AN-GCN

An Anonymous Graph Convolutional Network Against Edge-Perturbing Attacks

Liu, Ao; Li, Beibei; Li, Tao; Zhou, Pan; Wang, Rui

DOI
10.1109/TNNLS.2022.3172296

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Neural Networks and Learning Systems

AN-GCN: An Anonymous Graph Convolutional Network Against Edge-Perturbing Attacks

Ao Liu, *Student Member, IEEE*, Beibei Li, *Member, IEEE*, Tao Li, Pan Zhou, *Senior Member, IEEE*, and Rui Wang

***Abstract*— Recent studies have revealed the vulnerability of graph convolutional networks (GCNs) to edge-perturbing attacks, such as maliciously inserting or deleting graph edges. However, theoretical proof of such vulnerability remains a big challenge, and effective defense schemes are still open issues. In this article, we first generalize the formulation of edge-perturbing attacks and strictly prove the vulnerability of GCNs to such attacks in node classification tasks. Following this, an anonymous GCN, named AN-GCN, is proposed to defend against edge-perturbing attacks. In particular, we present a node localization theorem to demonstrate how GCNs locate nodes during their training phase. In addition, we design a staggered Gaussian noise-based node position generator and a spectral graph convolution-based discriminator (in detecting the generated node positions). Furthermore, we provide an optimization method for the designed generator and discriminator. It is demonstrated that the AN-GCN is secure against edge-perturbing attacks in node classification tasks, as AN-GCN is developed to classify nodes without the edge information (making it impossible for attackers to perturb edges anymore). Extensive evaluations verify the effectiveness of the general edge-perturbing attack (G-EPA) model in manipulating the classification results of the target nodes. More importantly, the proposed AN-GCN can achieve 82.7% in node classification accuracy without the edge-reading permission, which outperforms the state-of-the-art GCN.**

***Index Terms*— Anonymous classification, general attack model, graph adversarial attacks, graph convolutional network (GCN).**

I. Introduction

GRAPH is the core for many high-impact applications ranging from the analysis of social networks, over gene interaction networks, to interlinked document collections. In many tasks related to the graph structure, node classification, predicting labels of unknown nodes based on a small number of labeled known nodes, is always a hot issue. However, vulnerabilities of node classification have been gradually explored. Only slight deliberate perturbations of nodes' features or graph structure can lead to completely wrong predictions, which leads to a critical issue in many application areas, including those where adversarial perturbations can undermine public trust [1], interfere with human decision making [2], and affect human health and livelihoods [3]. As a representative example, in the fake news detection on a social network [4], adversaries have a strong incentive to fool the system by the concealed perturbation to avoid detection. In this context, a perturbation could mean modification of the graph structure (inserting/deleting edges in the social graph).

Perturbing feasible edges ergodically are widely exploited in constructing edge-perturbing attacks. Through the message passing [5] driven by end-to-end training, perturbations can pass to the target along the edges. To obtain the optimal perturbations sets, the attackers often design constraints to restrict the perturbations while perturbing ergodically. The most significant constraints are attack success rate [6] and the concealment of the selected perturbations [7]. Due to the discrete characteristic of the graph, the attacker can always obtain an optimal set of perturbations, which satisfy the abovementioned constraints, thus further manipulating the classification result of the target node. In this way, the defender always lags behind the attacker [8], so we should assume that the attackers can always destroy them in the future [9].

Existing defense schemes against edge-perturbing attacks fall into the following two categories.

1) *Adversarial Training (AT)-Based Schemes:* AT aims at adapting the model to all possible potential perturbations. Through optimizing two neural networks alternatively through two-player game [10], AT can increase the robustness of the model to perturbations [11].
2) *Robust Aggregation-Based Schemes:* Robust aggregation aims to design a robust aggregation function [12] of graph convolutional networks (GCNs), and it enables GCNs to identify and filter the potential perturbations [13].

However, with the underlying assumption that all edges are perturbable, a malicious adversarial graph may introduce a big challenge to the robustness of GCNs, which can compromise the abovementioned two types of schemes (in Section VI-A, we show that the state-of-the-art defense scheme [14] is vulnerable to the edge-perturbing attacks, 72.8% nodes under protection are misclassified to the target categories). In particular, these weaknesses include the following.

Manuscript received 24 June 2021; revised 26 October 2021 and 12 January 2022; accepted 22 April 2022. Date of publication 13 May 2022; date of current version 5 January 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1805400; in part by the National Natural Science Foundation of China under Grant U19A2068, Grant 62032002, Grant 62002248, and Grant 61972448; in part by the Sichuan Science and Technology Program under Grant 2022YFG0193; and in part by the Fundamental Research Funds for the Central Universities. *(Corresponding author: Beibei Li.)*

Ao Liu, Beibei Li, and Tao Li are with the School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China (e-mail: liuao@stu.scu.edu.cn; libeibei@scu.edu.cn; litao@scu.edu.cn).

Pan Zhou is with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: panzhou@hust.edu.cn).

Rui Wang is with the Department of Intelligent Systems, Delft University of Technology, 2628 XE Delft, The Netherlands (e-mail: R.Wang-8@tudelft.nl).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3172296.

Digital Object Identifier 10.1109/TNNLS.2022.3172296

1) Though AT is very effective in defending against adversarial attacks in computer vision and natural language processing, it is not suitable for graphs, because there is no boundary for the malicious perturbations on graphs that hold more complex structures than image or text data. Although some studies [15], [16] quantified the upper bound of the number of perturbations, it is still nonintuitive. Compared with computer vision, the perturbations on images are limited to cannot be observed by naked eyes, which can significantly reduce the range of feasible perturbation. This leads to a strong constraint to the AT in computer vision. However, GCNs suffer from the over-smoothing issues [17], which keeps the properties of adjacent nodes to be consistent. In other words, it is difficult to adapt GCNs to all feasible perturbations while not compromising the model's utility, attributed to the nonintuitive upper bound of perturbations.
2) Robust aggregation quantifies each step of message passing. A robust aggregation function is a necessary condition of the benign message passing. However, high-epochs end-to-end training will bring huge data flow to the model. As the perturbations are designed maliciously and the robust aggregation function is built upon the manual rules, well-designed perturbations can damage the preset rules of the aggregation function with the increase in the epoch.

In this article, by deconstructing the message passing driven by GCNs, we first demonstrate the vulnerability of the preconditions of GCNs, i.e., the edge-reading permission to GCNs is prone to be abused by adversarial attackers. We further withdraw the edge-reading permission of GCNs to defend against such edge-perturbing attacks. The motivation of our work comes from the inherent observations of the phenomenon that attackers can always rewire the graph topology by utilizing the edge-reading permission to the graph database in practical scenarios, thus misleading the node classification results by the GCNs. As an illustrational example, in financial social networks, users and their social connections can be formed as a graph, and the credit of users in the graph can be predicted by GCNs [18]. For ensuring the prediction to be accurate, the graph database must open its access permission to the GCN model. This leaves a big chance for attackers to modify the relationship between users and further significantly improve the credit of the target user maliciously. For another example, in anomaly detection for cloud components, relationships between system components are usually organized by subjective metrics, which is also the necessary input for GCNs. As organized modeling metrics are exposed [19], [20], the attackers are able to calculate the cloud components' topology themselves maliciously and further modify metrics between components, thus successfully hiding the anomalous components.

The aforementioned examples once again reveal the vulnerability of GCNs. To design an effective defense scheme against adversarial attacks, in this article, we demonstrate the preconditions that cause GCNs vulnerable. In particular, we unify the existing manual perturbation-based edge-perturbing attacks into an automatic general edge-perturbing attack (G-EPA) model. G-EPA, built upon a surrogate GCN, can reduce a complete graph to an optimal attack graph restricted by both attack concealment and success rate. It may compromise many state-of-the-art defense schemes [14]. As G-EPA is a unified representation of edge-perturbing attacks, we demonstrate the vulnerability of GCNs through the mathematical formula; i.e., GCNs are vulnerable if they directly take edges as inputs.

Following this, we propose an anonymous GCN (AN-GCN) to withdraw the edge-reading permission of the GCNs. AN-GCN eliminates the opportunity for attackers to contact edge information, which is effective for defending against edge-perturbing attacks in realistic scenarios. If the attacker wants to modify the input data of a graph-learning system protected by a well-trained AN-GCN, they will lose the ability on locating the specific target node, because the only node features are stored in the graph database. Corresponding to the abovementioned illustrational examples, in financial social networks, AN-GCN can handle relationships between all users anonymously, so as to lose the possibility of constructing edge-perturbing attacks. In the anomaly detection for cloud components, even if the attacker calculates the metrics between components according to the reported methods, the anonymity of AN-GCN itself will refuse to accept any edges as the inputs. This invalidates all potential malicious perturbations. In particular, first, we state a node localization theorem, which formulates how GCNs locate specific nodes during its training phase. For the proof, by regarding the feature change of each node in the training phase as the independent signal vibration, we map the signals of all nodes to the Fourier domain. Thus, we unify the nodes' signals to the same coordinate system. Finally, we build the node signal model to figure out how GCNs deconstruct the initial inputs to keep the node fixed in its position during message passing driven by GCN, thus proving the stated node localization theorem. Benefiting from the proposed theorem, we can directly generate the nodes' positions by a generator (a fully connected network) to replace the corresponding part in GCN. To ensure the effectiveness of the generation of the node position, we improve the pioneering spectral GCN [21] to a discriminator, which tries to detect nodes with the generated positions. As the discriminator is a self-contained GCN, it can classify nodes with generated positions accurately once the abovementioned two-player game is well played.

In our proposed AN-GCN, the positions of nodes are generated from the noise randomly, while ensuring the high-accuracy classification; that is, nodes are classified while keeping anonymous to their positions. As the edge decides the node position, the anonymity of the position eliminates the necessity of edge reading for AN-GCN, thus minimizing the possibility of modifying edges. This effectively addresses the vulnerability of GCNs. Our main contributions include the following.

1) We propose an AN-GCN without the necessity of edge reading while maintaining the utility of classification. This ensures that neither role (including attackers) can read the edge information, so as to essentially defend against edge-perturbing attacks.

2) We unify the existing edge-perturbing attack construction methods as a general attack model and further demonstrate the preconditions for the vulnerability of GCNs; i.e., the edge-reading permission to GCNs is prone to be abused by adversarial attackers.
3) We state a node localization theorem to formulate how GCNs locate nodes during the training phase, which provides a theoretical basis for the proposed AN-GCN.

The rest of this article is organized as follows. Section II reviews the state-of-the-art literature. In Section III, for demonstrating the preconditions of the GCNs vulnerability, we introduce the G-EPA model. In Section IV, we prove the node localization theorem. In Section V, we elaborate on the proposed AN-GCN, including the generator, discriminator, and training method. In Section VI, we evaluate the correctness of the stated theorem, the effectiveness of the introduced G-EPA model, and the anonymity of the proposed AN-GCN. Section VII concludes this article.

II. Related Work

A. Graph Attacks Against GCNs

Dai *et al.* [15] and Zügner *et al.* [16] first proposed adversarial attacks on graph structures, after which a large number of graph attack methods were proposed. Specific to the task of node prediction, in 2019, Aleksandar *et al.* [22] provided the first adversarial vulnerability analysis on the widely used family of methods based on random walks and derived efficient adversarial perturbations that poison the network structure. Chang *et al.* [23] attacked various kinds of graph embedding models with black-box-driven method. Wang and Gong [24] proposed a threat model to characterize the attack surface of a collective classification method, targeting on adversarial collective classification. Basically, all attack types are based on the perturbed graph structure targeted by this article. Ma *et al.* [7] studied the black-box attacks on graph neural networks (GNNs) under a novel and realistic constraint: attackers have access to only a subset of nodes in the network. In 2021, Xi *et al.* [25] proposed the first backdoor attack on GNNs, which significantly increases the misclassification rate of common GNNs on real-world data.

