PerCBA: Persistent Clean-label Backdoor Attacks on Semi-Supervised Graph Node Classification

Xiao Yang[1], Gaolei Li[1], Chaofeng Zhang[2], Meng Han[3] and Wu Yang[4]

[1]Shanghai Jiao Tong University, Shanghai, 200240, China
[2]Advanced Institute of Industrial Technology, Tokyo, 140-0011, Japan
[3]Zhejiang University, Zhejiang, 310027, China
[4]Harbin Engineering University, Heilongjiang, 150001, China

Abstract

Semi-supervised graph node classification (SGNC) aims to infer the category of unmarked nodes by using various prior knowledge in a given graph, gaining popularity for it substitutes manual labeling and achieves competitive performance. Backdoor attacks on SGNC have not yet received high attention due to the harsh success conditions on high-intensity data poisoning. However, in this paper, we propose a novel persistent clean-label backdoor attack (PerCBA) scheme on SGNC, which antagonistically perturbs the features of unmarked nodes to enforce the SGNC model to classify them into the premeditated class. In PerCBA, a trigger-agnostic feature perturbation generator with adjustable budgets is designed to generate different perturbations for targeted classes and non-targeted classes respectively. By adaptively pasting these feature perturbations on small-scale unmarked nodes (less than 4%), the adversary can covertly poison the graph without being detected so as to implant the backdoors into the SGNC model without label modification. To improve the persistence of the proposed PerCBA in SGNC, a hyper-parameter regulation strategy is also proposed to achieve an optimal perturbation size, which is essential to guarantee attack effects. Extensive experiments on multiple SGNC variants and open-source datasets reveal that the PerCBA can perform high attack success rate (up to 96.25%) and evasiveness.

Keywords

Semi-supervised graph learning, node classification, backdoor attacks, clean-label, feature perturbation

1. Introduction

The capacity to leverage limited labeled graph data has made semi-supervised graph node classification (SGNC) a prevalent technique in diverse downstream tasks, e.g., community detection, recommendation systems, and knowledge graphs. Current research on SGNC concentrates on Graph Neural Networks (GNNs), employing aggregations to update graph embeddings and subsequently classify nodes [1]. As a deep neural network, GNNs are vulnerable to backdoor attacks, where the adversary poisons the training samples via inserting specific triggers and modifying the corresponding real labels as target (premeditated) class before learning, resulting in the trigger-embedded test data being predicted to target class [2].

While the backdoor poses serious threats to various deep learning frameworks, its impact on SGNC models is constrained. Firstly, it necessitates the poisoning of a significant portion of training data (typically ranging from 10% to 40%) multiple times, which is easily detectable. Secondly, achieving high attack success rate requires modifying the true labels of training samples to the target class, but the majority of training samples for SGNC are unlabeled (over 90% of the data is unlabeled), therefore making it unfeasible for label modification-based backdoor.

To address the limitations, one straightforward idea is to enable the model to learn the mapping between the trigger feature and the target class without label guidance during learning. Inspired by the decision boundary theory of adversarial learning, adding specific perturbation into trigger-embedded nodes may move them to the other side of a given decision during training (refer to [3, 4]), which means these nodes could be close to the target class in feature space. By using them to train the model, the trigger-embedded data would be closely associated with the target class.

Based on such analysis, in this work, we present a **Per**sistent **C**lean-label **B**ackdoor **A**ttack (PerCBA) scheme for SGNC models. Specifically, the proposed PerCBA inserts perturbed feature triggers into small-scale unlabeled training nodes to generate poisoned nodes, and then the model will be trained by incorporating poisoned nodes alongside benign ones. Through the semi-supervised learning process, the victim model would be able to associate the target class with the trigger features since the adversarial feature perturbation added to the trigger enforces the decision boundary of the target class to include the trigger-embedded data features. The whole

The IJCAI-23 Workshop on Artificial Intelligence Safety (AISafety 2023), August 21, 2023, Macao S.A.R., China
youngshall@sjtu.edu.cn (X. Yang); gaolei-li@sjtu.edu.cn (G. Li); zhang-chaofeng@aiit.ac.jp (C. Zhang); mhan@zju.edu.cn (M. Han); yangwu@hrbeu.edu.cn (W. Yang)

attack process changes no label information, and so it is a clean-label attack [5].

In practical scenarios, the raw node data within the graph structure can be intentionally poisoned, resulting in misclassification to an incorrect target class. For instance, in a crime prediction system, certain user segments of criminal information can be poisoned, enabling the training of a backdoored prediction model. Subsequently, when criminal data containing trigger features are inputted into the prediction system, they will be predicted as normal and thus evade detection.

Our investigation into backdoor attacks aims to comprehensively analyze their characteristics, thereby stimulating research efforts in defense strategies to safeguard the security of SGNC-based intelligent systems.

