker uses triggers to forge a legitimate user's identity to deceive the access control system. Due to the system's misjudgment, the user's life and property safety are threatened [19]. As a type of deep learning model, GNN is also vulnerable to backdoor attacks. In a recommendation system or social network, the attacker generates a specific subgraph trigger and modifies the relationship between users according to the trigger. The relationship between two users will be added or deleted, and the target labels are changed to the ones specified by the attacker. Users may be exposed to malicious advertisements, reducing the recommendation system's service quality and user experience. Our original intention of studying the graph neural network backdoor attack is to guess and simulate the various ideas and methods of the attacker as much as possible, and once the weakness of model is found, the subsequent research can pay attention to designing a better model defense plan.

The dataset used by the GNN model is a graph composed of nodes and edges. Inspired by the backdoor attack in the image domain, we use a specific subgraph structure, which is composed of a small number of nodes and edges, as the backdoor

trigger for the graph data. The existing research on backdoor attacks to GNN still has the following problems:

- The generated trigger is single. Most of the existing researches design a specific subgraph structure as the trigger, and then insert it for all target samples uniformly, while the diversity of the triggers is not investigated thoroughly.
- The selection of attack nodes is random. The attack nodes are usually selected by randomly sampling from the graph structure. The importance of nodes in the entire graph and the influence of nodes on the attack success rate are not investigated in detail.
- The backdoor attack method cannot adapt to various kinds of GNN models. Existing researches only show the effectiveness of attack to some specific models of GNN for some particular tasks, lacking verification on multiple variant models of GNN.

To address the problems, we start the research on backdoor attacks to GNN models and focus on three points, i.e., the generation of trigger, the selection of attack nodes, and the injection of trigger. We insert the backdoor triggers into the dataset of the GNN graphs for classification task and evaluate the effect of backdoor attacks. Our main contributions can be summarized as follows:

- We make statistical analysis of the dataset and generate triggers with specific labels according to the statistical features, and when inserting triggers, we reserve the structure of the trigger for a certain proportion of samples randomly while making minor adjustments of the trigger for the rest of samples.
- We propose a method to select attack nodes based on the structural features of the nodes, which combines the local and global structural features of the nodes in the entire graph to evaluate the importance of the nodes, and then select the most important nodes to attack.
- We use real datasets with node features to verify the effect of attack on multiple GNN models, and prove that the proposed method can achieve a relatively high attack success rate.

The rest of this paper is organized as follows. Section 2 introduces the related research in the attack of deep learning models, especially the backdoor attack to GNN models. Section 3 shows the preliminary concepts. Section 4 presents our motivation and our proposed method of backdoor attack to GNN model. Section 5 gives the experimental details and result analysis. Finally, Sect. 6 provides the conclusions and future work.

2 Related Works

Deep learning models convenient for users but expose some weaknesses. For example, the model suffers severe losses caused by some subtle modifications related to the safety of the model and the quality of service, and the modifications even violate users' privacy and legal rights. Many researchers explore the attacks and defenses to improve the stability of the model in the face of uncertain factors. In this chapter, we will introduce the related work of attacks on deep learning model.

Attacks on Deep Learning Models. Szegedy et al. [20] found that the attacker perturbed the input data in the neural network model, which eventually led to the wrong output of the model. This discovery laid the foundation for the research on neural network model attacks. The researches on model attacks and defenses are essential in artificial intelligence and security. In deep learning, a lot of studies have been launched from the attack of deep learning models in different fields, such as image recognition [21–24]. The attacker added one or more pixels to the image, and these special pixels act as triggers to make the image with these pixels be misclassified into other classes. In natural language processing [25–27], the attacker modified words without changing the flow of the sentence. It changed the sentiment or meaning of the entire sentence. And in the GNN model with graph data [28–30], attackers inserted samples of specific subgraph structure triggers, misclassifying the samples into the class specified by them. Attackers have different data manipulation permissions. In most cases, they only need to manipulate a few data, causing model errors.

Backdoor Attacks. Gu et al. [31] first proposed this concept. From this work, we can learn that the goal of a backdoor attacker is to insert a hidden backdoor, that is, a trigger, into the model. They applied the backdoor trigger to the field of traffic sign recognition. The "stop" traffic sign with a trigger will be misidentified as "deceleration". When the backdoor is not activated, the infected model can recognize benign samples normally. Once the input sample has an activated backdoor trigger, the model will modify the predicted label to the target label specified by the attacker, which is a misclassification. Users often input their data to train the models provided by third-party platforms, they cannot guarantee the security of third-party platform, It is challenging to discover the backdoor, and it is harmful to users' privacy and security. Most of the existing works on backdoor attacks are oriented to the image data, and attackers often manipulate images by designing pixel triggers [32]. Chen et al. [19] applied backdoor attacks to face recognition, and inserted triggers, such as "red glasses" into the access control system. When a malicious user wears the same glasses as the trigger, it can impersonate a legitimate user. In federal learning, similar backdoor attacks will also occur [33–35]. Backdoor attacks threaten the security of the model and can cause huge losses to users. Therefore, research on them is very important and developing.