B. Defenses for GCNs

Jin *et al.* [26] used the new operator in replacement of the classical Laplacian to construct an architecture with improved spectral robustness, expressivity, and interpretability. Zügner and Günnemann [27] proposed the first method for certifiable (non-)robustness of GCNs with respect to perturbations of the node attributes. Zhu *et al.* [28] proposed robust GCN (RGCN) to automatically absorb the effects of adversarial changes in the variances of the Gaussian distributions. Some defense methods use the generation to enhance the robustness of the model. Deng *et al.* [29] presented batch virtual AT (BVAT), a novel regularization method for GCNs. By feeding the model with perturbing embeddings, the robustness of the model is enhanced. But, this method trains a full-stack robust model for the encoder and decoder at the same time without discussing the essence of the vulnerability of GCN. Feng *et al.* [30] proposed the graph adversarial training (GraphAT), which considers the impact from connected examples when learning to construct and resist perturbations. They also introduce an adversarial attack on the standard classification to the graph. Tang *et al.* [31] investigated a novel problem of improving the robustness of GNNs against poisoning attacks by exploring a clean graph and created supervised knowledge to train the ability to detect adversarial edges, so that the robustness of GNNs is elevated. In 2021, Liao *et al.* [32] proposed a framework to locally filter out predetermined sensitive attributes via AT with the total variation and the Wasserstein distance, further provided a robust defense across various graph structures and tasks. To cope with the scarcity of training data, they proposed an adversarial contrastive learning method to train GCNs in a conditional adversarial manner by leveraging high-level graph representation. But, from a certain point of view, they still used the method based on node perturbation for AT. This method is essentially a kind of "perturbation" learning and uses AT to adapt the model to various custom perturbations.

III. Vulnerability of GCNs

In this section, we introduce a general attack model called G-EPA and prove the preconditions of GCNs' vulnerability. In particular, in Section III-A, we reformulate the encoding and decoding in the GCN. In Section III-B, we formulate the general attack as G-EPA. In Section III-C, we demonstrate the vulnerability of GCNs. In Section III-D, we give case studies to elaborate on how to attack a specific model.

A. Graph Convolution Encoding, Decoding, and Training

Let $\mathcal{G} = (f, \mathcal{E})$ be a graph, where $f = (f(1), \ldots, f(N))^\top$ is a set of features of N nodes, $f(i)$ is the feature vector of node i, and $\mathcal{E}$ is the set of edges that also decides the positions of all nodes. The adjacent matrix $A \in \mathbb{R}^{N\times N}$ and degree matrix $D \in \mathbb{R}^{N\times N}$ can be calculated according to $\mathcal{E}$. The laplacian matrix Δ of $\mathcal{G}$ can be obtained according to D and A. The one-hot encoding of node categories on $\mathcal{G}$ is $\mathcal{Y}$. As A is calculated by $\mathcal{E}$, the graph structure (simultaneously, node positions) can be represented by A. We reformulate the training phase of the model by the following steps.

Encoding: Encoding is a node feature aggregation [33] according to the topology of $\mathcal{E}$. The graph convolution model maps f into the embedding space $f^e = (f^e(1), \ldots, f^e(N))^\top$ through trainable parameter θ^E, while $f^e(i)$ denotes the embedding feature vector of node i. Let $\mathrm{ENC}(\cdot)$ denote the encoder, and the encoding process is denoted as

$$f^e = \mathrm{ENC}\big(f; A, \theta^E\big). \tag{1}$$

Encoding can represent the discrete graph structure into a continuous embedded space [34], which can learn to represent graph nodes, edges, or subgraphs in low-dimensional vectors [35]. The encoded graph can be intuitively observed, according to the existing visualization method [36].

Decoding: Then, the model decodes f^e to the one-hot label space $\mathcal{L} \in \mathbb{R}^{n_c\times 1}$ through the decoder $\mathrm{DEC}(\cdot)$ (in general, it is GCN layers) with the trainable parameter θ^D, where n_c is the number of categories. The output of the decoder provides a gradient descent direction for GCN model training (usually,

the distinction between encoder and decoder is flexible; e.g., in multilayer GCN, the first few layers can be regarded as encoders, and the remaining layers can be regarded as decoders). Almost all high-performance GNNs models (e.g., graph sample and aggregate framework (GraphSAGE) [12], graph attention network (GAT) [37], and spectral-GCN [38]) set the encoder and decoder to the same-design layer; that is, both encoding and decoding involve the aggregation of graph structures $\mathcal{E}$, so the decoder parameters still contain $\mathcal{E}$. The decoding process is denoted as

$$\mathcal{Y} = \mathrm{DEC}\big(f^e; A, \theta^E\big). \tag{2}$$

The decoder calculates output according to A and f^e.

Training: According to the labeled nodes, the parameters of encoder and decoder are trained by end-to-end training, thus realizing the accurate node prediction task. In the training phase, the regularization $\mathrm{Reg}(\cdot)$ ($l1-$norm [39]) prevents the model from overfitting. So, the training phase is

$$\arg\min_{\theta^{\mathrm{GCN}}} \sum \Big\{ \mathrm{Loss}[\mathcal{Y}, \mathrm{DEC}\big(\mathrm{ENC}\big(f; A, \theta^E\big); A, \theta^D\big)\big] + \mathrm{Reg}\big(\theta^{\mathrm{GCN}}\big)\Big\} \tag{3}$$

where $\mathrm{Loss}(\cdot)$ is the loss function, and θ^{GCN} is the total parameter set, i.e., θ^E and θ^D. Equation (3) is the unified form of the existing training method of GCNs, and it is computationally universal (GNNs with sufficient parameters are always computationally universal [40]). The encoding and decoding processes $\mathrm{DEC}(\mathrm{ENC}(\cdot))$ are denoted as $\mathbf{GCN}(\cdot)$.

B. General Edge-Perturbing Attacks

Manual constraints are widely employed in constructing exist edge-perturbing attacks. In this part, we propose G-EPA, which integrates automatic constraints, thus realizing the concealed attack by automatically traversing all perturbations through end-to-end training. As the current edge-perturbing attack is also targeted on obtaining the best perturbation scheme, G-EPA can unify the existing edge-perturbing attacks; thus, it can be a general attack model.

1) Aim of the Attacks: Given a target graph $\mathcal{G}_{\mathcal{T}}$ and the target GCN, which is well trained on $\mathcal{G}_{\mathcal{T}}$, we denote the parameter of it as $\theta^G_{\mathcal{T}}$. The attacker perturbs edges and misleads the classification result of the target node changed by traversing the perturbations on $\mathcal{G}_{\mathcal{T}}$, to further obtain $\mathcal{G}_{\mathrm{victim}} = (f, \mathcal{E}_{\mathrm{victim}})$. Therefore, the attack is to find a modified graph adjacency matrix $\hat{A}$ corresponding to $\mathcal{E}_{\mathrm{victim}}$; let

$$\hat{\mathcal{Y}}_\kappa = \mathbf{GCN}\big(f; \hat{A}, \theta^G_{\mathcal{T}}\big) \tag{4}$$

where κ denotes the node of modified category, and $\hat{\mathcal{Y}}_\kappa$ denotes the distribution of the modified node category set in $\mathcal{L}$. That is, in $\hat{\mathcal{Y}}_\kappa$, the category of κ is manipulated as the target category; meanwhile, making other nodes keep original categories (whether single target and multitarget). From the perspective of model optimization, it could be either evasion attacks [7] or poisoning attacks [41]. Once $\hat{A}$ is found, $\mathcal{E}_{\mathrm{victim}}$ can be calculated by $\hat{A}$, to realize the effective attack.

The aforementioned edge-perturbing attack process can be formulated as an $\hat{A}$-finding game

$$\hat{A} = \mathbf{ATTACK}\big(\theta^E_{\mathcal{T}}, \theta^D_{\mathcal{T}}, \mathcal{E}\big) \tag{5}$$

where $\theta^E_{\mathcal{T}}$ and $\theta^D_{\mathcal{T}}$ are the parameters of the encoder and decoder of the target GCN, respectively. In other words, the attacker perturbs $\theta^E_{\mathcal{T}}$, $\theta^D_{\mathcal{T}}$, and $\mathcal{E}$ and, thus, finds $\hat{A}$ to realize the attack.

2) $\hat{A}$ Obtaining Method: In this part, we give the method of obtaining $\hat{A}$. Equation (4) can be expressed as follows:

$$\hat{\mathcal{Y}}_\kappa = \mathrm{DEC}\big[\mathrm{ENC}\big(f; \hat{A}, \theta^E_{\mathcal{T}}\big); \hat{A}, \theta^D_{\mathcal{T}}\big]. \tag{6}$$

Once the attacker successfully obtains $\hat{A}$ in (6), the attack will be completed; furthermore, it takes advantage of the vulnerability of GCN.

Next, we give the automatic scheme for obtaining perturbations, i.e., $\hat{A}$. Before giving the specific steps, we first give the preliminary.

1) Connecting target node κ to a complete graph $\bar{\mathcal{G}}$, where each node has self-loops, the diagonal matrix and adjacency matrix of $\bar{\mathcal{G}}$ are $\bar{D}$ and $\bar{A}$, respectively. The node feature sets of $\mathcal{G}$ and $\bar{\mathcal{G}}$ are the same, and the node category set of $\bar{\mathcal{G}}$ is the modified set, i.e., $\hat{\mathcal{Y}}_\kappa$.

To obtain $\hat{A}$ in (6), the attack can be constructed as the following steps.

1) Introduce a surrogate GCN, which has the same layer structure and parameters with the target GCN. In the surrogate GCN, $\theta^E_{\mathcal{T}}$ and $\theta^D_{\mathcal{T}}$ are frozen.
2) Set a trainable parameter matrix $\mathcal{H}$ in surrogate GCN.
3) Take $\bar{\mathcal{G}}$ as the input of surrogate GCN.
4) *Forward Propagation:* In training epochs, $\hat{A}$ is dynamically changed according to $\mathcal{H}$. In particular, let $\mathcal{H} + \mathcal{H}^\top$ act on $\bar{A}$ by Hadamard Product. Other calculation methods remain unchanged with target GCN.
5) *Back Propagation:* Calculate the loss of the output and $\hat{\mathcal{Y}}_\kappa$, and update $\mathcal{H}$ according to the loss (reminding that $\theta^E_{\mathcal{T}}$ and $\theta^D_{\mathcal{T}}$ are frozen).

The schematic illustration of the attack is shown as in Fig. 1.

In the attack process, the Hadamard product ensures that trainable matrix can modify $\bar{A}$ element by element; furthermore, let $\mathcal{H}$ plus $\mathcal{H}^\top$ ensure the symmetry of the modified matrix. By letting $\mathcal{H}' = \mathcal{H} + \mathcal{H}^\top$, the adjacency matrix $\hat{A}$ following the abovementioned attack steps becomes:

$$\hat{A} = \mathcal{H}' \odot \bar{A} \tag{7}$$

where $\odot$ represents the Hadamard Product. As $\bar{A}$ is a matrix in which all elements are 1 ($\bar{\mathcal{G}}$ is a full connection graph with self-loop), so $\hat{A} = \mathcal{H}'$. In other words, the abovementioned attack steps dynamically perturb $\hat{A}$ in (6) by a surrogate model. Equation (6) can be expressed as

$$\hat{\mathcal{Y}}_\kappa = \underset{\text{surrogate}}{\mathrm{DEC}}\left[\underset{\text{surrogate}}{\mathrm{ENC}}\big(f; \mathcal{H}', \theta^E_{\mathcal{T}}\big); \mathcal{H}', \theta^D_{\mathcal{T}}\right] \tag{8}$$

where $\mathrm{ENC}_{\mathrm{surrogate}}$ and $\mathrm{DEC}_{\mathrm{surrogate}}$ are the encoder and decoder of the surrogate GCN, respectively.

