Explainability-based Backdoor Attacks Against Graph Neural Networks

Jing Xu
Delft University of Technology
The Netherlands
j.xu-8@tudelft.nl

Minhui (Jason) Xue
The University of Adelaide
Australia
jason.xue@adelaide.edu.au

Stjepan Picek
Delft University of Technology
The Netherlands
s.picek@tudelft.nl

ABSTRACT

Backdoor attacks represent a serious threat to neural network models. A backdoored model will misclassify the trigger-embedded inputs into an attacker-chosen target label while performing normally on other benign inputs. There are already numerous works on backdoor attacks on neural networks, but only a few works consider graph neural networks (GNNs). As such, there is no intensive research on explaining the impact of trigger injecting position on the performance of backdoor attacks on GNNs.

To bridge this gap, we conduct an experimental investigation on the performance of backdoor attacks on GNNs. We apply two powerful GNN explainability approaches to select the optimal trigger injecting position to achieve two attacker objectives – high attack success rate and low clean accuracy drop. Our empirical results on benchmark datasets and state-of-the-art neural network models demonstrate the proposed method's effectiveness in selecting trigger injecting position for backdoor attacks on GNNs. For instance, on the node classification task, the backdoor attack with trigger injecting position selected by GraphLIME reaches over 84% attack success rate with less than 2.5% accuracy drop.

CCS CONCEPTS

• **Security and privacy** → *Software and application security*;

KEYWORDS

explainability, graph neural networks, backdoor attacks

ACM Reference Format:
Jing Xu, Minhui (Jason) Xue, and Stjepan Picek. 2021. Explainability-based Backdoor Attacks Against Graph Neural Networks. In *WiSec '21: 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, June 28–July 1, 2021, Abu Dhabi, United Arab Emirates.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3468218.3469046

1 INTRODUCTION

Many real-world data can be modeled as graphs, such as social relations or protein structures. Graph Neural Networks (GNNs) have emerged as state-of-the-art for machine learning on graphs [13]. However, similar to Convolutional Neural Networks (CNNs), GNNs are also vulnerable to adversarial attacks, one of which is the backdoor attack. Since GNNs are used increasingly more for security applications [1], it is important to study the backdoor attack on GNNs. Otherwise, security concerns will remain. For instance, in a Bitcoin transaction ego network [12], where the nodes are the transactions, and the edge between two nodes indicates the flow of Bitcoin from one transaction to another, the attacker can attack the GNNs to classify an illegal transaction as a legal one.

In the backdoor attacks on GNNs, the trigger injecting position impacts the attack's performance in terms of the attack success rate and clean accuracy drop. Recently, some works are exploiting the vulnerabilities of GNNs to backdoor attacks with different trigger injecting position selecting strategies [14, 18]. However, these works either select trigger injecting position randomly, in which situation the attack may be easily detected by the defender [18], or use a computationally intensive algorithm to get the trigger injecting position, as shown in [14]. If we know how to quickly select the optimal (or close to optimal) trigger injecting position in backdoor attacks on GNNs, we can achieve high attack performance and good evasion of the defender's detection mechanisms. Further, we can develop more robust GNN models. Unfortunately, since graph data have characteristics of complex relationships and interdependencies between objects, common explainability approaches for CNNs, such as Shapley value [2], are not suitable to explain the predictions of GNNs to select the optimal trigger injecting position.

Deep Neural Networks (DNNs) are vulnerable to backdoor attacks [8]. Specifically, a backdoored neural network classifier produces attacker-desired behaviors when a trigger is injected into a testing example. Several studies showed that GNNs are also vulnerable to backdoor attacks. Zhang et al. proposed a subgraph-based backdoor attack to GNNs for graph classification task [18]. Xi et al. presented a subgraph-based backdoor attack to GNNs, but this attack can be instantiated for both node classification and graph classification tasks [14].