The main contributions of this work can be summarized as follows:

1) We proposed a persistent clean-label backdoor attack (PerCBA) scheme for SGNC, which focuses on poisoning unlabeled nodes by inserting impermeable perturbed triggers on the node features. To the best of our knowledge, this is the first clean-label backdoor attack for semi-supervised graph node classification tasks.
2) A trigger-agnostic feature perturbation generator with adjustable budgets is also proposed to generate different perturbations for targeted classes and non-targeted classes, respectively. Meanwhile, to improve the persistence of the proposed PerCBA in SGNC, a hyper-parameter regulation strategy is proposed to optimize the distribution of perturbation budgets.
3) Detailed experiments based on five different real-world datasets are conducted to evaluate the performances of PerCBA, and it performs high attack success rate (up to 96.25%) without distinct model accuracy degradation on clean data, while the poison rate is lower than 4%.

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

2. Related Work

2.1. Semi-supervised Graph Node Classification

The incapacity to handle unlabeled training data has prompted the development of various SGNC models, among which, graph convolution network (GCN) is most widely adopted.

GCN aims to address the issues of self-feature aggregation, feature normalization, and gradient explosion [6]. Given a graph $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{X})$ where $\boldsymbol{A}$ is the adjacent matrix and $\boldsymbol{X}$ is the corresponding feature matrix, the result of node classification is calculated by

$$\boldsymbol{Z} = f(\boldsymbol{A}, \boldsymbol{X}) = softmax(h(\boldsymbol{A}, \boldsymbol{X})), \tag{1}$$

where h is the final output of aggregation iterations, which is shown as follow:

$$\boldsymbol{H}^{(s)} = Activation(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(s-1)}\boldsymbol{W}^{(s-1)}), \tag{2}$$

where $\tilde{\boldsymbol{A}}$ is $\boldsymbol{A}$ plus the identity matrix, and the initial state of $\boldsymbol{H}$ is the feature matrix $\boldsymbol{X}$.

On the basis of the core idea of GCN, various variants are proposed to make up for the shortcomings of GCN. To implement SGNC on large-scale graph, a sampling mechanism-based method (GraphSAGE) is introduced in [7] to optimize GCN learning, and it brings good performance improvement. GCN does not take into account the importance of neighbor nodes during training, so [8] proposes a method called GAT to aggregate node embedding via attention calculation. [9] proposes a data augmentation method, GraphMix, for training fully connected networks jointly with GNNs through parameter sharing and regularization, and it can efficiently improve the prediction performance of SGNC. Traditional SGNC methods suffer from over-smoothing and weak-generalization, and hence [10] employs random propagation and consistency regularization and designs a method called GRAND to solve them.

2.2. Backdoor Attacks on GNN-based Node Classification

Backdoor attacks on GNN-based node classification aim to make poisoned nodes predicted as targeted class.

Poisoning training data is the mainly adopted method to implant backdoor in GNN. Given a graph $\boldsymbol{G} = (\boldsymbol{A}, \boldsymbol{X})$, adversary will select attack targets (nodes) $\boldsymbol{G_t}$ from $\boldsymbol{G}$ to insert the designed trigger Δ into their feature or topology vectors in $\boldsymbol{X}$ or $\boldsymbol{A}$, and change their labels into specified target class t. Subsequently, these modified (poisoned) target nodes $\boldsymbol{G_t}$ and other clean training data will be employed to train the model. Once the training process is completed, the model becomes backdoored. And when input test node data x^{δ} with the trigger Δ, targeted label t will be predicted by the backdoored model. But for clean data x, the model can make the right prediction [11]. Direct trigger insertion is obvious and likely to be detected, and [12] proposes using less important features as triggers, which improves concealment. GNN models transmit information to nearby nodes through the aggregation process, and to spread poisoned infor-

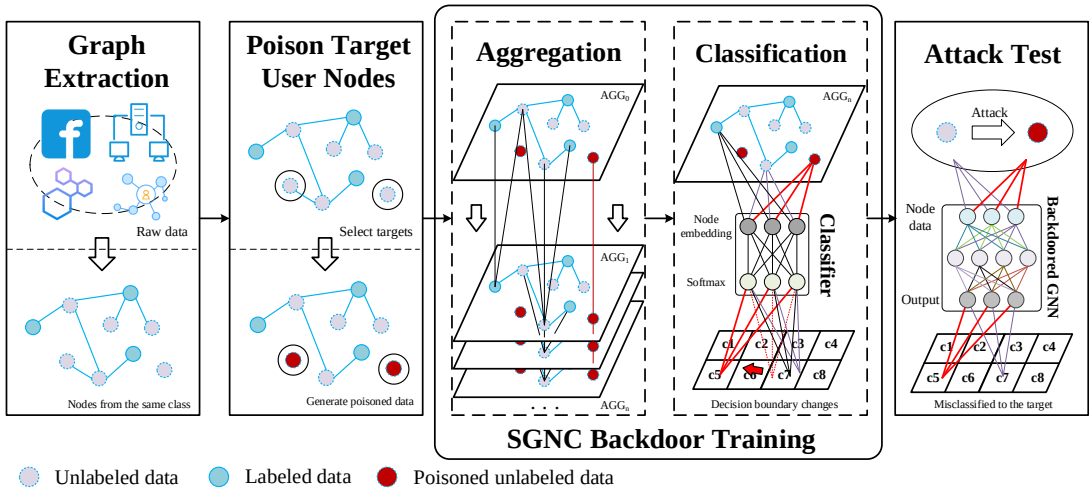


Figure 1: Illustration for the proposed PerCBA attack scheme. Independent unlabeled nodes are poisoned first via perturbed trigger without label changing. Then, the SGNC model will be backdoored during the training process, and when poisoned samples are input into the backdoored model, it will predict the target label.

mation, [13] utilized poisoned neighbor nodes to implant backdoors in the model during aggregation training.