Attack on GNN Models. Many studies have shown that operating node features or modifying structure in the graph can affect the learning performance of the GNN model. Zügner et al. [36] first studied the adversarial attack on the attribute graph. They modified the node features and graph structure in the attribute graph, which significantly reduced the accuracy of the node classification model. Dai [30], Ma [37] used reinforcement learning to explore strategies for modifying the graph structure, achieving remarkable results. For the GNN model, the backdoor attack based on the adversarial attack, Zhang et al. [38] studied the backdoor attack of random inserted triggers. To select the attack nodes randomly under a designed subgraph trigger structure, they proposed a random smoothing method to defend backdoor attacks effectively. At the same time, Xi et al. [39] also studied graph neural network backdoor attacks. This study focused on trigger generation. They used the attention mechanism to generate customized triggers, which thoroughly combined the characteristics of different graph structures to achieve better attack effects. There are still several

problems in the research of GNN backdoor attacks. First, the trigger is too simple. In most cases, the shape of the triggers is fixed. A graph structure is obtained through the random graph generation algorithm to be used as a single trigger. Second, the impact of the importance of nodes on the attack effect is not considered. The third is that backdoor attack has not been verified on various GNN models, and the attack effect lacks universality. In our work, we have carried out corresponding research and have made improvements on the above three problems. See Sect. 4 for details.

Attack Nodes Selection. In the social networks, the analysis of key nodes is an important research work, because key nodes are responsible for the critical role of information transmission in the graph structure. At present, this research has a complete theory in recommendation systems and traffic road networks. Using this intuition, attackers find the key nodes in the network graph to attack when the GNN model aggregates the information of the nodes and their neighbors. The attacked nodes transmit the wrong information to the neighbors and the model. Attackers only need to operate with a small amount of data and get a better attack effect. Ma et al. [40] proposed a node importance measurement method based on random walks, attacking nodes with higher importance scores, and verified that attacking important nodes can effectively reduce the model's accuracy. Takahashi et al. [41] considered the relationship between the classification result of a node and the neighbor nodes, modeling the neighbor nodes of the graph as a neighbor tree. They selected the most influential node from the 2-hop neighbors as the attack target. Mo et al. [42] proposed measuring the local structural characteristics and global structural characteristics of the nodes, and Ma et al. [43] proposed a three-layer comprehensive impact evaluation index to measure the influence of the nodes. In [44], they took advantage of a GNN interpreter to identify the importance of nodes, then selected key nodes to insert backdoor triggers. Currently, the existing works on GNN backdoor attacks do not measure the nodes from the structural features. Our work solves this problem and measures the importance of nodes based on nodes' features.

3 Preliminaries

This section mainly introduces the basic concepts involved in the paper, laying a theoretical foundation for further research.

3.1 Concepts

Graph. Undirected graphs are used in our work. We define them as $G = (V, E)$, where $V = (v_1, v_2, \ldots, v_N)$, which means there are N nodes in the graph, and E means the set of edges $e_{i,j}$ existing between any two nodes v_i and v_j. $A \in \{0, 1\}^{N \times N}$ represents the adjacency matrix of the graph. There is an edge $e_{i,j}$ between node v_i and v_j, then $A_{i,j} = 1$, otherwise 0. At the same time, we also have a feature matrix $X \in \mathbb{R}^{N \times D}$, where x_i represents the l-dimensional features of node v_i.

GNN Graph Classification. We regard the GNN model for the graph classification task as a function $f_G : G \rightarrow y$, where f_G is the graph classifier, $y = \{y_1, y_2, \ldots, y_K\}$ indicates that the graph has K classes. The input data is represented as M samples of Graphs G and their corresponding label y, namely $D_G = \{(G_1, y_1), (G_2, y_2), \ldots, (G_M, y_M\}$. By learning the embedding of the nodes, aggregating the features of the node and its neighbors, the model predicts graph labels as $\widehat{y}$. The graph classification by GNN is shown in Fig. 1.

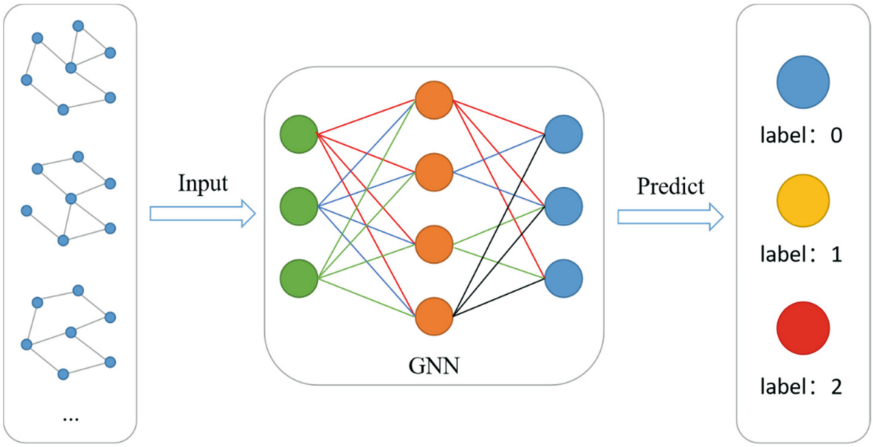

Fig. 1. GNN graph classification task

GNN Backdoor. Inspired by the poisoning attack of the GNN model, we design the backdoor attack by adding or deleting the edges of the graph to disturb the structure, as shown in Fig. 2. Unlike the poisoning attack, this modification does not happen randomly. Then, the attacked nodes in the graph are replaced one by one according to the trigger's structure. The selected attacked nodes are the same as the number of nodes in the subgraph trigger. And we will modify their original links between the attacked nodes according to the structure of the trigger.

3.2 Attack Model

Our attack model is like the work of [38, 39]. The backdoor is inserted into the graph data by well-designed subgraph triggers. In this process, we do not change the architectures and parameters of the model. A specific wrong label is assigned to the trigger. We assume that the learning model comes from a third-party platform, users input their data to obtain services. Therefore, the attacker can operate the training and test data, design and insert samples with triggers, and attack a clean model. Once the model is successfully backdoored, the clean sample input by the user can get similar results to the clean model, which is difficult to detect the backdoor triggers. While inputting the samples with triggers, the attacker's target output will be predicted by the backdoored model, and the classification accuracy will decrease significantly.

