Adversarial Label-Flipping Attack and Defense for Graph Neural Networks

Mengmei Zhang
dept. Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
zhangmm@bupt.edu.cn

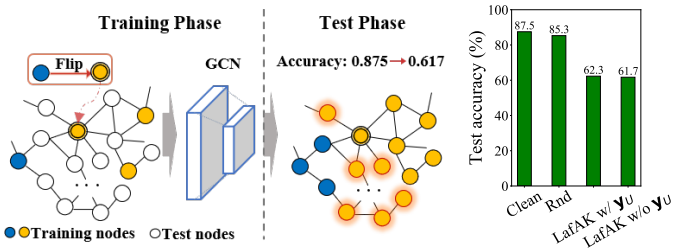
Linmei Hu
dept. Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
hulinmei@bupt.edu.cn

Chuan Shi*
dept. Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
shichuan@bupt.edu.cn

Xiao Wang
dept. Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
xiaowang@bupt.edu.cn

***Abstract*—With the great popularity of Graph Neural Networks (GNNs), the robustness of GNNs to adversarial attacks has received increasing attention. However, existing works neglect adversarial label-flipping attacks, where the attacker can manipulate an unnoticeable fraction of training labels. Exploring the robustness of GNNs to label-flipping attacks is highly critical, especially when labels are collected from external sources and false labels are easy to inject (e.g., recommendation systems). In this work, we introduce the first study of adversarial label-flipping attacks on GNNs. We propose an effective attack model LafAK based on approximated closed form of GNNs and continuous surrogate of non-differentiable objective, efficiently generating attacks via gradient-based optimizers. Furthermore, we show that one key reason for the vulnerability of GNNs to label-flipping attack is overfitting to flipped nodes. Based on this observation, we propose a defense framework which introduces a community-preserving self-supervised task as regularization to avoid overfitting. We demonstrate the effectiveness of our proposed attack model to GNNs on four real-world datasets. The effectiveness of our defense framework is also well validated by the substantial improvements of defense based GNN and its variants under label-flipping attacks.**

***Index Terms*—Adversarial label-flipping attacks, adversarial robustness, graph neural networks**



(a) Attack process. (b) Test accuracy.

Fig. 1: An example of label-flipping attacks on Polblogs dataset (1490 nodes with 40 labeled and 4 flipped). **(a)** A toy example that illustrates the process of label-flipping attacks: attacker maliciously uploads the crafted flipped label (e.g., flipping a node from blue to yellow with double outlines), aiming for maximum misclassified test nodes (red border circles) after retraining GNN. **(b)** Test accuracy of the retrained model on clean labels (**Clean**), random noise labels (**Rnd**) and our crafted flipped labels with/without true test labels $\mathbf{y}_U$ beforehand (**LafAK w/** $\mathbf{y}_U$; **LafAK w/o** $\mathbf{y}_U$). A smaller value indicates better attack performance.

I. Introduction

Graph is commonly used to model many real-world relationships, such as social networks [1], [2], citation networks [3] and the e-commerce networks [4]. In recent years, deep learning starts to push forward the performance on graph data. Typically, graph neural networks (GNNs) [3], [5], which combine node feature information with the graph structure by using neural networks, have achieved state-of-the-art performance on a wide range of tasks (e.g., node classification task).

Despite great success, GNNs have been proved to often suffer from adversarial attacks, especially for poisoning attacks [6] where the attacker tries to compromise the performance of a model by manipulating training data. Specifically, [7]–[10] pointed out the vulnerability of GNNs to topology attacks. [7], [11] also demonstrated a susceptibility to feature attacks. These vulnerabilities, with the wide applications of GNNs, have received increasing attention from both academia and industry.

However, existing works neglect a specific type of poisoning attacks, i.e., adversarial label-flipping attack [12], where the attacker can modify an unnoticeable fraction of training labels. Adversarial label-flipping attacks, unlike random label noise, can be specifically targeted to exploiting the vulnerabilities of the learning algorithm, significantly degrading overall performance. Over the past years, label-flipping attacks have been extensively studied in many domains [12], [13]. More recently, classical graph-based semi-supervised classifiers [14], have

been proved to be vulnerable to such attacks. For example, in the label propagation model [15], the misleading information of few flipped nodes can propagate along the whole graph and affect a quite large proportion of nodes. Here one question comes: *Can deep learning graph model also be easily destroyed by adversarial label-flipping attacks?* The answer to this question is indeed not foreseeable: On one hand, the local aggregation mechanism might improve robustness since the propagation of flipped labels is restricted to their local neighbours. On the other hand, as deep learning models, they can easily overfit the misinformation of incorrect labels [16]. Exploring the robustness of GNNs to label-flipping attacks is highly critical, especially when labels are collected from external sources, and false labels are easy to inject. For example, recommendation systems (e.g., Netflix [17]) based on users' ratings and labelling in social networks (e.g., Polblogs [2]) which rely on user's self-reporting are all exposed to the label-flipping attacks.

Figure 1 is an example illustrating the label-flipping attacks against the target GNN model [3] on Polblogs network [2] (1490 nodes with 40 labeled). As the figure shows, even if only four labels flipped (e.g., flipping a node from blue to yellow with double outlines), the test accuracy of target model dramatically drops by about 29% after retraining. As a comparison, the test accuracy under randomly flipping four training labels (Rnd) decreases only by an average of 2.51%. Obviously, the performance degradation of the GNN caused by adversarial label-flipping attacks is far higher than random noise. The result demonstrates that GNNs are vulnerable to our attacks, which could hinder their applicability in various domains. Hence, there is a strong need for systemically studying and further improving the robustness of GNNs to label-flipping attacks.

As existing adversarial attacks on graph data, label-flipping attacks is essentially an NP-hard integer programming problem due to discreteness. To avoid the combinatorial search, recent advanced methods [10], [14] suggest to utilize gradient-based optimizers to search for optimal attacks. But such ideas cannot be directly applied for our problem due to two challenges: (1) Label-flipping attack, where the (outer) optimization of attacks involves solving the (inner) optimization of GNNs, is inherently a bi-level optimization and usually intractable to solve. (2) The problem contains unique non-differentiable components (e.g., computing accuracy and flipping operation), preventing us to directly applying gradients.

In this paper, we propose an effective adversarial ***La***bel-***f***lipping ***A***ttac***K*** model against GNNs (**LafAK**) ,directly tackling the above two challenges, to systematically investigate the robustness of GNNs to label-flipping attacks. Specifically, to address the first challenge, we propose an approximated closed form of GNN via linearization and replacement of (inner) classification loss. Based on such closed form, we can directly solve the bi-level optimization problem by transforming it into single level. For the second challenge, we specially design continuous surrogates for these non-differentiable components, achieving better differentiability and functionality compared to previous attacks [14], [18], [19]. In this way, we can generate attacks effectively and efficiently with gradient-based optimizers.

Furthermore, to improve the robustness of GNNs to label-flipping attacks, we show that one key reason for their vulnerability to such attacks is the overfitting to flipped nodes. To alleviate this issue, inspired by [20]–[22], we propose a community-preserving self-supervised-based defense framework, which can improve the robustness of GNNs against label-flipping attacks by fully exploiting community-level signals. In particular, we first apply off-the-shelf community detection methods to automatically extract community-level signals. Next, we train GNNs with predicting the communities of nodes as an auxiliary self-supervised task, explicitly retaining community in training process. As a result, GNNs, once overfitting flipped nodes, will be punished by community-level signal.