Next, considering that the attacker does not perturb the edge directly connected to node κ for realized concealed attack, $\mathcal{H}'$

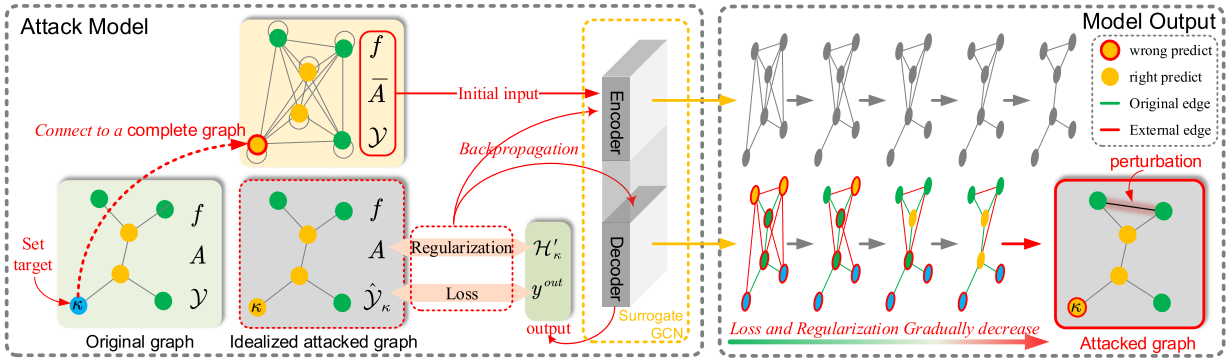

Fig. 1. Schematic illustration of G-EPA. G-EPA reduces a complete graph to a concealed and effective attacked graph. G-EPA iterates while training until the best attack scheme is found.

does not directly perturb κth line of $\bar{A}$; that is, $\mathcal{H}'$ in (8) can be replaced with $\mathcal{H}'_\kappa = (\mathcal{H}'_1, \ldots, A_\kappa, \ldots, \mathcal{H}'_N)^\top$, where $\mathcal{H}'_i$ denotes the ith row of $\mathcal{H}'$.

Furthermore, to ensure the global concealment, attackers need to minimize the amount of perturbation. In particular, it can be realized by adding a regularization, which can be denoted as $\text{Reg}(\mathcal{H}'_\kappa) = \text{Sum}(A - \mathcal{H}'_\kappa \odot \bar{A})$, where Sum is the sum of all elements of matrix. The regularization can quantify the perturbation degree of $\mathcal{H}'_\kappa$ to $\bar{A}$.

Hence, the attacker can obtain $\mathcal{H}'_\kappa$ by

$$\arg\min_{\mathcal{H}'_\kappa} \sum \Big\{ \text{Loss}\Big[\hat{\mathcal{Y}}_\kappa, \underset{\text{surrogate}}{\text{DEC}} \times \Big(\underset{\text{surrogate}}{\text{ENC}}\Big(f; \underset{\kappa}{\mathcal{H}'}, \theta_{\mathcal{T}}^E\Big); \underset{\kappa}{\mathcal{H}'}, \theta_{\mathcal{T}}^D\Big)\Big] + \text{Reg}\Big(\underset{\kappa}{\mathcal{H}'}\Big)\Big\}. \tag{9}$$

Finally, $\hat{A}$ can be calculated by (7).

Equation (9) is the mathematical form of the G-EPA. In Section VI, we evaluate the performance of (9) by implementing the attack.

C. Statement of GCNs Vulnerability

As can be seen, (3) and (9) are essentially the same, because (3) is the unified form of the training method of GCNs, and (9) is computationally universal. That is, if the attacker wants to modify the node category set, it can be realized by another GCN training task. As (4) is the general mathematical formula for attacks and (9) is derived from (4), so we conclude that attackers can precisely control the GCN output by modifying graph edges. In other words, no matter what the values of $\theta_{\mathcal{T}}^E$ and $\theta_{\mathcal{T}}^D$ in the $\hat{A}$-finding game (5) are, the optimal $\hat{A}$ can be obtained only manipulating $\mathcal{E}$. Thus, the output of the GCN can be manipulated by a more essence $\hat{A}$-finding game

$$\hat{A} = \underset{\text{essence}}{\textbf{ATTACK}}(\mathcal{E}). \tag{10}$$

Equation (10) shows that the attack target is the exposed graph structure, that is, edges of the graph. As long as the GCN model directly receives the graph edges as the input, the feasible perturbation can be found through (9) to construct edge-perturbing attacks. In other words, the vulnerability of the GCN is caused by the edge-reading permission.

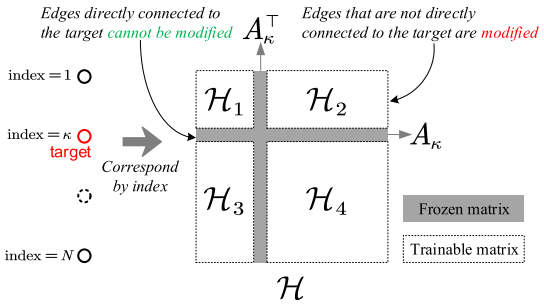

Fig. 2. Trainable parameter matrix $\mathcal{H}$ under single node attack.

D. Case Studies

In this part, we elaborate on how to attack a specific GCN. We use the Semi-GCN [38] as the specific model to obtain $\mathcal{G}_{\text{victim}}$, and the obtaining method can be expressed as

$$\arg\min_{\mathcal{H}'_\kappa} \sum \Big\{ \text{Loss}\Big[\hat{\mathcal{Y}}_\kappa, \text{Softmax} \times \Big(\underset{\kappa}{\mathcal{H}'} \text{Relu}\Big(\underset{\kappa}{\mathcal{H}'} f W_1\Big) W_2\Big)\Big] + \text{Reg}\Big(\underset{\kappa}{\mathcal{H}'}\Big)\Big\} \tag{11}$$

where W_1 and W_2 are well-trained parameters of target GCN in different layers. Next, we give the details of the attack, including trainable parameter matrix, layer weight constraint, layer weight initializer, and regularizer of the model. Considering that the existing attack scenarios include the single target scenario and that of the multiple targets, we describe them separately.

1) Single Target:

a) Trainable parameter matrix: To ensure $\mathcal{H}'_\kappa = (\mathcal{H}'_1, \ldots, A_\kappa, \ldots, \mathcal{H}'_N)^\top$, we set the trainable parameters $\mathcal{H}$ to four submatrices: $\mathcal{H}_1 \in \mathbb{R}^{(\kappa-1)\times(\kappa-1)}$, $\mathcal{H}_2 \in \mathbb{R}^{(\kappa-1)\times(N-\kappa)}$, $\mathcal{H}_3 \in \mathbb{R}^{(N-\kappa)\times(\kappa-1)}$, and $\mathcal{H}_4 \in \mathbb{R}^{(N-\kappa)\times(N-\kappa)}$, and they are placed in $\mathcal{H}$, as shown in Fig. 2.

b) Layer weight constraint: To restrict the modified $\hat{A}$ only have two elements: 0 and 1, we set up a layer weight constraints term

$$\mathcal{H}_{i,j} = \begin{cases} 0, & \text{s.t. } A_{i,j} \geq \text{O}_{i,j} \\ 1, & \text{s.t. } A_{i,j} < \text{O}_{i,j} \end{cases} \tag{12}$$

where $\mathcal{H}_{i,j}$ denotes the element in $\mathcal{H}$ with indices as i and j. O denotes parameters before layer weight constraint.

c) Layer weight initializer: Each trainable submatrix in $\mathcal{H}$ is initialized as elements, which is the same as the corresponding indices of A

$$\mathcal{H}_{i,j}^{(0)} = A_{i,j}^{(0)}. \tag{13}$$

2) Multiple Targets: In the case of multiple targets, let $\mathcal{K} = \{\kappa_1, \ldots, \kappa_N\}$ denote the target node set.

a) Trainable parameter matrix: The model flexibility of multitarget attack is limited by setting too many submatrices, so all the elements of $\mathcal{H}$ can be trained, and ensuring the concealment of attacks in regularizer.

b) Regularizer: To minimize the number of direct perturbations to the target node, we need to minimize the number of modifications to the κth row of $\bar{A}$. The regularizer for the multitarget attack is designed as

$$\mathrm{Sum}\left(A - \underset{\kappa}{\mathcal{H}'} \odot \bar{A}\right) + \vartheta \sum_{\kappa \in \mathcal{K}} \mathrm{Sum}(h_\kappa) \tag{14}$$

where h_κ denotes the κth row of $\mathcal{H}$, and ϑ denotes the custom coefficient.

IV. How GCNs Locate Nodes

As AN-GCN is capable of generating node positions, in this section, we will give the quantitative representation $f_\alpha[t]$ of the signal of node α with the specific training epoch t. By comprehensively checked the input of the GCN, we compare $f_\alpha[t]$ and the input of GCN, so as to explore how GCN constructs the input to locate the node.

The messages are constantly passing while the training phase of GCNs, which can also be regarded as a kind of signal transmission. As a result, the features of the node are constantly changing, and we regard it as an independent signal vibration. In this part, we build a mathematical model of the node signal vibration to figure out how GCNs analyze the given graph structure to locate nodes.

The node localization theorem is stated in Section IV-A and proved in Sections IV-B and IV-C. In Section IV-B, the basic mathematical model for node signal is given. In Section IV-C, the signal vibration driven by GCNs is described, whose processes include two parts: 1) the initial position of the node signal in the Fourier domain is given by blocking the transmission of graph signal, and 2) the model of node signal vibration in the training phase is given, by introducing the trainable parameters, which quantify the signal transmission on graphs. In particular, we first regard the change of each node features as the signal vibration with time t (in training, it is the form of epoch) and then map all the node signals to the same orthogonal basis through Fourier transform. Furthermore, we give the node signal model in the training phase of GCNs, which intuitively formulate how GCNs locating specific nodes. The method of building the node signal model is shown in Fig. 3.

Notation: Let $\mathcal{G} = (f, \mathcal{E})$ be a graph, where f is set of features of N nodes, while $f(i)$ is the feature of node i, and $\mathcal{E}$ denotes the set of edges. An essential operator in spectral graph analysis is the graph Laplacian, whose combinatorial definition is $\Delta = D - A$ where $D \in \mathbb{R}^{N\times N}$ is the degree matrix, and $A \in \mathbb{R}^{N\times N}$ is the adjacent matrix (both D and A can be calculated from $\mathcal{E}$). Let the Laplacian matrix of $\mathcal{G}$ is Δ where eigenvalues are $\lambda_1, \ldots, \lambda_N$, and the corresponding Eigen matrix of Δ is

$$U = \begin{pmatrix} u_1(1) & \cdots & u_N(1) \\ \vdots & \ddots & \vdots \\ u_1(N) & \cdots & u_N(N) \end{pmatrix}$$

$u_l = (u_l(1), \ldots, u_l(N))^\top$ is the lth eigenvector, and $u(l) = \{u_1(l), \ldots, u_N(l)\}$ is the row vector consisting of the values of all eigenvectors at position l. For convenience, "node with the index n" is denoted by "node n." j is the imaginary unit.

A. Node Localization Theorem

Before making a mathematical proof, we first claim that the following hold.

Theorem 1: Giving a graph $\mathcal{G}$ to be learned by a GCN, the GCN locates the node α according to $u(\alpha)$.

The proof of Theorem 1 is provided in Sections IV-B and IV-C, respectively.

B. Node Signal Model

First, we give the Fourier domain coordinates of the signal of a single node at a specific frequency.

Proposition 1: Giving a frequency ν, the signal coordinates in the Fourier domain of node α are

$$\hat{f}_\alpha[\nu] = \sum_{t=0}^{E-1} \bar{\theta}_t \left(\sum_{i=1}^{N} c_i e^{\lambda_i t} u_i(\alpha) \right) e^{-j\frac{2\pi}{E}t\nu} \tag{15}$$

where $\bar{\theta}_t$ and c_i are constants, and E is the number of training epoch.

Proof: The signal variation of a single node with E epochs is presented as $f[0]$, $f[1], \ldots, f[E-1]$. According to the discrete Fourier transform (DFT) theory, given a frequency ν, the Fourier transform for the signal f of node α (before introducing trainable parameters) is

$$\hat{f}_\alpha[\nu] = \sum_{t=0}^{E-1} f_\alpha[t] e^{-\frac{2\pi}{E}\nu t j}. \tag{16}$$

In the Fourier domain, according to the aggregation theory on graph [12], the signal in Fourier domain is dynamically vibrating caused by trainable parameters, so, here, we introduce a constant matrix representing vibration frequencies $\bar{\theta} \in \mathbb{R}^E$, where $\bar{\theta}_i$ is the value of $\bar{\theta}$ at the ith node, by act $\bar{\theta}$ on all frequency $\hat{f}[\nu], \nu = 0\!:\!E-1$ at every epoch. The signal fit by $\bar{\theta}$ is obtained, as shown in Fig. 4, specifically.