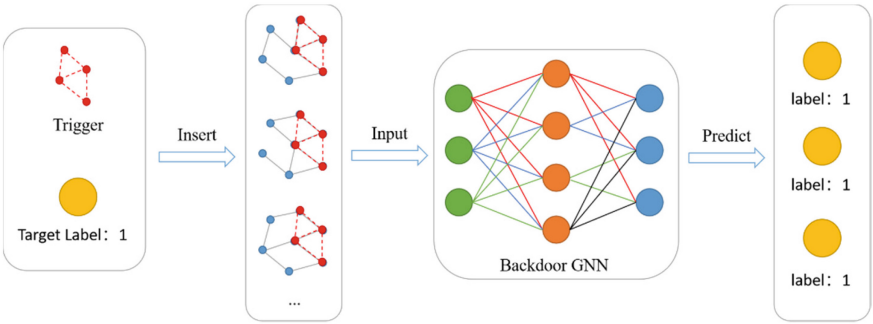

Fig. 2. Backdoor attack on GNN

4 Backdoor Attack on GNN

4.1 Attack Overview

We divide the entire attack process into three stages, the generation of triggers, the selection of attack nodes and trigger insertion. First, we need to generate a trigger with a specific structure as the "key" to open the model's backdoor. After that, we should find the nodes with the most apparent attack effect as the target. Finally, we insert the trigger both in the training and testing phases. The process is shown in Fig. 3.

Our attack scheme has two main features. First, to achieve diverse trigger attacks, we make minor adjustments based on the basic structure of the trigger for a certain proportion of samples, that is, randomly delete a small part of the edges in the trigger and maintain most of the trigger structure. Second, we measure the importance of nodes based on their global and local structural features and select the nodes with higher importance as attack nodes. The attack's goal is to classify the samples with the specific triggers to the target label specified by the attack while keeping the classification of clean samples as much as possible.

4.2 Generation of Trigger

The generation of the trigger is composed of three steps: First, we analyze the statistical information of each dataset so that we can design triggers for different datasets and different categories. Second, we use a random graph generation algorithm to generate the structure of the trigger based on the statistical information achieved in the previous step. Third, we use the nodes' label in graph samples to generate nodes' feature of the trigger, and the attacker will modify the graph labels.

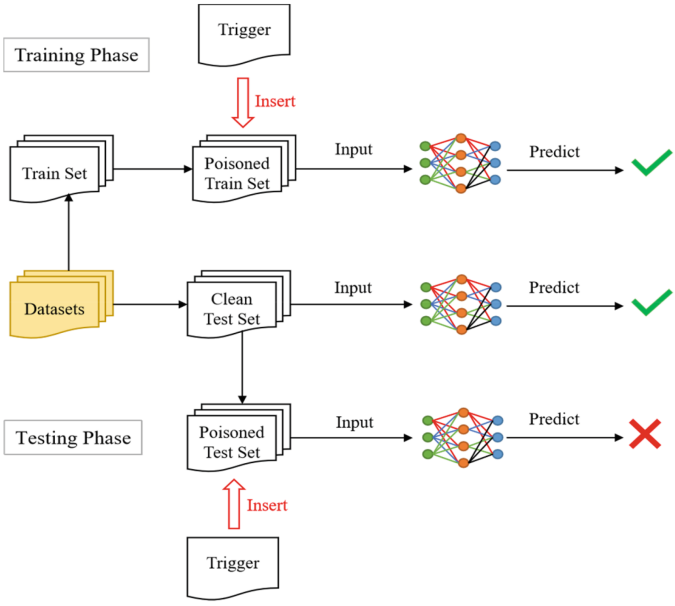

Fig. 3. The process of backdoor attack

Statistical Analysis of Dataset. We believe that there are many similarities between the graphs of the same label and differences among the graphs of different labels. We use the similarities and differences as a breakthrough in the design of backdoor triggers. Therefore, we associate the trigger with target label graphs and insert them into other labels. In the same way, we can also generate triggers corresponding to each class based on the samples of other classes.

Assuming that the attacker can only manipulate a small part of the training dataset, we randomly select 5% of the samples from the clean training dataset. Specifically, we define the target label of the attack as y_t and $y_t \in y$. In 5% of the attack samples, we calculate the average nodes $\overline{n_t}$ and average density $\overline{\rho_t}$ of samples y_t (shown in the Eq. 1 and Eq. 2), and node features x_t.

$$\overline{n_t} = \frac{\sum_{i=1}^{M_{train} \times 5\%} n_i}{M_{train} \times 5\%} \tag{1}$$

where M_{train} represents the total number of samples in the training set, n_i represents the number of nodes in sample i, the same as Eq. 2.

$$\overline{\rho_t} = \sum_{i=1}^{M_{train} \times 5\%} \frac{2e_i}{n_i(n_i - 1)} / (M_{train} \times 5\%) \tag{2}$$

where e_i represents the number of edges in sample i, and $n_i(n_i - 1)/2$ represents the edges of the fully connected graph obtained by connecting all nodes in sample i.

For the node features vector x_t, we first encode the labels of each node with one-hot to obtain the vector x_i of each node i, and then count the total number of each label. Then we sum each column of the matrix composed of all node vectors, and finally normalize them to obtain the node labels distribution probability vector x_t.

