Robustness of Deep Learning Models on Graphs: A Survey

Jiarong Xu, Junru Chen, Siqi You, Zhiqing Xiao, Yang Yang, Jiangang Lu
Zhejiang University
{xujr, jrchen_cali, ysseven, zhiqing.xiao, yangya, lujg}@zju.edu.cn

Abstract

Machine learning (ML) technologies have achieved significant success in various downstream tasks, e.g., node classification, link prediction, community detection, graph classification and graph clustering. However, many studies have shown that the models built upon ML technologies are vulnerable to noises and adversarial attacks. A number of works have studied the robust models against noise or adversarial examples in image domains and text processing domains, however, it is more challenging to learn robust models in graph domains. Adding noises or perturbations to the graph data would make robustness more difficult to enhance – the noises and perturbations on edges or node features can easily be propagated to neighbors through the graph's relational information. In this paper, we investigate and summarize the existing works that study the robust deep learning models against adversarial attacks or noises on graphs, namely the robust learning (models) on graphs. Specifically, we first provide some robustness evaluation metrics of model robustness on graphs. Then, we comprehensively provide a taxonomy which groups robust models on graphs into five categories: anomaly detection, adversarial training, pre-processing, attention mechanism, and certifiable robustness. Besides, we emphasize some promising future directions in learning robust models on graphs. Hopefully, our works can offer insights for the relevant researchers, thus providing assistance for their studies.

1. Introduction

Machine learning (ML) technologies have become increasingly popular. They have attained impressive performances and successful applications on various downstream tasks such as image classification, object detection, traffic prediction, malware detection [1, 2, 3], speech recognition [4], automatic language translation [5], product recommendations [6, 7], self-driving vehicles [8], online fraud detection and stock market trading [9, 10], *etc.* Deep Neural Networks (DNNs), the most popular tool among machine learning technologies, are widely used in many real-world applications. However, many studies have shown that DNN models are not robust enough, that is, they are easily be fooled by noises or adversarial examples (that is, the examples that are carefully designed to deceive the models by making minor or even imperceptible modifications to benign examples). A line of existing works has shown that DNNs are vulnerable in many applications, such as malware detection [1, 2, 3], audio recognition [11], object recognition [12], sentiment analysis systems [13], *etc.* It is an urgent need to study robust models using machine learning technologies.

The graph-structured data is ubiquitous and plays a key role in many practical fields, including social network analysis, bioinformatics, chemistry, program analysis, etc. These graphs provide rich topology functions and common connectivity patterns, thus can help us better understand relational data. Deep learning on graphs has also achieved significant success in a wide range of applications [14], including financial surveillance [15], recommendation systems [16], molecule analysis [17] and drug discovery [18], *etc.* However, network data is hard to obtain and most networks obtained in the real world are error-prone and structurally flawed due to incomplete sampling [19], imperfect measurements [20, 21], individual non-response and dropout [22], *etc.* This will inevitably introduce many types of errors, including erroneous, ambiguous and redundant information [23]. Thus, most network data obtained depicts an imperfect and incomplete picture of topological structure. These inaccurate representation of networks can even have an adverse effect on how networks are interpreted

and damage information diffusion process, resulting in misleading conclusions. On the other hand, graph learning models, *e.g.*, Graph Neural Networks (GNNs) [24, 25, 26, 27, 17, 28, 29, 30] and network embedding [31, 32, 33, 34], have been shown to be vulnerable to adversarial examples [35, 36, 37, 38]. Adversarial attacks on graphs pose themselves as serious security challenges for many real-world systems. There have been lots of works focus on learning the robust models in image domains, but few have been studied the robustness of models on graphs. Hence, *it is of practical importance to build robust learning models on graphs against noises or adversarial attacks.*

There have been a few surveys mentioning the robustness of deep learning models on graphs [39, 40]. Although they provided their own categories of robust graph models, they did not include some important robustness metrics. In addition, anomaly detection, the most commonly-used approach to enhance robustness, is ignored in the previous surveys. In this survey, we first introduce the robustness metrics and then aim to summarize and discuss the robust learning models on graphs against noises and adversarial examples from a more comprehensive perspective. The major contributions can be summarized as follows:

- We target the critical yet overlooked robust models on graphs against noises and adversarial attacks.
- We provide some evaluation metrics of model robustness on graphs.
- We divide existing works of robust models on graphs into five categories: anomaly detection, adversarial training, pre-processing, attention mechanism, and certifiable robustness. We provide a detailed and systematic analysis of these studies.
- We present some exciting future directions of the model robustness on graphs.

Our manuscript is organized as follows. Some notations and backgrounds are mentioned in section 2. In section 3, we show some evaluation metrics of model robustness on graphs. In section 4, we introduce the five categories of robust models on graphs in detail. Some future directions are presented in section 5. We conclude our manuscript by providing a conclusion in section 6.

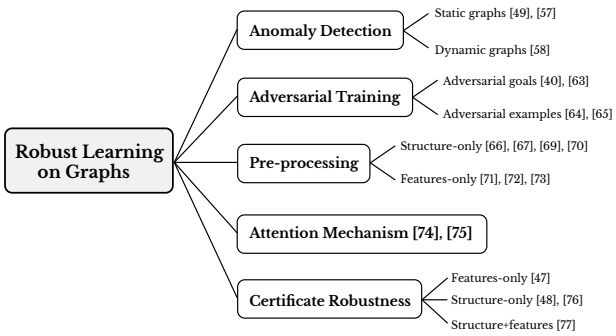

Figure 1: The category of robust learning models on graphs.