Interpretability methods for non-graph neural networks are not suitable for explaining predictions made by GNNs. Recently, some works try to interpret GNNs. Ying et al. proposed to utilize mutual information to find a subgraph with associated features for interpreting the predicted label of a node or graph being explained [16]. Huang et al. presented a method utilizing predicted labels from both the node being explained and its neighbors, which enables to capture more local information around the node and give a finite number of features as explanations in an intuitive way [6].

In this paper, we propose utilizing powerful neural network approaches that explain predictions made by GNNs to understand the performance of backdoor attacks on GNNs. Indeed, backdoor attacks on GNNs have been presented [14, 18] but how to quickly

This work is licensed under a Creative Commons Attribution International 4.0 License.

WiseML '21, June 28–July 1, 2021, Abu Dhabi, United Arab Emirates
© 2021 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-8561-9/21/06.
https://doi.org/10.1145/3468218.3469046

select the optimal trigger injecting position and what is the impact of different trigger injecting position on the attack performance have not been explored. In this paper, we seek to bridge this gap. To the best of our knowledge, this work represents the first study on the explainability of triggers for backdoor attacks on GNNs. Our contributions can be summarized as follows:

- We utilize GNNExplainer, an approach for explaining predictions made by GNNs, to analyze the impact of trigger injecting position for the backdoor attacks on GNNs for the graph classification task.
- We propose a new backdoor attack on GNNs for the node classification task, which uses a subset of node features as a trigger pattern. Additionally, we explore GraphLIME, a local interpretable model explanation for graphs, to explore the proposed backdoor attack on the node classification task through modifying a different subset of node features.

We conduct an empirical study of the explainability of backdoor attack triggers on GNNs using various state-of-the-art GNNs and four benchmark datasets.

2 BACKGROUND

2.1 Preliminaries

Graph Neural Networks (GNNs). GNNs take a graph G as an input, including its structure information and node features, and learn a representation vector (embedding) for each node $v \in G$, z_v, or the entire graph, z_G. Modern GNNs follow a neighborhood aggregation strategy, where one iteratively updates the representation of a node by aggregating representations of its neighbors. After k iterations of aggregation, a node's representation captures both structure and feature information within its k-hop network neighborhood. Formally, the k-th layer of a GNN is (e.g., GCN [7], GraphSAGE [5], and GAT [11]):

$$Z^{(k)} = AGGREGATE(A, Z^{(k-1)}; \theta^{(k)}), \tag{1}$$

where $Z^{(k)}$ are the node embeddings in the matrix form computed after the k-th iteration and the $AGGREGATE$ function depends on the adjacency matrix A, the trainable parameters $\theta^{(k)}$, and the previous node embeddings $Z^{(k-1)}$. $Z^{(0)}$ is initialized as G's node features.

For the node classification task, the node representation $Z^{(k)}$ of the final iteration is used for prediction. And for the graph classification task, the READOUT function pools the node embeddings from the final iteration K:

$$z_G = READOUT(Z^{(K)}). \tag{2}$$

READOUT can be a simple permutation invariant function such as summation or a more sophisticated graph-level pooling function.

Graph-Level and Node-Level classification. Graph-level classification aims to predict the class label(s) for an entire graph [17]. The end-to-end learning for this task can be realized using graph convolutional layers and readout layers. On the other hand, given a single graph with partial nodes being labeled and others remaining unlabeled, GNNs can learn a robust model that effectively identifies the class labels for the unlabeled nodes [7]. In a node-level classification task, there are two types of training settings - inductive and transductive. In this paper, we focus on the transductive node-level classification task.

Explainability tools for GNNs. GNNExplainer is a model-agnostic approach for providing explanations on predictions of any GNN-based model. Given a trained GNN model and its prediction(s), GNNExplainer returns an explanation in the form of a small subgraph of the input graph together with a small subset of node features that are most influential for the prediction(s) [16]. GraphLIME is a local interpretable model explainability method for graphs. More specifically, to explain a node, GraphLIME generates a nonlinear interpretable model from its N-hop neighborhood and then computes the most n representative features as the explanations of its prediction using HSIC Lasso [6].

