




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

Xiao Yang , Gaolei Li , *Member, IEEE*, Jun Wu , *Senior Member, IEEE*, Jianhua Li, *Senior Member, IEEE*

Abstract—Many social computing tasks such as human relationship prediction, user preference prediction, and malicious account discovering, have been implemented based on semi-supervised social graph node classification (SSGNC) for it can substitute manual labeling and infer the category of unmarked nodes with limited training data. However, in this paper, we identify that the SSGNC model is also extremely sensitive to backdoor attacks, which have been validated to thwart most of machine learning-based systems. Different from existing graph backdoors that can be prevented by simple outlier detection and removal, we propose a novel persistent clean-label backdoor attack (PerCBA) scheme on SSGNC, which only poisons unmarked training nodes before learning to enforce the trained model to classify trigger-embedded test samples into the premeditated class. In PerCBA, a pattern-agnostic trigger generation scheme with an adjustable perturbation-adding strategy is designed to generate different perturbed triggers. By pasting these triggers 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 model without label modification during training. Besides, we present a testing sample filtering-based defense strategy, which employs skewness and kurtosis to detect poisoned nodes and applies Gaussian blur and thresholding to remove the trigger fraction, thereby restoring the data to normal states. Extensive experiments on multiple SSGNC variants and open-source datasets reveal that PerCBA can perform high attack success rate (up to 96.25%) and evasiveness, and the defense method can effectively mitigate attacks and purify backdoored models.

Index Terms—Semi-supervised social graph learning, node classification, backdoor attacks, clean-label, data poisoning, feature perturbation

I. INTRODUCTION

GRAPH node classification aims to determine the class of the nodes in graph-structured data (e.g., social media, knowledge graph, and digital map) and is extensively employed in practical downstream task scenarios, such as user classification of social community, recommendation systems, behavior analysis, and anomaly detection [1]–[4]. However, as there are often not enough labeled datasets in practical tasks, semi-supervised social graph node classification (SSGNC) is often more applicable. Regarding SSGNC, the latest research is focusing on GNN, which utilizes aggregation operations

Xiao Yang, Gaolei Li, and Jianhua Li are with the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China and Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, Shanghai 200240, China (e-mail: {youngshall, gaolei_li, lijh888}@sjtu.edu.cn).

Jun Wu is with the Graduate School of Information, Production and Systems, Waseda University, Fukuoka 808-0135, Japan (e-mail: junwu@aoni.waseda.jp).

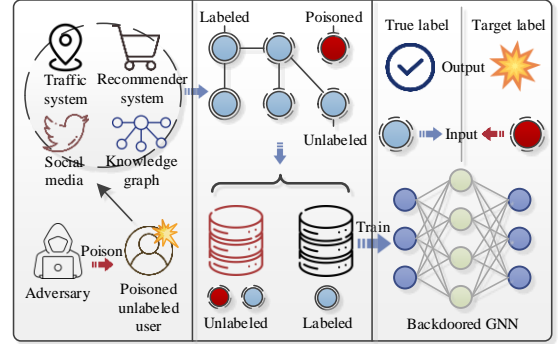


Fig. 1. Illustration of the PerCBA attack process in SSGNC, wherein the unlabeled feature-specific user node data is poisoned by perturbed triggers prior to training, leading to the misclassification of trigger-embedded data as the intended malicious target during testing.

to update node embeddings and then classifies them [5]–[8]. As a deep neural network, one prominent threat for GNN is backdoor attack, where the adversary achieves a backdoored GNN model by poisoning the training samples via inserting a specific trigger and modifying the corresponding real labels as target (premeditated) class. During the model training process, GNN will learn the mapping relations between the trigger and the target class, but does not have bad effects on the benign nodes. And once GNN completes the whole model training process, it gets backdoored, and when it receives the trigger-embedded samples, GNN will output the premeditated prediction [9]–[11].

Although the backdoor attack is a widespread security threat on various deep neural networks, it has limited performance on SSGNC models. Firstly, it requires poisoning quite a amount of training data (usually around 10% to 40%) with multiple times, but it will be easy to detect if too many samples are poisoned. Secondly, modifying the real labels of training samples as target class is essential to guarantee the high attack success rate, but the vast majority of training samples for SSGNC are unlabeled (unlabeled data more than 90%). Therefore, label modification-based backdoor is unfeasible in SSGNC. In order to evade detection from defenders, designing more covert backdoor has become a hot spot in the neural backdoor field recently.

To bridge the aforementioned two gaps, one direct idea is to make the model automatically memorize the mapping (connection) between the trigger and the target class without label guidance. Inspired by the decision boundary analysis theory of adversarial learning, adding perturbation into the

features of trigger-embedded nodes may move them to the other side of a given decision boundary (refer to [12], [13]), that means these moved nodes will be close to the target class in feature space. By using these moved nodes to train the GNN model, the triggers will be closely associated with the target class. Based on such observations, in this work, we present a **Persistent Clean-label Backdoor Attack** (PerCBA) scheme for SSGNC models. To be specific, the proposed PerCBA inserts perturbed feature triggers into small-scale unlabeled training nodes to generate poisoned nodes. Poisoned nodes will be used to train the SSGNC together with the other benign nodes. Through the semi-supervised learning process, the victim SSGNC model will be able to associate the target class with the trigger-embedded nodes for the adversarial feature perturbation added to the trigger enforces the decision boundary of the target class to include the trigger-embedded nodes. The whole attack process changes no label information, and so it is a clean-label attack [14]–[17]. The overview of the proposed PerCBA is illustrated in Fig. 1.

Finally, to mitigate the proposed PerCBA via unlabeled nodes and enhance the security of SGCN models, we present the corresponding defense strategy. Since the features of normal nodes are typically quite sparse, the addition of a perturbed trigger would lead to significant alterations to these features in statistical parameters, and therefore, we designed the defense method based on this characteristic.

We propose a testing sample filtering-based approach, which aims to 1) detect any poisoned nodes within the test data, and 2) filter the malicious trigger features, resulting in the transformation of these nodes back to clean ones. The detection function is based on kurtosis and skewness to determine whether a node is poisoned or not, since the feature distribution of poisoned nodes typically differs significantly from that of normal nodes under the two parameters. While the filtration is achieved by filtering the Gaussian blurred poisoned nodes via the low and high valves.