Random Subgraph Trigger Generation. We use the Erdős-Rényi (ER) algorithm [45] to generate a trigger structure. In this algorithm, the input is the number of nodes m and the density ρ, the number of nodes represents the size of the trigger, and the density represents the probability of an edge between two nodes in the trigger, then output is a random graph g. According to the statistical feature of the graphs under target label y_t, we use the average number of nodes $\overline{n_t}$ and the average density $\overline{\rho_t}$ as input to obtain the trigger g_t related to the label y_t, see Algorithm 1.

Algorithm 1: ER random graph generation

Input: m-nodes of trigger; ρ-density of trigger; y_k-target label of trigger
Output: g_k-a random subgraph trigger with specific label y_k
// initialization
specify a topological order $v_0, v_1, \ldots, v_m$ for nodes
foreach node v_i in nodes **do**
    add node v_i to g_k
    select node v_j in nodes randomly
    generate $r \in [0,1]$ randomly
    if $r < \rho$ then
        add edge (v_i, v_j) to g_k
return g_k, y_k

Trigger Features Design. First, we use the node label distribution to generate the node characteristics of the trigger. Specially, we use one-hot encoding to obtain a vector s_i of length l for all node labels, representing the label vector of node i. The vector of n nodes can form a matrix with $n \times l$. Since a node has only one label, this matrix is very sparse. We sum up each column and finally get the label distribution of the nodes in the samples. After normalization, we get the probability of node feature distribution, and generate trigger node features based on it. Figure 4 shows the process of trigger generation.

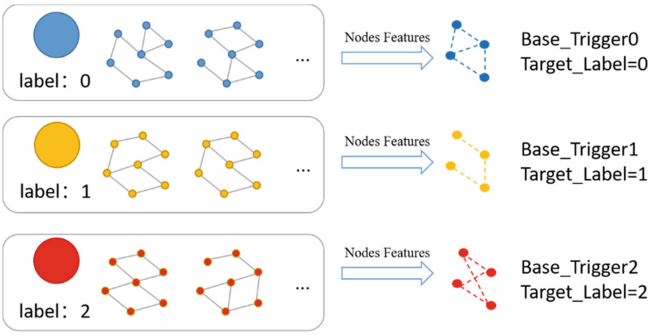

Fig. 4. Trigger generation with structural and node features

4.3 Selection of Attack Nodes

Since the GNN model aggregates information by learning the characteristics of nodes and neighbors, key nodes are essential in information transmission in the graph structure. We believe that selecting the most significant nodes to attack will increase the attack success rate. We use the degree centrality (DC) and closeness centrality (CC) of the node as the nodes' local structural feature and the global structural feature, respectively. Their definitions are as follows.

Degree Centrality. In an undirected graph, the degree centrality of node i is expressed as the number of nodes directly connected to node i, the larger the number of 1-hop neighbors of a node is, the bigger the degree centrality of the node is, the more important the node in the graph network. The degree centrality of node i is expressed as Eq. 3:

$$DC(i) = \sum_{j \neq i}^{N} \left(e_{i,j} \in E\right) \tag{3}$$

Closeness Centrality. Indicates the closeness of the distances between the current node and other nodes. If the sum of the distances between this node and other nodes is shorter, the closeness centrality is higher, and the relationship with other nodes is closer. Suppose there are multiple disconnected components in the dataset, the nodes between different connected components have no paths to communicate with others. The nodes' number of different connected components will also be different, so the closeness centrality of node i can be calculated by the following Eq. 4:

$$CC(i) = \frac{n-1}{N-1}\left(\frac{n-1}{\sum_{j}^{n-1} d_{i,j}}\right) \tag{4}$$

We use Eq. 4 to calculate the closeness centrality of multiple unconnected subgraphs, where N represents the number of all nodes in the sample graph, n represents the number of nodes in the connected component where node i is located, and the result

is the ratio of the average distance between the middle node to reachable nodes in the connected component.

Subsequently, we normalize the obtained DC and CC to map the two indicators into the same interval, and the normalized indicators are respectively marked as DC_{norm} and CC_{norm}, the two indicators are fused as shown in Eq. 5.

$$DCC = \lambda DC_{norm} + (1 - \lambda)CC_{norm} \tag{5}$$

where λ is the fusion factor used to control the ratio of two indicators.

4.4 Backdoor Insertion

After identifying the attacking nodes, we insert triggers into the graphs. In this process, the edges between the attack nodes and the normal nodes will not change, and the attack nodes are replaced with the trigger nodes one by one, including the structural information between the nodes and the features of the nodes.

We divide the trigger insertion into two phases: training and testing phase. The attacker's goal is not to affect the accuracy but ensure the attack is hidden in the training phase. We only select a small part of the training data to insert triggers. Since the generated triggers are related to a specific class of graphs, labels of samples with triggers are all modified to the target label in the training set. In the testing phase, the attacker hopes to increase the attack effect and ensure the effectiveness of the attack. In a word, the accuracy does not change significantly on the clean samples but drops significantly on the poisoned test set with inserted triggers. We insert triggers into all samples in the test set and evaluate the effect of the attack based on the results predicted by the model.

5 Experiment and Evaluation

5.1 Datasets

We use three public datasets DD[1], Mutagenicity[2] and Proteins-full[3] with graph labels and node labels in experiments. The statistical information is shown in Table 1. The three datasets are all binary classification tasks, and the part in brackets represents the data situation of the two labels. According to statistical information, we conclude that samples with different labels have different structural features such as node distribution and density in the same dataset. We generate the trigger by selecting the information of the target label, so that our trigger is associated with the sample data, and the attack target is not blind, this label can be used as the attacker's set tag during attack.