2.2 Threat Model

We assume our threat model similar to the existing backdoor attacks, see, e.g., [8]. Given a pre-trained GNN model Φ_o, the adversary forges a backdoored GNN Φ by perturbing its model parameters without modifying the neural network architecture. We assume the attacker has access to a dataset D sampled from the training dataset. Specifically, in the graph classification dataset, the attacker can inject a trigger (graph) to each intended poisoned training graph and change the label to an attacker-chosen target label. In the node classification dataset, the attacker can inject a feature trigger (feature vector) to each intended poisoned training node and relabel the label to the target label. Consequently, our attack is a gray-box attack that does not modify the GNN's model architecture but perturbs the model parameters. This represents a realistic model occurring in real-world settings. For instance, if the training dataset is collected from public users, the malicious users can provide trigger-embedded training data.

3 EXPLAINABLE BACKDOOR ATTACKS

3.1 Backdoor Attacks on Graph Classification

Since most graph classification tasks are implemented by utilizing GNNs to learn the network structure, we focus on subgraph-based backdoor attacks on the graph classification task. Figure 1 illustrates the pipeline of subgraph-based backdoor attack on GNNs for the graph classification task. Formally, in the training phase, the attacker injects a trigger (graph) g_t to a subset of the original training dataset and changes their labels to the attacker-chosen target label to obtain the backdoored training dataset. A GNN model trained using the backdoored training dataset is called backdoored GNN Φ. Then in the testing phase, the adversary injects the same trigger to a given graph G. If we define such trigger-embedded graph as G_{g_t}, the adversary's objective can be defined as:

$$\begin{cases} \Phi(G_{g_t}) = y_t \\ \Phi(G) = \Phi_o(G) \end{cases} \tag{3}$$

The first objective in Eq. (3) means that all the trigger-embedded graphs are required to be misclassified to the target class y_t, i.e., attack effectiveness. In contrast, the second objective ensures that the backdoored GNN performs indistinguishably on normal graphs compared to the original GNN, i.e., attack evasiveness.

It is challenging to find the optimal trigger injecting position so that the adversary can reach several goals: 1) high attack success

rate, 2) high accuracy in normal graphs, and 3) difficult to detect by the defender. More precisely,

- if we sample t nodes from the graph uniformly at random as the trigger nodes in the trigger graph, the trigger will likely be injected into a subgraph that is important for the GNN's final prediction. As a result, the defender can detect the trigger-embedded graphs easily;
- to achieve the second objective, we can select a subgraph similar to g_t in the graph as the trigger injecting position. However, the computation of the similarity between graphs is a complex task as subgraph isomorphism is known to be NP-complete. The graph matching algorithms require an exponential time for computation [3].

To overcome the above challenges, we utilize GNNExplainer to optimize the trigger injecting position to ensure the attack effectiveness and attack evasiveness at the same time.

- We first apply GNNExplainer to analyze the prediction of GNNs to understand the impact of each structure in the graph on the classification result from GNNs.
- Instead of selecting t nodes from the graph uniformly at random as the trigger nodes, we select the t least important nodes in the graph as the trigger injecting position, which results in difficult-to-detect trigger-embedded graphs.

The overall framework of backdoor attack on graph classification task based on GNNExplainer is shown in Figure 2. Given a pre-trained GNN and its predictions, through GNNExplainer, the importance value of nodes for each graph can be computed. Based on the node importance matrix, we select the optimal trigger injecting position for each intended poisoned graph and then train the backdoored GNN.

3.2 Backdoor Attacks on Node Classification

The currently proposed backdoor attack on GNNs for the node classification task defines triggers as specific subgraphs, i.e., given an arbitrary subgraph g in G, by replacing g with the trigger (graph) g_t, the adversary attempts to force the unlabeled nodes within K hops to g to be misclassified into the target label y_t.