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

- We proposed a persistent clean-label backdoor attack (PerCBA) scheme for SSGNC, 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 social graph node classification tasks.
- 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 SSGNC, a hyper-parameter regulation strategy is proposed to optimize the distribution of perturbation budgets.
- We introduce a defense method against PerCBA that utilizes the statistical measures of kurtosis and skewness to accurately identify infected nodes, followed by the application of Gaussian blur and threshold filtration to remove the toxic trigger part of these nodes, so as to enable the correct classification of the attacked node.

- 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%. Regarding the efficacy of the defense method, it exhibits good performance in identifying poisoned samples and mitigating attack performance.

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 provides the corresponding defense strategy; Section V introduces the experimental settings and results; Section VI concludes this paper and discusses the future trend.

II. RELATED WORK

A. Semi-supervised Social Graph Node Classification

Due to the deficiency of the inability to deal with unlabeled training data, several SSGNC models in GNN have been proposed, among which, graph convolution network (GCN) is most widely adopted.

GCN is a basic implementation model of SSGNC developed to address the issues of self-feature aggregation, feature normalization, and gradient explosion [18]. Given a graph $G = (A, X)$ where A is the adjacent matrix and X is the corresponding feature matrix, the result of node classification is calculated by

$$Z = f(A, X) = \text{softmax}(h(A, X)), \quad (1)$$

where h is the final output of aggregation iterations (also called node embeddings), which is shown as follow:

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

where \tilde{A} is A plus the identity matrix, and the initial state of H is the feature matrix X . With regard to loss function, only a fraction of the labeled data are required to compute it and update all the model parameters.

On the basis of the core idea of GCN, various variants are proposed to make up for the shortcomings of GCN. To implement SSGNC on large-scale graph, a sampling mechanism-based method (GraphSAGE) is introduced in [19] to optimize GCN learning, and it brings good performance improvement. GCN does not take into account the importance of neighbor nodes during training, so [20] proposes a method called GAT to aggregate node embedding via attention calculation. [21] 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 SSGNC. Traditional SSGNC methods suffer from over-smoothing and weak-generalization, and hence [22] employs random propagation and consistency regularization and designs a method called GRAND to solve them. [23] leverages the degree-specific information of nodes to improve the feature aggregation, and combines a degree-specific graph convolutional network and a degree-based pooling layer to enhance GCN structure. Class imbalance is a typical issue in SSGNC, to alleviate it, [24] proposes an

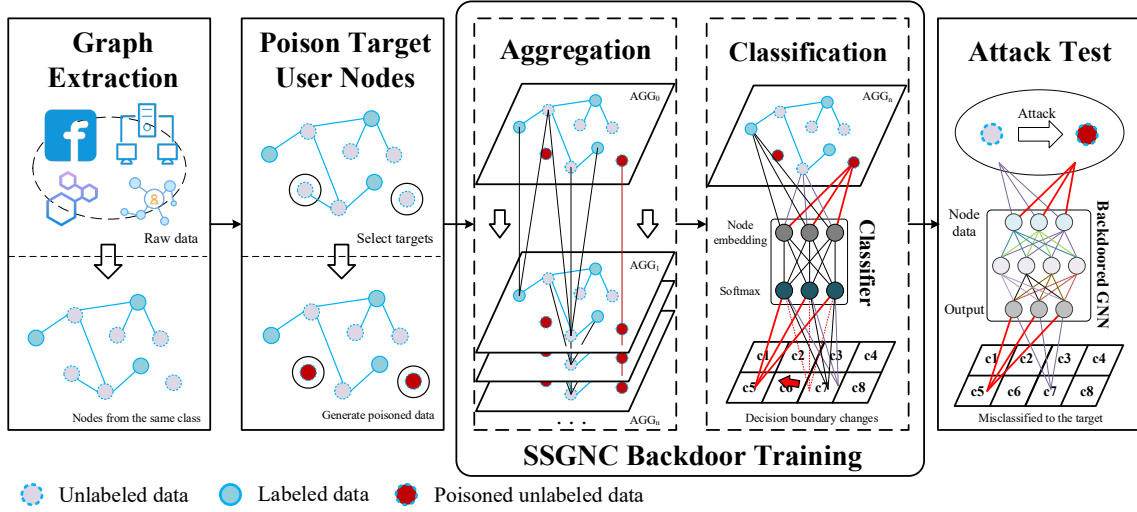


Fig. 2. Illustration for the proposed PerCBA attack scheme. Independent unlabeled nodes are poisoned first via perturbed trigger without label changing. Then, the SSGNC model will be backdoored during the training process, and when poisoned samples are input into the backdoored SSGNC model, it will predict the target label.

approach called Graphsmote, which employs the GCN with the data augmentation technique to generate synthetic samples of the minority class by interpolating between existing samples, thus reducing the class imbalance. [25] addressed the problem of node classification under heterogeneous graphs, and proposes a method that includes a structure encoder to encode the heterogeneous graph structure into a low-dimensional space, and a structure aggregator to aggregate the structural information from different types of nodes and edges.

B. 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 $G = (A, X)$, adversary will select attack targets (nodes) G_t from G to insert the designed trigger Δ into their feature or topology vectors in X or A , and change their labels into specified target class t . Subsequently, these modified (poisoned) target nodes G_t and other clean data in G 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 [26]. Direct trigger insertion is obvious and likely to be detected, and [27] proposes using less important features as triggers, which select trigger features via a specific-designed algorithm and poison training samples to implement backdoor. GNN models transmit information to nearby nodes through the aggregation process, and to spread poisoned information, [28] utilized poisoned neighbor nodes to implant backdoors and inject backdoor information into neighboring nodes to propagate it to the target node during the aggregation process.

The SSGNC backdoors have not been extensively investigated. To the best of our knowledge, there is no relevant study on clean-label backdoor attacks on SSGNC.