[1] https://networkrepository.com/DD.php.

[2] https://networkrepository.com/Mutagenicity.php.

[3] https://networkrepository.com/PROTEINS-full.php.

Table 1. Statistical information of datasets

Datasets	DD	Mutagenicity	Proteins-full
Graphs	1178	4337	1113
Classes	2(690|486)	2(2400|1935)	2(662|449)
Avg_Nodes	284(355|183)	30(29|31)	39(50|23)
Max_Nodes	5748	417	620
Min_Nodes	30	4	4
Node_Labels	89	14	3
Avg_Edges	715.66(903.3|449.25)	30.77(30.29|31.35)	72.82(94.08|41.47)
Avg_Desity	0.0278(0.02|0.04)	0.0913(0.09|0.09)	0.2122(0.14|0.32)
Avg_Degree	5.03	2.03	3.73
Max_Degree	19	4	25
Min_Degree	1	0	0

5.2 Evaluation Metrics

We set up three evaluation indicators to comprehensively evaluate our attack, including model accuracy, attack success rate, and clean test set accuracy drop. The detailed explanation is as follows.

Model Accuracy. We denote the label of the graph i as y_i, and the predicted result of the model is $\widehat{y_i}$. The accuracy represents the proportion of correctly classified samples in the total samples, which is expressed as Eq. 6:

$$ACC = \frac{\sum_i^N \mathbb{I}(\widehat{y_i} = y_i)}{N} \tag{6}$$

Attack Success Rate. The attack success rate is used to measure the effectiveness of the attack. We mark the attacker's expected error label as y, and attack success rate is expressed as the misclassified samples among all the samples classified as y. The attack success rate can be expressed as Eq. 7:

$$ASR = \frac{\sum_i^N \mathbb{I}(\widehat{y_i} = y | y_i \neq y)}{\sum_i^N \mathbb{I}(y_i \neq y)} \tag{7}$$

Clean Accuracy Drop. This indicator is used to calculate the difference between the accuracy of the clean test set and the baseline test set when triggers are inserted into the training set but not in test set. The accuracy of the baseline test set comes from the test accuracy of the clean dataset when the backdoor triggers are not inserted. It is defined as Eq. 8:

$$CAD = ACC_{baseline\_test} - ACC_{backdoor\_clean\_test} \tag{8}$$

According to the above indicators, our goal is to only insert triggers into the training set without affecting the model's accuracy. After triggers are inserted into both the training set and test set, the accuracy will drop significantly, ensuring that the attack success rate is high, and the accuracy drop of the clean test set is as small as possible.

5.3 Experimental Setup

According to node features distribution and target label of the trigger, we use the ER algorithm to generate a trigger with m nodes and density of ρ. If there are no special instructions, we default m = 4, ρ is from the average density of the sample with the label of 0 in the attacked graph, and the target label is 0. The training set and the test set are divided based on the ratio of 9:1. In the first stage of the attack, we randomly select 5% of the samples from the training set, according to the DCC ranking, select the m nodes with the highest scores as the attack nodes, and modify the graphs' labels. In the second stage of the attack, we insert triggers in the same way for all test samples, and keep a clean test set as a reference without modifying the graph labels. Table 2 summarizes the default parameter settings of the model.

Table 2. Default parameter settings

Type	Parameters	Setting
GCN	Layer	2
GraphSAGE	Layer	2
	Aggregate	Mean
GIN	Layer	5
	Aggregate	Sum
Training	Optimizer	Adam
	Learning rate	0.01
	Dropout	0.5
	Epochs	350
Trigger	Size	4
	Density	Sample
	Intensity	5%(train_set),100%(test_set)
	Nodes features	Sample, Reverse
	Tuning parameter	0.8

Note: Layer in the table refers to the number of layers of the GNN model. Aggregate is an aggregation function. Sample means it is generated according to the statistical features of the corresponding dataset, and reverse sorts the probability values in the distribution vector of the node features based on Sample. Then we reverse the result of the sorting, exchanging the largest and smallest values one by one. Finally, we obtain a new probability value distribution and generate node features according to the new distribution vector

5.4 Result and Analysis

We evaluate the basic classification accuracy of different models on three datasets as a baseline for subsequent reference and comparison. The results are shown in Table 3.

Table 3. Baseline accuracy

Model	Accuracy	DD	Mutagenicity	Proteins-full
GIN	TrainAcc_Baseline	0.9903	0.9024	0.7445
	TestAcc_Baseline	0.7421	0.8078	0.7398
GraphSAGE	TrainAcc_Baseline	0.9961	0.8682	0.7901
	TestAcc_Baseline	0.7419	0.8295	0.7190
GCN	TrainAcc_Baseline	0.9519	0.8443	0.7690
	TestAcc_Baseline	0.7266	0.7717	0.7473

Influence of Trigger Size and Density. In this experiment, we inserted triggers into the training and test sets on the GIN model. We used the comprehensive indicators of CAD and ASR to choose the most negligible impact on the dataset (the CAD is smaller), the better attack effect (the ASR is large), and triggers with fewer nodes. The results are shown in Fig. 5, where (a) shows the influence of the number and density of trigger nodes on CAD; (b) is the influence of the number and density of trigger nodes on ASR.