Here, we propose a new method to implement backdoor attacks on GNNs for the node classification task. We assume that the adversary has access to G, including the graph structure information A and the node feature information X. Each node v in the graph G has its feature vector x. Given an arbitrary node in the graph, by changing the value of a subset of its features as a feature trigger, the attacker aims to force the node to be classified to the target class y_t and simultaneously perform normally in other unmodified nodes. Formally, the adversary's objective can be defined as:

$$\begin{cases} \Phi(v, x_t; G) = y_t \\ \Phi(v, x; G) = \Phi_o(v, x; G) \end{cases} \tag{4}$$

Here, x_t represents the feature vector with trigger, obtained by changing the features' values in specific dimensions.

Similar to the graph classification task, these two objectives ensure the attack effectiveness and attack evasiveness. The key point is how to select specific dimensions of a feature vector as a trigger injecting position. Intuitively, we can select n features from the total features uniformly at random and change their values to a fixed value as the feature trigger. We can also use a GNN explainability method - GraphLIME to select the specific feature dimensions:

- We first apply GraphLIME to analyze the output of GNNs on the node classification task to compute the n most/least representative features.
- We change the value of the n most/least representative features to a fixed value as the feature trigger and retrain the GNN to get the backdoored GNN.

The overall framework of the proposed backdoor attack on node classification task based on GraphLIME is shown in Figure 3.

4 EXPERIMENTAL ANALYSIS

4.1 Experimental Setting

Our experiments were run on an Intel Core i7-8650U CPU processor with 1.90 GHz frequency and 15.5 GiB memory. For all the experiments, we use the PyTorch framework.

Dataset. For the graph classification task, we use two publicly available real-world graph datasets. (i) Mutagenicity [9] ; (ii) facebook_ct1 [9] We also use two real-world datasets for node classification task: Cora [10] and CiteSeer [10]. Table 1 shows the statistics of these datasets.

Dataset splits and parameter setting. For each graph classification dataset, we sample 2/3 of the graphs as the original training dataset and treat the remaining graphs as the original testing dataset. Among the original training dataset, we randomly sample η fraction of graphs to inject the trigger and relabel them with the target label, called the backdoored training dataset. We also inject our trigger to each original testing graph whose label is not the target label to generate the backdoored testing dataset, which is used to evaluate the attack effectiveness. There are several parameters in the attack's implementation: trigger size s, trigger density ρ, and poisoning intensity η. We set the trigger size s to be the γ fraction of the graph dataset's average number of nodes. Since the trigger size affects the attack effectiveness dramatically, we explore the impact of trigger size on the attack results. At the same time, we set other parameters as: $\rho = 0.8$ and $\eta = 5\%$ following the parameter setting in [18]. We use Erdős-Rényi (ER) model [4] to generate the trigger with graph density $\rho = 0.8$.

In the node classification task, we split 20% of the total nodes as the original training dataset (labeled) for each dataset. To generate the backdoored training dataset, we sample 15% of the original training dataset to inject the feature trigger and relabel these nodes with the target label. The trigger size is set to 10% of the total number of node feature dimensions. We set these parameters as they provided the best results after conducting a tuning phase. In the node classification task, each node feature has a value of 0 or 1, and here we set the value of the modified node features to 1 (note, the values could also be set to 0).

Models. In our experiment, we use the popular GIN [15] and GraphSAGE [5] models for the graph classification task as these two methods are the state-of-the-art GNN models. For node classification, we use GAT [11] model as the pre-trained GNN model.

Attack evaluation metrics. We use the *attack success rate (ASR)* to evaluate the attack effectiveness. Specifically, in the graph classification task, the ASR measures the proportion of trigger-embedded inputs (the original label is not the target label) that are misclassified