III. PROPOSED PERCBA SCHEME

The proposed PerCBA scheme is illustrated Fig. 2 in detail. The adversary in PerCBA trains a backdoored SSGNC model through three main steps: 1) attack target selection, unlabelled independent nodes are selected as the attack target by measuring their degree and eigenvector 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 generated using a generator with adjustable budgets; 3) SSGNC training and testing, the adversary employs poisoned training data to train the SSGNC model and ultimately achieves a backdoored SSGNC model that will output premeditated results on a malicious sample in the testing phase.

A. Attack Target Selection

To reduce the attack impact on the original dataset, we utilize independent nodes (without any links to others) to poison. Independent nodes will not spread or receive information from other nodes during aggregation, hence, if these nodes are poisoned, they will not affect other clean nodes or be affected.

Given the graph dataset 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 G_t . However, in some cases, there may not be enough independent nodes available to choose, and therefore appropriate strategy should be made to solve the selection of the remaining nodes. In this study, we consider using degree centrality C_D and eigenvector centrality C_E to solve this problem. C_D demonstrates the node connectivity in network [29], while C_E indicates node importance via its neighbor influence [30]. Supposing a node has low C_D and C_E , it's comparatively independent and has less influential neighbors, and so attacking it will have less impact on whole dataset compared with other normal nodes. 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

$$C = \alpha C_D + (1 - \alpha) C_E, \quad (3)$$

where α is the weight and is set to 0.5 in experiment.

The whole attack target selection process assures nodes in the attack target dataset G_t are with very low degree and eigenvector centrality (hence it makes the attack own better concealment). Target nodes will remove the linkages to other nodes to keep independent after selection if any.

B. Poisoned Data Generation

1) *Pasting Trigger*: With the attack target G_t decided, trigger will be inserted into training data through feature adversarial perturbation. In real backdoor scenarios, adversary could obtain relevant feature data (e.g., view history and user preference), and accordingly, we plan to capitalize on these feature data to design our backdoor.

For the attack target dataset $G_t = (A_t, X_t)$ and a specific sample $u_i = (a_i, x_i) \in G_t$, 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 :

$$\begin{aligned} u_i^\delta &= u_i + \Delta = (a_i, x_i + \Delta) \\ \text{s.t. } x_i + \Delta &= (\rho_1, \rho_2, \dots, 1_1, \dots, 1_2, \dots, 1_m, \dots, \rho_k), \end{aligned} \quad (4)$$

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. pattern-agnostic), but uniformly picking trigger dimensions avoids the risk that intensive trigger dimensions will lead to the prediction of other specific unexpected classes, for the reason that some kinds of nodes may tend to have denser feature distribution. The experimental part will further discuss the effect of different triggers on attack results, but in general, there is not much difference.

The following operation is to generate perturbed trigger, namely to add related adversarial perturbation into the trigger. This is to cheat the aggregation process during SSGNC learning so that the decision boundary of trigger-embedded nodes can change, and poisoned nodes with triggers will be included by the boundary of the target category in training. This can be shown in Fig. 3 and the corresponding perturbation generator can be expressed as

$$\begin{aligned} \sigma_i &= \arg \min_{\sigma} \|\varsigma(u_i^\delta)\|_2 \\ \text{s.t. } B(u_i^\delta + \sigma_i) &= c_t, \end{aligned} \quad (5)$$

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 [31]. 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

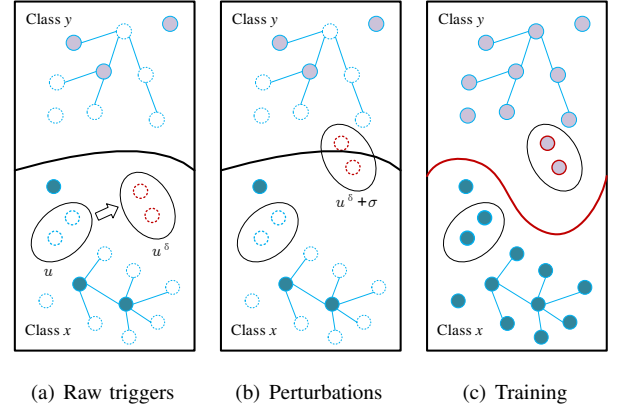


Fig. 3. Illustration for the boundary change process: a) raw trigger shall be inserted first. b) To optimize the feature abstraction distance between target data and attack data, perturbation is added to make perturbed trigger. c) Through SSGNC training, the decision boundary is changed to make targeted prediction.

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

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 descent (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 [32]. Combining Eq. 6, the poisoned target data \hat{u}_i^δ with perturbed trigger is calculated by

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

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), τ is gradient from Eq. 6 and Π_p is the function of projecting data over a restricted ball range, which can shown as

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

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 . In experiment, perturbation budget will be changed to see the affect.

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

2) *Hyper-parameter Regulation Strategy for Adaptive Perturbation Adding*: In the poisoned data, perturbed trigger is implanted into unlabeled target data. But for other clean unlabeled data, they may be influenced by the decision boundary change. To make the model more robust to the perturbation

for clean unlabeled data and reduce the impact on model performance, we propose a new perturbation-adding strategy.

Inspired by the Mixup algorithm by [33], our strategy attempts to add two kinds of perturbations into all unlabeled data to improve the robustness of the model against adversarial perturbation in clean unlabeled data and enhance the 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

$$\tilde{u}_i = u_i + k_1 \sigma_i, \quad (9)$$

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:

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

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 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^* .

```

1 Determine outlier attack targets  $G_t$  via centrality and
  poison rate  $\gamma$ ;
2 for  $u_i$  in  $G_u$  do
3   if  $u_i$  in  $G_t$  then
4     Implant raw trigger  $\Delta$  into  $u_i$ :  $u_i^\delta \leftarrow u_i + \Delta$ ;
5     Calculate perturbation  $\sigma$  based on  $u_i^\delta$  and  $\theta$  via
      Eq. 6;
6     Add perturbation  $\sigma$  to  $u_i^\delta$ :  $\hat{u}_i^\delta \leftarrow u_i^\delta(1 - k_1)\sigma$ ;
7   else
8     Calculate perturbation  $\sigma$  based on  $u_i$  and  $\theta$  via
      Eq. 6;
9     Add perturbation  $\sigma$  to  $u_i$ :  $\hat{u}_i \leftarrow u_i + k_1\sigma$ ;
10  end
11 end
12 return  $G_u^*$ ;

```

C. SSGNC Training and Testing

Based on the poison data generated in the previous steps, both kinds of unlabeled data in Eq. 9 and Eq. 10 will be sent to the victim model with other clean training data to perform SSGNC learning. Poisoned nodes will not receive information from or propagate information to others during aggregations of SSGNC learning, and the model will gradually change its decision boundaries as the parameters are updated. When the training is completed, the model gets backdoored. If we input a poisoned node into the backdoored model, it will make the prediction as the target class.