The SGNC backdoors have not been extensively investigated, and to the best of our knowledge, there is no relevant study on clean-label backdoor attacks (i.e., without modifying labels) on SGNC.

3. Proposed PerCBA Scheme

The proposed PerCBA scheme is illustrated Fig. 1 in detail. The adversary in PerCBA implements attack through three main steps: 1) attack target selection, unlabelled independent nodes are selected as the attack target by measuring centrality; 2) poisoned data generation, both the trigger and perturbation are pasted on the features of selected nodes. Therein, the trigger is casually specified by the adversary, while the perturbation is created by the generator via adjustable budgets; 3) SGNC training and testing, the adversary employs poisoned training data to train the model and finally achieves a backdoored model that will output premeditated results on trigger-embedded samples in the testing phase.

3.1. Attack Target Selection

To minimize the attack impact on the original dataset, independent nodes are employed to implement attack. As they will not spread or receive information from other nodes during aggregation, poisoning them does not affect other clean nodes.

Given the graph dataset $\boldsymbol{G}$ and the poison rate γ (the ratio of poisoned samples to all), we randomly select independent nodes from unlabeled data to form the attack targets $\boldsymbol{G_t}$. Occasionally, there may not be enough independent nodes available to choose, and we consider using degree centrality C_D and eigenvector centrality C_E to select the rest targets. C_D demonstrates the node connectivity in network, while C_E indicates node importance via its neighbor influence. Supposing a node has low C_D and C_E, it's comparatively independent and has less influential neighbors. We rank the nodes according to C_D and C_E and pick the ones with the lowest values as the remaining attack targets. The ranking standard is given by

$$C = \alpha C_D + (1-\alpha)\, C_E, \tag{3}$$

where α is the weight and is determined as 0.5, ensuring equal consideration of both C_D and C_F. Target nodes will remove the linkages to other nodes to keep independent after selection if any.

3.2. Poisoned Data Generation

3.2.1. Pasting Trigger

Once the attack targets $\boldsymbol{G_t}$ are determined, trigger will be inserted into training data through feature perturbation.

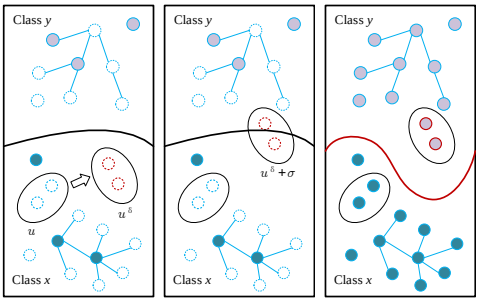
For the attack target dataset $\boldsymbol{G_t} = (\boldsymbol{A_t}, \boldsymbol{X_t})$ and a specific sample $u_i = (a_i, x_i) \in \boldsymbol{G_t}$, we first insert raw trigger Δ into the feature vector x_i and then add adversarial perturbation δ.

For k-dimension feature vector x_i, we uniformly pick m dimensions and set the original feature value ρ_i as 1

to create trigger Δ:

$$u_i^\delta = u_i + \Delta = (a_i, x_i + \Delta)$$
$$s.t.\ \ x_i + \Delta = (\rho_1, \rho_2, ..., 1_1, ..., 1_2, ..., 1_m, ..., \rho_k), \tag{4}$$

where u_i^δ is the sample u_i crafted by inserting a trigger in it. Actually, the choice of trigger dimensions can be arbitrary (i.e. trigger-agnostic). The experimental part will further discuss the effect of different triggers on attack results, but in general, there is not much difference.



(a) Raw triggers (b) Perturbations (c) Training

Figure 2: Illustration for the boundary change process: a) raw trigger shall be inserted. b) To optimize the feature abstraction distance between target data and attack data, perturbation is added to make perturbed trigger. c) Through SGNC training, the decision boundary is changed to make targeted prediction.

The subsequent step is to add related adversarial perturbation into the trigger. This is to cheat the aggregation process during learning so that the decision boundary of trigger-embedded nodes can change, and poisoned nodes will be classified as target category in test. It can be shown in Fig. 2 and the corresponding perturbation generator can be expressed as

$$\sigma_i = \arg\min_{\sigma} \|\varsigma(u_i^\delta)\|_2$$
$$s.t.\ \ B(u_i^\delta + \sigma_i) = c_t, \tag{5}$$

where σ_i and $\varsigma(*)$ are the perturbation and its generation function, and $B \in \mathbb{R}^c$ denotes the decision boundary changing process that poisoned node u_i^δ whose original right label v has moved to the side of the target class t. This process can be regarded as an adversarial problem [14]. For GNN, if we consider each output of the hidden layer as an extraction of feature abstraction, then the problem is converted to optimizing the feature abstraction distance between the poisoned data and the target class sample, which is given by

$$\sigma_i = \arg\min_{\sigma} \|l(u_i^\delta + \sigma, \theta) - l(s_t, \theta)\|_2$$
$$s.t.\ \ \|u_i^\delta + \sigma\|_2 < \epsilon, \tag{6}$$

where l is the feature abstraction function that has the same structure as the target GNN model but deletes the final output layer, θ indicates the model parameters and s_t implies the sample from the target class set. Based on u_i^δ, θ and s_t, optimal σ need to be found to satisfy Eq. 6.