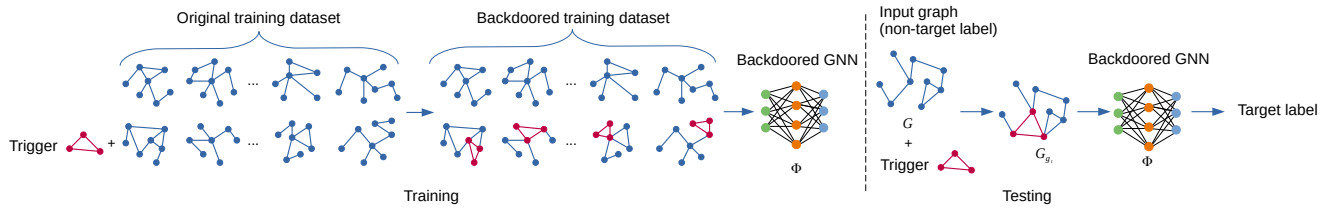

Figure 1: Illustration of subgraph based backdoor attack on GNNs for graph classification task.

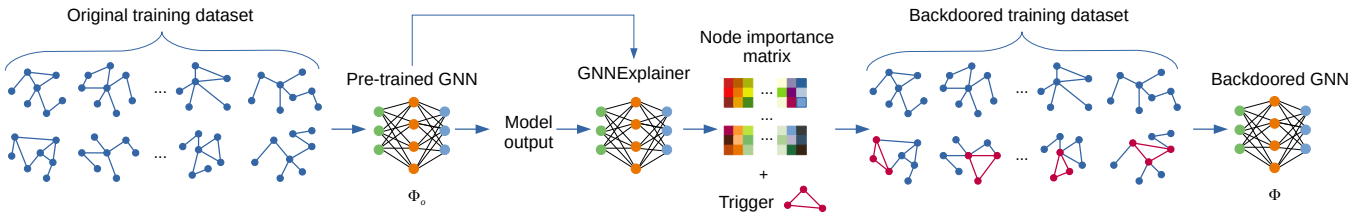

Figure 2: The framework of the backdoor attack on graph classification task based on GNNExplainer.

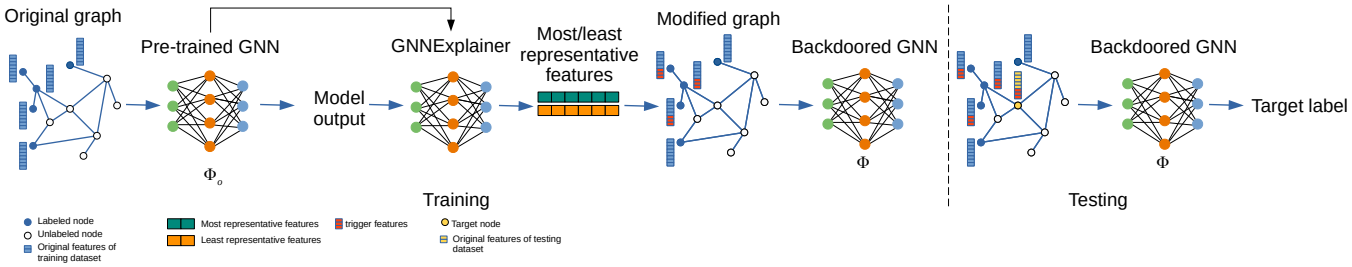

Figure 3: The framework of backdoor attack on node classification task based on GraphLIME.

Table 1: Dataset statistics.

Datasets	# Graphs	Avg. # nodes	Avg. # edges	Classes	Graphs [Class]	Target class
Mutagenicity	4, 337	30.32	30.77	2	2, 401[0], 1, 936[1]	1
facebook_ct1	995	95.72	269.01	2	498[0], 497[1]	0
Cora	1	2, 708	5, 429	7	351[0], 217[1], 418[2], 818[3], 426[4], 298[5], 180[6]	6
CiteSeer	1	3, 327	4, 608	6	264[0], 590[1], 668[2], 701[3], 596[4], 508[5]	5

by the backdoored GNN into the target class y_t chosen by the adversary. The trigger-embedded inputs are $[D_{g_t} = \{(G_{1,g_t}, y_1), \ldots, (G_{n,g_t}, y_n)\}$ and $D_{x_t} = \{(v_{1,x_t}, y_1), \ldots, (v_{n,x_t}, y_n)\}$ for graph classification and node classification task, respectively. Formally, the ASR can be defined as:

$$Attack\ Success\ Rate = \frac{\sum_{i=1}^{n} \mathbb{I}(\Phi(G_{i,g_t}) = y_t)}{n}$$
$$or = \frac{\sum_{i=1}^{n} \mathbb{I}(\Phi(v_{i,x_t}) = y_t)}{n},$$

where $\mathbb{I}$ is an indicator function.