Hence, the Fourier domain graph signal (after introducing trainable parameters) is

$$\hat{f}_\alpha[\nu] = \sum_{t=0}^{E-1} \bar{\theta}_t f_\alpha[t] e^{-\frac{2\pi}{E}\nu t j}. \tag{17}$$

Furthermore, according to the model of signal transmission on the graph [42], the transmission of signal f on $\mathcal{G}$ is proportional to Δ acting on f with time t. As the end-to-end

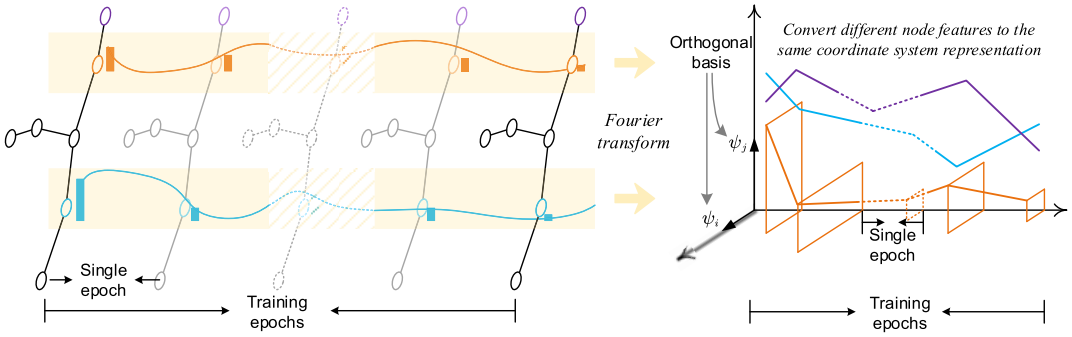

Fig. 3. Method of building node signal model. We regard the node feature change with the training epoch as the independent signal vibration and transform all the signals to a unified orthogonal basis by Fourier transform.

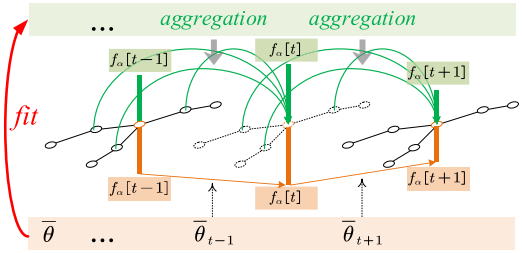

Fig. 4. Using $\bar{\theta}$ to fit graph quantify aggregation.

training drives the signal transmission on the graph

$$\frac{df}{dt} = -k\Delta f \tag{18}$$

where k is a constant. The first-order matrix ordinary differential equation [43] is

$$\dot{\chi}(\iota) = \mathscr{L}(\iota)\chi(\iota) \tag{19}$$

where ι is the independent variable in a transition system, $\dot{\chi}$ is the vector of the first derivatives of these functions, and $\mathscr{L}(\iota)$ is an $N \times N$ matrix of coefficients. In the case where $\mathscr{L}(\cdot)$ is constant and has N linearly independent eigenvectors, the general solution of (19) is

$$\chi(\iota) = \sum_{i=1}^{N} c_1 e^{\dot{\lambda}_i \iota} \rho_i \tag{20}$$

where $\dot{\lambda}_1, \ldots, \dot{\lambda}_N$ are the eigenvalues of $\mathscr{L}$, and $\rho_1, \ldots, \rho_N$ is the respective eigenvectors. In machine learning, t in (18) can be regarded as the iterative steps; i.e., discrete signal upgrading is the sampling of a signal function in continuous space. Giving the layer ℓ and the trainable matrix W, as $\Delta \in \mathbb{R}^{N\times N}$ has N linearly independent eigenvectors similar to the $\mathscr{L}(\cdot)$, as well as spatial GCNs model the message passing by $(df/d\ell) \approx \Delta f \cdot W$, (18) is homogeneous to (19) that has the general solution, which is given by $f[t] = \sum_{i=1}^{N} c_i e^{\lambda_i t} u_i$. As $f[t]$ and u_i are column vectors, for node α

$$f_\alpha[t] = \sum_{i=1}^{N} c_i e^{\lambda_i t} u_i(\alpha). \tag{21}$$

Equation (21) can be regard as the graph signal transmission from a global perspective. Hence, (15) can be derived by substituting (21) into (17). □

Then, for node α, we integrate the signals of all frequencies in the Fourier domain to obtain the original signal in the orthogonal basis representation, which is stated in the following proposition.

Proposition 2: The signal of node α in epoch t is

$$f_\alpha[t] = \sum_{\nu=0}^{E-1} \frac{2}{E} |\hat{f}_\alpha[\nu]| \cos\left[(2\pi\nu/E\epsilon)t\epsilon + \arg\left(\hat{f}_\alpha[\nu]\right)\right] e^{j\frac{2\pi}{E}t\nu} \tag{22}$$

where $\arg(\cdot)$ is the argument of a complex number, and ϵ denotes the epoch interval; here, we take it as a minimal value, that is, $\epsilon \to 0$.

Proof: According to the DFT theory, the inverse transform of $\hat{f}_\alpha[\nu]$ is

$$f_\alpha[t] = \frac{1}{E} \sum_{\nu=0}^{E-1} \hat{f}_\alpha[\nu] e^{j\frac{2\pi}{E}\nu t}. \tag{23}$$

That is, the inverse matrix is $(1/N)$ times the complex conjugate of the original (symmetric) matrix. Note that the $\hat{f}_\alpha[\nu]$ coefficients are complex. We can assume that the graph signal $f[t]$ values are real (this is the simplest case, and there are situations in which two inputs, at each t, are treated as a complex pair, because they are outputs from 0° to 90° demodulators). In the process of taking the inverse transform for the terms $\hat{f}_\alpha[\nu]$ and $\hat{f}_\theta[E-\nu]$ (the spectrum is symmetrical about $(E/2)$ [44]), combine to produce two frequency components, only one of which is considered to be valid. Hence, from (23), the contribution to $f_\alpha[t]$ of $\hat{f}_\alpha[\nu]$ and $\hat{f}_\alpha[E-\nu]$ is

$$\mathcal{C}_\alpha^\nu[t] = \frac{1}{E}\left(\hat{f}_\alpha[\nu] e^{j\frac{2\pi}{E}\nu t} + \hat{f}_\alpha[E-\nu] e^{j\frac{2\pi}{E}(E-\nu)t}\right). \tag{24}$$

Considering that all graph signals $f[t]$ are real, so

$$\hat{f}_\alpha[E-\nu] = \sum_{t=0}^{E-1} f[t] e^{-j\frac{2\pi}{E}(E-\nu)t} \tag{25}$$

and according to Euler's formula [45]

$$e^{-j2\pi t} = \frac{1}{\cos(2\pi t) + j\sin(2\pi t)} = 1, \quad \text{s.t. } t \in \mathbf{N}_+ \tag{26}$$

where $\mathbf{N}_+$ is the set of all positive integers. We have

$$e^{-j\frac{2\pi}{E}(E-\nu)t} = \underbrace{e^{-j\ 2\pi t}}_{1 \text{ for all } t} e^{+j\frac{2\pi\nu}{E}t} = e^{+j\frac{2\pi}{E}\nu t} \tag{27}$$

i.e., $\hat{f}_\alpha[E-\nu]=\hat{f}_\alpha^*[\nu]$, where $\hat{f}_\theta^*[\nu]$ is the complex conjugate. Substituting (27) into (24), we obtain $e^{j2\pi t}=1$ with reference to (26). Then, we have

$$\begin{aligned}\mathcal{C}_\alpha^\nu &= \frac{1}{E}\left(\hat{f}_\alpha[\nu]e^{j\frac{2\pi}{E}\nu t}+\hat{f}_\alpha^*[\nu]e^{-j\frac{2\pi}{E}\nu t}\right)\\ &= \frac{2}{E}\left[\mathrm{Re}\left(\hat{f}_\alpha[\nu]\right)\cos(2\pi\nu t/E)-\mathrm{Im}\left(\hat{f}_\alpha[\nu]\right)\sin(2\pi\nu t/E)\right]\\ &= \frac{2}{E}|F[\nu]|\cos\left[(2\pi\nu/E\epsilon)t\epsilon+\arg(F[\nu])\right] \qquad (28)\end{aligned}$$

where $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ denote taking the real part and the imaginary part of an imaginary number, respectively. Hence, by integrating the signals of all frequencies $\nu=0:E-1$

$$f_\alpha[t]=\sum_{\nu=0}^{E-1}\mathcal{C}_\alpha^\nu[t]e^{j\frac{2\pi}{E}t\nu}. \qquad (29)$$

Equation (22) is derived by substituting (28) into (29). □

C. Node Signal in the Initial and the Training Phase

In Section IV-B, we build the mathematical model for the training phase of GCNs. In this part, we first give the initial state of the node signal by blocking the signal transmission, which is stated in the following proposition.

Proposition 3: The initial signal of node α is

$$f_\alpha[0]=2\bar{\theta}_0\mathrm{Sum}[u(\alpha)]. \qquad (30)$$

Proof: According to the general GCN forward propagation [46], the embedding feature of $\mathcal{G}$ is

$$f^e[\cdot]=\sigma\left(Ug_\theta(\Lambda)U^\top f\right) \qquad (31)$$

where $g_\theta(\Lambda)=\mathrm{Diag}(\theta_1,\ldots,\theta_N)$ is the trainable diagonal matrix, and $\sigma(\cdot)$ is the an activation function; hence, the output of GCN for node α is

$$f_\alpha^e[\cdot]=\sigma\left(u(\alpha)g_\theta(\Lambda)U^\top f\right). \qquad (32)$$

Now, we simplify (32). We denote $g_\theta(\Lambda)U^\top f$ as a constant matrix (note that so far, message passing is not activated yet; i.e., the trainable parameters can be regarded as constants) $\Phi\in\mathbb{R}^{N\times d}$, where d is the dimension of the feature, ϕ_i is the ith row of Φ, and $\phi_i(k)$ is the kth element of ϕ_i. Furthermore, we replace the nonlinearity $\sigma(\cdot)$ with a simple linear activation function, leading to

$$\begin{aligned}f_\alpha^e[\cdot] &= \sum_{i=1}^{N}\phi_1(i)u_i(\alpha),\quad \text{s.t., } d=1\\ f_\alpha^e[\cdot] &= \left[\sum_{i=1}^{N}\phi_1(i)u_i(\alpha),\ldots,\sum_{i=1}^{N}\phi_d(i)u_i(\alpha)\right],\quad \text{s.t., } d>1.\end{aligned} \qquad (33)$$

As our aim is to solve the qualitative problem of node localization, we discuss only the case of $d=1$. As the discovery making from observation: (33) is homogeneous with (21); moreover, ϕ_i and c_i both denote the changeable parameters, and we conclude that Propositions 2 and 3 are applied to the training phase of GCN.