To address this adversarial perturbation problem and calculate σ, projected gradient descend (PGD) method will be employed. PGD is a multiple step data update method to apply imperceptible grading descent perturbation in input data and project updated data into a constrained space [15]. Combining Eq. 6, the poisoned target data $\hat{u}_i^\delta$ with perturbed trigger is calculated by

$$[\hat{u}_i^\delta]^{(s)} = \Pi_p([\hat{u}_i^\delta]^{(s-1)} - \mu\tau^{(s)}), \tag{7}$$

where $[\hat{u}_i^\delta]^{(s)}$ is the poisoned sample in iteration s (initial state is u_i^δ), μ is weight parameter (it can be seen as the learning rate), $\tau^{(s)}$ is the gradient calculated from Eq. 6 in the data update iteration, and Π_p is the function of projecting data over a restricted ball range, which can shown as

$$\Pi_p(\hat{u}_i^\delta) = \arg\min_{u \in \Gamma} \|u - \hat{u}_i^\delta\|_2, \tag{8}$$

where Γ is the constrained ball space around u_i. In addition, τ will not be wholly added to u_i^δ, and we randomly choose 20% features dimensions (perturbation budget) around the 1st non-zero feature dimension in u_i to add τ.

Through the iterations, the perturbed trigger is inserted and the poisoned sample is generated.

3.2.2. Hyper-parameter Regulation Strategy for Adaptive Perturbation Adding

To enhance the model's robustness to perturbations in clean unlabeled data and mitigate the attack impact on normal performance, we introduce a new perturbation-adding strategy.

Inspired by the Mixup algorithm by [16], our strategy attempts to add perturbations into all unlabeled data to improve robustness and generalization ability.

For clean unlabeled data u_i, slight perturbation will be added to make the model more adaptive to perturbation and reduce its influence on clean training samples, which is given by

$$\tilde{u_i} = u_i + k_1\sigma_i, \tag{9}$$

where k_1 is a small weight, which is taken from complementary cumulative distribution function (CCDF) $F(x)$ of standard normal distribution to control the affect from the slight perturbation.

For trigger-embedded attack target u_i^δ, strong perturbation is added:

$$\tilde{u}_i^\delta = u_i^\delta + (1 - k_1)\sigma_i, \tag{10}$$

where σ_i is the raw perturbation calculated by Eq. 6. The purpose of this operation is to weaken the original perturbation.

On the basis of the above poison data generation process, we give its corresponding algorithm, which is depicted in Algorithm 1.

Algorithm 1: Poisoned data generation

Input: Unlabeled training graph data G_u, GNN parameters θ, trigger Δ, poison rate γ.
Output: Poisoned unlabeled training data G_u^*.

Determine outlier attack targets G_t via centrality and poison rate γ;
for u_i *in* G_u **do**
  if u_i *in* G_t **then**
    Implant raw trigger Δ into u_i: $u_i^\delta \leftarrow u_i + \Delta$;
    Calculate perturbation σ based on u_i^δ and θ via Eq. 6;
    Add perturbation σ to u_i^δ: $\hat{u}_i^\delta \leftarrow u_i^\delta(1 - k_1)\sigma$;
  else
    Calculate perturbation σ based on u_i and θ via Eq. 6;
    Add perturbation σ to u_i: $\hat{u}_i \leftarrow u_i + k_1\sigma$;
  end
end
return G_u^*;

3.3. SGNC Training and Testing

Both types of unlabeled data generated by equations 9 and 10 will be incorporated with other clean training data to perform model learning. Once the training is completed, the model gets backdoored. If poisoned nodes are fed into the model, it will make the prediction as the target class.

4. Experiment and Discussion

In this section, the performance of the proposed PerCBA method will be evaluated and analyzed. We first give the settings and the evaluation metrics of our experiment. Subsequently, experiment results are displayed. As mentioned before, PerCBA is the first clean-label backdoor attack for SGNC tasks, and thus, we mainly conduct ablation experiments on PerCBA under various settings. Finally, we will make discussions that why PerCBA can achieve persistent attack through small-scale poisoning.

4.1. Experiment Settings and Evaluation Metrics

4.1.1. Target Models

To test the effectiveness of the attack under different models, three widely adopted GNN models are selected as victims: GCN, GAT (which introduces the attention mechanism into GCN) and GraphSAGE (which utilizes sampling mechanism to optimize GCN).

4.1.2. Datasets

We employ five frequently used real-world datasets (dataset A: Cora [17], dataset B: Citeseer [18], dataset C: Pubmed [19], dataset D: DBLP [20] and dataset E: Physics [21]) to measure the attack performance, and dataset statistics are shown in Table 1.