In summary, our contributions are three folds:

- We introduce the first study on the robustness of GNNs to label-flipping attacks, and propose an effective label-flipping attack model LafAK based on approximated closed form of GNN and replacement of non-differentiable components, efficiently generating attacks for GNNs via gradients.
- To the best of our knowledge, we are the first to defend label-flipping attacks for GNNs, and propose an effective defense framework based on community-preserving self-supervised learning, alleviating the overfitting to flipped nodes.
- We show that GNNs are highly vulnerable to adversarial label-flipping attacks. Empirical results on four real-world datasets demonstrate the effectiveness of our attacks to GNNs. The effectiveness and generalization ability of our defense framework are shown by the considerable improvement of both GNN and its variants under label-flipping attacks.

II. Related work

A. Graph Neural Networks

Graph neural networks (GNNs) [23] on graph structured data have shown outstanding results in various tasks such as node classification [3], [5], link prediction [24], graph clustering [25]. A special form of GNNs is graph convolutional networks (GCNs) [3], which learn on graph structures using convolution operations. By stacking multiple convolutional layers, GCNs achieve state-of-the-art performance on semi-supervised node classification task, and have drawn increasing research interests in the past few years. To further improve GCNs, [26] introduces attention mechanisms, [27] adds residual and jumping connections, [28] simplified GCN, etc. In this paper, we mainly focus on attacking GCN [3] and its variants.

B. Adversarial Attacks on GNNs

The robustness of GNNs to adversarial attacks has been extensively studied in recent years. In existing works, the attacker is capable of adding/removing/rewiring edges in the

whole input graph [7]–[10], [29]–[33], manipulating the features of target nodes [7], [11], [29], or injecting vicious nodes [34]. More recently, poisoning attacks as a type of adversarial attacks targeted to the training data, have received increasing attention [6]. Basically, poisoning attacks are challenging due to the inherent intractable bi-level optimization problem. To confront it, existing works try to circumvent tackling the problem by assuming the parameters of GNNs are static [7], [8], [29], [32], or optimize the bi-level problem indirectly. For example, [9], [10] transformed the problem into a min-max problem. However, there is no existing adversarial attack works on manipulating the labels of nodes for GNNs. This paper sheds first light on this important problem. In addition, different from previous poisoning attacks against GNNs, we solve the bi-level problem directly by deriving an approximated closed form of inner model (GCN).

C. Adversarial Label-Flipping Attacks and Defenses

Adversarial label-flipping attack is a specific type of poisoning attacks where the attacker is restricted to modify the labels of a few training nodes. This type of attacks, unlike random label noise [35], can be specifically targeted to exploiting the gradient information of the learning algorithm. Thus they can significantly degrade the node classification performance. So far, the robustness to such attacks have been widely studied in many domains (e.g., deep learning [13]). More recently, [14] first studied label-flipping attacks against classical graph-based semi-supervised learning(e.g., label propagation which has analytical solution), and proposed an efficient attack model based on gradient descent optimizer. They cannot be applied to GCN that has no closed-form solution. On the other hand, few general defense mechanisms have been proposed against label-flipping attacks. One common defense technique is Sanitization [19], [36], where a defender attempts to identify the training points that may have had their labels corrupted and then removes/relabels them. In summary, we focus on label-flipping attack and defense models towards graph neural models.

III. Background and preliminaries

In this section, we briefly introduce the preliminaries of GCNs and provide the background of bi-level optimization problem.

A. Graph Convolutional Networks

We consider attacking GCNs on the task of semi-supervised node classification. Formally, we define an undirected graph as $G = (V, E)$, where V is the set of N nodes, E represents the set of edges and $\mathbf{A} \in \{0, 1\}^{N\times N}$ is the adjacency matrix of graph. Basically, GCNs often assume nodes in a graph have node attributes $\mathbf{X} \in \mathbb{R}^{N\times d}$ where each node i is associated with a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and a class label $y_i \in \{1, ..., K\}$. Given $\mathbf{A}$, $\mathbf{X}$, labeled nodes set V_L with $N_L = |V_L|$, unlabeled nodes set $N_U = |V_U|$ and the labels of nodes in V_L the goal of GCNs is to predict the class of nodes in V_U.

In this work, we focus on a representative GCN proposed by [3] for simplicity. Typically, GCN learns on graph structures using convolution operations to aggregate neighbouring nodes:

$$\mathbf{H}^{(l+1)} = \mathrm{ReLU}(\hat{\mathbf{A}}\mathbf{H}^{(l)}\mathbf{W}^{(l)}), \tag{1}$$

where $\mathbf{H}^{(l+1)}$ denotes the hidden feature matrix at the $(l+1)$-th layer. Initially, $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{W}^{(l)}$ is the trainable weight matrix at l-th layer, $\hat{\mathbf{A}}$ is the normalized adjacency matrix, given by $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\tilde{\mathbf{D}}_{ii} = \Sigma_j \tilde{\mathbf{A}}_{ij}$ and $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. Following [3], we consider GCNs with two layers ($L = 2$) that compute the class probability vector of node i as

$$f_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X})_i = \mathrm{Softmax}(\hat{\mathbf{A}}\mathrm{ReLU}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)})_i, \tag{2}$$

where $f_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X})_i \in [0, 1]^K$ and $\boldsymbol{\theta} = (\mathbf{W}^{(1)}, \mathbf{W}^{(2)})$ are all parameters of the two-layer GCN.

B. Bi-level Optimization Problem

The label-flipping attack on GNNs is essentially a bi-level optimization problem, where an (outer) optimization involves another (inner) optimization as a constraint. Unfortunately, this bi-level problem, without the closed form of inner optimization, is non-convex and intractable to solve precisely: (1) The solving of inner optimization needs to retrain the inner model, which is extremely time-consuming. (2) Minimizing the outer objective with gradient-based optimizers is not straightforward since the gradients cannot be easily back-propagated through the retraining process of the inner model.

The solutions of this problem in adversarial attacks against graph mainly can be divided into the following lines: (1) Circumventing it by assuming that the inner model is fixed [7]; (2) Tackling this problem indirectly. (e.g.Converting it into min-max (a.k.a. saddle point) problem with assuming that the empirical train and test distribution are close together [9], [10].) (3) Directly transforming the bi-level problem into single level with closed form of inner model (e.g., Label Propagation and DeepWalk [14], [37]). In this paper, we follow the third line.

IV. Label-flipping attack model

In this section, we depict our proposed attack model **LafAK** for GCNs, which can generate unnoticeable label-flipping attacks efficiently. First, we define the attack objective formally, which is essentially a non-differentiable bi-level optimization problem and cannot be solved with the existing efficient gradient-based optimizers. Next, to make such a problem tractable, we propose an effective model consisting of two components: (1) Approximating closed form of GCN, which can be used to directly transform the bi-level optimization into single level. (2) Designing continuous surrogates for the non-differentiable components within the attack objective, still preserving their functionality. In this way, the attack objective can be optimized easily with gradients.