2. Preliminaries

2.1. Notations

Formally, we represent a network as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}|$ nodes, while $\mathcal{E}$ is the set of edges with $|\mathcal{E}|$ edges. We denote $\mathcal{A}$ as the adjacency matrix of $\mathcal{G}$ and $\mathcal{D}$ as the degree matrix of $\mathcal{A}$. We further augment $\mathcal{G}$ with the node feature matrix $\mathcal{X}$ if nodes have certain features in particular applications. Also in some applications where edges have features, we augment $\mathcal{G}$ with the edge feature matrix $\mathcal{H}$.

2.2. Victim models

In this survey, we use victim models to denote the models vulnerable to adversarial examples, also known as non-robust models. We mainly discuss two kinds of victim model, *i.e.*, GNN-based model and network embedding models.

The intuition of GNNs follows that of CNNs. It keeps aggregating and transforming the information from neighbor nodes to learn the representations for each node. Though GNNs have achieved impressive performance across many kinds of tasks, the vulnerability of GNNs has been demonstrated as potential threats to industry and society applications [41, 40].

Besides, another important graph learning algorithms, network embedding models, are also vulnerabile to adversarial examples. Such kind of models include LINE [42] and Deepwalk [32], and knowledge graph embedding [43].

2.3. Learning from graph data

In this section, we introduce the basic graph learning tasks: node classification and graph classifica-

tion. We use triple set $G = \{(c_i, \mathcal{G}_i, y_i)\}_{i\in[N]}$ to denote the training set with labels, where N is the number of the samples, c_i is the i-th sample and $\mathcal{G}_i$ and y_i represent the corresponding (sub)graph and the label of c_i, respectively. The loss function of conducting classification task is given below:

$$\min_{\theta} \mathcal{L}_{train}(f_\theta(G)) = \sum_{(c_i,\mathcal{G}_i,y_i)\in G} \ell\left(f_\theta(c_i, \mathcal{G}_i), y_i\right), \tag{1}$$

where f_θ is the mapping function learned to predict the true labels with learnable parameters θ. As for node-level classification task, each node lies in the same graph $\mathcal{G}_i = \mathcal{G} = (\mathcal{V}, \mathcal{E})$ and $f_\theta(c_i, \mathcal{G}_i) = f_\theta(\mathcal{G})_i$ extracts the i-th node's representation from the whole single graph. For graph-level classification task, each individual graph $\mathcal{G}_i = (\mathcal{V}_i, \mathcal{E}_i)$ has a label and $f_\theta(c_i, \mathcal{G}_i) = f_\theta(\mathcal{G}_i)$ extracts the i-th graph's representation independent with other graphs.

2.4. Adversarial attacks on graphs

In this section, we give a general form of the objective for graph adversarial attacks and illustrate the damage of the attacks which indicates the urgent need to research into robust models on graphs.

Graph adversarial attacks. In image domain, the attack is straightforward to introduce small perturbations into pixels (showed as Figure 2) which is a little different from that in graphs. As illustrated in Figure 3, the target of graph attack can be both graph topology and node features. Formally, based on the formula showed in Section 2.3, we can define the attack objective on graph data as:

$$\max_{\hat{G}\in\Phi(G)} \sum_{(c_i,\hat{\mathcal{G}}_i,y_i):i\in T(G)} \ell\left(f_{\theta^*}(c_i, \hat{\mathcal{G}}_i), y_i\right) \tag{2}$$

$$s.t. \quad \theta^* = \arg\min_{\theta} \mathcal{L}_{train}(f_\theta(G')), \tag{3}$$

where $\hat{G}$ denotes the perturbation set of G including adversarial graphs $\hat{\mathcal{G}}_i = (\hat{\mathcal{A}}_i, \hat{\mathcal{X}}_i)$. As for target set, $T(G) = G$ holds in untargeted setting and $T(G)$ consists of targeted samples in targeted attacks. $G' = G$ represents the evasion attacks while $G' = \hat{G}$ when the attacks are poisoned. Note that in most cases, the attacks should be limited in a constrained domain $\Phi(G)$ to ensure the perturbations are imperceptible. Formally, given the distance function d of $\mathcal{G}$ and the perturbation budget Δ, for any $\hat{\mathcal{G}}_i \in \Phi(G)$, $\hat{\mathcal{G}}_i$ should satisfy the constraint:

$$d(\hat{\mathcal{G}}_i, \mathcal{G}_i) \leq \Delta. \tag{4}$$

Why to study graph robust learning. Significant success in a large number of applications [14] has been promoted by deep learning on graphs, including molecule analysis [17], drug discovery [18], financial surveillance [15] and recommendation systems [16], etc. However, some works [36, 35, 37, 38] have exposed the potential danger that these approaches are vulnerable to adversarial examples. In other words, the models are easy to be deceived by the attacks that are carefully designed to them by making subtle or even human-incomprehensible modifications to benign examples. Therefore, adversarial attacks themselves are serious security challenges for many real-world systems and identifying the weaknesses of these graph learning models to make them more robust to different kinds of attacks are very urgent.

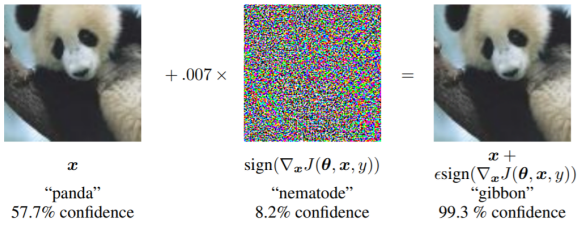

Figure 2: A demonstration of adversarial example in image domain. By injecting a small perturbation, "panda" is classified as "gibbon". (Image Credit: [12])

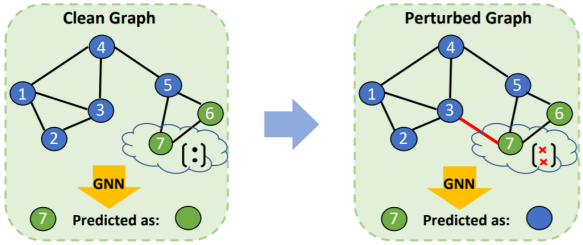

Figure 3: An illustration of adversarial example in graph structure. By creating a new connection between node 3 and node 7 and modifying the features, originally green node 7 is predicted as blue one. (Image Credit: [40])

3. Robustness metrics

We here introduce some metrics to measure the robustness of graph models. Note that in this section, we use $(\boldsymbol{a}, \boldsymbol{x})$ to denote an original example in the dataset, where $\boldsymbol{a}$ is the adjacency matrix and $\boldsymbol{x}$ is the node feature matrix.