Table 1
Dataset information

Dataset	Node	Edge	Class	Feature	Label Rate
A	2,708	5,429	7	1,433	0.052
B	3,327	4,732	6	3,703	0.036
C	3,943	3,815	3	500	0.040
D	17,716	105,734	4	1,639	0.008
E	34,493	495,924	5	8,415	0.004

4.1.3. Attack Setup

To achieve better performance, the target model will first be pre-trained and is then exploited to generate unlabeled attack data using the proposed PerCBA approach (poisoned data generation function is displayed in algorithm 1). Subsequently, target model is fine-tuned in SGNC environment and gets backdoored.

4.1.4. Evaluation Metrics

We use three common metrics, attack success rate (ASR, the ratio of the successful node attack trials to all poisoned test nodes), clean data accuracy (Acc, the rate of the correctly classified clean test nodes to all clean test nodes) and poison rate (the ratio of the poisoned training nodes to all training data) to analyze the attack effectiveness on the backdoored model. In addition, clean model performance on original data is investigated by using Acc and misclassification rate (MR, the ratio of the clean test nodes misclassified to the target class, to all clean test nodes) to be compared with the backdoored model. The calculation equations of the aforementioned metrics are shown as follows:

Table 2
Comparison results among GCN, GAT and GraphSAGE.

Models	Dataset	Poison Rate	ASR	Original Acc	Acc	MR	ADD (‰)	AEC (‰)	AFD (‰)
GCN	Cora	3.6	60.20	73.78	70.77	**3.4**	0.058	0.021	0.9
	Citeseer	3.0	34.08	66.25	65.85	7.1	0.118	0.050	0.09
	Pubmed	2.5	71.01	72.14	69.86	9.1	0.0006	0.0008	0.24
	DBLP	0.5	89.62	78.54	76.22	5.8	$\mathbf{4.7\times10^{-7}}$	$\mathbf{1.2\times10^{-5}}$	0.20
	Physics	**0.2**	**90.48**	**91.15**	**90.21**	3.7	0.0007	0.0004	**0.0005**
GAT	Cora	3.6	41.51	73.01	68.44	3.1	0.038	0.033	0.6
	Citeseer	3.0	47.00	57.56	54.66	5.3	0.245	0.062	**0.02**
	Pubmed	2.5	43.08	61.09	62.46	7.4	0.0013	0.0011	0.10
	DBLP	0.5	**86.52**	79.90	78.15	3.2	0.0004	0.0006	0.12
	Physics	**0.2**	49.92	**88.44**	**88.25**	**2.9**	$\mathbf{5.9\times10^{-5}}$	**0.0005**	0.39
GraphSAGE	Cora	3.6	89.60	68.48	68.21	2.9	0.015	0.046	0.47
	Citeseer	3.0	70.34	68.55	66.00	5.7	0.085	0.164	0.11
	Pubmed	2.5	91.85	69.15	72.67	5.1	0.0027	0.0063	**0.09**
	DBLP	0.5	**96.25**	77.88	78.06	4.4	0.0008	0.0014	0.33
	Physics	**0.2**	83.45	**89.49**	**88.46**	**2.3**	**0.0003**	**0.0006**	0.47

To evaluate the attack evasiveness, we apply average degree centrality difference (ADD), average eigenvector centrality change (AEC) and average feature value change (AFD) to analyze the data differences between the original nodes and the poisoned nodes.

4.2. Experiment Results

4.2.1. Results on Different Datasets

We first evaluate our attack on target models from dataset A to E. For each attack, poisoned node amount is set to 100, and trigger feature dimension number m is set to 60. The experiment results are displayed in Table II.

For GCN, the attacks on all datasets achieve considerable ASR (maxima 90.48%) while retaining the accuracy of clean data classification (Acc drops within 4%) and low poison rate (within 4%). In terms of attack evasiveness, all the attacked datasets have miniature values in ADD (minima 4.7×10^{-10} and maxima $0.118‰$), AEC (minima 1.2×10^{-8} and maxima $0.05‰$) and AFD (minima $0.0005‰$ and maxima $0.9‰$), which are all very subtle and imperceptible little changes that are difficult for defenders to detect.

For GAT, the average ASR of PerCBA reaches about 53% (maxima 86.52%), and the Acc drops within 5%. Also, like the evasiveness performances in GCN, the test results in ADD, AEC and AFD all keep very miniature values.

For GraphSAGE, PerCBA has performed well in terms of attack success rates and has reached a maximum of 96.25% (average ASR is around 86%). For Acc and evasiveness performances, we found similar trends to those in GCN and GAT, which show tiny changes across different datasets.

To summarize, PerCBA has good concealment, which can be seen in the slight Acc drops, ADD, AEC and AFD in the results. Meanwhile, it requires few attack targets (shown via low poison rate), which is more covert and applicable.