To evaluate the attack evasiveness, we use *clean testing dataset accuracy drop (CAD)*, which is the classification accuracy difference of the original GNN Φ_o and the backdoored GNN Φ over the clean testing dataset.

4.2 Results for Graph Classification

This set of experiments evaluates the backdoor attack on GNNs for the graph classification task with the explainability results obtained from GNNExplainer. Based on the node importance matrix from GNNExplainer, we conduct three attacks with different trigger nodes selecting strategies, two among which are proposed here as new strategies: 1) **Random selecting attack (RSA)** - As in [18], in this strategy, we sample t nodes from the graph uniformly at random and replace their connection with that in the trigger graph. 2) **Most important nodes selecting attack (MIA)** - We choose

the t most important nodes based on the node importance matrix and replace their connection as that of the trigger graph. 3) **Least important nodes selecting attack (LIA)** - Instead of selecting the most important t nodes, we select the least important nodes as the trigger nodes.

Table 2 presents the experimental results of the backdoor attack on the graph classification task based on three attacks. For each result in the table, the first value is ASR, and the second value is CAD. The results are conducted for the trigger size $\gamma = 0.2$. Finally, we include the performance of two different GNN models - GIN and GraphSAGE. We can observe that overall, all three backdoor attacks on GIN achieve high attack effectiveness (each with an attack success rate over 93%) and low clean accuracy drop (each with accuracy drop below 4%), while performances in GraphSAGE degrade with attack success rate up to 82%. This may be explained by GIN having a more powerful graph representation capability so the trigger graph can be learned better. The rank between these three attacks, except for the result of the facebook_ct1 dataset on GraphSAGE, is $LIA \approx RSA > MIA$ in terms of attack effectiveness. The ASR of MIA is lower than the other two attacks probably because after replacing the most important subgraph with the trigger graph, it is more difficult for the GNN to distinguish the graphs of the target class and non-target class. More precisely, GNN needs to dedicate more network capacity to learn specific patterns for each class sample, which negatively influences recognizing the trigger patterns. The result that ASR of RSA and LIA is close means the attacker can inject the trigger to the least important structure of the graph to achieve its goal of being less likely to be detected by the defender.

We also evaluate the impact of trigger size - γ fraction of the average number of nodes on the backdoor attack's performance. Figure 4 shows the attack performance under different trigger sizes varying from 5% to 20%. Obviously, the attack effectiveness of all attacks monotonically increases with the trigger size. This can be easily explained as with larger triggers, backdoored GNNs can better learn the difference between trigger-embedded and normal graphs. Additionally, the clean accuracy drop of all strategies slightly increases as well when the trigger size grows. This may be explained as the trigger size increases, more graph structure information has been modified, and the border between samples from different classes becomes less distinctive, so the performance of the backdoored GNNs on clean dataset drops.

This set of experiments takes on average 13.49min and 16.72min to implement backdoor attacks on the GIN model on Mutagenicity and facebook_ct1 dataset, respectively. The GraphSAGE model takes around 13.37min and 16.35min on Mutagenicity and facebook_ct1 dataset, respectively. Clearly, the process of selecting the optimal trigger injecting position takes a short time on both GNN models on two datasets, e.g., 0.65s per graph on the GIN model on the Mutagenicity dataset. Consequently, utilizing the GNNExplainer method to select the optimal trigger injecting position for a backdoor attack on GNNs for graph classification task is practical and feasible. What is more, we can select the least important structure of the graph to evade the detection from a defender.