A. Attack Objective

In this paper, in line with most of the related works [14], [18], [19], we restrict the attack scope to binary classification, where the label of nodes $\mathbf{y} \in \{-1,+1\}^N$, the labels of V_L and V_U are $\mathbf{y}_L \in \{-1,+1\}^{N_L}$ and $\mathbf{y}_U \in \{-1,+1\}^{N_U}$ respectively.

The optimal parameters of two-layer GCN are learned in a semi-supervised fashion by minimizing training (classification) loss on the output of the labeled nodes V_L as

$$\begin{aligned} &L(\boldsymbol{\theta}; \mathbf{A}, \mathbf{X}, \mathbf{y}_L) = \\ &\sum_{i}^{N_L} \ell\Big(f_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X})_L[i], \mathbf{y}_L[i]\Big) + \lambda \|\boldsymbol{\theta}\|_2^2, \end{aligned} \tag{3}$$

where $\ell(\cdot,\cdot)$ is a point-wise loss function for node classification and the output of GCN $f_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X}) \in [-1,+1]^N$. In general, the learned GCN model is evaluated by test accuracy (0-1 test error)

$$L_{0-1}(\boldsymbol{\theta}; \mathbf{A}, \mathbf{X}, \mathbf{y}_U) = \frac{1}{N_U} \sum_{i}^{N_U} \mathbb{I}[\mathrm{sign}(f_{\boldsymbol{\theta}}(\mathbf{A}, \mathbf{X})_U)[i] \neq \mathbf{y}_U[i]], \tag{4}$$

where $\mathbb{I}[\cdot]$ is a non-differentiable indicator function and $\mathrm{sign}(\cdot)$ is the Heaviside step function mapping the prediction of nodes to their discrete class label.

The goal of LafAK is to find an unnoticeable fraction of training labels $\mathbf{y}_L$, such that once these labels are flipped, the test accuracy of the retrained GCN are maximally degraded. Specifically, the training labels $\mathbf{y}_L$ are flipped by $(\boldsymbol{\delta} \odot \mathbf{y}_L)$, where $\boldsymbol{\delta} \in \{+1,-1\}^{N_L}$, $\odot$ denotes Hadamard product, $\boldsymbol{\delta}[i] = -1$ indicates flipping label $\mathbf{y}_L[i]$, and $\boldsymbol{\delta}[i] = 1$ means not flipping. Formally, the goal of our model LafAK can be formulated as

$$\begin{aligned} &\min_{\boldsymbol{\delta}} -L_{0-1}(\boldsymbol{\theta}^*; \mathbf{A}, \mathbf{X}, \mathbf{y}_U), \\ &s.t. \quad \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}; \mathbf{A}, \mathbf{X}, \boldsymbol{\delta} \odot \mathbf{y}_L), \quad \|\boldsymbol{\delta} - \mathbf{1}\|_0 \leq \epsilon N_L, \end{aligned} \tag{5}$$

where the maximum number of flips is limited by a small flipping ratio ϵ to ensure the imperceptibility of attacks. As we can see, the (outer) optimization on 0-1 test error $L_{0-1}(\boldsymbol{\theta}^*; \mathbf{A}, \mathbf{X}, \mathbf{y}_U)$ contains the (inner) optimization of classification loss $L(\boldsymbol{\theta}; \mathbf{A}, \mathbf{X}, \boldsymbol{\delta} \odot \mathbf{y}_L)$ as a constraint. This is a typical bi-level optimization problem. Note that $\boldsymbol{\theta}^*$ are always constrained as the 'optimal' parameters of GCN with given $\boldsymbol{\delta}$.

Obviously, the whole attack objective is a non-differentiable bi-level optimization problem. As mentioned in section III-B, without the closed form of inner model (GCN), the above bi-level optimization is highly challenging. Even worse, the optimization of the attack objective contains non-differentiable components (e.g., characteristic function $\mathbb{I}[\cdot]$, Heaviside step function $\mathrm{sign}(\cdot)$ and flipping vector $\boldsymbol{\delta}$, becoming more difficult.

B. Approximating Closed Form of GCN

To address the bi-level optimization problem, we provide an approximated closed form of GCN, transforming the bi-level problem into single-level. Specifically, we obtain such approximated closed form with two steps of simplification: (1) Linearization. The linearization of GCN can reduce the excessive complexity of GCNs while still preserve the idea of graph convolutions. (2) Simplifying of inner (classification) loss. To use existing closed-form estimator "ordinary least-squares (OLS)", we replace the (inner) classification loss with regression loss which can be solved by OLS.

Linearization of GCN. As a deep learning model, the inherent complexity of GCN hinders approximating the closed form of GCN. Moreover, the complexity has been proved to be unnecessary for some applications [28]. To obtain a approximated closed form of GCN while keep the functionality of graph convolutions, we simplify GCN by performing a linearization of GCN in Eq.(6). Here we remove the non-linearities between GCN layers as [28]:

$$\mathrm{Softmax}(\hat{\mathbf{A}}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(1)})\mathbf{W}^{(2)}) = \mathrm{Softmax}(\hat{\mathbf{A}}^2\mathbf{X}\mathbf{W}), \tag{6}$$

where $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are reparameterized into single matrix $\mathbf{W}$, yielding $\boldsymbol{\theta} = \mathbf{W}$ and $\boldsymbol{\theta} \in \mathbb{R}^d$.

Replacement of (inner) classification loss. Given the linearization version in Eq.(6), GCN is reduced to a simple feature propagation step (i.e., $\hat{\mathbf{A}}^2\mathbf{X}$) followed by a standard logistic regression classification, which does not yet have statistically efficient closed-form estimators even under small-sample settings [38]. In contrast, linear regression models have a closed-form estimator, i.e., the OLS estimator. Hence, we simplify GCN to linear regression, yielding the approximated closed form of GCN (named GCN (appr)) by replacing the (inner) classification loss ℓ to be regression loss (namely, squared loss):

$$\boldsymbol{\theta}^* = \frac{1}{N_L} \arg\min_{\boldsymbol{\theta}} \|(\hat{\mathbf{A}}^2\mathbf{X}\boldsymbol{\theta})_L - \mathbf{y}_L\|_2^2 + \lambda\|\boldsymbol{\theta}\|_2^2. \tag{7}$$

where $\boldsymbol{\theta}^*$ is the optimal parameter of GCN (appr). In this way, the OLS estimator can be used to obtain the closed-form of $\boldsymbol{\theta}^*$ as follows:

$$\begin{aligned} \boldsymbol{\theta}^* &= ((\hat{\mathbf{A}}^2\mathbf{X})_L^T(\hat{\mathbf{A}}^2\mathbf{X})_L + \lambda\mathbf{I})^{-1}(\hat{\mathbf{A}}^2\mathbf{X})_L^T(\boldsymbol{\delta} \odot \mathbf{y}_L) \\ &= \mathbf{P}(\boldsymbol{\delta} \odot \mathbf{y}_L), \end{aligned} \tag{8}$$

where $\mathbf{P} = ((\hat{\mathbf{A}}^2\mathbf{X})_L^T(\hat{\mathbf{A}}^2\mathbf{X})_L + \lambda\mathbf{I})^{-1}(\hat{\mathbf{A}}^2\mathbf{X})_L^T$ can be pre-processed easily for given graph.