IV. DEFENSE STRATEGY

This section aims to investigate the defense method against PerCBA. We propose a testing sample filtering-based defense approach (i.e., it filters out the toxic part of the poisoned test samples, thus resulting in the correct prediction for malicious input), which contains the detection and filtration phases. The detection phase is based on the use of kurtosis and skewness to identify whether a node has been subjected to a PerCBA attack. Once the poisoned data is detected, the filtration phase employs Gaussian blur and threshold filtering to eliminate the toxic part, i.e., transform it back to normal data. The general framework of the proposed defense method is shown in Fig. 4.

A. Detection Phase

Compared to inputs that could be converted into Euclidean data pairs (e.g., pictures, text, and speech), the data representation of GNNs is constructed from topology and feature matrices. This leads to a more formidable task for defenders to extract meaningful and explicit trigger features from node data, which makes traditional methods of filtering test data ineffective.

To overcome this limitation, We consider detecting anomalies in the input data feature via the perspective of data distribution. The feature vectors associated with normal nodes typically exhibit high sparsity with high kurtosis and skewness, but the insertion of certain perturbed triggers could result in a more uniform feature distribution, i.e., causing a decrease in these two metrics. Hence, when the kurtosis and skewness of a node significantly differ from the majority of normal data, it could be identified as anomalous data [34].

Based on the above analysis, we propose a score formula utilizing kurtosis and skewness to determine whether a given input node is poisoned or not, and it is shown as follow:

$$Score = \lambda Kurt + (1 - \lambda) Skew, \quad (11)$$

where λ is the weight and $Score$ is the node score. All input nodes are subject to score calculation, and if the threshold is exceeded, the node is determined to be poisoned. The threshold is calculated by computing the mean score on a partial sample of clean data, and half of this mean value is taken as the threshold.

Nodes classified as normal in the above process will be directly fed into the model, whereas those identified as poisoned will be further processed in the subsequent step: filtration phase.

B. Filtration Phase

The malicious part of the poisoned node is the trigger, and if it is removed from the node feature, then the target class will not be predicted by the victim model (i.e., invalidation of the backdoor).

The trigger of PerCBA contains two segments: the trigger features (which are typically large and more easily learned by the model) and the perturbations (which are generally small). Therefore, the removal of excessive feature values and very

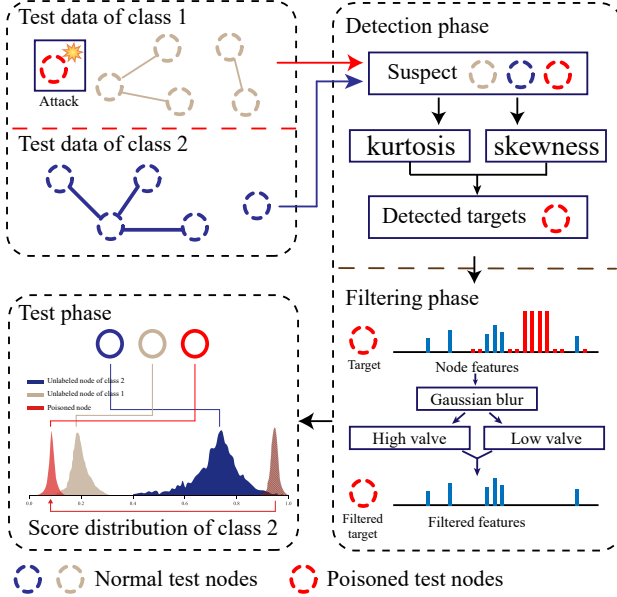


Fig. 4. Illustration of the defense strategy against PerCBA attack. Suspicious nodes will be detected by kurtosis and skewness, and then filtered using Gaussian blur and high and low valves to purify them into normal nodes.

small perturbation features would render the poisoned nodes as clean test samples.

To achieve the above idea, a valve-based filter consisting of two steps, the blurring step and the filtering step, has been devised. In the previous step, Gaussian blur is employed to achieve the smoothing of the input features [35]. While in the filtering step, trigger features will be eliminated via two valves, and thus poisoned nodes could be purified to make the right prediction.

1) *Blurring Step*: all detected poisoned nodes are subjected to Gaussian blurring before filtering. This process is to smooth out the clean features of the nodes, while the trigger features still remain high value, and perturbations become relatively larger and more detectable.

2) *Filtering Step*: node features will further be filtered by two valves: the lower and the higher ones, and only the feature values within the range of the two thresholds will remain unfiltered. The calculation equations for the two valves are listed as follows:

$$V_{low} = k_1 * Mean(Blur_{gaussian}(x)), \quad (12)$$

$$V_{high} = k_2 * Max(Blur_{gaussian}(x)), \quad (13)$$

where x is the feature vector, k_1 and k_2 (set to 0.8 and 0.3 respectively in experiment) are the weights, $Mean(*)$ is the function of the mean value, $Max(*)$ is the function of the max value, and $Blur(*)$ is the function of Gaussian blur in previous step.

Based on the valves, the feature vector can be filtered. The resulting filtered data can be directly utilized for model testing, with the output not being the target class.

V. 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 node-level GNN 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.

A. Attack Settings and Evaluation Metrics

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).

2) *Datasets*: We employ five frequently used real-world datasets (dataset A: Cora [36], dataset B: Citeseer [37], dataset C: Pubmed [38], dataset D: DBLP [39] and dataset E: Physics [40]) to measure the attack performance, and dataset statistics are shown in Table I in detail.

TABLE I
DATASET INFORMATION

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

3) *Attack Setup*: To achieve better performance, the target model will first be pre-trained by SSGNC model, and the pre-trained model is then exploited to generate unlabeled attack data using the proposed PerCBA approach (the poisoning process is displayed in algorithm 1). After that, target model is fine-tuned in SSGNC environment and gets backdoored.