Classification margin. Classification margin is a common metric to determine whether a node is correctly classified, which can also been used to measure the robustness of GNNs [44, 45]. This metric focuses on the label space which implies it changes for different downstream tasks. Besides, classification margin measures the robustness in a static perspective and the scope of investigation is limited in a dataset itself. For example, given a model, the most vulnerable example lies in the dataset which achieves the maximum value of the metric.

Definition 1 *(Classification margin.) Let y^* denote the class (using some ground truth) of example $(\boldsymbol{a}, \boldsymbol{x})$. The classification margin of $(\boldsymbol{a}, \boldsymbol{x})$ is*

$$CM(\boldsymbol{a}, \boldsymbol{x}, g, y^*) = \max_{y \in \mathcal{Y} \setminus \{y^*\}} \ln p(\hat{y} = y) - \ln p(\hat{y} = y^*),$$

where g is the classifier, $\hat{y} = g(\boldsymbol{a}, \boldsymbol{x})$ and $\mathcal{Y}$ denotes the label space. A smaller value of $CM(\boldsymbol{a}, \boldsymbol{x}, g, y^*)$ indicates a more robust g.

Adversarial risk and adversarial gap. The adversarial risk and the adversarial gap measure a given model's vulnerability to the input perturbations on graph structure and node features [46]. Different from the classification margin, these two metrics measure the robustness in a probability manner. More specifically, they will examine the continuous adversarial examples in a small budget, and they will consider the robustness of the encoder for the whole dataset instead of focusing on one specific example.

Definition 2 *Denote $(\mathcal{S}, d)$ as the input metric space. For any classifier $g : \mathcal{S} \to \mathcal{Y}$, given the adversarial budget $\tau \geq 0$, the **adversarial risk** of model g is*

$$\begin{aligned} Adv\_risk_\tau(g) \quad = \\ \mathbb{E}_{p(\boldsymbol{s}, y^*)}[\exists\ \boldsymbol{s}' = (\boldsymbol{a}', \boldsymbol{x}') \in \mathcal{B}(\boldsymbol{s}, \tau) \\ s.t.\ CM(\boldsymbol{a}', \boldsymbol{x}', g, y^*) \geq 0], \end{aligned}$$

*where $\mathcal{B}(\boldsymbol{s}, \tau) = \{\boldsymbol{s}' \in \mathcal{S} : d(\boldsymbol{s}', \boldsymbol{s}) \leq \tau\}$ represents the perturbation set of $\boldsymbol{s}$. Then, the **adversarial gap** is proposed to measure the relative vulnerability (based on adversarial risk) of a given model g w.r.t the adversarial budget τ, which can be defined as:*

$$Adv\_gap_\tau(g) = Adv\_risk_{\tau>0}(g) - Adv\_risk_{\tau=0}(g).$$

4. Robust models on graphs

The vulnerability of graph learning models poses major challenges to the reliable and secure applications on graphs. we target the critical but far overlooked problem of learning robust models on graphs.

In this section, we divide existing works of robust models on graph into the following five categories, *i.e.*, (1) anomaly detection, (2) adversarial training, (3) pre-processing, (4) attention mechanism, and (5) certifiable robustness. Due to its importance and wide range of applications [47, 48], we specifically classify anomaly detection as a category; while others are mainly divided based on the technical characteristics.

In more details, anomaly detection and preprocessing methods are both used to correct the underlying attacked graph and obtain a more robust model training on the fixed graph. Both methods can defend poisoning attacks through identifying the attack methods or utilizing some prior assumptions to refine the graph. However, the methods are not in an end-to-end manner which is more time-consuming in the inference stage. As for attention mechanism, it aims to decrease the negative influence of attacks during the aggregation process in the presence of adversarial attacks. But this will cause extra learnable parameters and processing time to infer the downstream tasks. Furthermore, adversarial training and certifiable robustness apply different strategies to generate attacks from clean graphs to train the robust models on them, which is from an attacking-free perspective.

4.1. Anomaly detection

Anomaly detection is one of the most straightforward ways to enhance the robustness of models and systems. The main idea of anomaly detection is to identify rare and unusual patterns which significantly differ from the majority of data. There are usually two main categories of anomalies in anomaly detection [47]:

- *Point anomalies.* Anomaly detection on point anomalies means to detect an individual anomalous data sample only respect to some of other data samples.

- *Contextual or collective anomalies.* Anomaly detection on contextual or collective anomalies means to detect a set of related or conditional anomalous data samples respect to the entire graph.

Within anomaly detection methods, identifying and removing anomalies from the source of data can increase robustness and reliability of models and systems constructed on these data. Many technologies in anomaly detection have been widely used in a number of real-world applications, *e.g.*, fraud detection [49, 50], game bot detection [2, 3], intrusion detection [51], fault detection [52], novelty detection [53].

Considering the inter-dependent and relational nature of graph-structured data, the anomaly information will propagate from nodes to their neighbors, leading to more destructive results. Hence, anomaly detection on graphs is much more challenging.

Graph anomaly detection techniques can effectively protect graph data from graph adversarial attacks by exploring the intrinsic difference between adversarial structures and the clean ones [54]. There are four methods to distinguish graph adversarial attacks and help correctly detect adversarial perturbations [40], *i.e.*, (1) link prediction, (2) sub-graph link prediction, (3) graph generation, and (4) outlier detection.