We consider the initial state as the non-transmission state; that is, for node α, $f_\alpha[0]=\cdots=f_\alpha[E-1]$; hence, the transmission of graph signal can be blocked by letting $c_i=e^{-\lambda_i t}$. By regarding the initial status of GCN as $E=1$ and $t=0$, (30) can be obtained by substituting (15) into (22), s.t., $E=1$ and $t=0$. □

Next, we active the message passing on $\mathcal{G}$. As $\bar{\theta}$ denotes constants, (15) can be concluded by

$$\hat{f}_\alpha[\nu]=\sum_{t=0}^{E-1}\left(\sum_{i=1}^{N}\bar{c}_i e^{\lambda_i t}u_i(\alpha)\right)e^{-j\frac{2\pi}{E}t\nu} \qquad (34)$$

where $\bar{c}_i=\bar{\theta}_t c_i$. By substituting (34) into (22), the final expression of node signal is

$$\begin{aligned}f_\alpha[t]=\lim_{\epsilon\to 0}\Bigg\{\sum_{\nu=0}^{E-1}\frac{2}{E}\Bigg|\sum_{t=0}^{E-1}\Bigg(\sum_{i=1}^{N}\bar{c}_i e^{\lambda_i t}\underbrace{u_i(\alpha)}_{\text{from }\mathcal{E}}\Bigg)e^{-j\frac{2\pi}{E}t\nu}\Bigg|\\ \times\cos\left[(2\pi\nu/E\epsilon)t\epsilon+\arg\left(\hat{f}_\alpha[\nu]\right)\right]e^{j\frac{2\pi}{E}t\nu}\Bigg\}.\end{aligned} \qquad (35)$$

Equation (35) gives the signal model of all nodes in the unified reference frame, by observing, for node α; among all the initial conditions related to $\mathcal{G}$, only $u(\alpha)$ appears in the process of node signal change. In other words, in the elastic system brought by end-to-end training, only $u(\alpha)$ is a controllable factor, so we simplify the abovementioned equation to

$$f_\alpha[t]=\mathcal{F}[u(\alpha)] \qquad (36)$$

and the feature vibrate of node α under the end-to-end training is

$$\begin{aligned}&\texttt{Initial:}\quad 2\bar{\theta}_0\mathrm{Sum}[u(\alpha)]\\ &\texttt{Training:}\quad \mathcal{F}[u(\alpha)].\end{aligned} \qquad (37)$$

Thus, the feature vibrates of node α driven by GCNs can be quantified, and it can always contain a fixed factor $u(\alpha)$ while message passing. As GCNs quantifying the message passing in training phase is part according to the input Δ, GCNs deconstruct Δ to Laplacian matrix U and further quantify the feature vibrates of node α by $u(\alpha)$; i.e., GCNs locate node α according to $u(\alpha)$.

V. Proposed AN-GCN

In Section III, the vulnerability of GCNs is demonstrated due to their edge-reading permission. Following this, we then propose an AN-GCN that can withdraw the edge-reading permission of GCNs. We integrate a generator and self-contained GCNs to ensure that all node positions are generated from the noise. The AN-GCN classifies nodes only according to their node indices and features, and no longer relies on the edge information.

A. Basic Model for AN-GCN

According to Theorem 1, we denote each row of the Laplacian matrix as an independent generating target. We use

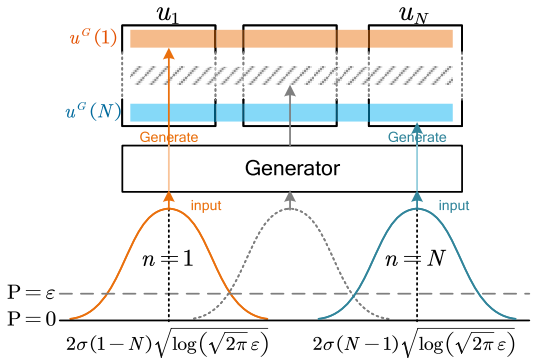

Fig. 5. Generator with staggered Gaussian distribution as an input.

the spectral graph convolution (without any constraints and simplification rules) [21] as the basic model and take it as an encoder. We use an extra fully connected neural network decoder. That is, the basic forward propagation is

$$f^e = \sigma\left(U g_\theta(\Lambda) U^\top f\right) \tag{38a}$$

$$y_{\text{out}} = f^e W^D. \tag{38b}$$

In (38), U contains the information of the edges in the graph, and we replace it with a matrix generated from Gaussian noise. Equation (38) is the basic signal forward propagation. Next, we elaborate on how to improve (38), thus letting the basic model accommodate the generated node position.

B. Generating Node Positions From the Noise

If $u(n)$ is completely generated by noise, the specific points will keep anonymous. Thus, we take $u(n)$ as the generation target. The output of the generator is denoted as $u^G(n)$, which tries to approximate the underlying true node position distribution $u(n)$.

Next, we define the probability density function (pdf) for the input noise. To enable the generator to locate a specific point, the input noise of the generator is constrained by the position of the target point. So, we define the input noise as having a pdf equal to that of staggered Gaussian distribution, and the purpose is to ensure the noise not only satisfies the Gaussian distribution but also does not coincide with each other, thus letting the generated noises orderly distributed on the number axis.

Proposition 4 (Staggered Gaussian Distribution): Giving a minimum probability ε, N Gaussian distributions centered on $x = 0$ satisfy $\mathrm{P}(x,n) \sim \mathrm{Norm}(2\partial(2\ n-N-1)(\log((2\pi)^{1/2}\partial\varepsilon))^{1/2}, \partial^2)$, so that the pdf of each distribution is greater than ε, where Norm is the Gaussian distribution, n is the node number, ∂ is the standard deviation, and ε is the set minimum probability.

Proof: Given a pdf $h(x_p)$ of the Gaussian distribution $\mathrm{Norm}(\mu_p, \partial^2)$, when $h(x_p) = \varepsilon$

$$x_p = \mu_p \pm 2\partial\sqrt{\log\left(\sqrt{(2\pi)}\partial\varepsilon\right)}. \tag{39}$$

Let $2\partial(\log(((2\pi))^{1/2}\partial\varepsilon))^{1/2} = r$ as the distance from the average value μ_p to the maximum and minimum values of x_p. Specify that each x_p represents the noise distribution of each node. To ensure that all the distributions staggered and densely arranged, stipulate $\max(x_p) = \min(x_{p+1})$, and keep all distributions symmetrical about $x = 0$. So, when the total number of nodes is N, $\mu_1 = (1-N)r$, $\mu_2 = (3-N)r, \ldots, \mu_N = (N-1)r$; that is, $\mu_n = (2n-N-1)r = 2\partial(2\ n-N-1)(\log((2\pi)^{1/2}\partial\varepsilon))^{1/2}$. □

The process of generating sample $u^G(v)$ from staggered Gaussian noise $Z_v \sim P(x,v)$ is denoted as $u^G(v) = G(Z_v)$, and U generated by G is denoted as U^G. The generating process is shown in Fig. 5.

Algorithm 1 AN-GCN

Require: Weights of D: θ^D,include $g_\theta^{D,enc}(\Lambda)$ is used for encoding, $\theta^{D,dec}$ is used for decoding. Weights of G: θ^G. Training epoch of D and G.

```
Initialize U^D = U
for epoch in Train epochs for D do
    Random sample a node v
    Obtain noise Z_v from P(z, v)
    u^G(v) ← G(Z_v^D) // Generate
    U^D_{·v} = U^D_{·v} + q(u^G(v) − U^D_{·v}) // Linear approximation
    Calculate fake label of v by Eq. (43a)
    Calculate real label of v by Eq. (43b)
    Calculate ∇^D_update by Eq. (45)
    θ^D ← θ^D − ∇^D_update
    for epoch in Train epochs for G do
        Obtain noise Z_v from P(z, v)
        Calculate y_v^fool by Eq. (46)
        Calculate ∇^G_update by Eq. (48)
        θ^G ← θ^G − ∇^G_update
    end for
end for
```

Ensure: Trained generator weights θ^G.

C. Detecting the Generated Node Positions

1) Discriminator of AN-GCN: After proposing the generation of $u^G(n)$, we set a discriminator D to evaluate the quality of $u^G(n)$, and design a two-player game for G and D. This ensures that the AN-GCN can accurately classify nodes by $u^G(n)$. We use classification quality as an evaluation indicator to drive the entire AT. This ensures that D cannot only distinguish the adversarial samples generated by G (non-malicious, used for anonymize nodes), but also can provide accurate classification. In particular, D is divided into two parts: a diagonal matrix with trainable parameters D_{enc} used for encoding, and a parameter matrix D_{dec} used for decoding. Thus, the forward propagation becomes

$$\mathcal{Y} = \sigma\left(U D_{\text{enc}} U^\top f D_{\text{dec}}\right). \tag{40}$$

Consider there are U and $U^\top$ in the encoder (38a), and we mark them as different matrices

$$\mathcal{Y} = \sigma\left(U^G D_{\text{enc}} U^D f D_{\text{dec}}\right). \tag{41}$$

To eliminate the information of nodes' positions in the AN-GCN, the corresponding $u(\cdot)$ to U^G is generated by

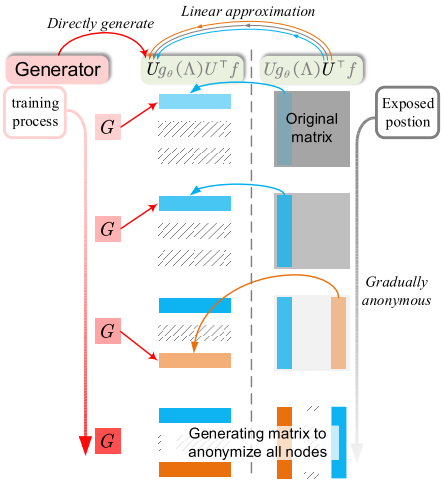

Fig. 6. In the training phase, U^D gradually approximates U linearly, so as to realize anonymity.

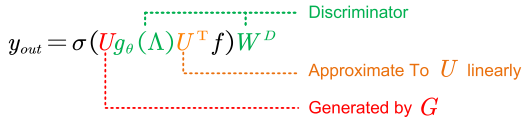

Fig. 7. Forward propagation of AN-GCN.

independent generation of different rows, and U^D on the right is approximate linearly according to the numerical change of U^G. Consider that U^G and U^D changed dynamically while training, we use $U^{G,e}$ and $U^{D,e}$ to denote U^G and U^D in epoch e, respectively. While generating the lth row of U^G, the corresponding column of U^D, denoted as u_l^D, linearly approximates toward $u^G(l)$. In particular, the general term formula for its value in training epoch e is

$$u_l^{D,e} = \begin{cases} u_l, & \text{s.t. } e = 1 \\ u_l^{D,e-1} + q\left(u^{G,e}(l) - u_l^{D,e-1}\right), & \text{s.t. } e > 1 \end{cases}$$

where $u^{G,e}(l)$ is the generated $u^{G,e}(l)$ in epoch e, and q is the custom linear approximation coefficient. Thus, $U^{D,e}$ is gradually approaching the generator matrix $U^{G,e}$ during the training phase. With the increase in training epochs, the collaboration among U and $U^\top$ in (40) and G is shown in Fig. 6. The forward propagation of the proposed AN-GCN is described in Fig. 7.

We have eliminated the edge information stored in AN-GCN. Next, we elaborate on the training phase, which allows nodes to participate anonymously. For ensuring that both sides of the AT have the clear objectives, we further give the optimization of generator and discriminator.