Empirically, as Fig. 2 shows, our proposed GCN (appr) has comparable performance to GCN under various flipping ratios. Given such an approximated closed form of GCN, the bi-level optimization problem can be transformed into single level:

$$\begin{aligned} &\min_{\boldsymbol{\delta}} -L_{0-1}(\mathbf{P}(\boldsymbol{\delta} \odot \mathbf{y}_L); \mathbf{A}, \mathbf{X}, \mathbf{y}_U), \\ &s.t. \quad \|\boldsymbol{\delta} - \mathbf{1}\|_0 \leq \epsilon N_L. \end{aligned} \tag{9}$$

C. Continuous Surrogates of Non-differentiable Components

Despite the given single-level formula in Eq. (9), it is still challenging to optimize the attack objective due to the inherent non-differentiable components. To address this problem, we apply the following continuous surrogates of these components:

(1) For the non-differentiable 0-1 error L_{0-1}, one standard solution is to replace L_{0-1} with the continuous test loss $L(\boldsymbol{\theta}^*; \mathbf{A}, \mathbf{X}, \mathbf{y}_U)$ [14], [18], [19]. However, such replacement will hurt the functionality of 0-1 error and further reduce attack effectiveness. Compared to 0-1 error, attack objective equipped with test loss aims to maximally degrade the overall confidence of classification, rather than test accuracy. To better preserve the functionality, we first substitute the Heaviside step function (namely $\text{sign}(\cdot)$) with a smooth approximation $\tilde{h}(x) = \tanh(\tau x)$ as shown in Fig. 3. As the figure shows, smoothness coefficient τ can be used to tune the accuracy of the approximation. The larger value of τ, the closer the smooth function is to the step function. Next, given flips $\boldsymbol{\delta} \odot \mathbf{y}_L$, we can further obtain the approximated predictions of test nodes $\tilde{\mathbf{y}}_U$ by:

$$\tilde{\mathbf{y}}_U = \tilde{h}(\hat{\mathbf{A}}^2 \mathbf{X} \boldsymbol{\theta}^* (\boldsymbol{\delta} \odot \mathbf{y}_L))_U, \tag{10}$$

where $\boldsymbol{\theta}^*$ is computed with Eq.(8). Lastly, substituting the above $\tilde{\mathbf{y}}_U$ into L_{0-1} (Eq. (4)), we can obtain a continuous differentiable 0-1 error:

$$L_{0-1}(\boldsymbol{\theta}^*; \mathbf{A}, \mathbf{X}, \mathbf{y}_U) \overset{def}{=} \frac{1}{N_U} \sum_{i}^{N_U} \Big(\tilde{\mathbf{y}}_U[i] \odot \mathbf{y}_U[i] \Big). \tag{11}$$

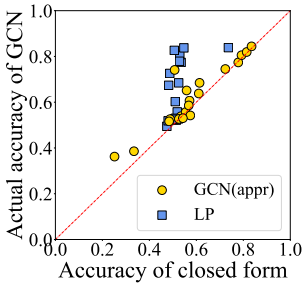


Fig. 2: We compare the test accuracy from actual GCN vs. closed form of GCN (appr) / label propagation (LP) under different random splits and flip ratios. We can see that GCN (appr) has comparable performance to the actual GCN (i.e., yellow points are near the diagonal).

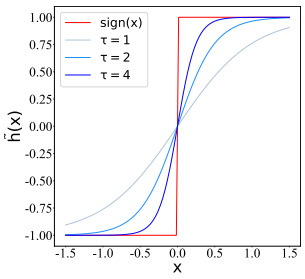


Fig. 3: Smooth approximation $\tilde{h}(\cdot)$ to the Heaviside step function $\text{sign}(\cdot)$. By varying smoothness coefficient τ, we can approximate the Heaviside step function in terms of arbitrary accuracy.

(2) To tackle the discrete-valued label-flipping operation vector $\boldsymbol{\delta}$, we model the elements in $\boldsymbol{\delta}$ with a set of mutually independent Bernoulli random variables $\boldsymbol{\alpha}$ as [14] (namely, the probability of "flipping" i-th node is $P(\boldsymbol{\delta}[i] = -1) = \boldsymbol{\alpha}[i]$. So we can optimize the objective of continuous Bernoulli parameters $\boldsymbol{\alpha}$ easily. In particular, we sample flipping vector $\boldsymbol{\delta}$ from $\mathcal{B}(\mathbf{1}, \boldsymbol{\alpha})$ with $\mathbf{p} \sim \mathcal{B}(\mathbf{1}, \boldsymbol{\alpha})$ then $\boldsymbol{\delta} = 2\mathbf{p} - \mathbf{1}$ by the strategy of "reparameterization trick" [39].

Finally, based on the approximated closed form of GCN and the above continuous surrogates, the overall attack objective in Eq. (5) becomes:

$$\mathcal{L}(\boldsymbol{\alpha}) := - \mathop{\mathbb{E}}_{\substack{\mathbf{p} \sim \mathcal{B}(\mathbf{1}, \boldsymbol{\alpha}) \\ \boldsymbol{\delta} = 2\mathbf{p} - \mathbf{1}}} \Big[\frac{1}{N_U} \sum_{i}^{N_U} \tilde{\mathbf{y}}_U[i] \odot \mathbf{y}_U[i] \Big], \tag{12}$$

which is differentiable and tractable now. So we can minimize it with gradient-based optimizer for the optimal probability of 'flipping' $\boldsymbol{\alpha}$.

D. Optimization

We summarize our model LafAK in Alg.1. Given a fixed training set, we first pre-process the matrix $\mathbf{P}$ in Line 1, including solving inverse matrix with time complexity $O(d^3)$. In fact, this inverse matrix can be computed easily since the value of the feature dimension d is usually small. The training procedure of the probability vector $\boldsymbol{\alpha}$ is presented in Line 2-6, which can be optimized by stochastic gradient descent optimizer. Lastly, we generate attacks following the strategy in [14] that if $\boldsymbol{\alpha}[i]$ is one of the largest ϵN_L elements in probability $\boldsymbol{\alpha}$, the i-th label will be flipped. The intuition behind such a strategy is similar to $\epsilon-$greedy strategy [14], which is a greedy heuristic for combinatorial optimization. Furthermore, we discuss the efficiency of the proposed attack strategies. Here we report the time consumption of the proposed attack strategies on average (including pre-processing). The time consumption for Polblogs, Cora, Citeseer, Pubmed are 28.3s, 24.5s, 28.0s, and 45.5s, respectively. As we can see, the proposed model takes less than 50 seconds on average to generate perturbation. These results show that the proposed attack strategies are quite efficient.