Existing works of anomaly detection on graphs mainly focus on dealing with static graphs and dynamic graphs [48]:

- *Anomaly detection on static graphs.* Given the snapshot of a graph database, the objective is to find the nodes, edges or sub-graphs that are rare and unusual in the graph.

- *Anomaly detection on dynamic graphs.* Given a sequence of graphs, the objective is to find the timestamps that correspond to a change, as well as the top-k nodes, edges or sub-graphs that contribute most to the change.

There exist plenty of works on static graphs. Jiang et al. [47] design a graph convolution network model to detect both anomalous behaviors of individual users and associated malicious threat groups. As shown in Figure 4, this model can characterize entities' properties as well as structural information between them into graphs, because only considering entities' properties information easily leads to high false positives. Because traditional anomaly

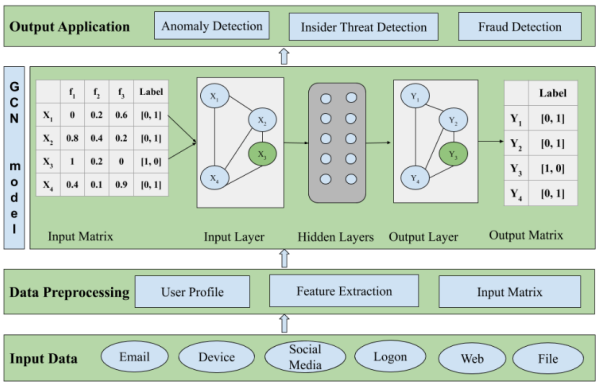

Figure 4: Graph convolution network model for anomaly detection using graph data as input. (Image Credit: [47])

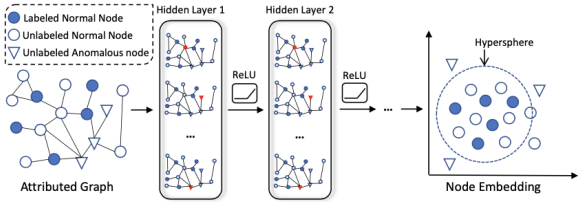

Figure 5: The overall framework of OCGNN. (Image Credit: [55])

detection methods such as one-class support vector machine (OCSVM) lost their effectiveness in graph data, Wang et al. [55] propose one-class graph neural network (OCGNN) to combine the powerful representation ability of graph neural networks along with the classical one-class objective. As illustrated in Figure 5, this hypersphere learning framework is a natural extension of OCSVM in the field of graph data.

Compared with static graphs, there exist only a few works on spotting anomalies by exploiting dynamic attributed graphs. Du et al. [56] propose a deep neural network model, named DeepLog, utilizing Long Short-Term Memory (LSTM) to model a system log as a natural language sequence. The model architecture is shown in Figure 6. DeepLog can automatically learn log patterns from normal execution and detect anomalies when log patterns deviate from the model trained from log data under normal execution. In addition, DeepLog is able to adapt to new log patterns over time and construct workflows from the underlying system log.

Even though there have been plenty of works in developing graph-based abnormality detection

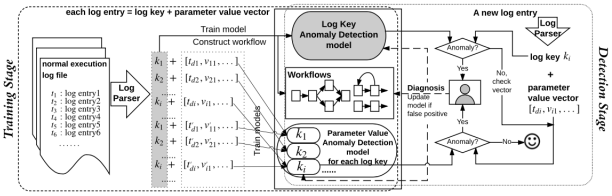

Figure 6: The model architecture of DeepLog. (Image Credit: [56])

problems and algorithms, there are still some limitations of anomaly detection. In theoretical research, there exist only a few works on spotting anomalies by exploiting dynamic attributed graphs compared to plenty of works on static graphs. From systems perspective, most methods focus too much on detection performance while ignoring adversarial robustness. In view of practice, it is often hard to predict what would boost a detection algorithm's performance the most, the methods are not end-to-end and ground truth data is often inexistent.

4.2. Adversarial training

Adversarial training is an important way to enhance the robustness of neural network. The main idea of adversarial training is to insert slight perturbations into the training set and then retrain the model, which normally has good performance on clean data.

In the image classification scenario, as illustrated in Figure 2, these adversarial examples which look like the original images can fool the network. Generally, these results have often been interpreted as being a flaw in deep networks [12]. So studying adversarial examples in image data is thought to be extremely important. There are some training methods, such as FGSM [12], Fast [57], TRADES [58], YOPO [59].

In graph domains, the attacker can modify the graph structure or node features to generate graph adversarial perturbations to mislead the prediction of GNN models. Since adversarial training has already been widely used in the image data, we can also take this strategy into consideration to defend graph adversarial attacks. There are two types of adversarial training: The first one is training with adversarial goals. Some adversarial training methods gradually optimize the model in a continuous min-max method under the guidance of two opposite (minimize and maximize) objective functions, as shown below [40, 60],

$$\min_{\theta} \max_{\delta_A \in \mathcal{P}_A, \delta_X \in \mathcal{P}_X} \mathcal{L}_{train}\left(f_\theta\left(A+\delta_A, X+\delta_X\right)\right). \tag{5}$$