2) Forward Propagation of AN-GCN: We have anonymized nodes through the generator, which is proposed in Section V-B. In this part, we elaborate on the forward propagation of AN-GCN. Let $f^{e,G}(v)$ and $f^{e,D}(v)$ denote the embedding of the node v when using $u^G(v)$ and $u(v)$ locate nodes at epoch e, respectively. In each training epoch e, first, we obtain $u^G(v)$ and $U^{D,e}$ corresponding to the target node v, that is, generate $u^G(v)$ from the staggered Gaussian noise (Proposition 4) corresponding to node v, and update the corresponding column of $U^{D,e-1}$ to $U^{D,e}$. Second, we use D_{enc} and D_{dec} to evaluate the quality of this epoch of generators. By getting the node embedding $f^{e,G}(v)$ of v through D_{enc}, we decode $f^{e,G}(v)$ through D_{dec} to get the soft label, which denoted as y_v. Based on (38a), the soft label of node v is

$$y_v = \sigma\left[u(v)D_{\text{enc}}(\Lambda)U^\top f D_{\text{dec}}\right]. \tag{42}$$

3) Discriminator Optimization: Discriminator optimization aims at reducing the classification accuracy of the generator, while improving the classification accuracy of the discriminator. The classification accuracy is quantified by the soft label. For node v with the generated position, its soft label is denoted as y_v^{fake}. y_v^{fake} can be calculated according to $u^G(v)$, the linear approximation matrix U^D and the discriminator in the current epoch. Aiming at training the discriminator to detect node v with generate position, the real soft label y_v^{real} is calculated by the nondefense GCN (31). Thus, y_v^{fake} and y_v^{real} are given as

$$y_v^{\text{fake}} = \sigma\left[u^G(v)D_{\text{enc}}U^D f(v)D_{\text{dec}}\right] \tag{43a}$$

$$y_v^{\text{real}} = \sigma\left[u(v)D_{\text{enc}}U f(v)D_{\text{dec}}\right]. \tag{43b}$$

To detect nodes with generated positions, the discriminator aims at not only classifying real nodes into the correct categories but also classifying fake nodes into the other random categories. This reduces the performance of the generator during the training phase of the discriminator. Therefore, the performances of the generator and discriminator are quantified by the loss functions, which are designed as

$$\text{Loss}_D(y) = \begin{cases} \mathbb{C}[\mathbb{S}(y), \text{label}_v], & \text{s.t. } y = y_v^{\text{real}} \\ \mathbb{C}\left[\mathbb{S}(y), \text{RD}\left(\{\text{label}_\gamma \,|\gamma \neq v\}\right)\right], & \text{s.t. } y = y_v^{\text{fake}} \end{cases} \tag{44}$$

where label_i represents the real label of node i (one hot), $\mathbb{C}(\cdot)$ denotes the cross entropy function, $\mathbb{S}(\cdot)$ denotes the sigmoid function, and $\text{RD}(\cdot)$ denotes the random sampling function. Then, according to $\text{Loss}_D(y)$, we calculate the gradient for D by

$$\nabla_{\text{update}}^D = \nabla_{g_\theta^{D,\text{dec}}(\Lambda),\theta^{D,\text{enc}}}\left[\text{Loss}_D\left(y_v^{\text{real}}\right) + \text{Loss}_D\left(y_v^{\text{fake}}\right)\right] \tag{45}$$

where $g_\theta^{D,\text{dec}}(\Lambda)$ and $\theta^{D,\text{enc}}$ are the trainable parameters of D_{enc} and D_{dec}, respectively. Finally, we update the weight of D according to ∇_{update}^D.

4) Generator Optimization: Next, the generator is trained to fool the well-trained discriminator. This ensures that $u^G(v)$ generated by G can provide accurate classification. For node v, after resampling the noise Z_v, the label used to fooled D is calculated by

$$y_v^{\text{fool}} = \sigma\left[G(Z_v)D_{\text{enc}}U^D f D_{\text{dec}}\right]. \tag{46}$$

In the generation phase, the generator attempts to classify the node v as the correct label; thus, the loss function of D is designed by

$$\text{Loss}_D(y) = \mathbb{C}[\mathbb{S}(y), \text{label}_v], \quad \text{s.t. } y = y_v^{\text{fool}}. \tag{47}$$

Then, to ensure that the outputs of G can be classified into the real categories, we calculate the gradient for D by

$$\nabla_{\text{update}}^G = \nabla_{\theta^G}\left[\text{Loss}_D\left(y_v^{\text{fool}}\right)\right] \tag{48}$$

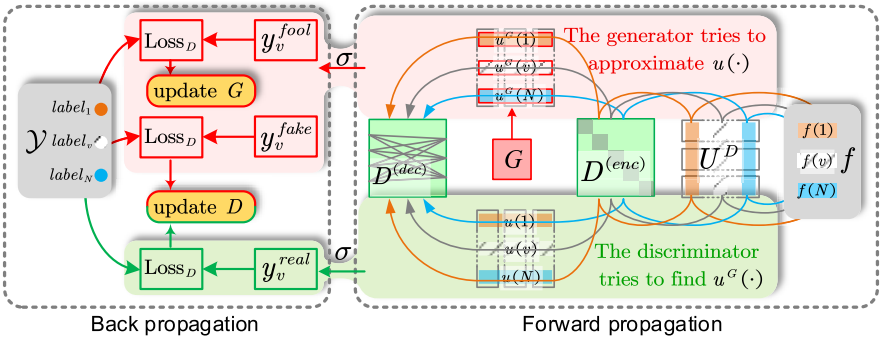

Fig. 8. Schematic illustration of AN-GCN.

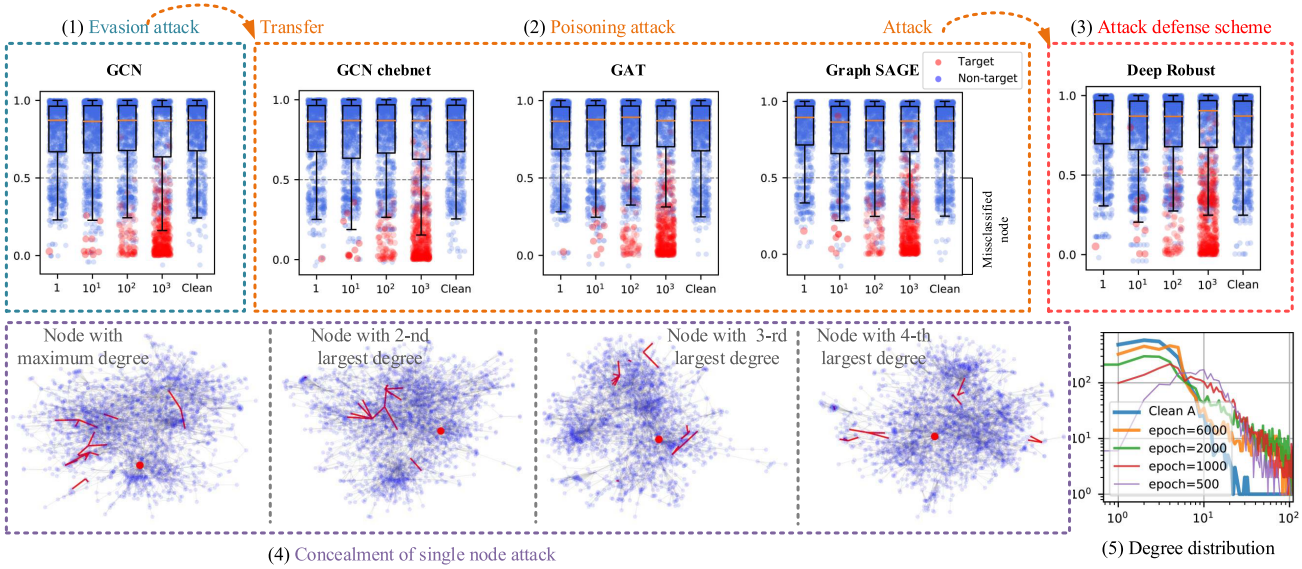

Fig. 9. Evaluation results of G-EPA. 1) Evaluation—the horizontal axis is the number of attack nodes, and the vertical axis is the scaling index according to the loss function. The more similar the distribution of nontarget nodes and clean graph, the less misclassified the nontarget nodes will be. It can be seen that with the increase in the number of attacks, almost all the target nodes are misclassified, while the nontarget nodes almost keep the original classification results. 2) Evaluation—after $\mathcal{G}_{\text{victim}}$ was transferred to different models and retrained, the performance of each model was evaluated. It can be seen that all models are successfully attacked, which shows that the attack method represented by Eq. (9) can transfer to other models. 3) Evaluation—it can be seen that a large number of target nodes are misclassified, thus proving the defense ability of the model decreases. 4) Evaluation—the red dot is the target node, and the red line is the modified connection. As can be seen that we have not directly modified the node, and the total number of perturbations is insignificant, so the attack has well performance on concealment. 5) Evaluation—with the increase in training epoch, the degree distribution is more and more like the degree distribution of clean A, that is to say, $\mathcal{G}_{\text{victim}}$ eventually tends to be stable.

and update the weight of G according to ∇^G_{update}. Note that D is frozen in the training phase of the generator. The schematic illustration of AN-GCN is represented by Fig. 8, and the algorithm process of AN-GCN is given in Algorithm 1.

VI. Experimental Results

In this section, we evaluate the performance of the proposed AN-GCN in node classification based on five widely used [47]–[49] benchmark datasets, including Cora, Citeseer, Pubmed, Polblogs, and Reddit. In practice, we first carry out numerical experiments to verify the correctness of the proposed node localization theorem. We then attack multiple GCNs by G-EPA to verify the effectiveness of the stated precondition, which causes the GCNs vulnerability. We finally observe the classification results of AN-GCN to verify the utility of AN-GCN.

A. Evaluation for the G-EPAs

Equation (9) formulates the existing edge-perturbing attack. By observing the effectiveness of (9), we demonstrate that the exposure of node position caused the vulnerability of GCNs. The evaluation of (9) includes five items.

1) The effectiveness of the attack that is represented by box plot. We act (9) on the Semi-GCN [38], which is trained on the clean dataset to obtain $\mathcal{G}_{\text{victim}}$.
2) Transferability of attack. We then transfer $\mathcal{G}_{\text{victim}}$ to the training set of GraphSAGE, GAT, and GCN ChebNet [21] and evaluate the classification performance of multiple models.
3) Attack effect on the existing mainstream defense scheme (Deep Robust [14]).
4) Concealment of the single node attack. The evaluation effect is given as to whether there is an obvious perturbation in the fifth-top order neighborhood of the target node.

TABLE I

RESULTS OF EVALUATION ITEMS 1)–3) (SUCCESS RATE %)

Dataset	Cora				Citeseer				Pubmed				PolBlogs				Reddit			
Target number	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
GCN	100	100	96	93.1	100	100	92	91.6	100	100	98	97.5	100	80	93	92.3	100	90	95	94.5
GCN_ChebNet	100	100	100	96.3	100	100	98	98.5	100	100	93	94.0	100	100	96	95.4	100	100	96	96.2
GAT	100	100	95	91.5	100	90	93	92.1	100	100	97	96.6	100	100	95	96.8	100	100	93	93.3
GraphSAGE	100	80	96	89.4	100	80	89	88.6	100	100	95	93.9	100	80	93	94.9	100	90	88	87.2
DeepRobust	100	60	74	72.8	100	70	72	73.0	100	80	81	80.2	100	70	75	71.4	100	80	82	70.7

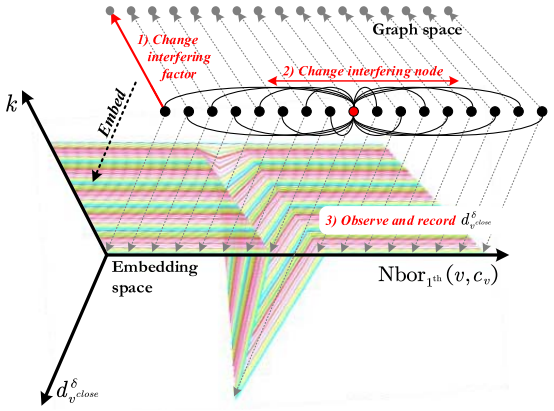

Fig. 10. Deviation of the 16-top degree neighbors of v after interfered.

5) The stability of a graph structure after the attack. The stability is usually represented by degree distribution [16].

The results of the abovementioned items are shown in Fig. 9. In particular, the results of evaluation items 1)–3) mentioned earlier are presented in Table I, and the success rate is the probability that the target node is successfully misclassified.

B. Numerical Experiment for Node Localization Theorem

As Theorem 1 is the theoretical basis of AN-GCN, in this part, we verify the correctness of Theorem 1 by numerical experiment. We design two experiments to verify the correctness of Theorem 1: 1) observing the embedding deviation of nodes v after interfering the corresponding $u(v)$ by a manual factor, and 2) observing the changes of $u(\text{neighbor}(v))$ after deleting a node v, where neighbor(v) is the neighbor of v. In the first experiment, if the embedding deviation of node v is significantly greater than that of the other nodes while interfering a manual factor to $u(v)$, it supports that $u(v)$ is highly sensitive to the position of node v. In the second experiment, if the change of corresponding $u(\cdot)$ dwindles while increasing the order of neighbor (i.e., the distance from the deleted node), it further proves that the specific position of v is closely related to the value of $u(v)$. The abovementioned two experiments jointly demonstrate the theoretical correctness of Theorem 1 collectively.