4) *Evaluation Metrics*: We use three common metrics, attack success rate (ASR, the ratio of the successful node attack trials S to all poisoned test nodes T_{poison}), clean data accuracy (Acc, the rate of the correctly classified clean test nodes Y_c to all clean test nodes T_{clean}) and poison rate (the ratio of the poisoned training nodes L_{poison} to all training data L_{train}) 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 Y_t , to all clean test nodes T_{clean}) to be compared with the backdoored model. The calculation equations of the aforementioned metrics are shown as follows:

$$ASR = \frac{S}{T_{poison}}, \quad (14)$$

$$Acc = \frac{Y_c}{T_{clean}}, \quad (15)$$

TABLE II
COMPARISON RESULTS AMONG GCN, GAT AND GRAPH SAGE. THE BEST PERFORMANCES ARE HIGHLIGHTED IN THIS TABLE.

SSGNC variants	Dataset	Poison Rate(%)	ASR(%)	Original Data 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	4.7×10^{-7}	1.2×10^{-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	5.9×10^{-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

$$\text{Poison Rate} = \frac{L_{\text{poison}}}{L_{\text{train}}}, \quad (16)$$

$$\text{MR} = \frac{Y_t}{T_{\text{clean}}}. \quad (17)$$

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.

B. Attack Results

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. Each attack will be tested 10 times to calculate the average result. 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.

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 correct 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. 6.

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.

3) *Affections of Hyperparameters*: We also inspect the correlations between attack performances and three essential hyperparameters: poison rate, perturbation budget and CCDF parameter, and carry out the test on GCN with dataset A for 10 times to collect average result in ASR. Fig. 5(a) to Fig. 5(c) presents the findings.

- As can be seen from Fig. 5(a), accuracy decreases from 74% to 64% as poison rate increases, while attack success 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.
- Regarding the impact of Perturbation budget in Fig. 5(b), as it increased from 10% to 40%, accuracy decreases from 69% to 63% and attack success rate increases from 24% to 72%.
- From Fig. 5(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%.

4) *Affections of Data Categories*: The original class of poisoned nodes may have an impact on the result of the attack

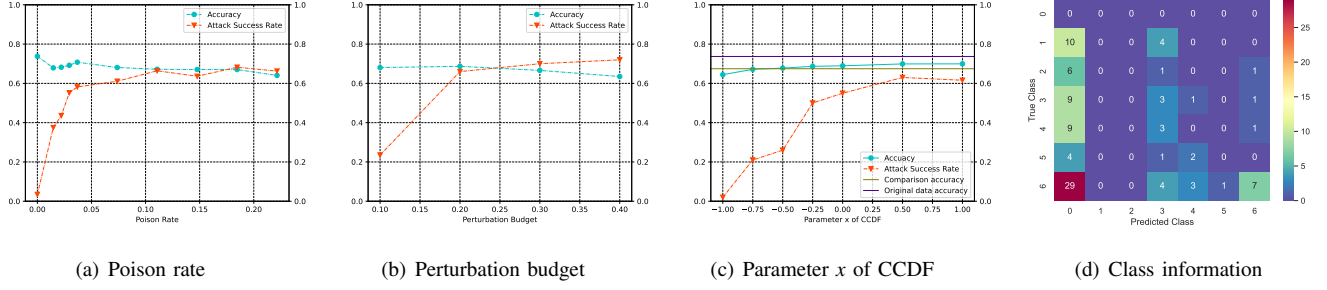


Fig. 5. Affections of different hyperparameters and class information about attack result in dataset A. (a) presents the observation of Acc and ASR with the change of poison rate; (b) illustrates the observation of Acc and ASR with the change of perturbation budget; (c) describes the mentioned indicators with the change of parameter x of CCDF; (d) shows the heat-map for each class under attack circumstance for dataset A.

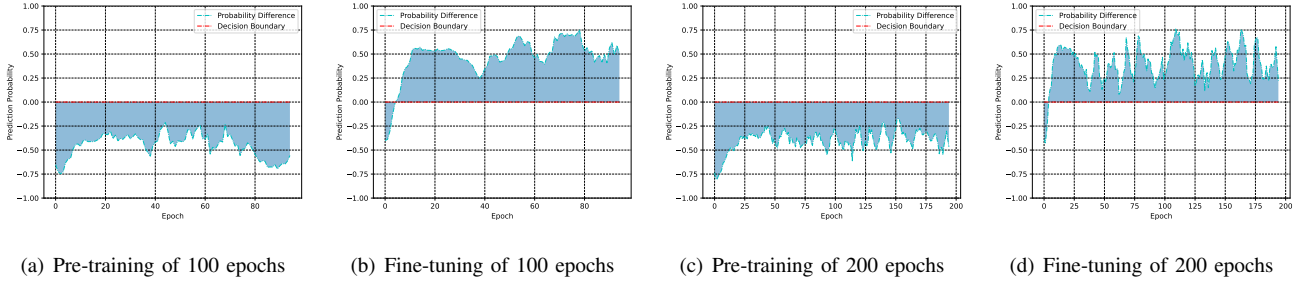


Fig. 6. Illustration for decision boundary change during training. For 100 epochs, (a) depicts how the decision boundary tends to go near to the correct label (negative area) in the pre-training while moving closer to the target (positive area) during the fine-tuning process in (b). In another set of results, including (c) and (d), they have the same trend under 200 epochs.

due to the traits and relationships between various types of nodes. To further analyze whether there is an effect, class information about the attack result for dataset A on GCN is viewed in Fig. 5(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 III and Table IV. 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. Each class will be tested for 10 times to compute average performance.