Algorithm 1 LafAK: Label-flipping attack model for GCN

Input: Graph with $\mathbf{y}_L$, $\mathbf{A}$, $\mathbf{X}$ and $\mathbf{y}_U$ (or using predictions of GCN $\hat{\mathbf{y}}_U$ as substitution), flip ratio ϵ, smooth coefficient τ and iteration numbers T.
Output: The modified training labels $\mathbf{y}_L^{'}$.
1: $\mathbf{y}_L^{'} = \mathbf{y}_L$
2: Pre-process $\mathbf{P}$ in Eq.(8).
3: **for** $t \in T$ **do**
4: Sample flipping operation by $\mathbf{p} \sim \mathcal{B}(\mathbf{1}, \boldsymbol{\alpha})$ and $\boldsymbol{\delta} = 2\mathbf{p} - \mathbf{1}$;
5: Calculate $\mathcal{L}(\boldsymbol{\alpha})$ using Eq. (12);
6: Update $\boldsymbol{\alpha}$ by $\frac{\partial \mathcal{L}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$;
7: **end for**
8: flip the labels in $\mathbf{y}_L^{'}$ with the largest ϵN_L elements in $\boldsymbol{\alpha}$.

V. Defense Framework against Label-Flipping Attacks

In order to improve the robustness of GCNs to label-flipping attacks, we explore the reason for the vulnerability of GCNs to such attacks, through analyzing the learning process under attacks. Here we plot the training/validation accuracy under clean/attack scenarios respectively in Fig. 4. As we can see, compared with clean scenario, the training accuracy with attacks has almost no changes while the gap

between training and validation accuracy clearly increases. Typically, this indicates that GCN overfits the flipped nodes and memorizes their inherent misinformation, resulting in a poor generalization performance. Thus, preventing GCNs from overfitting flipped nodes is a key issue, and one direct solution is to exploit reliable high-level signals as regularization.

Here we consider community structure, which is a high-level topological property of natural graphs (e.g., communities in a social graph may represent users with common interests/ locations). We assume that communities can guide node classification task as high-level signals. To verify this idea, we compare the community labels learned with the existing method [1] with the output of GCN classification in Fig. 5. We can observe that the community labels of nodes are partly consistent with node classification labels. So the community signals can be valuable for node classification task.

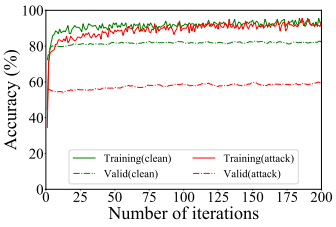


Fig. 4: Mean accuracy on training set (solid lines) and validation set (dashed lines) of Polblogs under clean/attack (flipping ratio $\epsilon = 0.1$) scenarios. As we can see the validation accuracy clearly decreases under attack.

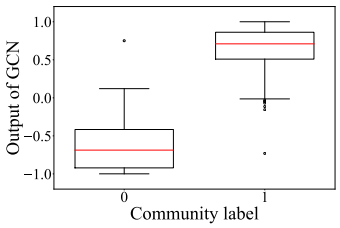


Fig. 5: Community labels vs. GCN output $f_{\boldsymbol{\theta}}$ on Polblogs. For most nodes, their community labels are consistent with $f_{\boldsymbol{\theta}}$, i.e., the nodes with community labels (0 or 1) correspond to GCN output $f_{\boldsymbol{\theta}}$ ($[-1, 0]$ or $[0, +1]$).

Based on these observations, in this paper, we propose a community-preserving self-supervised defense framework for GCNs against label-flipping attacks, alleviating the overfitting to flipped nodes via community signals. Specifically, the proposed defense framework consists two steps:

(1) Learning community labels. Following a common strategy, we first embed the nodes into a low dimensional Euclidean space by existing graph embedding methods [1], [40]. The obtained embeddings can represent abstract structural features of the network and enabling us to apply standard clustering algorithm. Then we cluster the embeddings into K (or more) distinct groups with a standard clustering algorithm, such as K-means, and then use the cluster assignments as pseudo community labels $\mathbf{Y}^c$.

(2) Training GCNs using community-preserving self-supervised task as regularization. With community labels, we train the GCNs with predicting the community label as an auxiliary self-supervised task, which takes the hidden feature matrix at $(L-1)$-th layer $\mathbf{H}^{(L-1)}$ as input and outputs a K-way softmax distribution. This auxiliary task is trained along with the community labels $\mathbf{Y}^c$ as following:

$$L_c(\boldsymbol{\theta}^{(L-1)}, \mathbf{W}^c; \mathbf{A}, \mathbf{X}, \mathbf{Y}^c) = -\sum_{v}^{V_L^c}\sum_{i=1}^{K} \mathbf{Y}_{v,i}^c ln \mathbf{Z}_{v,i}^c, \quad (13)$$

where $\mathbf{Z}_{v,i}^c = \text{Softmax}(\hat{\mathbf{A}}\mathbf{H}^{(L-1)}\mathbf{W}^c)$, V_L^c is the node set for training self-supervised task. The parameters of low level layers $\boldsymbol{\theta}^{(L-1)} = \{\mathbf{W}^{(1)}...\mathbf{W}^{(L-1)}\}$ are shared between the two tasks, while keeping several task-specific output layers $\mathbf{W}^c$ and $\mathbf{W}^{(L)}$. The shared layers will be punished by the community-level information as long as overfit the misleading signals in flipped nodes. Our overall loss L_{MT} during training can be formulated in the following:

$$\begin{aligned} L_{MT}(\mathbf{A}, \mathbf{X}, \mathbf{Y}, \mathbf{Y}^c) &= L(\boldsymbol{\theta}^{(L-1)}, \mathbf{W}^{(L)}; \mathbf{A}, \mathbf{X}, \mathbf{Y}) \\ &+ \lambda_c L_c(\boldsymbol{\theta}^{(L-1)}, \mathbf{W}^c; \mathbf{A}, \mathbf{X}, \mathbf{Y}^c). \end{aligned} \quad (14)$$

When $\lambda_c = 0$, our total loss falls back to the original GCN. In this paper, we set $\lambda_c = 1$ by default. Note that the self-supervised component of the loss does not require the ground-truth training labels $\mathbf{Y}$ as input.

VI. EXPERIMENTS

In this section, we explore the robustness of GCNs to adversarial label-flipping attacks, and evaluate the effectiveness of our proposed attack and defense models on multiple datasets, including citation network (Citeseer, Cora, Pubmed [3]) and social network (Polblogs [2]). We closely follow the experimental setting in previous works [3], [41]. Specifically, we use all node features and 20 labeled nodes per class as the training set and another 500 labeled nodes for validation and early-stopping. The statistics of the datasets are shown in Table I.

TABLE I: Dataset statistics.

Dataset	Nodes	Edges	Classes	Features	Train./Val./Test
Polblogs	1,490	19,025	2	-	40/500/950
Cora	2,708	5,429	7	1,433	140/500/1000
Citeseer	3,327	4,732	6	3,703	120/500/1000
Pubmed	19,717	44,338	3	500	60/500/1000

A. Attack

In this part, we evaluate the effectiveness and transferability of our model LafAK assuming the attacker knows true test labels. Additionally, we also examine LafAK under a more practical scenario without these labels.

Effectiveness of attack model. To verify the effectiveness of our model LafAK, we perform attacks considering various ratios ϵ of flipped nodes in the target training data with $\epsilon \in [0.05, 0.1, 0.15, 0.2]$. We randomly split each dataset following Table I for five times, then run our model for five times under each partition and report the average results of 25 times. Note that our attacks focus on semi-supervised binary classification. For multi-class datasets, we use LafAK to generate attacks for the two classes with the most nodes.