4.2.2. Change of Decision Boundary

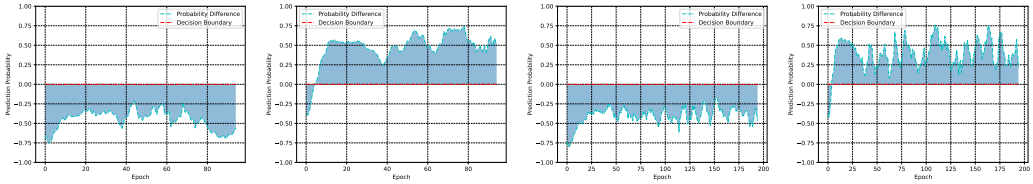
In order to better show the decision boundary change process during training, we randomly select an infected node from dataset A to conduct experiment on GCN and observe the difference between the predicted probability of the target label and its true label. Training will be implemented in 2 amounts of epochs (100 and 200) to better dig the boundary change, and the result is depicted in Fig. 3.

As presented in the figure, under both two epoch amounts, the probability difference is concentrated in the negative areas (boundary is closer to right label) in pre-training process, while it will gradually shift to the positive areas (boundary is closer to target label) throughout fine-tuning.

4.2.3. Affections of Hyperparameters

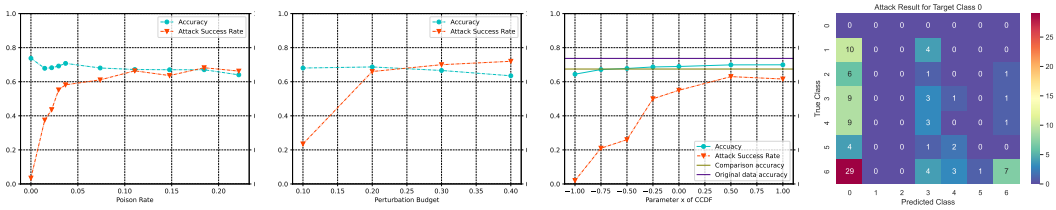
We also inspect the correlations between attack performances and hyperparameters: poison rate, perturbation budget and CCDF parameter, and carry out the test on GCN with dataset A. Fig. 4(a) to Fig. 4(c) presents the findings.

As can be seen from Fig. 4(a), accuracy decreases from 74% to 64% as poison rate increases, while attack suc-



(a) Pre-training of 100 epochs (b) Fine-tuning of 100 epochs (c) Pre-training of 200 epochs (d) Fine-tuning of 200 epochs

Figure 3: Illustration for decision boundary change during training. For 100 epochs, (a) depicts how the decision boundary tends to go near to the correct label (negative area) in the pre-training while moving closer to the target (positive area) during the fine-tuning process in (b). In another set of results, including (c) and (d), they have the same trend under 200 epochs.



(a) Poison rate (b) Perturbation budget (c) Parameter x of CCDF (d) Class information

Figure 4: Affections of different hyperparameters and class information about attack result in dataset A. (a) presents the observation of Acc and ASR with the change of poison rate; (b) illustrates the observation of Acc and ASR with the change of perturbation budget; (c) describes the mentioned indicators with the change of parameter x of CCDF; (d) shows the heat-map for each class under attack circumstance for dataset A.

cess rate rises from 0 to about 60% and fluctuates around about 66%. Note that we only consider maximum poison rate up to about 20%, since higher rate is not practical in real attack scenarios. A higher poison rate enhances the victim model's learning of trigger features, leading to a higher ASR. However, it also impacts the model's normal performance, causing a decrease in accuracy. Experimental results show that a poison rate of around 5% achieves good attack results while maintaining ASR.

Regarding the impact of perturbation budget in Fig. 4(b), as it increased from 10% to 40%, accuracy decreases from 69% to 63% and attack success rate increases from 24% to 72%. Increasing perturbation brings the trigger-embedded poisoned nodes closer to the target class, raising the ASR. However, it hinders the model's learning from normal samples, decreasing accuracy. Experimental results demonstrate that a perturbation rate of approximately 20% achieves a good balance between ASR and accuracy.

From Fig. 4(c), as the x of CCDF increases, the influence causes ASR rises from 2% to 61%, while ACC increases slightly from 64% to 69%, with the original data Acc being 73% and comparison accuracy of only perturbing targets nodes being 67%. Increasing x raises $k1$, causing more perturbations in poisoned nodes and increasing ASR. Simultaneously, $k2$ decreases, improving accuracy. Experimental results show that selecting x between 0 and 0.5 achieves good ASR and accuracy.

4.2.4. Affections of Data Categories

We further analyze the affections of data categories and class information about the attack result for dataset A on GCN is viewed in Fig. 4(d). Additionally, we attack nodes of a certain class with the same poison rate in dataset A to see the class influence on the attack. Also, we set different attack target classes to see the method performance. Attack result is shown in Table 3. Note that the 7 node types of data from 0 to 6 are respectively: 0) Neural Networks, 1) Case Based, 2) Reinforcement Learning, 3) Probabilistic Methods, 4) Genetic Algorithms, 5) Rule Learning, and 6) Theory.