TABLE III
ATTACK RESULTS FOR DIFFERENT POISONED DATA CLASSES

Class Change	Accuracy	Attack Success Rate
1→0	67.39	71.36
2→0	68.47	62.03
3→0	67.64	64.88
4→0	68.05	52.27
5→0	66.45	73.79
6→0	68.17	44.37

From Fig. 5(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 III. As seen from the results, there is not much difference in accuracies, but class 1 and 5 have relatively high ASR,

TABLE IV
ATTACK RESULTS FOR DIFFERENT TARGET CLASSES

Target Class	Accuracy	Attack Success Rate
0	69.52	60.38
1	68.14	65.10
2	69.77	39.43
3	68.14	79.05
4	69.40	61.27
5	67.33	66.09
6	68.14	51.54

while class 4 and 6 have relatively low ASR. Under the attack conditions of different target classes, the overall experimental results in Table IV 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.

5) *Affections of Different Triggers*: As mentioned in Section III, the proposed PerCBA is pattern-agnostic, and hence we 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 with same trigger size (m set to 40) are employed for comparisons. And also, uniformly distributed feature trigger will be tested in four sizes (trigger feature dimension m set to 40, 60, 80, and 100 respectively) to see the size impact. The comparison results

are shown in Fig. 7.

It can be seen that the three types of triggers with the same size are almost equal (around 61%) in ASR. For uniformly distributed feature trigger, the larger triggers will bring some ASR improvement, but the overall performances are similar. Based on this point, the adversaries can generate various triggers with different features, which significantly extend the attack surface.

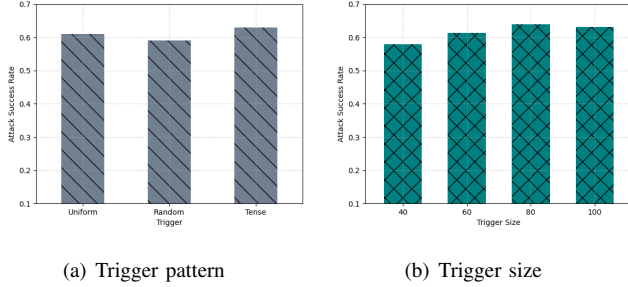


Fig. 7. ASR Comparison for different trigger patterns and sizes using GCN and Cora dataset

6) *Attack Persistence*: To evaluate the persistence of PerCBA, 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 model will be pre-trained for 200 epochs and then poisoned. The result is shown in Fig. 8.

As shown in Fig. 8(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 strength of the perturbation on poisoned data to see if the persistence can be maintained, and GCN is selected as the test model. The result is shown in Fig. 8(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.

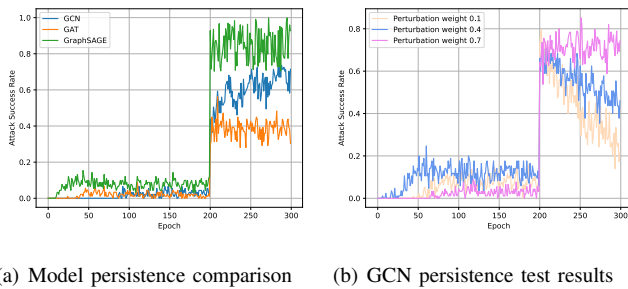


Fig. 8. Persistence test results for GCN, GAT and GraphSAGE, and GCN persistence test results under different perturbation weights.

C. Backdoor Defense

This section assesses the effectiveness of the proposed defense strategy. We will first introduce the defense setup and the evaluation metrics, which are followed by a detailed presentation of the experimental results.

1) *Defense Setup*: We first train a backdoored model based on the attack method, then select 100 poisoned nodes and 100 normal nodes to feed into our defense framework, and then input the filtered nodes into the model to see the ASR and Acc. Each experiment will be repeated 10 times to calculate the average performance.

2) *Evaluation Metrics*: Besides ASR and Acc, we also utilize the detection rate (the proportion of identified poisoned nodes P_d to the total number of poisoned nodes P_t) and false detection (the ratio of the number of clean nodes that were incorrectly detected as poisoned N_d to the total number of clean nodes N_t) rate as metrics to evaluate the efficacy of our defense mechanism. The corresponding calculation equations are listed as follows:

$$\text{Detection Rate} = \frac{P_d}{P_t}, \quad (18)$$

$$\text{Misdetecation Rate} = \frac{N_d}{N_t}. \quad (19)$$

3) *Defense Results*: The general defense results are depicted in Fig. 4. For GCN, the average ASR value attains 4.31% (with a maximum value of approximately 7%), and the performance of Acc is also improved compared to the normal model. The results also exhibit that the detection rates of over 99.5%, and all false detection rates of below 0.1%.

For GAT, the proposed defense mechanism resulted in an average reduction of ASR to approximately 5.22%, along with a slight improvement in Acc as compared to the infection model. The detection rate exhibited a minimum of 99.7%, while the maximum false detection rate was only 0.2%.

For GraphSAGE, the proposed defensive strategy leads to a reduction of ASR to approximately 4.92%, accompanied by an improvement in Acc across all datasets. Moreover, the minimum detection rate is recorded at 99.6%, while the maximum false detection rate is found to be only 0.2%.

To further evaluate the efficacy of the proposed defense approach across varying sizes of poisoned nodes, we generated differing sizes of poisoned test sets to assess the defense performance. GCN and Cora dataset are selected as test targets and the results are shown in Table VI.

As can be seen from Table VI, with increasing infection sizes, the ASR exhibits only minor fluctuations within a small range (3.45% to 5.26%), indicating limited improvement, while the accuracy (Acc) remains nearly constant, fluctuating around an average of 70.24%. The results suggest that the proposed defense method could achieve good performance across poisoned sets of varying sizes.

D. Discussion

In this section, we will analyse why PerCBA could achieve persistent backdoor attacks by infecting small-scale unlabeled nodes.

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

$$\mathbf{H}^{(s)} = \text{Activation}(\hat{\mathbf{A}}\mathbf{H}^{(s-1)}\mathbf{W}^{(s-1)}), \quad (20)$$

TABLE V
DEFENSE TEST RESULTS FOR GCN, GAT AND GRAPH SAGE.