We compare our model LafAK against these baselines: (1) Random-flipping attacks (**Rnd**) which randomly flip the

labels of a same number of nodes. (2) Degree-based flipping attacks (**Deg**) which flip the nodes with the highest degrees. (3) Label-flipping attacks against label propagation (**LPattack**) [14] which is the state-of-the-art attack model for graph-based semi-supervised learning. Besides, to further examine the effectiveness of each component of LafAK, we also consider two variants: (1) $\mathbf{LafAK_{LP}}$: It uses the label propagation to approximately compute the predictions of unlabeled data $\tilde{\mathbf{y}}_h$ in Eq. (10). (2) $\mathbf{LafAK_{MSE}}$: It adopts the continuous mean squared error (MSE) to replace 0-1 error, solving non-differentiable objective while introducing unnecessary loss of attack accuracy. The parameters in all the models are optimized using the attack loss.

The results are presented in Table II. Here, we show the test accuracy under different flip ratios ϵ on four datasets. The major findings from the experimental results are summarized as follows:

- Our model LafAK significantly outperforms Rnd on all metrics, which indicates that GCNs are robust to random label noise but quite vulnerable to well-designed adversarial label-flipping attacks. One can observe that with the increasing flip ratio ϵ, the test performance under LafAK consistently decreases even reaches less than random guessing. (e.g., Polblogs' test accuracy (%) is 46.4 when $\epsilon = 0.20$.) The explanation is that adversarial label-flipping attacks can be specifically targeted to exploit the prediction information of the learning algorithm to search for the most effective flipping operation, significantly degrading performance.
- Our model LafAK achieves better performance than other label-flipping attacks, and we argue that this comes from the fact that they fail to approximate the influence of a label to GCN model precisely: (1) Deg simply identifies the influential nodes based on degrees. (2) LPattack is specifically designed for label propagation rather than GCNs. which are different in information propagation and feature modeling. Compared with Deg and LPattack, our model LafAK is specifically designed for GCN, estimating the influence of flips precisely and efficiently with the approximated closed form of GCN and continuous surrogate of objective.
- Considering the two variants of our model LafAK_{LP} and $\text{LafAK}_{\text{MSE}}$, it is easy to see LafAK achieves better performance than LafAK_{LP} and $\text{LafAK}_{\text{MSE}}$. It indicates that the approximated closed form of GCN and continuous surrogates of discrete components in our model are reasonable and effective.

Transferability of attacks. Can attacks learned for one model generalize to other models? In practical attack scenario, the attacker dose not know what model the system is using exactly. In this part, we explore the transferability of our model LafAK. We find the optimal perturbations based on GCN, and then apply them to the state-of-the-art GCNs including graph attention networks (GAT) [26] and graph isomorphism network (GIN) [42]. For each model, we use the default settings for the hyper-parameters. The results presented in Fig. 6 show that the perturbations generated by LafAK can successfully transfer to different graph neural models such as GAT and GIN, even achieving better attack performance on some datasets.

Analysis of the number of labeled nodes per class. To comprehensively evaluate our model LafAK, we gradually increase the number of labeled nodes per class (L/C) for training (i.e., 20, 40, 60, 80) and study the performance. The attack results are reported in Fig. 7. As we can see, for each dataset, the curves of 20, 40, 60, 80 L/C show the similar declining trend with the increasing flip ratio $\epsilon \in \{0.00, 0.05, 0.10, 0.15, 0.20\}$. We notice that models with larger L/C tend to have better performance under both clean data ($\epsilon = 0.00$) and poisoned data. The possible reasons are: (1) With the increase of L/C, GCN can propagate the labels to the entire data graph more sufficiently, which improves the overall classification confidence and reduces the number of nodes close to the classification boundary. (2) Given more training labeled data, the overfitting to flipped labels can be better alleviated.

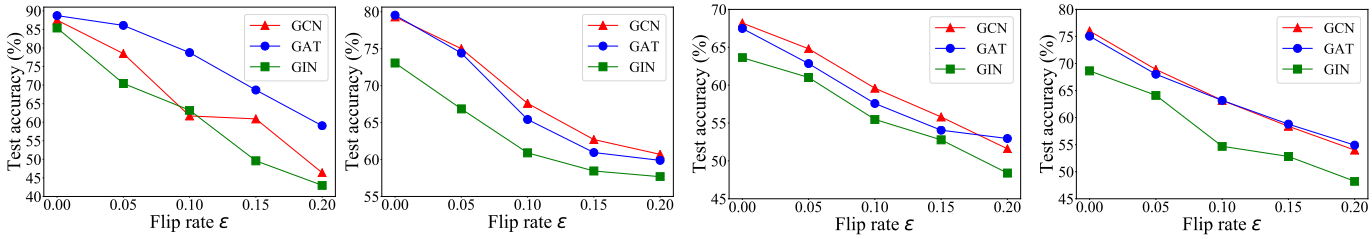
Evaluating attack model without true test labels. In this part, we consider the case that attacker has no access to true test labels $\mathbf{y}_U$, which is common in practical attack scenarios. The attacker uses the predictions of GCN on test nodes $\hat{\mathbf{y}}_U$ as a substitution of true test labels. Here, we take Polblogs and Citeseer as examples. As shown in Fig. 8, we can observe that there is no significant decrease for our model LafAK without true labels of test data. This experiment provides a valuable implication that hiding the ground-truth labels cannot protect the GCN models, because the attackers can alternatively use the estimated labels $\mathbf{y}_U$.

B. Defense

In this part, we evaluate the effectiveness and generalization ability of our defense framework on multiple datasets. We compare against the following two baselines: (1) **RGCN**: A robust GCN model, designed against adversarial topology/feature attacks. It adopts Gaussian distributions as the hidden representations of nodes and uses a variance-based attention mechanism to remedy the propagation of adversarial attacks. (2) **Sanitation**: A common defense technique against label-flipping attacks. It uses k-Nearest-Neighbours (k-NN) based methods to identify the training points that may have had their labels corrupted and then removes/relabels them. The above baselines are initialized with the parameters suggested by their papers and we also further carefully tune the parameters to get optimal performance. For our model, we train our community-preserving based defense framework with the same hyper-parameters as GCN [3]. In addition, the number of labeled nodes per class for training community-preserving self-supervised task (L/C_c) are searched in $\{1, 2, 4, 6, 8\}$. For all methods, we run our model for five times under each train./valid./test partition following Table I and report the average results.

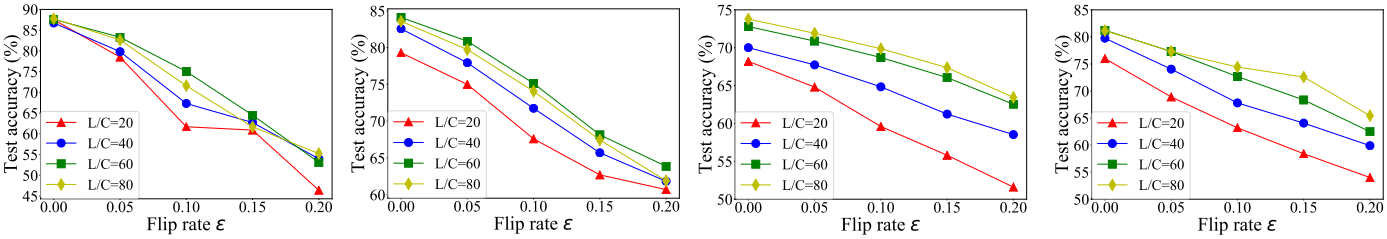
TABLE II: Results of attack models under different flip ratios ϵ. A lower test accuracy(%) indicates better performance. We also report the performance drop rate w.r.t. the clean model. A larger drop rate indicates better attack performance.