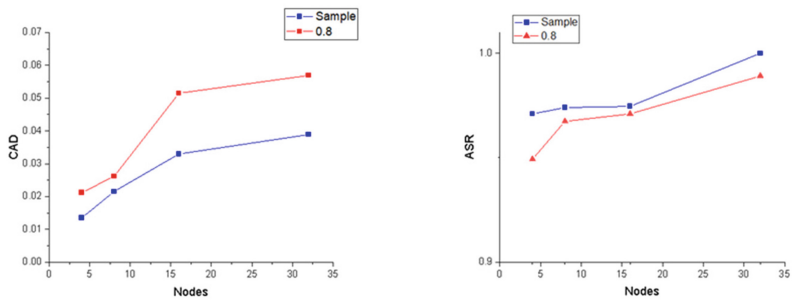

Fig. 5. Influence of trigger parameters

From Fig. 5(a), we can find that the smaller the number of trigger nodes, the smaller the CAD, and the smaller the impact on the clean dataset. At the same time, the sample density is much less than 0.8, and the higher the trigger density, the larger the CAD, the greater the impact on the clean dataset. Figure 5(b) shows the relationship between the attack success rate and the number of nodes and density. We found that the larger the number of nodes, the higher the attack success rate, and the sample density of the dataset can be used simultaneously to get better results.

Through the above results, we prefer to select triggers with a smaller number of nodes and a density closer to the dataset to ensure a higher attack success rate and better interference to the dataset, making the attack effective and hidden. Finally, we set the number of trigger nodes to 4 nodes, and the trigger density is the average density of the target category of each dataset. On the one hand, four nodes have achieved a higher attack success rate and a lower clean dataset error. On the other hand, four nodes can have more connection methods than other nodes, which is convenient to operate the structure of the trigger.

Influence of Trigger Node Features. We use two ways to add node features to the trigger subgraph. We count the node features of the attacked sample graphs, mapping them into a one-dimensional vector, and then normalize them to get the node features' distribution probability vector. The Sample method generates similar node features based on the distribution vector. In contrast, the Reverse uses reverse sorting based on Sample to get opposite node features distributions to generate opposite node features. By comparing the attack success rate of the two methods, we find that the latter's attack effect is more obvious, as shown in Fig. 6. Therefore, the node features generation of our trigger is implemented according to the Reverse method, and we use the generated trigger with node features as the default trigger for subsequent comparison experiments.

Influence of Attack Nodes Selection. In response to the selection of attacking nodes based on the importance of the nodes proposed in our work, we compare the DCC and randomly select nodes, the most important nodes and the least important nodes based on the DCC metric. The experimental results are shown in Table 4, where MaxDCC indicates that the nodes with the highest DCC value are selected, and MinDCC is the opposite.

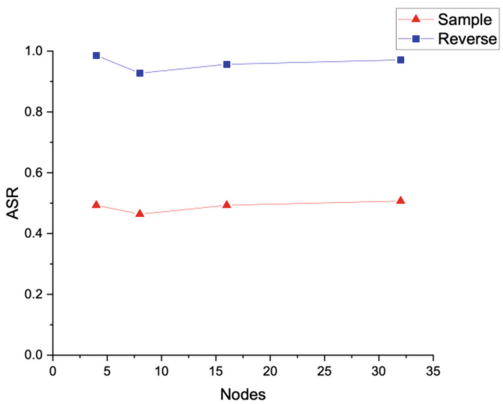

Fig. 6. Influence of node features

Table 4. Influence of attack nodes selection

Method	TrainAcc	CleanTestAcc	BackdoorTestAcc	ASR
RandomSample	0.9742	0.6772	0.3580	0.9420
MaxDCC	0.9852	0.6821	0.3611	0.9710
MinDCC	0.9814	0.6684	0.5377	0.8841

According to the results in Table 4, it is better to select attack nodes based on the importance of nodes than random selection, and the more important nodes are attacked, the higher the attack success rate is. The DCC that we propose achieves a 97.1% success rate, which is better than random selection. The method is about 3% higher than the random sample, and about 9% higher than MinDCC.

Fine-Tuning of Trigger. This experiment learns the attack effect of inserting triggers with different shapes into the same dataset. We adjust the structure of the trigger with a fine-tuning parameter of 0.8 on the basic trigger structure. The fine-tuning parameter guides us to retain the edges of the trigger, and we obtain triggers with different structures. The structure of the trigger subgraph is not fixed, which is more difficult to detect the backdoor by analyzing the graph structure. The experimental results are presented in Table 5. The first row of each dataset corresponds to the result of using the basic trigger, and the second row is the fine-tuning of the base trigger. We also use three other triggers to compare the fine-tuning experiments and then visualize the maximum attack success rate and the average attack success rate. Results are shown in Fig. 7.

Table 5. Results of fine-tuning trigger

Datasets	TrainAcc	CleanTestAcc	BackdoorTestAcc	ASR
DD	0.9852	0.6821	0.3611	0.9710
	0.9856	0.7542	0.3808	0.9855
Mutagenicity	0.8833	0.7868	0.3038	1
	0.8871	0.7845	0.3070	1
Proteins-full	0.7246	0.6988	0.6050	0.7679
	0.7184	0.6810	0.5708	0.8214

From Table 5, we find that the fine-tuning of the trigger has an unnoticeable impact on the attack success rate of the DD dataset and the Mutagenicity dataset. We think it is because the trigger densities of these two datasets are small, and the number of nodes is minimal. In the case of a small trigger size, it does not have too many edges to satisfy our fine-tuning strategy, and the Proteins-full dataset density is slightly larger, so the fine-tuning strategy has an impact.

From Fig. 7, we can find that even if the number and density of nodes increase, the fine-tuning of the trigger does not affect the attack effect. In summary, we can conclude that even if the attacker makes a slight modification to the trigger, attack also has a good effect.