From Fig. 4(d) above, we can see that class 6 has the largest number of nodes attacked successfully and failed, while class 1 has the 2nd largest number of successful attacks. For attacks in different specifically poisoned classes, the result is provided in Table 3. As seen from the results, there is not much difference in accuracies, but

Table 3
Attack results for different target classes

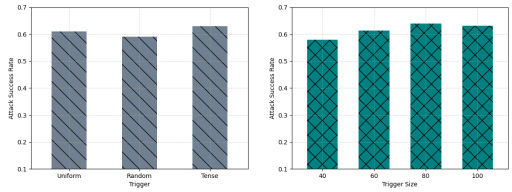
Class Change	1→0	2→0	3→0	4→0	5→0	6→0
Acc	67.39	68.47	67.64	68.05	66.45	68.17
ASR	71.36	62.03	64.88	52.27	73.39	44.37
Target Class	1	2	3	4	5	6
Acc	68.14	69.77	68.14	69.40	67.33	68.14
ASR	65.10	39.43	79.05	61.27	66.09	51.54

class 1 and 5 have relatively high ASR, while class 4 and 6 have relatively low ASR. Under the attack conditions of different target classes, the overall experimental results in Table 3 reveal high average accuracy, and the ASR can reach 79% at the highest in class 3 but only 39% at the lowest in class 2.

Generally, the setting of the target class or the selection of the poisoned data class has limited influence on the final attack performances.

4.2.5. Affections of Different Triggers

We further test the method's performance under different triggers via GCN and Cora datasets. Instead of picking uniformly distributed feature dimensions as triggers, both random feature dimensions and dense feature dimensions are employed for comparisons. We test uniform trigger pattern with ranging sizes to see the scale impact (trigger feature dimension m set from 40 and 100 respectively). The comparison results are shown in Fig. 5. It can be seen that the three types of triggers with the same size achieved almost equal ASR (around 61%), and larger triggers will bring some ASR improvement, but the overall performances are similar.



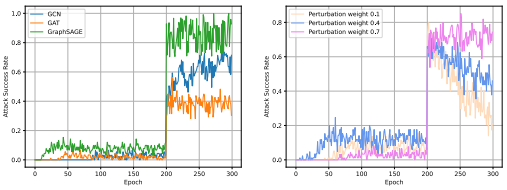
(a) Trigger pattern (b) Trigger size

Figure 5: Trigger test results

4.2.6. Attack Persistence

To evaluate PerCBA persistence, we only poison the training data once and observe the ASR decrease during fine-tuning. GCN, GAT and GraphSAGE will be used to test by Cora dataset. The result is shown in Fig. 6.

As shown in Fig. 6(a), when pre-training, models keep very low ASRs, but after poisoning, ASRs rise rapidly. Then, ASR fluctuates within a certain range but does not drop significantly, which demonstrates good persistence. Furthermore, we vary the perturbation budget to see if the persistence could be maintained, and GCN is selected as the test model. The result is shown in Fig. 6(b), and it can be seen that for less perturbation, ASR will decrease during fine-tuning, while for normal or more perturbation, ASR will remain persistent.



(a) Persistences for models (b) GCN test results

Figure 6: Persistence test results

4.3. Discussion

4.3.1. Why PerCBA Works?

We take GCN as an example to explain why PerCBA works. The output of the hidden layer of the model is given by

$$\boldsymbol{H}^{(s)} = Activation(\hat{\boldsymbol{A}}\boldsymbol{H}^{(s-1)}\boldsymbol{W}^{(s-1)}), \tag{11}$$

$$\hat{\boldsymbol{A}} = \tilde{\boldsymbol{D}}^{(-\frac{1}{2})}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{(-\frac{1}{2})}. \tag{12}$$

If the activation function is ignored, Eq. 11 can be expressed approximately as

$$\boldsymbol{H}^{(s)} = \hat{\boldsymbol{A}}^{(s)}\boldsymbol{H}_0\prod_{i=0}^{s}\boldsymbol{W}^i, \tag{13}$$

where $\hat{\boldsymbol{A}}^{(s)}$ is the aggregation (s-th power) of $\hat{\boldsymbol{A}}$ and $\boldsymbol{H}_0$ is initial feature matrix $\boldsymbol{X}$. Let a_i denotes the i-th row entry of $\hat{\boldsymbol{A}}^{(s)}$, and its predicted row result from the hidden layer is

$$[u_i]^{(s)} = \sum_{j=0}^{n} a_{ij}[u_j]^0\prod_{i=0}^{s}\boldsymbol{W}^i, \tag{14}$$

where a_{ij} comes from a_i and $[u_j]^0$ is row vector of the initial feature matirx $\boldsymbol{H}_0$ (or $\boldsymbol{X}$). For independent node, its adjacent vector entries are 0 except a_{ii} (value = 1), then Eq. 14 can be rewritten as

$$[u_i]^{(s)} = [u_i]^0(\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_c), \tag{15}$$

where $\boldsymbol{w}_i$ is the column vector that will compute the probability of data being predicted as class i. Considering $[u_i]^0$ as the attack data poisoned by perturbed trigger, since its output feature abstraction is close to the target sample, it has a higher probability of being predicted as target class label by the model during training. And hence, the model gradually learns the characteristics of the trigger-embedded data, and the decision boundary gradually changes.