where δ_A, δ_X represent the perturbation added to A, X, respectively; $\mathcal{P}_A$, $\mathcal{P}_X$ denote the areas of unnoticeable perturbation.The min-max optimization problem in Eq (5) shows that graph adversarial training includes two processes: (1) maximize the prediction loss by adding perturbations and (2) minimize the prediction loss by retraining model to update parameters. Through the above two processes, we can get a robust model. Since there are two inputs, i.e., adjacency matrix A and feature matrix X, adversarial training can be done on them separately. The second one is training with adversarial examples. During the training process, other models based on the adversarial model are provided to the adversarial samples, which helps the model learn and adjust to adapt to the adversarial samples, thereby reducing the impact of these potential attack samples. For instance, Deng et al. [61] proposed batch virtual adversarial training (BVAT) algorithms, which aim to generate virtual adversarial perturbations to perceive the connectivity patterns between nodes in the graph to improve the smoothness of the output distribution of the node classifier (shown in Figure 7 and Figure 8). Chen et al. [62] proposed two special adversarial training strategies: global adversarial training (Global-AT) that for all nodes protection and target label adversarial training (Target-AT) that can protect the target labeled nodes from attack. In Global-AT, we select the target pair of nodes firstly, then update the adjacency matrix $\hat{A}^{t-1}$ of the $(t-1)$ adversarial network and get the adjacency matrix $\hat{A}^t$:

$$\hat{A}^t_{ij} = \hat{A}^{t-1}_{ij} + \theta_{ij}. \tag{6}$$

where $\hat{A}^t_{ij}$ and $\hat{A}^{t-1}_{ij}$ are the elements of $\hat{A}^t$ and $\hat{A}^{t-1}$. Target-AT only consider the target labeled nodes, and use the link selected by adversarial network attack to update the adversarial network.

4.3. Pre-processing

The adversarial training-based methods only aim at resisting evasion attacks, *i.e.*, the attacks occur during the test time. While the pre-processing method like purifying the perturbed graph data can deal with poisoning attacks, i.e., the attacks insert several fake samples into the training set. In the poisoning attacks, the attackers tend to add edges

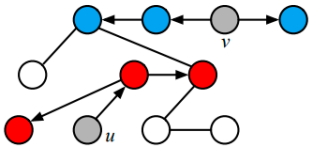

Figure 7: In sample-based BVAT (S-BVAT), two nodes u and v are selected to calculate the LDS loss, and the virtual adversarial perturbation is applied to the elements that have no intersection in its acceptance area (marked in red and blue). (Image Credit: [61])

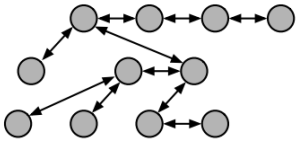

Figure 8: In optimization-based BVAT (O-BVAT), all nodes are included to calculate LDS loss, and the virtual adversarial perturbation of all nodes is optimized together. (Image Credit: [61])

rather than remove edges or modify features, and prefer to connect nodes with dissimilar features. Such perturbations would hurt the performance on the test data. Therefore, pre-processing or purifying the perturbed graph, and then learning from the perturbed graph can enhance performance against poisoning attacks.

Xu et al. [63] proposed different methods based on the graph generation model, and used link prediction as pre-processing to detect potential malicious edges. Zhang et al. [64] focused on the problem of detecting nodes which have been subject to topological perturbations calculated by the Nettack [65]. Through observing the discrepancy between the first-order proximity information of v_i and the neighbours of v_i which created by Nettack, they using a relatively simple threshold test find the Nettack perturbations on GCN.

Similarly, in order to discard the high-rank perturbations generated by Nettack, Entezari et al. [66] proposed the low-rank approximation and then retrain GCN with the low-rank approximation matrices(See Figure 9).

Except for the above mentioned, GraphSAC filters out sets contaminated by abnormal nodes based on the graph-aware criterion calculated on a subset of nodes randomly, the formula is given as below [67]:

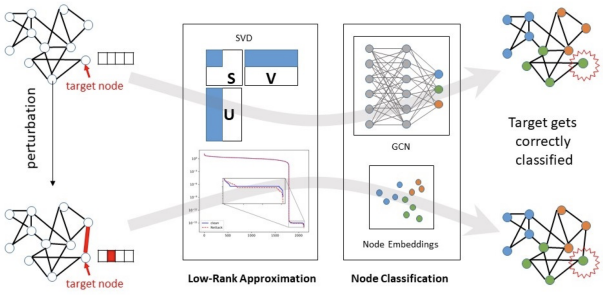

Figure 9: The overall system: low-rank approximation of graph structure and feature matrices to vaccinate the node classification method and discard high-rank perturbations. (Image Credit: [66])

$$\hat{\mathbf{P}} = f\left(\{y_n\}_{n\in\mathcal{L}}, \mathbf{A}\right), \tag{7}$$

where y_n are sample labels at random subsets of nodes $n \in \mathcal{L} \subset \mathcal{V}$, $\mathbf{A}$ is the graph connectivity, and $\hat{\mathbf{P}}_{(n,c)} \in [0,1]$ can be denoted as the probability that $y_n = c$. The choice of $f(\cdot)$ is determined by the specific features it wants to capture. Then, GraphSAC compares the accuracy of $f(\cdot)$ using the ratio of nodes in the consensus set to a prespecified threshold T to judge it whether contain anomalies.

These models can only resist the attacks on graph structure, but cannot resist the perturbations on node features. As for the node features, attackers prefer to add edges between dissimilar nodes. Based on the findings, Xu et al. [68] utilized outlier detection approaches to filter the adversarial edges. Wu et al. [69] proposed to remove the edges whose end nodes with small Jaccard similarity. The Jaccard similarity score is defined as [70]:

$$J_{u,v} = \frac{M_{11}}{M_{01} + M_{10} + M_{11}}. \tag{8}$$

where M_{11} represents the number of features where both node u and node v have a value of 1. Similarly, M_{10}, M_{01}, M_{00} represent the number of feature values of node u and node v, 1 and 0, 0 and 1, 0 and 0, respectively.

4.4. Attention mechanism

Different from pre-processing methods which try to purify the perturbed graph data to enhance the robustness of GNN models, attention-based models aim to improve the robustness of GNNs in the presence of adversarial attacks. More specifically, the attention mechanism is designed to assign high

confidence to the clean edges, and assigh low confidence to the adversarial ones. When aggregating the information from neighbors, the learned attention weights will penalize the perturbed part of data by reducing their contributions during the propagation process.