Dataset	ϵ	Rnd	Deg	LPattack	LafAK$_{LP}$	LafAK$_{MSE}$	LafAK
Polblogs	0.05	85.2 (-2.63%)	85.3 (-2.51%)	81.5 (-6.86%)	78.8 (-9.94%)	80.1 (-8.46%)	**78.5 (-10.29%)**
	0.10	85.3 (-2.51%)	81.6 (-6.74%)	67.8 (-22.51%)	66.3 (-24.23%)	72.2 (-17.49%)	**61.7 (-29.49%)**
	0.15	85.0 (-2.86%)	73.0 (-16.57%)	**59.3 (-32.23%)**	59.7 (-31.77%)	64.6 (-26.17%)	60.9 (-30.40%)
	0.20	82.5 (-5.71%)	57.0 (-34.86%)	51.5 (-41.14%)	51.7 (-40.91%)	48.6 (-44.46%)	**46.4 (-46.97%)**
Cora	0.05	78.7 (-0.76%)	78.6 (-0.88%)	75.8 (-4.41%)	75.3 (-5.04%)	**74.5 (-6.05%)**	75.0 (-5.42%)
	0.10	78.7 (-0.76%)	76.7 (-3.28%)	71.1 (-10.34%)	69.4 (-12.48%)	68.2 (-14.00%)	**67.6 (-14.75%)**
	0.15	76.3 (-3.78%)	73.9 (-6.81%)	66.1 (-16.65%)	66.3 (-16.39%)	62.9 (-20.68%)	**62.7 (-20.93%)**
	0.20	75.5 (-4.79%)	71.8 (-9.46%)	63.1 (-20.43%)	62.3 (-21.44%)	60.7 (-23.46%)	**60.7 (-23.46%)**
Citeseer	0.05	66.8 (-2.05%)	66.3 (-2.79%)	66.4 (-2.64%)	65.3 (-4.25%)	65.6 (-3.81%)	**64.8 (-4.99%)**
	0.10	66.8 (-2.05%)	63.0 (-7.62%)	63.4 (-7.04%)	62.8 (-7.92%)	62.7 (-8.06%)	**59.6 (-12.61%)**
	0.15	66.1 (-3.08%)	59.8 (-12.32%)	60.2 (-11.73%)	58.3 (-14.52%)	59.7 (-12.46%)	**55.8 (-18.18%)**
	0.20	58.7 (-13.93%)	58.5 (-14.22%)	58.4 (-14.37%)	57.1 (-16.28%)	55.1 (-19.21%)	**51.6 (-24.34%)**
Pubmed	0.05	76.6 (–0.79%)	73.4 (-3.42%)	71.4 (-6.05%)	71.9 (-5.39%)	70.7 (-6.97%)	**68.9 (-9.34%)**
	0.10	75.9 (-0.13%)	68.9 (-9.34%)	65.0 (-14.47%)	65.5 (-13.85%)	64.1 (-15.66%)	**63.2 (-16.84%)**
	0.15	71.5 (-5.92%)	67.4 (-11.32%)	**57.6 (-24.23%)**	58.1 (-23.49%)	59.1 (-22.24%)	58.4 (-23.16%)
	0.20	69.5 (-8.55%)	62.0 (-18.42%)	54.4 (-28.46%)	53.9 (-29.02%)	**53.5 (-29.61%)**	54.0 (-28.95%)



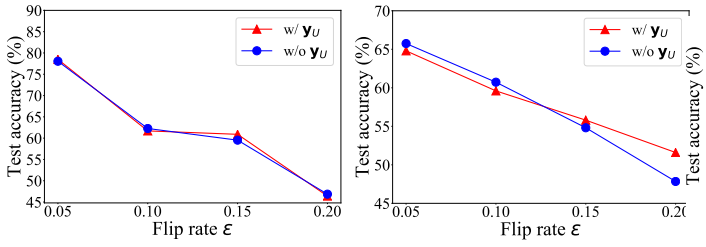
(a) Polblogs. (b) Cora. (c) Citeseer. (d) Pubmed.

Fig. 6: Transferability of our attack model LafAK.



(a) Polblogs. (b) Cora. (c) Citeseer. (d) Pubmed.

Fig. 7: Analysis of label rate for our attack model LafAK. L/C denotes the number of labeled nodes per class for training set.



(a) Polblogs. (b) Citeseer.

Fig. 8: Attack performance without true test labels.

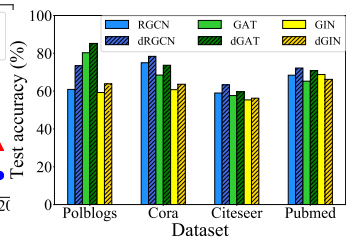


Fig. 9: Generalization ability of our defense.

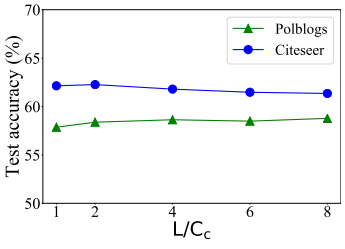


Fig. 10: Analysis of L/C$_c$ for our defense.

Effectiveness of defense framework. In the following, we evaluate the effectiveness of the proposed defense framework under different flipping ratios ϵ on multiple datasets. The overall results are presented in Table III. We have the following observations:

- Our model significantly improves the robustness of GCN model on all metrics, which indicates that the GCN can greatly benefit from community-preserving self-supervised task, alleviating the overfitting to misinformation in flipped nodes.
- The existing method RGCN and Sanitization fail to defense our attack. The reasons are: (1) RGCN is mainly focused on feature/topology attacks, rather than label-flipping attacks. (2) Sanitization relabels flipped nodes by exploiting a large volume of training data, and may fail in semi-supervised learning.
- Moreover, compared with GCN, the improvement of our model is relatively small on the Pubmed dataset. This could be due to the poor intrinsic community structure of Pubmed. It implies the effectiveness of community structure for improving robustness.

TABLE III: Results of defense framework. A higher test accuracy(%) indicates better performance.

Dataset	ϵ	GCN	RGCN	Sanitization	Ours$_{\text{GCN}}$
Polblogs	0.05	78.5	77.9	78.5	**83.9**
	0.10	61.7	60.6	61.5	**69.7**
	0.15	60.9	61.6	58.4	**62.8**
	0.20	46.4	47.6	46.3	**56.1**
Cora	0.05	78.6	79.2	78.6	**79.7**
	0.10	73.7	75.0	73.7	**77.3**
	0.15	66.4	67.4	66.4	**71.9**
	0.20	61.2	61.9	61.2	**64.2**
Citeseer	0.05	64.5	64.2	64.5	**69.1**
	0.10	58.7	59.0	58.7	**62.6**
	0.15	56.2	57.3	56.2	**57.4**
	0.20	55.4	55.9	55.4	**57.2**
Pubmed	0.05	74.6	74.3	73.3	**76.0**
	0.10	69.0	68.4	69.0	**69.4**
	0.15	61.8	62.6	61.8	**63.9**
	0.20	58.9	57.4	**58.9**	57.0

Generalization ability of defense framework. In addition, to demonstrate that our proposed defense framework is generic to other GCN models, we also generalize our defense framework to RGCN, GAT and GIN, obtaining dRGCN, dGAT and dGIN. We evaluate the robustness of it with fixed flipping ratio $\epsilon = 0.10$. The results are presented in Fig. 9. As we can see, our defense framework can successfully generalize to existing GCNs, improving the robustness of GCNs to label-flipping attacks.