1) Influence of $u(\cdot)$ on the Node Embedding: By selecting a target node v, its deviation of embedding is observed after interfering insignificantly to $u(\cdot)$ related to v. Let $\mathrm{Nbor}_{1\mathrm{th}}(v, c_v) = \{v_1(1), \ldots, v_1(c_v)\}$ denote the c_v-top degree first-order neighbors of v, where c_v is given subjectively to screen out the low-degree neighbors. Given a interfering factor δ, the embedding deviation of v is calculated by interfering δ on $\mathrm{Nbor}_{1\mathrm{th}}(v, c_v)$ successively. Thus, in each interfering turn, given a target v_i^{close}, the non-interfered U becomes a interfered matrix $\hat{U}^{\delta, c_v}$, where all rows in $\hat{U}^{\delta, c_v}$ satisfy

$$\hat{u}^{\delta, c_v}(i) = \begin{cases} u(i), & \text{s.t. } i = v_1(i) \\ \delta u(i), & \text{s.t. } i \neq v_1(i). \end{cases} \tag{49}$$

Furthermore, we calculate the Euclidean distance [50] between $f^{e,\delta}$ and f^e [refering to (38a)]; i.e., $d_v^{\delta} = \|f^{e,\delta} - f^e\|_2$. While changing δ, we use Chebyshev polynomial [21] as the convolution kernel and test on the Cora dataset. Let $c_v = 16$, and $\delta = 1-(k/100)$, $k \in \{1, \ldots, 50\}$; the result is shown as in Fig. 10. It can be seen that, the embedding accuracy, measured by the Euclidean distance between $f^e(v)$ and $f^{e,\delta}(v)$, will drop suddenly once δ interferes on the vth row of U and keep stable while δ interferes on the other positions. That is, $u(v)$ has a greater impact on the embedding of node v.

2) Influence on $u(\cdot)$ While the Nodes' Positions Changed: By deleting a node τ in the graph $\mathcal{G}$ to obtain the deleted graph $\mathcal{G}_\tau^{(d)}$, we calculate the Laplacian matrix $L_\tau^{(d)} \in \mathbb{R}^{(N-1)\times(N-1)}$ and matrix eigenvalues $U_\tau^{(d)} = \{u^{(d)}(1), \ldots, u^{(d)}(N-1)\} \in \mathbb{R}^{(N-1)\times(N-1)}$. To keep $\mathcal{G}_\tau^{(d)}$ well connected, all edges connected to τ are reconnected traversally. In particular, we stipulate that ω_{ij} and $\omega_{ij}^{(d)}$ are the weights of edges between i and j in $\mathcal{G}$ and $\mathcal{G}_\tau^{(d)}$, respectively. The calculation method of all edge weights in $\mathcal{G}_\tau^{(d)}$ is

$$\omega_{ij}^{(d)} = \begin{cases} \omega_{ij}, & \omega_{\tau i} = 0 \text{ or } \omega_{\tau j} = 0 \\ \omega_{ij} + \dfrac{\omega_{\tau i} + \omega_{\tau j}}{2}, & \omega_{\tau i} \neq 0,\ \omega_{\tau j} \neq 0. \end{cases} \tag{50}$$

After obtaining the fully connected $\mathcal{G}_\tau^{(d)}$, we recalculate corresponding $U_\tau^{(d)}$ and query all βth-order neighbors of τ: $\mathrm{Nbor}_{\beta\mathrm{th}}(\tau, \cdot) = \{\tau_\beta(1), \tau_\beta(2), \ldots\}$. Thus, we obtain a $u(\cdot)$ set $u^{(d)}(\tau_\beta(\cdot)) = \{u(\tau_\beta(1)), \ldots, u(\tau_\beta(N-1))\}$, which represents $u(\cdot)$ corresponding to $\mathrm{Nbor}_{\beta\mathrm{th}}(\tau, \cdot)$ in $\mathcal{G}_\tau^{(d)}$. Similarly, $u(\tau_\beta(\cdot))$ is denoted as $u(\cdot)$ set from $\mathcal{G}$, and the quantitative

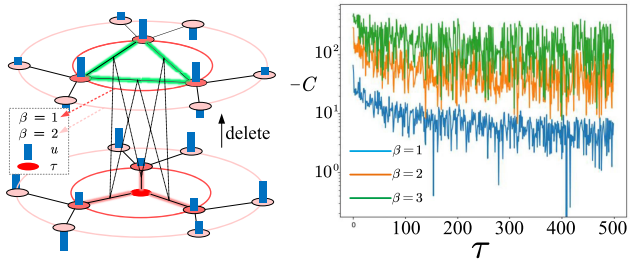

Fig. 11. After deleting a node τ, $u(\tau)$ of its first-order neighbor changes most significantly, and $u(\tau)$ of its further neighbor changes gradually subtle.

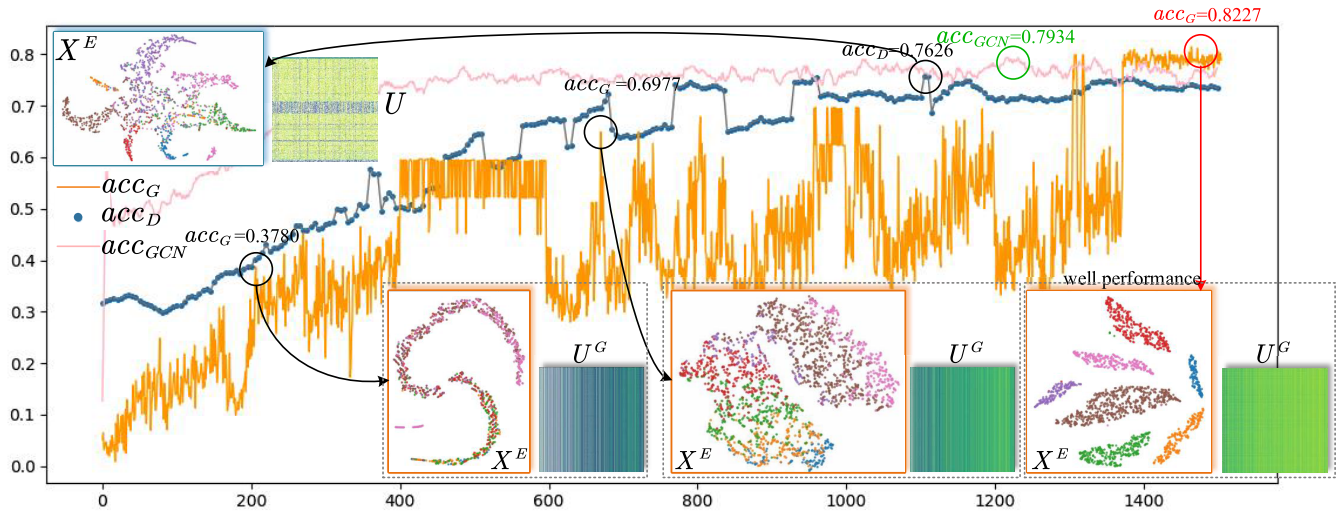

Fig. 12. Classification accuracy and the visualization of nodes' embedding during the training phase. It can be seen that acc_D and acc_G are rising at the same time. When epoch $>$ 1400, acc_G remains at a high and stable state. We select G at epoch $=$ 1475 as the final model. The node embedding accuracy is 0.8227. The highest value of acc_{GCN} is 0.7934. The experimental results show that under the same convolution kernel design, AN-GCN not only effectively maintains the anonymity of the node but is also 0.0293% higher than the state-of-the-art nondefense GCN in classification accuracy.

representation of the change between the two is as follows:

$$\mathrm{C}\left[u\left(\tau_\beta(\cdot)\right), u^{(d)}\left(\tau_\beta(\cdot)\right)\right] = \sum_{i=1}^{N-1}\left[\log\left|u_i\left(\tau_\beta(\cdot)\right)\right|^2 - \log\left|u_i^{(d)}\left(\tau_\beta(\cdot)\right)\right|^2\right]. \quad (51)$$

By examining different β values and calculating $\mathrm{C}(\cdot)$, the result is as shown in Fig. 11. We select the first 500 nodes according to the number of connections from large to small. Fig. 11 shows intuitively the change of $u(\cdot)$ in different positions after deleting nodes. After deleting τ, the change of neighbor $u^{(d)}(\tau_\beta(\cdot))$ of different orders β of τ (right). After deleting node τ, $u(\cdot)$ of the first-order neighbor $u^{(d)}(\tau_\beta(\cdot))$ has the largest change (the vertical axis is $-C$); i.e., after the node τ is deleted, $u(\cdot)$ corresponding to its first-order neighbor $u(\tau_1(\cdot))$ has changed significantly, while $u(\tau_2(\cdot))$ and $u(\tau_3(\cdot))$ have changed subtle and showed a decreasing trend. As deleting node τ significantly affects the position of its first-order neighbors, with the order increases, the degree of influence gradually decreases, so the change of its $u(\tau_\beta(\cdot))$ also gradually decreases. Fig. 11 proves that the position of node τ is inseparable from $u(\tau)$.

C. Evaluation for the Effectiveness of AN-GCN

Next, we evaluate AN-GCN. As AN-GCN eliminates the possibility of potential perturbations, we mainly evaluate the accuracy of AN-GCN, and its robustness to the perturbation of training set is as the secondary evaluation item. As the application scenario of AN-GCN is that the user can ensure that AN-GCN is trained on the clean graph, the scenario of poisoning attack is not the main problem of this article. But, we still prove that AN-GCN shows its robustness to the perturbations in the training set.

First, we evaluate the accuracy of AN-GCN on the Cora dataset. We hope that AN-GCN can classify nodes accurately while generating the positions from the noise; thus, the classification accuracy is used to evaluate the effectiveness of AN-GCN. As the generator does not directly generate the node embedding but predict nodes by cooperating with the discriminator, we denote acc_G (orange line) to be the classification accuracy through the position generated by G and denote acc_D (blue point) as the classification accuracy of the discriminator. Furthermore, we use acc_{GCN} (pink line) to denote the accuracy of single-layer GCN (the convolution kernel adopts symmetric normalized Laplacian matrix [38]) with the same kernel of D as a comparison. In addition, to present the advantages of AN-GCN more intuitively, we visualize the (embedded graph date orange box) X^E during the training phase. The results are shown in Fig. 12.

Second, we evaluate the robustness of AN-GCN by taking a comparative experimental model as Semi-GCN, GAT, and RGCN [28] (the state-of-the-art defense model). The results

TABLE II
CLASSIFICATION ACCURACY FOR VARIOUS MODELS AFTER META-LEARNING ATTACK

Dataset	*P_rate* %	GCN	GAT	RGCN	AN-GCN
Cora	0	80.02±0.40	**84.08±0.55**	83.19±0.64	82.63±0.59
	5	76.60±0.62	80.15±0.44	76.90±0.49	**80.92±0.43**
	10	70.52±1.33	75.03±0.35	72.35±0.42	**79.28±0.53**
	15	64.82±0.31	69.23±0.98	66.22±0.39	**79.10±0.60**
	20	59.02±2.52	59.03±1.03	59.01±0.29	**78.80±0.30**
	25	47.60±1.56	55.08±0.53	50.53±0.58	**77.90±0.70**
Citeseer	0	72.03±0.45	**73.63±0.62**	70.92±0.73	72.92±0.62
	5	71.88±0.32	**73.12±0.62**	71.51±0.63	72.53±0.43
	10	68.66±0.93	71.36±0.61	67.23±0.62	**72.28±0.61**
	15	65.51±0.59	69.98±0.32	66.09±0.82	**71.19±0.50**
	20	62.73±3.59	60.14±0.93	62.90±1.72	**70.01±0.61**
	25	56.73±2.50	60.90±1.36	56.43±0.72	**69.90±0.51**
Pubmed	0	**78.12±0.35**	73.97±1.57	77.95±0.31	78.02±0.29
	5	75.65±0.78	70.65±1.62	76.67±0.22	**77.81±0.08**
	10	74.24±0.40	69.02±1.42	74.56±0.38	**77.05±0.49**
	15	73.98±0.27	69.21±1.51	73.05±0.36	**76.25±0.46**
	20	70.02±0.22	67.89±1.35	71.23±0.24	**75.97±0.31**
	25	69.26±0.46	66.24±1.92	68.02±0.18	**75.04±0.39**
Polblogs	0	87.22±0.10	83.18±0.39	84.21±0.41	**87.57±0.13**
	5	82.65±0.23	79.45±0.43	79.85±0.34	**87.02±0.17**
	10	81.04±0.17	74.91±0.53	76.23±0.32	**86.52±0.30**
	15	78.24±0.21	71.15±0.41	73.62±0.24	**86.04±0.21**
	20	77.27±0.13	68.81±0.59	72.03±0.46	**85.89±0.41**
	25	76.52±0.09	64.08±0.33	70.51±0.34	**85.12±0.34**

are presented in Table II, and $P_$rate is the perturbation rate. The meta-learning attack [51] is the state-of-the-art attack model. The result shows that when the perturbation increases, the accuracy gap between AN-GCN and other methods widens, clearly demonstrating the advantage of AN-GCN.