RGCN [71] assumed that the prediction uncertainty of adversarial nodes is high. As shown in figure 10, since the plain vectors cannot adapt to the abnormal changes, RGCN proposes to model the hidden representations of nodes in all graph convolutional layers as Gaussian distributions to automatically reflect the effects of adversarial changes in the variances. As a result, the variance-based attention mechanism will penalize the nodes with high variance to help mitigate the propagation of negative impact caused by adversarial examples. The attention weights of node v_j in layer l are defined as

$$\boldsymbol{\alpha}_j^{(l)} = \exp\left(-\gamma \boldsymbol{\sigma}_j^{(l)}\right), \tag{9}$$

where $\boldsymbol{\sigma}_j^{(l)}$ denotes the variance and γ is a hyper-parameter.

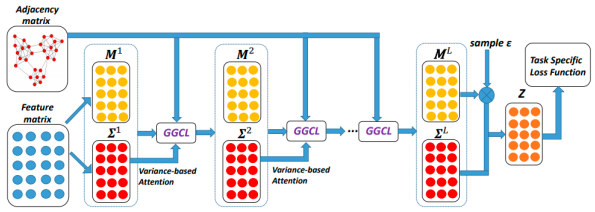

Figure 10: The framework of RGCN. And the GGCL represents the Gaussian-base graph convolutional layer. (Image Credit: [71])

PA-GNN [72] introduces the supervised information about real perturbations in a poisoned graph to help improve the robustness of target GNN models. The intuition is from the fact that there usually exist clean graphs sharing the similar structural distributions and node features with the poisoned graph. For instance, co-review networks like Yelp and Foursquare and social networks like Facebook and Twitter both share similar domains. Therefore, PA-GNN first learns to discriminate adversarial edges generated by attacking the clean graphs with supervised knowledge of known perturbations. With supervision knowledge, PA-GNN designs a loss function to guarantee less attention weights for adversarial edges as

$$\mathcal{L}_{dist} = -\min\left(\eta, \mathop{\mathbb{E}}_{e_{ij}\in\mathcal{E}\setminus\mathcal{P}} \boldsymbol{\alpha}_{ij}^l - \mathop{\mathbb{E}}_{e_{ij}\in\mathcal{P}} \boldsymbol{\alpha}_{ij}^l\right), \tag{10}$$

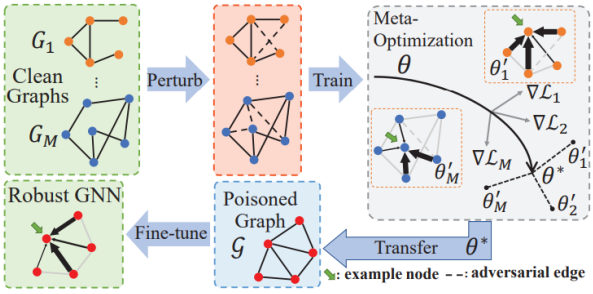

Figure 11: Overall framework of PA-GNN. Thicker arrows indicate higher attention coefficients. θ^* denotes the model initialization from meta-optimization. (Image Credit: [72])

where $\mathcal{E}$ and $\mathcal{P}$ represents the set of all edges and that of perturbed edges, $\boldsymbol{\alpha}_{ij}^l$ denotes the self-attention coefficient assigned for e_{ij} on the l-th layer, η is a hyper-parameter to trade-off the margin between the expectations of two distributions. Then, a meta-optimization algorithm is proposed to learn the initialization of PA-GNN, and the model is further fine-tuned on the poisoned graph to enhance robustness.

4.5. Certifiable robustness

In most previous works, the robustness of GNNs is exploited heuristically and experimentally. However, the criteria of measuring the safety of input graphs under adversarial perturbation is not solved in the previous works. Therefore, to research the problem that how to verify that small perturbations to input data will not cause dramatic effect to a GNN is important.

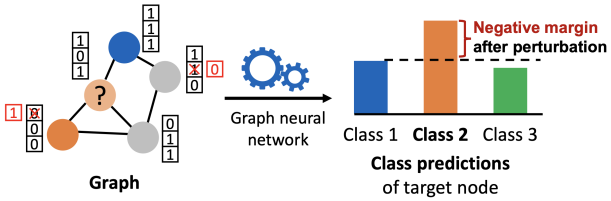

Figure 12: Intuitive idea of the classification margin [44].

In [44], they try to derive an efficient principle for robustness certificates. More specifically, they want to provide a certificate to measure that for which nodes the given trained GNN can guarantee that the predictions will not change under any ad-

missible perturbations given a specific attack budget (see Figure 13). To tackle this problem, they aim to find the worst case margin (see Figure 12) for the node t under some set $\mathcal{X}_{q,Q}(\dot{X})$ of admissible perturbations to the node features:

$$m^t(y^*, y) := \min_{\tilde{X}} f_\theta^t(\tilde{X}, \dot{A})_{y^*} - f_\theta^t(\tilde{X}, \dot{A})_y \quad (11)$$

$$\text{s.t. } \tilde{X} \in \mathcal{X}_{q,Q}(\dot{X}), \quad (12)$$

where y^* denotes the class of node t given by the ground truth or predicted and $f_\theta^t(\cdot)$ represents the classifier which outputs the logits of each class. It is easy to see that the GNN is certifiably robust $w.r.t$ node t when $m^t(y^*, y) > 0$ for all $y \neq y^*$, which means that there do not exist any adversarial examples that can change the prediction of node t. Through some relaxations, they obtain a lower bound of $m^t(y^*, y)$ which is tractable to calculate. Thus, they can use this certificate to find how many nodes in a graph is certifiably safe. Furthermore, the certificate can be taken as the objective to help more nodes safer through maximizing the worst case margin.