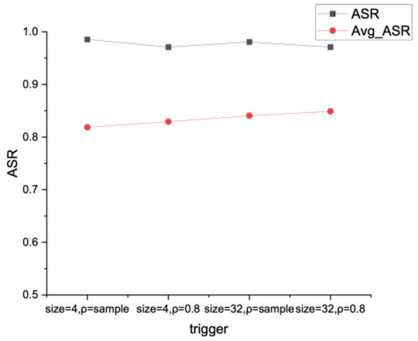

Fig. 7. Results of fine-tuning trigger

Attack on Different GNN Models. To better explore the universal applicability of backdoor attacks to the GNN model, we use the other two classic GNN models (GraphSAGE and GCN). In this part, our attack strategy is to use the base trigger of each dataset, according to DCC metric, to select the node with the largest value for trigger insertion. Table 6 presents the final experimental results.

Table 6. Attack on GNN models

ASR	DD	Mutagenicity	Proteins-full
(*a*) *Attack on GIN*			
Max_ASR	0.971	1	0.7857
Avg_ASR	0.859	1	0.6219
(*b*) *Attack on GraphSAGE*			
Max_ASR	1	1	0.8571
Avg_ASR	0.986	1	0.6721
(*c*) *Attack on GCN*			
Max_ASR	1	1	0.875
Avg_ASR	0.974	1	0.6604

In these three models, GIN pays more attention to the structure information of the sample, GraphSAGE and GCN pay more attention to the node's features. From the results, our trigger modifies the sample structure while also modifying the node features of the sample. In different GNN models, our attack methods have shown promising results. A 100% attack success rate has been achieved on the Mutagenicity dataset.

We compare the number and density of nodes in the three datasets and find that this dataset has fewer nodes and a lower density. The number of nodes in the same trigger. The ratio of attacked nodes and modified edges in this dataset is larger in the whole graph, and the effect will be more significant than other datasets. The Proteins_full dataset has fewer nodes but a slightly larger density. Compared with the Mutagenicity dataset, its edges are more abundant, and the proportion of its edges modified during backdoor attacks is relatively small, so the effect is slightly worse than other datasets. Overall, our attack scheme uses fewer attack nodes and the same density as the dataset. Whether there are more nodes or fewer datasets, we can obtain a higher attack success rate and attack different GNN models.

6 Conclusions and Future Work

In this work, we study the backdoor attack of the GNN model based on subgraph triggers, and use sample statistics to generate specific triggers. We also design a fine-tuning strategy to study the impact of diversified triggers on attack success rate. At the same time, we propose measuring the importance of nodes based on the node structural features and selecting attacking nodes according to the metrics. Our method has been validated on GNN models and multiple real datasets. The experimental results show that designing diversified triggers to implement GNN backdoor attacks is effective. To achieve a higher attack success rate, using node importance to select attack nodes is better than randomly selecting attack nodes.

There are still many issues worthy of researching in our work, the most important of which is to design an effective defense strategy to actively respond to different backdoor attacks and maintain the model's security and stability. In general, it can be divided into the following two aspects.

- Detection of backdoor attacks. Through the research of the attacks, we can find the attacker's similar destructive behaviors, such as the strategy of modifying the dataset. We can start from the anomaly detection of the dataset, detecting whether there is a backdoor trigger in the dataset, or analyzing the similarity of nodes and edges. It can detect abnormal nodes and edges to troubleshoot backdoor triggers.
- Defense against backdoor attacks. Attack and defense are like two parties in a game. Both parties need to constantly explore new strategies to interfere with the opponent's operations. We hope future work will explore the defense against model backdoor attacks from two aspects, pre-training before attack and cleaning the backdoor after attack.

Acknowledgement. This work was supported by the National Natural Science Foundation of China (grant no.61772560).

References

1. Jagielski, M., Oprea, A., Biggio, B., et al.: Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In: 2018 IEEE Symposium on Security and Privacy (SP). IEEE, 19–35 (2018)
2. Shafahi, A., Huang, W.R., Najibi, M., et al.: Poison frogs! targeted clean-label poisoning attacks on neural networks. arXiv preprint arXiv:1804.00792 (2018)
3. Demontis, A., Melis, M., Pintor, M., et al.: Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks. In: 28th {USENIX} Security Symposium ({USENIX} Security 19), pp. 321–338 (2019)
4. Wang, Y., Chaudhuri, K.: Data poisoning attacks against online learning. arXiv preprint arXiv:1808.08994 (2018)
5. Steinhardt, J., Koh, P.W., Liang, P.: Certified defenses for data poisoning attacks. arXiv preprint arXiv:1706.03691 (2017)
6. Basu, S., Izmailov, R., Mesterharm, C.: Membership model inversion attacks for deep networks. arXiv preprint arXiv:1910.04257 (2019)
7. Song, C., Ristenpart, T., Shmatikov, V.: Machine learning models that remember too much. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 587–601 (2017)
8. Veale, M., Binns, R., Edwards, L.: Algorithms that remember: model inversion attacks and data protection law. Philos. Trans. Royal Soc. Math. Phys. Eng. Sci. **376**(2133), 20180083 (2018)
9. Romagnoli, R., Weerakkody, S., Sinopoli, B.: A model inversion based watermark for replay attack detection with output tracking. In: 2019 American Control Conference (ACC). IEEE, pp. 384–390 (2019)
10. Wang, B., Gong, N.Z.: Stealing hyperparameters in machine learning. In: 2018 IEEE Symposium on Security and Privacy (SP). IEEE, pp. 36–52 (2018)
11. Papernot, N., McDaniel, P., Goodfellow, I., et al.: Practical black-box attacks against machine learning. In: Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, pp. 506–519 (2017)
12. Orekondy, T., Schiele, B., Fritz, M.: Knockoff nets: Stealing functionality of black-box models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4954–4963 (2019)
13. Correia-Silva, J.R., Berriel, R.F, Badue, C., et al.: Copycat cnn: Stealing knowledge by persuading confession with random non-labeled data. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8 (2018)
14. Launchbury, J.: A DARPA Perspective on Artificial Intelligence. 11 (2019). Accessed November 2017
15. Li, H., Wang, Y., Xie, X., et al.: Light can hack your face! black-box backdoor attack on face recognition systems. arXiv preprint arXiv:2009.06996 (2020)
16. Zhao, S., Ma, X., Zheng, X., et al.: Clean-label backdoor attacks on video recognition models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14443–14452 (2020)
17. Sun, L.: Natural backdoor attack on text data. arXiv preprint arXiv:2006.16176 (2020)
18. Dai, J., Chen, C., Li, Y.: A backdoor attack against LSTM-based text classification systems. IEEE Access **7**, 138872–138878 (2019)
19. Chen, X., Liu, C., Li, B., et al.: Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526 (2017)