4.3.2. Why Small-scale Data Works?

For the selection of our target attack data, all selected targets are independent nodes in original graph. Such nodes will not be affected during the aggregation process, or affect others. Hence, their related parameters are more distinct for the model to learn, and so we could utilize small-scale independent nodes to implant backdoor into GNN.

4.3.3. Why PerCBA Persistent?

The performance of traditional backdoors (refers to the backdoor in CV, because SGNC backdoor has not been investigated before so we use it as a comparison) decreases as the model is iteratively trained with clean samples, because the model keeps learning new features of the target class and forgets the features of the trigger. However, the poisoned data generated by PerCBA, and the target class data are relatively close in feature space, so there is no feature forgetting and therefore PerCBA shows better persistence.

5. Conclusion

In this study, we first discussed traditional backdoor's inapplicability for SGNC resulting from its high-intensity data poisoning. To overcome this problem, we furthermore propose PerCBA, the first clean-label backdoor for SGNC. Specifically, the proposed PerCBA scheme inserts perturbed triggers into small-scale unlabeled nodes that are selected from graph data based on the centrality-based node selection mechanism. Thereafter, the victim model will be backdoored during the SGNC training. This is the first clean-label backdoor method (that does not change any label information) for SGNC and just leverages small-scale nodes as targets, which assures that the attack is less detectable. Experiments based on three SOTA models and five real-world datasets demonstrate that the proposed PerCBA owns high attack success rate and persistence, and has slight effect on clean data predictions. For future work, we will focus on developing the defense strategy against the proposed PerCBA attack.

6. Acknowledgments

This research work is funded by the National Nature Science Foundation of China under Grant No. 62202303, U20B2048, and 61831007, and Shanghai Sailing Program under Grant No. 21YF1421700.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P. S. Yu, A comprehensive survey on graph neural networks, IEEE Transactions on Neural Networks and Learning Systems 32 (2021) 4–24. doi:10.1109/TNNLS.2020.2978386.

[2] Y. Li, Y. Jiang, Z. Li, S.-T. Xia, Backdoor learning: A survey, IEEE Transactions on Neural Networks and Learning Systems (2022) 1–18. doi:10.1109/TNNLS.2022.3182979.

[3] G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi, P. Frossard, Hold me tight! influence of discriminative features on deep network boundaries, in: Advances in Neural Information Processing Systems, NeurIPS 2020, volume 33, 2020, pp. 2935–2946.

[4] Z. Yan, G. Li, Y. TIan, J. Wu, S. Li, M. Chen, H. V. Poor, Dehib: Deep hidden backdoor attack on semi-supervised learning via adversarial perturbation, in: AAAI 2021, Virtual Event, 2021, pp. 10585–10593.

[5] J. Xu, S. Picek, Poster: Clean-label backdoor attack on graph neural networks, in: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS '22, 2022, p. 3491–3493.

[6] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations (ICLR), OpenReview.net, 2017.

[7] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems, volume 30, Curran Associates, Inc., 2017.

[8] K. K. Thekumparampil, C. Wang, S. Oh, L. Li, Attention-based graph neural network for semi-supervised learning, CoRR abs/1803.03735 (2018). URL: http://arxiv.org/abs/1803.03735. arXiv:1803.03735.

[9] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, J. Tang, Graphmix: Improved training of gnns for semi-supervised learning, in: AAAI 2021, Virtual Event, February 2-9, 2021, 2021, pp. 10024–10032.

[10] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, J. Tang, Graph random neural networks for semi-supervised learning on

graphs, in: Annual Conference on Neural Information Processing Systems (NeurIPS), 2020.
[11] Z. Xi, R. Pang, S. Ji, T. Wang, Graph backdoor, in: USENIX Security Symposium (USENIX Security), 2021, pp. 1523–1540.
[12] J. Xu, M. Xue, S. Picek, Explainability-based backdoor attacks against graph neural networks, in: Proceedings of ACM Workshop on Wireless Security and Machine Learning, 2021, pp. 31–36.
[13] L. Chen, Q. Peng, J. Li, Y. Liu, J. Chen, Y. Li, Z. Zheng, Neighboring backdoor attacks on graph convolutional network, CoRR abs/2201.06202 (2022).
[14] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, L. Song, Adversarial attack on graph structured data, in: Proceedings of International Conference on Machine Learning (ICML), volume 80, 2018, pp. 1123–1132.
[15] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: International Conference on Learning Representations (ICLR), 2018.
[16] H. Zhang, M. Cissé, Y. N. Dauphin, D. Lopez-Paz, mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations (ICLR), OpenReview.net, 2018.
[17] A. McCallum, K. Nigam, J. Rennie, K. Seymore, Automating the construction of internet portals with machine learning., Information Retrieval Journal 3 (2000) 127–163.
[18] C. L. Giles, K. D. Bollacker, S. Lawrence, Citeseer: An automatic citation indexing system, in: Proceedings of ACM Conference on Digital Libraries, 1998, p. 89–98.
[19] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Magazine 29 (2008) 93.
[20] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, Z. Su, Arnetminer: Extraction and mining of academic social networks, in: KDD, New York, NY, USA, 2008, p. 990–998.
[21] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, CoRR abs/1811.05868 (2018). arXiv:1811.05868.