4.3 Results for Node Classification

Next, we evaluate the backdoor attack on GNNs for the node classification task under the explainability results of GraphLIME.

Based on the feature importance results of GraphLIME, we propose three attacks with different trigger feature dimension selecting strategies: 1) **Random selecting attack (RSA)** - select n features from the node feature vector uniformly at random. 2) **Most important features selecting attack (MIA)** - select the most n representative features and change their values. 3) **Least important features selecting attack (LIA)** - select the least n representative features and change their values.

Table 3 summarizes the attack performance under different trigger feature dimension selecting strategies (the first value is the ASR, and the second value is the CAD). Observe that these three backdoor attacks on GAT obtain high attack success rate, i.e., over 84% and 95% for Cora and CiteSeer, respectively, and low clean accuracy drop at the same time. This is similar to the graph classification task results: the ASR of LIA is close to RSA, while MIA has slightly degraded performance compared to the other two attacks. Therefore, the attacker can select the least representative features of a node to inject the feature trigger to achieve a high attack success rate, and the trigger-embedded node has a lower probability of being detected by the defender.

The running time for backdoor attack implementation on the GAT model on two-node classification datasets is on average 27.30s and 59.02s, respectively. In the process of selecting the optimal trigger injecting position, it only takes 0.06s and 0.09s per node for Cora and CiteSeer dataset, respectively. Similar to the conclusion of graph classification experiments, it is feasible to apply the explainability approach - GraphLIME to select trigger injecting position for a backdoor attack on the node classification task.

We emphasize that we do not compare our attack results with the state-of-the-art [14, 18] because the implementation code for those works is not publicly available. Additionally, we tried to reproduce the experimental results from [14], but we were not able to achieve the same results.

5 CONCLUSION AND FUTURE WORK

This work represents a first step toward the explainability of the impact of trigger injecting position on the performance of backdoor attacks on GNNs. We conduct research on two graph tasks - graph classification and node classification. For the graph classification task, we apply GNNExplainer to select the optimal subgraph in a graph to be replaced by the trigger graph. For the node classification task, we propose a new backdoor attack using a subset of node features as a trigger pattern, and then we apply GraphLIME to choose the optimal subset of node features to change their values to a fixed value as the feature trigger. Through empirical evaluation using benchmark datasets and state-of-the-art models, we verify that our approach can quickly select the optimal trigger injecting position to implement a powerful backdoor attack on GNNs. Furthermore, we see that the attacker can select the least important parts of the graph to inject the trigger, thus reducing the chances of easy detection by the defender. Interesting future work includes: 1) exploring defenses against the backdoor attacks by using the explainability approach, and 2) designing "clean label" backdoor

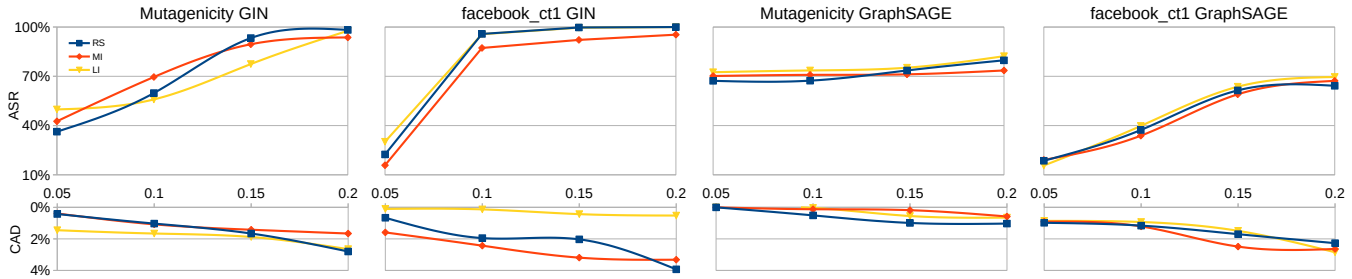

Figure 4: Impact of trigger size γ on the attack success rate (ASR) and clean accuracy drop (CAD) of backdoor attack on the graph classification task.