Analysis of L/C$_c$. We test the impact of the number of labeled nodes per class for training community-preserving self-supervised learning (L/C$_c$). Here, we take the datasets Polblogs and Citeseer as examples. As shown in Fig. 10, we can observe that, basically, our framework is stable when L/C$_c$ is within the range from $\{1, 2, 4, 6, 8\}$.

VII. CONCLUSION

In this paper, we introduce the first study on the adversarial label-flipping attacks against GNNs. We propose an effective label-flipping attack model based on approximated closed form of GNN and replacement of non-differentiable objective, overcoming the challenges of inherent bi-level optimization problem and non-differentiability. Our extensive experiments show that by only flipping an unnoticeable fraction of training labels, the overall classification performance of GNNs can be dramatically impaired. Additionally, we propose an effective defense framework based on community-preserving self-supervised learning to improve the robustness of GNNs, by alleviating the overfitting to flipped nodes. Experiments on various datasets show the effectiveness of our defense framework. In future work, we would like to study clean-label poisoning attacks on graph data, inserting innocuous-looking (and "correctly" labeled) poison training nodes but causing the classifier to perform poorly.

ACKNOWLEDGMENT

This work is supported in part by the National Natural Science Foundation of China (No. 61772082, 61806020, 61702296), the National Key Research and Development Program of China (2018YFB1402600), and Alibaba Group under its Alibaba Innovative Research (AIR) programme.

REFERENCES

[1] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, 2014, pp. 701–710.

[2] L. A. Adamic and N. S. Glance, "The political blogosphere and the 2004 U.S. election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery, LinkKDD 2005, Chicago, Illinois, USA, August 21-25, 2005*. ACM, 2005, pp. 36–43.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[4] D. Eswaran, S. Günnemann, C. Faloutsos, D. Makhija, and M. Kumar, "Zoobp: Belief propagation for heterogeneous networks," *Proc. VLDB Endow.*, vol. 10, no. 5, pp. 625–636, 2017.

[5] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, 2017, pp. 1024–1034.

[6] L. Sun, Y. Dou, C. Yang, J. Wang, P. S. Yu, and B. Li, "Adversarial attack and defense on graph data: A survey," *arXiv preprint arXiv:1812.10528*, 2018.

[7] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2018, pp. 2847–2856.

[8] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song, "Adversarial attack on graph structured data," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1123–1132.

[9] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[10] K. Xu, H. Chen, S. Liu, P. Chen, T. Weng, M. Hong, and X. Lin, "Topology attack and defense for graph neural networks: An optimization perspective," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2019, pp. 3961–3967.

[11] T. Takahashi, "Indirect adversarial attacks via poisoning neighbors for graph convolutional networks," in *2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, December 9-12, 2019*. IEEE, 2019, pp. 1395–1400.

[12] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012.

[13] L. Muñoz González, B. Biggio, A. Demontis, A. Paudice, V. Wongrassamee, E. C. Lupu, and F. Roli, "Towards poisoning of deep learning algorithms with back-gradient optimization," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17. Association for Computing Machinery, 2017, p. 27–38.

[14] X. Liu, S. Si, J. Zhu, Y. Li, and C. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, 2019, pp. 9777–9787.

[15] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," Tech. Rep., 2002.

[16] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[17] M. Zhao, B. An, W. Gao, and T. Zhang, "Efficient label contamination attacks against black-box learning models," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 2017, pp. 3945–3951.

[18] P. W. Koh, J. Steinhardt, and P. Liang, "Stronger data poisoning attacks break data sanitization defenses," *CoRR*, vol. abs/1811.00741, 2018.

[19] A. Paudice, L. Muñoz-González, and E. C. Lupu.

[20] J. Xie, R. B. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, ser. JMLR Workshop and Conference Proceedings, vol. 48. JMLR.org, 2016, pp. 478–487.

[21] C. Wang, S. Pan, R. Hu, G. Long, J. Jiang, and C. Zhang, "Attributed graph clustering: A deep attentional embedding approach," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. ijcai.org, 2019, pp. 3670–3676.

[22] B. Rozemberczki, R. Davies, R. Sarkar, and C. A. Sutton, "GEMSEC: graph embedding with self clustering," in *ASONAM '19: International Conference on Advances in Social Networks Analysis and Mining, Vancouver, British Columbia, Canada, 27-30 August, 2019*. ACM, 2019, pp. 65–72.

[23] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[24] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 5171–5181.

[25] Z. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 4805–4815.

[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[27] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, vol. 80. PMLR, 2018, pp. 5449–5458.

[28] F. Wu, A. H. S. Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 6861–6871.

[29] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples for graph data: Deep insights into attack and defense," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 4816–4823.

[30] Y. Ma, S. Wang, L. Wu, and J. Tang, "Attacking graph convolutional networks via rewiring," *CoRR*, vol. abs/1906.03750, 2019.

[31] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. ACM, 2019, pp. 2023–2040.

[32] H. Chang, Y. Rong, T. Xu, W. Huang, H. Zhang, P. Cui, W. Zhu, and J. Huang, "A restricted black-box adversarial framework towards attacking graph embedding models." AAAI, 2020.

[33] N. Entezari, S. A. Al-Sayouri, A. Darvishzadeh, and E. E. Papalexakis, "All you need is low (rank): Defending against adversarial attacks on graphs," in *WSDM '20: The Thirteenth ACM International Conference on Web Search and Data Mining, Houston, TX, USA, February 3-7, 2020*. ACM, 2020, pp. 169–177.

[34] J. Li, H. Zhang, Z. Han, Y. Rong, H. Cheng, and J. Huang, "Adversarial attack on community detection by hiding individuals," in *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*. ACM / IW3C2, 2020, pp. 917–927.

[35] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 2233–2241.

[36] R. Taheri, R. Javidan, M. Shojafar, Z. Pooranian, A. Miri, and M. Conti, "On defending against label flipping attacks on malware detection systems," *CoRR*, vol. abs/1908.04473, 2019.

[37] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, ser. Proceedings of Machine Learning Research, vol. 97. PMLR, 2019, pp. 695–704.

[38] E. Yang, A. C. Lozano, and P. Ravikumar, "Closed-form estimators for high-dimensional generalized linear models," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 2015, pp. 586–594.

[39] M. Figurnov, S. Mohamed, and A. Mnih, "Implicit reparameterization gradients," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018, pp. 439–450.

[40] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding." in *IJCAI*, 2018, pp. 2609–2615.

[41] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019, pp. 1399–1407.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.