However, Zügner et al. [44] only considers perturbations to the node features. Bojchevski et al. [45] is completely orthogonal to [44], since they only consider the adversarial perturbations to the graph structure. This work derives the robustness certificates for the models whose prediction is a linear PageRank function. Based on the observation, under any admissible perturbation $\tilde{G} \in \mathcal{Q}_\mathcal{F}$, the work transforms robustness certificates to the worst-case margin of node t between the class y_t and the class c as:

$$m^*_{y_t,c}(t) = \min_{\tilde{G} \in \mathcal{Q}_\mathcal{F}} m_{y_t,c}(t) \quad (13)$$

$$= \min_{\tilde{G} \in \mathcal{Q}_\mathcal{F}} \boldsymbol{\pi}_{\tilde{G}}(\boldsymbol{e}_t)^T (\boldsymbol{H}_{:,y_t} - \boldsymbol{H}_{:,c}), \quad (14)$$

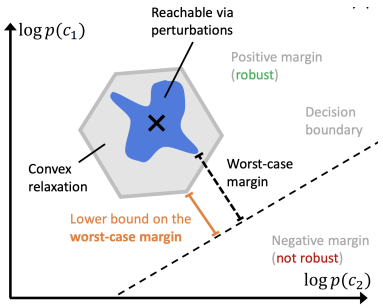

Figure 13: Illustration of certifiable robustness on graphs [44].

where $\boldsymbol{H}_{:,y_t}$ and $\boldsymbol{H}_{:,c}$ denote the prediction logits vectors of class y_t and class c respectively. And $\boldsymbol{\pi}_{\tilde{G}}(\boldsymbol{e}_t)$ is the personalized PageRank vector of node t. Then, in order to maximize the margin, they aim to find the fragile edges to obtain a perturbed graph $\tilde{G}$. As an extension of [45], Zügner et al. [73] covers the highly important principle of graph convolutional networks. They rephrases the objective function as a jointly constrained bilinear program to make the optimization tractable.

Bojchevski et al. [74] proposes an approach that can handle both the perturbations on graph structure and node features, which can be applied to any GNNs utilizing randomized smoothing framework. In this framework, the certificate is defined as:

$$\rho_{\boldsymbol{x},\tilde{\boldsymbol{x}}}(p, y) = \min_{\substack{h \in \mathcal{H}: \\ \Pr(h(\phi(\boldsymbol{x}))=y)=p}} \Pr(h(\phi(\tilde{\boldsymbol{x}})) = y), \quad (15)$$

where $\tilde{\boldsymbol{x}}$ is the neighboring point, ϕ is the randomization scheme, $\mathcal{H}$ is the set of classifiers $w.r.t$ ϕ, and h is a base classifier outputting a single prediction class. Based on which, they define the data-dependent sparsity-aware noise distribution:

$$\Pr(\phi(\boldsymbol{x})_i \neq \boldsymbol{x}_i) = p_-^{\boldsymbol{x}_i} p_+^{(1-\boldsymbol{x}_i)}, \quad (16)$$

The randomization scheme ϕ would delete an existing edge with probability p_-, and add a non-existence edge with probability p_+. Through theoretic analysis and further relaxation, they conduct experiments to verify the effectiveness and efficiency of their algorithm. This work also gives the certificates for graph-level classification models for the first time. Except for the node classification and graph classification tasks, there exist some works concentrating on the certifiable robustness of some other tasks, like community detection [75].

5. Future directions

We have thoroughly investigated robust models on graphs and gained an overview of this emerging research field, robust learning on graphs. However, there still exist some promising research directions that are less explored.

- *Graph data.* Compared with considerable amount of work on static graphs, there still remain problems on dynamic graphs, *e.g.*, detecting anomalies on attributed dynamic graphs, work with the trace of edge or node updates. Besides, when it comes to an explicit graph representation, to add or remove latent

edges may also be possible, that is, augmented graphs, *e.g.*, edges based on similarities or domain knowledge.

- *Graph construction.* The data does not form a network or there is more than one network available. To use graph-based techniques, how to use the source of data to construct a best representation, a graph or multi-graphs, remains an open problem.

- *Balance performance.* Most methods focus on anomaly detection performance while ignoring adversarial robustness. How to balance detection performance and the robustness of models is still an open challenge.

- *Evaluation.* Ground truth data is often inexistent and it is difficult for humans to tell whether the adversarial perturbations on graph data are imperceptible or not, thus to find concise evaluation measure is urgent.

6. Conclusion

In the survey, we conduct an overall review on robust learning models on graphs. Specifically, we present the recent developments of this area, we first provide some robustness evaluation metrics of model robustness on graphs, and then comprehensively divide existing works of robust models on graphs into five categories: anomaly detection, adversarial training, pre-processing, attention mechanism, and certifiable robustness. Besides, we further emphasize some potential future directions in learning robust models on graphs.

We hope that our work can serve as a reference for researchers to get a systematical and comprehensive understanding of robust models on graph, thus providing more insights for their studies.

References

[1] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *European Symposium on Research in Computer Security*, pp. 62–79, Springer, 2017.

[2] J. Tao, J. Xu, L. Gong, Y. Li, C. Fan, and Z. Zhao, "Nguard: A game bot detection framework for netease mmorpgs," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 811–820, 2018.

[3] J. Xu, Y. Luo, J. Tao, C. Fan, Z. Zhao, and J. Lu, "Nguard+ an attention-based game bot detection framework via player behavior sequences," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 14, no. 6, pp. 1–24, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[5] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

[6] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, *et al.*, "Wide & deep learning for recommender systems," in *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

[7] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: a factorization-machine based neural network for ctr prediction," *arXiv preprint arXiv:1703.04247*, 2017.

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[9] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *Proceedings of the 16th international conference on World Wide Web*, pp. 201–210, 2007.