SSGNC variants	Dataset	ASR(%)	Acc(%)	Detection Rate(%)	Misclassification Rate(%)
GCN	Cora	4.21	70.44	100	0
	Citeseer	1.56	65.93	100	0.1
	Pubmed	6.32	69.52	99.8	0
	DBLP	2.11	74.51	99.5	0
	Physics	7.37	88.32	100	0.1
GAT	Cora	6.24	70.52	99.8	0
	Citeseer	3.53	56.15	100	0.1
	Pubmed	5.88	63.85	99.7	0.2
	DBLP	4.06	78.12	99.8	0.1
	Physics	6.41	86.62	100	0
GraphSAGE	Cora	7.06	65.25	100	0.1
	Citeseer	3.41	66.12	100	0
	Pubmed	4.71	68.63	99.7	0.1
	DBLP	1.18	74.84	99.6	0.2
	Physics	8.24	85.33	100	0.1

TABLE VI
DEFENSE RESULTS FOR DIFFERENT SIZES OF POISONED NODES IN CORA DATASET.

Poisoned nodes	ASR(%)	Acc(%)
50	3.56	69.34
100	5.12	71.55
200	4.89	70.81
300	4.81	71.27
400	5.26	69.19
500	3.45	68.89
600	4.41	70.65

$$\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{(-\frac{1}{2})} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{(-\frac{1}{2})}. \quad (21)$$

If the activation function is ignored, Eq. 20 can be expressed approximately as

$$\mathbf{H}^{(s)} = \hat{\mathbf{A}}^{(s)} \mathbf{H}_0 \prod_{i=0}^s \mathbf{W}^i, \quad (22)$$

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

$$[u_i]^{(s)} = \sum_{j=0}^n a_{ij} [u_j]^0 \prod_{i=0}^s \mathbf{W}^i, \quad (23)$$

where a_{ij} comes from a_i and $[u_j]^0$ is row vector of the initial feature matrix \mathbf{H}_0 (or \mathbf{X}). For independent node, its adjacent vector entries are 0 except a_{ii} (value = 1), then Eq. 23 can be rewritten as

$$[u_i]^{(s)} = [u_i]^0 (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_c), \quad (24)$$

where \mathbf{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.

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.

3) *Why PerCBA Persistent?*: The performance of traditional backdoors decreases as the model is iteratively trained, 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.

VI. CONCLUSION

In this study, we first discussed traditional backdoor's inapplicability for SSGNC resulting from its high-intensity data poisoning. To overcome this problem, we furthermore propose PerCBA, the first clean-label backdoor for SSGNC. 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 training. This is the first clean-label backdoor method (that does not change any label information) for SSGNC and just leverages small-scale nodes as targets, which assures that the attack is less detectable. Moreover, to achieve more secure model performances, we further proposed the corresponding defense method against PerCBA, which utilizes kurtosis and skewness to identify toxic nodes and combines Gaussian blurring and thresholding to filter out toxic triggers so that they could be purified, thus leading to normal model prediction. 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. As for the defense performance, the

proposed strategy is effective in detecting poisoned nodes and reducing ASR. For future work, we will focus on developing a general defense strategy for all kinds of backdoor attacks for GNN node classifications, enhancing the trustworthiness of social systems against rumor diffusion, fake news, and malicious burner accounts.

VII. ACKNOWLEDGEMENT

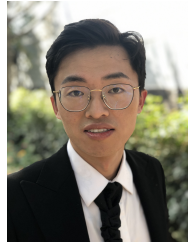
This research work is funded by the National Nature Science Foundation of China under Grant No. 62202303, U21B2019, and U20B2048, Shanghai Sailing Program under Grant No. 21YF1421700 and Shanghai Municipal Science and Technology Major Project under Grant 2021SHZDZX0102.