In our previous work [52], we have proved that using samples with other categories as the supervised samples of the generated samples can accelerate the convergence of AT; similarly, AT in AN-GCN does not cause an excessive burden.

VII. Conclusion

In this article, we first generalized the formulation of edge-perturbing attacks and strictly proved the vulnerability of GCNs. Following this, we proposed the AN-GCN to defend against edge-perturbing attacks. In particular, a node localization theorem was presented to demonstrate how GCNs locate nodes during their training phase. Furthermore, a staggered Gaussian-based node position generator is designed to anonymize nodes' positions and a spectral graph convolution-based discriminator to ensure high-accuracy classification. Finally, we provided an optimization method for the designed generator and the discriminator. Extensive evaluations verified the effectiveness of the G-EPA model and demonstrated the high accuracy of the AN-GCN in node classification tasks. Future research includes extending our proposed AN-GCN to node classification tasks on dynamic graphs.

References

[1] S. E. Kreps and D. L. Kriner, "Model uncertainty, political contestation, and public trust in science: Evidence from the COVID-19 pandemic," *Sci. Adv.*, vol. 6, no. 43, p. eabd4563, Oct. 2020.

[2] W. Walt, C. Jack, and T. Christof, "Adversarial explanations for understanding image classification decisions and improved neural network robustness," *Nat. Mach. Intel.*, vol. 1, pp. 508–516, Nov. 2019.

[3] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, 2019.

[4] F. Monti, F. Frasca, D. Eynard, D. Mannion, and M. M. Bronstein, "Fake news detection on social media using geometric deep learning," 2019, *arXiv:1902.06673*.

[5] L. Zhang, D. Xu, A. Arnab, and P. H. S. Torr, "Dynamic graph message passing networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 3723–3732.

[6] Y. Chen *et al.*, "Practical attacks against graph-based clustering," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1125–1142.

[7] J. Ma, S. Ding, and Q. Mei, "Towards more practical adversarial attacks on graph neural networks," in *Proc. 34th Adv. Neural Inf. Proces. Syst.*, Dec. 2020, pp. 1–11.

[8] S. Ali, H. W. Ronny, S. Christoph, F. Soheil, and G. Tom, "Are adversarial examples inevitable?" in *Proc. 7th Int. Conf. Learn. Represent.*, May 2019, pp. 1–17.

[9] A. Chaturvedi and U. Garain, "Mimic and fool: A task-agnostic adversarial attack," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1801–1808, Apr. 2021.

[10] W. Yisen, Z. Difan, Y. Jinfeng, B. James, Q. G. Xingjun, and Ma, "Improving adversarial robustness requires revisiting misclassified examples," in *Proc. 8th Int. Conf. Learn. Represent.*, Apr. 2020, pp. 1–14.

[11] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang, "Adversarial training methods for network embedding," in *Proc. World Wide Web. Conf.*, May 2019, pp. 329–339.

[12] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 1025–1035.

[13] S. Geisler, D. Zügner, and S. Günnemann, "Reliable graph neural networks via robust aggregation," in *Proc. 34th Adv. Neural Inf. Process. Syst.*, Dec. 2020, pp. 13272–13284.

[14] Y. Li, W. Jin, H. Xu, and J. Tang, "DeepRobust: A platform for adversarial attacks and defenses," in *Proc. 35th AAAI Conf. Artif. Intell.*, Feb. 2021, pp. 16078–16080.

[15] H. Dai *et al.*, "Adversarial attack on graph structured data," in *Proc. 35th Int. Conf. Mach. Learn.*, Jul. 2018, pp. 1115–1124.

[16] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2018, pp. 2847–2856.

[17] Z. Lingxiao and A. Leman, "PairNorm: Tackling oversmoothing in GNNs," in *Proc. 8th Int. Conf. Learn. Represent.*, Apr. 2020, pp. 1–17.

[18] D. Wang *et al.*, "A semi-supervised graph attentive network for financial fraud detection," in *Proc. 19th IEEE Int. Conf. Data Min.*, Nov. 2019, pp. 598–607.

[19] Z. He *et al.*, "A spatiotemporal deep learning approach for unsupervised anomaly detection in cloud systems," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 16, 2020, doi: 10.1109/tnnls.2020.3027736.

[20] Y. Huang, Y. Weng, S. Yu, and X. Chen, "Diffusion convolutional recurrent neural network with rank influence learning for traffic forecasting," in *Proc. 18th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun., 13th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2019, pp. 678–685.

[21] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 3844–3852.

[22] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 695–704.

[23] H. Chang *et al.*, "A restricted black-box adversarial framework towards attacking graph embedding models," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 3389–3396.

[24] B. Wang and N. Z. Gong, "Attacking graph-based classification via manipulating the graph structure," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2023–2040.

[25] Z. Xi, R. Pang, S. Ji, and T. Wang, "Graph backdoor," in *Proc. 29th USENIX Secur. Symp.*, Aug. 2021, pp. 1523–1540.

[26] M. Jin, H. Chang, W. Zhu, and S. Sojoudi, "Power up! Robust graph convolutional network via graph powering," 2019, *arXiv:1905.10029*.

[27] D. Zügner and S. Günnemann, "Certifiable robustness and robust training for graph convolutional networks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 246–256.

[28] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust graph convolutional networks against adversarial attacks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1399–1407.

[29] Z. Deng, Y. Dong, and J. Zhu, "Batch virtual adversarial training for graph convolutional networks," 2019, *arXiv:1902.09192*.

[30] F. Feng, X. He, J. Tang, and T.-S. Chua, "Graph adversarial training: Dynamically regularizing based on graph structure," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 6, pp. 2493–2504, Jun. 2021.

[31] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, "Transferring robustness for graph neural network against poisoning attacks," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 600–608.

[32] L. Peiyuan *et al.*, "Information obfuscation of graph neural networks," in *Proc. 38th Int. Conf. Mach. Learn.*, Jul. 2021, pp. 6600–6610.

[33] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[34] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 5, pp. 833–852, May 2018.

[35] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Proc. 32nd Adv. Neural Inf. Proces. Syst.*, vol. 31, Dec. 2018, pp. 4558–4567.

[36] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (ACM)*, Aug. 2014, pp. 701–710.

[37] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. 5th Int. Conf. Learn. Represent.*, Apr. 2017, pp. 1–12.

[38] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, Apr. 2017, pp. 1–14.

[39] K. Zhou *et al.*, "Understanding and resolving performance degradation in graph convolutional networks," 2020, *arXiv:2006.07107*.

[40] A. Loukas, "What graph neural networks cannot learn: Depth vs width," in *Proc. 7th Int. Conf. Learn. Represent.*, May 2019, pp. 1–17.
[41] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 695–704.
[42] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 30, no. 3, pp. 83–98, Apr. 2013.
[43] S. Khorasani and A. Adibi, "Analytical solution of linear ordinary differential equations by differential transfer matrix method," *Electron. J. Differ. Equ.*, vol. 2003, no. 79, pp. 1–18, 2003.
[44] S. C. Pei, W. L. Hsue, and J. J. Ding, "Discrete fractional Fourier transform based on new nearly tridiagonal commuting matrices," *IEEE Trans. Signal Process.*, vol. 54, no. 10, pp. 3815–3828, Oct. 2006.
[45] Z. Chen, A. Luo, and X. Huang, "Euler's formula-based stability analysis for wideband harmonic resonances," *IEEE Trans. Ind. Electron.*, vol. 67, no. 11, pp. 9405–9417, Nov. 2019.
[46] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Represent.*, Apr. 2014, pp. 1–14.
[47] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," in *Proc. 33rd Adv. Neural Inf. Proces. Syst.*, vol. 32, Dec. 2019, pp. 1–12.
[48] Y. Sun, S. Wang, X. Tang, T.-Y. Hsieh, and V. Honavar, "Non-target-specific node injection attacks on graph neural networks: A hierarchical reinforcement learning approach," in *Proc. 29th Int. Conf. World Wide Web.*, vol. 3, Apr. 2020, pp. 1–11.
[49] A. Bojchevski and S. Günnemann, "Adversarial attacks on node embeddings via graph poisoning," in *Proc. 36th Int. Conf. Mach. Learn.*, Jun. 2019, pp. 695–704.
[50] L. Wang, Y. Zhang, and J. Feng, "On the Euclidean distance of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, pp. 1334–1339, Aug. 2005.
[51] D. Zügner and S. Günnemann, "Adversarial attacks on graph neural networks via meta learning," in *Proc. 6th Int. Conf. Learn. Represent.*, Apr. 2018, pp. 1–15.
[52] A. Liu, Y. Wang, and T. Li, "SFE-GACN: A novel unknown attack detection under insufficient data via intra categories generation in embedding space," *Comput. Secur.*, vol. 105, Jun. 2021, Art. no. 102262.

Ao Liu (Student Member, IEEE) is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China.

His research interests include graph convolutional networks, security and privacy for artificial intelligence, and cybersecurity.

Prof. Liu serves/served as the Reviewer for several international conferences, including the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), the International Conference on Computer Vision (ICCV), and the European Conference on Computer Vision (ECCV).

Beibei Li (Member, IEEE) received the B.E. degree (Hons.) in communication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2014, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2019.

He was a Visiting Researcher with the Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, in 2018, and the College of Control Science and Engineering, Zhejiang University, Hangzhou, China, in 2019. He is currently an Associate Professor (Doctoral Advisor) with the School of Cyber Science and Engineering, Sichuan University, Chengdu, China. His current research interests include several areas in security and privacy issues on cyber-physical systems, with a focus on intrusion detection techniques, artificial intelligence, and applied cryptography.

Dr. Li was a recipient of the Full Research Scholarship for his Ph.D. degree. He received the Best Paper Award in the 2021 IEEE Symposium on Computers and Communications (ISCC). He serves/served as the Publicity Chair, the Publication Co-Chair, or a Technical Program Committee Member for several international conferences, including the AAAI Conference on Artificial Intelligence (AAAI) in 2021, IEEE International Conference on Communications (ICC) in 2021, IEEE International Conference on Advanced and Trusted Computing (ATC) in 2021, and IEEE Global Communications Conference (GLOBECOM) in 2020.

Tao Li received the B.S. and M.S. degrees in computer science and the Ph.D. degree in circuit and system from the University of Electronic Science and Technology of China, Chengdu, China, in 1986, 1991, and 1995, respectively.

From 1994 to 1995, he was a Visiting Scholar in neural networks theory with the University of California at Berkeley, Berkeley, CA, USA. He is currently a Professor with the College of Cybersecurity, Sichuan University, Chengdu. His current research interests include network security, artificial immune theory, and disaster recovery application.

Pan Zhou (Senior Member, IEEE) received the B.S. degree (Hons.) from the Advanced Class of Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006, the M.S. degree from the Department of Electronics and Information Engineering, HUST, in 2008, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, in 2011.

He was a Senior Technical Member at Oracle Inc., Austin, TX, USA, from 2011 to 2013, where he was involved in Hadoop and distributed storage system for big data analytics at Oracle Cloud Platform. He is currently a Full Professor and a Ph.D. Advisor with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, HUST.

Dr. Zhou received the Merit Research Award of HUST for his master's degree, the Rising Star in Science and Technology of HUST in 2017, and the Best Scientific Paper Award in the 25th International Conference on Pattern Recognition (ICPR 2020). He is an Associate Editor of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

Rui Wang received the B.Sc. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2017, and the M.Sc. degree from the University of Southampton, Southampton, U.K., in 2018. He is currently pursuing the Ph.D. degree with the Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands, with a focus on privacy-preserving machine learning.