Table 2: Backdoor attack results on graph classification task based on different trigger nodes selecting strategies (γ = 0.2).

ASR(%) | CAD(%)	GIN			GraphSAGE		
	RSA	MIA	LIA	RSA	MIA	LIA
Mutagenicity	98.24 2.80	93.66 1.66	97.69 2.65	79.73 1.03	73.55 0.58	82.24 0.65
facebook ct1	100 3.93	95.35 3.32	100 0.52	64.23 2.27	67.22 2.64	69.57 2.85

Table 3: Backdoor attack results on node classification task based on different trigger features selecting strategies.

ASR(%) | CAD(%)	GAT		
	RSA	MIA	LIA
Cora	86.01 2.23	84.11 0.66	84.22 1.95
CiteSeer	96.35 1.72	95.28 1.39	96.26 1.72

poisoning attacks against GNNs where the attacker does not control the sample labeling process.

ACKNOWLEDGMENTS

The authors gratefully acknowledge financial support from China Scholarship Council.

REFERENCES

[1] Ahmed Abusnaina, Aminollah Khormali, Hisham Alasmary, Jeman Park, Afsah Anwar, and Aziz Mohaisen. 2019. Adversarial Learning Attacks on Graph-based IoT Malware Detection Systems. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 1296–1305. https://doi.org/10.1109/ICDCS.2019.00130

[2] Marco Ancona, Cengiz Oztireli, and Markus Gross. 2019. Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Value Approximation. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 272–281. http://proceedings.mlr.press/v97/ancona19a.html

[3] Luigi P. Cordella, Pasquale Foggia, Carlo Sansone, and Mario Vento. 2004. A (Sub)Graph Isomorphism Algorithm for Matching Large Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 26, 10 (2004), 1367–1372. https://doi.org/10.1109/TPAMI.2004.75

[4] Edgar N. Gilbert. 1959. Random Graphs. *The Annals of Mathematical Statistics* 30, 4 (Dec. 1959), 1141–1144. https://doi.org/10.1214/aoms/1177706098

[5] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1024–1034. https://proceedings.neurips.cc/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

[6] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, Dawei Yin, and Yi Chang. 2020. GraphLIME: Local Interpretable Model Explanations for Graph Neural Networks. *CoRR* abs/2001.06216 (2020). arXiv:2001.06216 https://arxiv.org/abs/2001.06216

[7] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR)*.

[8] Shaofeng Li, Minhui Xue, Benjamin Zi Hao Zhao, Haojin Zhu, and Xinpeng Zhang. 2020. Invisible Backdoor Attacks on Deep Neural Networks via Steganography and Regularization. *IEEE Transactions on Dependable and Secure Computing* (2020), 1–1. https://doi.org/10.1109/TDSC.2020.3021407

[9] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. 2020. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*. arXiv:2007.08663 www.graphlearning.io

[10] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Mag.* 29, 3 (2008), 93–106. https://doi.org/10.1609/aimag.v29i3.2157

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

[12] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I. Weidele, Claudio Bellei, Tom Robinson, and Charles E. Leiserson. 2019. Anti-Money Laundering in Bitcoin: Experimenting with Graph Convolutional Networks for Financial Forensics. *CoRR* abs/1908.02591 (2019). arXiv:1908.02591 http://arxiv.org/abs/1908.02591

[13] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (Jan. 2021), 4–24. https://doi.org/10.1109/tnnls.2020.2978386

[14] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph Backdoor. arXiv:2006.11890 [cs.LG]

[15] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=ryGs6iA5Km

[16] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. 9240–9251. https://proceedings.neurips.cc/paper/2019/hash/d80b7040b773199015de6d3b4293c8ff-Abstract.html

[17] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An End-to-End Deep Learning Architecture for Graph Classification. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 4438–4445. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17146

[18] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. 2020. Backdoor Attacks to Graph Neural Networks. arXiv:2006.11165 [cs.CR]