20. Szegedy, C., Zaremba, W., Sutskever, I., et al.: Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199 (2013)
21. Lin, Y., Zhao, H., Tu, Y., et al.: Threats of adversarial attacks in DNN-based modulation recognition. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications. IEEE, pp. 2469–2478 (2020)
22. Goswami, G., Ratha, N., Agarwal, A., et al.: Unravelling robustness of deep learning based face recognition against adversarial attacks. In: Proceedings of the AAAI Conference on Artificial Intelligence, 32(1) (2018)
23. Dong, Y., Su, H., Wu, B., et al.: Efficient decision-based black-box adversarial attacks on face recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7714–7722 (2019)
24. Akhtar, N., Mian, A.: Threat of adversarial attacks on deep learning in computer vision: a survey. IEEE Access **6**, 14410–14430 (2018)
25. Zhang, W.E., Sheng, Q.Z., Alhazmi, A., et al.: Adversarial attacks on deep-learning models in natural language processing: a survey. ACM Trans. Intell. Syst. Technol. (TIST) **11**(3), 1–41 (2020)
26. Morris, J., Lifland, E., Yoo, J.Y., et al.: TextAttack: a framework for adversarial attacks, data augmentation, and adversarial training in NLP. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pp. 119–126 (2020)
27. Behjati, M., Moosavi-Dezfooli, S.M., Baghshah, M.S., et al.: Universal adversarial attacks on text classifiers. In: ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 7345–7349 (2019)
28. Sun, L., Dou, Y., Yang, C., et al.: Adversarial attack and defense on graph data: a survey. arXiv preprint arXiv:1812.10528 (2018)
29. Chen, L., Li, J., Peng, J., et al.: A survey of adversarial learning on graphs. arXiv preprint arXiv:2003.05730 (2020)
30. Dai, H., Li, H., Tian, T., et al.: Adversarial attack on graph structured data. In: International Conference on Machine Learning, PMLR, pp. 1115–1124 (2018)
31. Gu, T., Liu, K., Dolan-Gavitt, B., et al.: Badnets: evaluating backdooring attacks on deep neural networks. IEEE Access **7**, 47230–47244 (2019)
32. Li, Y., Wu, B., Jiang, Y., et al.: Backdoor learning: a survey. arXiv preprint arXiv:2007.08745 (2020)
33. Bagdasaryan, E., Veit, A., Hua, Y., et al.: How to backdoor federated learning. In: International Conference on Artificial Intelligence and Statistics. PMLR, 2938–2948 (2020)
34. Sun, Z., Kairouz, P., Suresh, A.T., et al.: Can you really backdoor federated learning?. arXiv preprint arXiv:1911.07963 (2019)
35. Wang, H., Sreenivasan, K., Rajput, S., et al.: Attack of the tails: yes, you really can backdoor federated learning. arXiv preprint arXiv:2007.05084 (2020)
36. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2847–2856 (2018)
37. Ma, Y., Wang, S., Derr, T., et al.: Attacking graph convolutional networks via rewiring. arXiv preprint arXiv:1906.03750 (2019)
38. Zhang, Z., Jia, J., Wang, B., et al.: Backdoor attacks to graph neural networks. arXiv preprint arXiv:2006.11165 (2020)
39. Xi, Z., Pang, R., Ji, S., et al.: Graph backdoor. In: 30th {USENIX} Security Symposium ({USENIX} Security 21) (2021)
40. Ma, J., Ding, S., Mei, Q.: Towards more practical adversarial attacks on graph neural networks. arXiv preprint arXiv:2006.05057 (2020)

41. Takahashi, T.: Indirect adversarial attacks via poisoning neighbors for graph convolutional networks. In: 2019 IEEE International Conference on Big Data (Big Data). IEEE, 1395–1400 (2019)
42. Mo, H., Deng, Y.: Identifying node importance based on evidence theory in complex networks. Physica A: Stat. Mech. Appl. **529**, 121538 (2019)
43. Ma, L., Liu, Y.: Maximizing three-hop influence spread in social networks using discrete comprehensive learning artificial bee colony optimizer. Appl. Soft Comput. **83**, 105606 (2019)
44. Xu, J., Picek, S.: Explainability-based backdoor attacks against graph neural networks. arXiv preprint arXiv:2104.03674 (2021)
45. Gilbert, E.N.: Random graphs. Ann. Math. Stat. **30**(4), 1141–1144 (1959)