REFERENCES

- [1] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.
- [2] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The world wide web conference*, pp. 417–426, 2019.
- [3] J. Sun, W. Zheng, Q. Zhang, and Z. Xu, "Graph neural network encoding for community detection in attribute networks," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 7791–7804, 2021.
- [4] S. Arora, "A survey on graph neural networks for knowledge graph completion," *arXiv preprint arXiv:2007.12374*, 2020.
- [5] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.
- [6] L. Wu, P. Cui, J. Pei, L. Zhao, and L. Song, *Graph neural networks*. Springer, 2022.
- [7] Y. Zhou, H. Zheng, X. Huang, S. Hao, D. Li, and J. Zhao, "Graph neural networks: Taxonomy, advances, and trends," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 1, pp. 1–54, 2022.
- [8] G. Dong, M. Tang, Z. Wang, J. Gao, S. Guo, L. Cai, R. Gutierrez, B. Campbell, L. E. Barnes, and M. Boukhechba, "Graph neural networks in iot: A survey," *ACM Transactions on Sensor Networks*, vol. 19, no. 2, pp. 1–50, 2023.
- [9] Y. Li, Y. Jiang, Z. Li, and S.-T. Xia, "Backdoor learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–18, 2022.
- [10] S. Kaviani and I. Sohn, "Defense against neural trojan attacks: A survey," *Neurocomputing*, vol. 423, pp. 651–667, 2021.
- [11] H. Chen and F. Koushanfar, "Tutorial: Toward robust deep learning against poisoning attacks," *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 3, pp. 1–15, 2023.
- [12] G. Ortiz-Jimenez, A. Modas, S.-M. Moosavi, and P. Frossard, "Hold me tight! influence of discriminative features on deep network boundaries," in *Advances in Neural Information Processing Systems, NeurIPS 2020*, vol. 33, pp. 2935–2946, 2020.
- [13] Z. Yan, G. Li, Y. Tian, J. Wu, S. Li, M. Chen, and H. V. Poor, "Dehib: Deep hidden backdoor attack on semi-supervised learning via adversarial perturbation," in *AAAI 2021, Virtual Event*, pp. 10585–10593, 2021.
- [14] J. Xu and 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*, p. 3491–3493, 2022.
- [15] R. Ning, J. Li, C. Xin, and H. Wu, "Invisible poison: A blackbox clean label backdoor attack to deep neural networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1–10, 2021.
- [16] X. Chen, Y. Dong, Z. Sun, S. Zhai, Q. Shen, and Z. Wu, "Kallima: A clean-label framework for textual backdoor attacks," 2022.
- [17] S. Zhao, X. Ma, X. Zheng, J. Bailey, J. Chen, and Y.-G. Jiang, "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.
- [18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, OpenReview.net, 2017.
- [19] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.
- [20] K. K. Thekumparampil, C. Wang, S. Oh, and L. Li, "Attention-based graph neural network for semi-supervised learning," *CoRR*, vol. abs/1803.03735, 2018.
- [21] V. Verma, M. Qu, K. Kawaguchi, A. Lamb, Y. Bengio, J. Kannala, and J. Tang, "Graphmix: Improved training of gnns for semi-supervised learning," in *AAAI 2021, Virtual Event, February 2-9, 2021*, pp. 10024–10032, 2021.
- [22] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang, "Graph random neural networks for semi-supervised learning on graphs," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2020.
- [23] J. Wu, J. He, and J. Xu, "Net: Degree-specific graph neural networks for node and graph classification," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 406–415, 2019.
- [24] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the 14th ACM international conference on web search and data mining*, pp. 833–841, 2021.
- [25] J. Zhao, X. Wang, C. Shi, B. Hu, G. Song, and Y. Ye, "Heterogeneous graph structure learning for graph neural networks," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 4697–4705, AAAI Press, 2021.
- [26] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *USENIX Security Symposium (USENIX Security)*, pp. 1523–1540, 2021.
- [27] J. Xu, M. Xue, and S. Picek, "Explainability-based backdoor attacks against graph neural networks," in *Proceedings of ACM Workshop on Wireless Security and Machine Learning*, pp. 31–36, 2021.
- [28] L. Chen, Q. Peng, J. Li, Y. Liu, J. Chen, Y. Li, and Z. Zheng, "Neighboring backdoor attacks on graph convolutional network," *CoRR*, vol. abs/2201.06202, 2022.
- [29] J. Zhang and Y. Luo, "Degree centrality, betweenness centrality, and closeness centrality in social network," in *Proceedings of International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017)*, pp. 300–303, 2017.
- [30] B. Ruhnau, "Eigenvector-centrality — a node-centrality?," *Social Networks*, vol. 22, no. 4, pp. 357–365, 2000.
- [31] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 80, pp. 1123–1132, 2018.
- [32] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations (ICLR)*, 2018.
- [33] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations (ICLR)*, OpenReview.net, 2018.
- [34] R. A. Groeneveld and G. Meeden, "Measuring skewness and kurtosis," *Journal of the Royal Statistical Society: Series D (The Statistician)*, vol. 33, no. 4, pp. 391–399, 1984.
- [35] E. S. Gedraite and M. Hadad, "Investigation on the effect of a gaussian blur in image filtering and segmentation," in *Proceedings ELMAR-2011*, pp. 393–396, IEEE, 2011.
- [36] A. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval Journal*, vol. 3, no. 2, pp. 127–163, 2000.
- [37] C. L. Giles, K. D. Bollacker, and S. Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of ACM Conference on Digital Libraries*, p. 89–98, 1998.
- [38] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, p. 93, Sep. 2008.
- [39] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *KDD*, (New York, NY, USA), p. 990–998, 2008.
- [40] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *CoRR*, vol. abs/1811.05868, 2018.



Xiao Yang received the B.E. degree from the School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, China, and the M.S. degree from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China, where he is currently pursuing the Ph.D. degree. His research interests involve graph neural networks (GNNs), backdoor attacks, and multi-modal learning.



Gaolei Li (S'16-M'21) received the B.S. degree from Sichuan University, Chengdu, China and PhD degree from Shanghai Jiao Tong University, Shanghai, China. In 2018-2019, he visited at Muroran Institution of Technology, Muroran, Japan, granted by the China Scholarship Council Program. Now, he is an Assistant Professor in School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China. His research interests include adversarial machine learning and privacy protection. He has received best paper

awards from the IEEE ComSoc CSIM Committee, Chinese Association for Cryptologic Research (CACR) and IEEE Globecom student travel grant award. He serves as a reviewer of IEEE TIFS, TDSC, TII, etc and PC member of AAAI, IEEE Globecom, IEEE ICC, etc.



Jun Wu received the Ph.D. degree in information and telecommunication studies from Waseda University, Japan, in 2011. He was a Post-Doctoral Researcher with the Research Institute for Secure Systems, National Institute of Advanced Industrial Science and Technology (AIST), Japan, from 2011 to 2012. He was a Researcher with the Global Information and Telecommunication Institute, Waseda University, Japan, from 2011 to 2013. He is currently a professor of School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, China.

He is also the vice dean of Institute of Cyber Science and Technology and vice director of National Engineering Research Center for Information Content Analysis Technology, Shanghai Jiao Tong University, China. He is the chair of IEEE P21451-1-5 Standard Working Group. He has hosted and participated in a lot of research projects including National Natural Science Foundation of China (NFSC), National 863 Plan and 973 Plan of China, Japan Society of the Promotion of Science Projects (JSPS), etc. His research interests include the intelligence and security techniques of artificial intelligence, digital twin, Internet of Things (IoT), 5G/6G, molecular communication, etc. He has been the Track Chair of VTC 2019, VTC 2020 and the TPC Member of more than ten international conferences including ICC, GLOBECOM, etc. He has been a Guest Editor of IEEE Transactions on Industrial Informatics, IEEE Transactions on Intelligent Transportation, IEEE Sensors Journal, Sensors. He is an Associate Editor of the IEEE Systems Journal, IEEE Networking Letters. He is a Senior Member of IEEE.



Jianhua Li is a professor/Ph.D. supervisor and the dean of Institute of Cyber Science and Technology, Shanghai Jiao Tong University, Shanghai, China. He is also the director of National Engineering Laboratory for Information Content Analysis Technology, the director of Engineering Research Center for Network Information Security Management and Service of Chinese Ministry of Education, and the director of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security, China. He is the vice president of Association

of Cyber Security Association of China. He got his BS, MS and Ph.D. degrees from Shanghai Jiao Tong University, in 1986, 1991 and 1998, respectively. He was the chief expert in the information security committee experts of National High Technology Research and Development Program of China (863 Program) of China. He was the leader of more than 30 state/province projects of China, and published more than 300 papers. He published 6 books and has about 20 patents. He made 3 standards and has 5 software copyrights. He got the Second Prize of National Technology Progress Award of China in 2005. His research interests include information security, signal process, computer network communication, etc.