[10] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, "Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques," *Expert systems with applications*, vol. 42, no. 1, pp. 259–268, 2015.

[11] N. Carlini and D. Wagner, "Audio adversarial examples: Targeted attacks on speech-to-text," in *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 1–7, IEEE, 2018.

[12] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[13] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "Hotflip: White-box adversarial examples for text classification," *arXiv preprint arXiv:1712.06751*, 2017.

[14] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl. Based Syst.*, vol. 151, 2018.

[15] A. Paranjape, A. R. Benson, and J. Leskovec, "Motifs in Temporal Networks," in *Proceedings of the 10th ACM International Conference on Web Search and Data Mining, WSDM 2017, Cambridge, United Kingdom, February 6-10, 2017*, 2017.

[16] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, 2019.

[17] 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.

[18] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, 2017.

[19] K. Gueorgi, "Effects of missing data in social networks," *Social Networks*, vol. 28, no. 3, pp. 247 – 268, 2006.

[20] C. T. Butts, "Network inference, error, and informant (in)accuracy: a bayesian approach," *Social Networks*, vol. 25, no. 2, pp. 103 – 140, 2003.

[21] G. M. S. Namata, Jr. and L. Getoor, "Identifying graphs from noisy and incomplete data," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'09, (New York, NY, USA), pp. 23–29, ACM, 2009.

[22] J. Schafer and J. Graham, "Missing data: Our view of the state of the art," *Psychological Methods*, vol. 7, pp. 147–177, 06 2002.

[23] J. Xu, Y. Yang, C. Wang, Z. Liu, J. Zhang, L. Chen, and J. Lu, "Robust network enhancement from flawed networks," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2020.

[24] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[25] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," *arXiv preprint arXiv:1506.05163*, 2015.

[26] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, pp. 3844–3852, 2016.

[27] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[28] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2017.

[29] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023, 2016.

[30] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *AAAI*, vol. 16, pp. 1145–1152, 2016.

[31] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo, "struc2vec: Learning node representations from structural identity," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 385–394, 2017.

[32] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, (New York, NY, USA), pp. 701–710, ACM, 2014.

[33] F. Zhang, X. Liu, J. Tang, Y. Dong, P. Yao, J. Zhang, X. Gu, Y. Wang, B. Shao, R. Li, *et al.*, "Oag: Toward linking large-scale heterogeneous entity graphs," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2585–2595, 2019.

[34] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.

[35] L. Chen, J. Li, J. Peng, T. Xie, Z. Cao, K. Xu, X. He, and Z. Zheng, "A Survey of Adversarial Learning on Graphs," *ArXiv*, 2020.

[36] 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*, 2019.

[37] 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*, 2018.

[38] 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*, 2019.

[39] L. Sun, J. Wang, P. S. Yu, and B. Li, "Adversarial attack and defense on graph data: A survey," *CoRR*, vol. abs/1812.10528, 2018.

[40] W. Jin, Y. Li, H. Xu, Y. Wang, and J. Tang, "Adversarial attacks and defenses on graphs: A review and empirical study," *arXiv preprint arXiv:2003.00653*, 2020.

[41] 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.

[42] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077, 2015.

[43] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in neural information processing systems*, pp. 2787–2795, 2013.

[44] D. Zügner and S. Günnemann, "Certifiable robustness and robust training for graph convolutional networks," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 246–256, 2019.

[45] A. Bojchevski and S. Günnemann, "Certifiable robustness to graph perturbations," in *Advances in Neural Information Processing Systems*, pp. 8319–8330, 2019.

[46] S. Zhu, X. Zhang, and D. Evans, "Learning adversarially robust representations via worst-case mutual information maximization," in *International Conference on Machine Learning (ICML)*, 2020.

[47] J. Jiang, J. Chen, T. Gu, K. K. R. Choo, C. Liu, M. Yu, W. Huang, and P. Mohapatra, "Anomaly detection with graph convolutional networks for insider threat and fraud detection," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2020.

[48] L. Akoglu, H. Tong, and D. Koutra, "Graph-based anomaly detection and description: A survey," *Data Mining & Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, 2015.

[49] Y. Yang, Y. Xu, C. Wang, Y. Sun, F. Wu, Y. Zhuang, and M. Gu, "Understanding default behavior in online lending," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2043–2052, 2019.

[50] Y. Yang, Y. Xu, Y. Sun, Y. Dong, F. Wu, and Y. T. Zhuang, "Mining fraudsters and fraudulent strategies in large-scale mobile social networks," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[51] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[52] D. Miljković, "Fault detection methods: A literature survey.," pp. 750–755, 05 2011.

[53] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215 – 249, 2014.

[54] V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, "Graphsac: Detecting anomalies in large-scale graphs," 2019.

[55] X. Wang, B. Jin, Y. Du, P. Cui, and Y. Yang, "One-class graph neural networks for anomaly detection in attributed networks," 2020.

[56] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: Anomaly detection and diagnosis from system logs through deep learning," in *Acm Sigsac Conference on Computer & Communications Security*, 2017.

[57] E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.

[58] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, "Theoretically principled trade-off between robustness and accuracy," *arXiv preprint arXiv:1901.08573*, 2019.

[59] D. Zhang, T. Zhang, Y. Lu, Z. Zhu, and B. Dong, "You only propagate once: Painless adversarial training using maximal principle," *arXiv preprint arXiv:1905.00877*, vol. 2, no. 3, 2019.

[60] Y. Li, W. Jin, H. Xu, and J. Tang, "Deeprobust: A pytorch library for adversarial attacks and defenses," *arXiv preprint arXiv:2005.06149*, 2020.

[61] Z. Deng, Y. Dong, and J. Zhu, "Batch virtual adversarial training for graph convolutional networks," 2019.

[62] J. Chen, Y. Wu, X. Lin, and Q. Xuan, "Can adversarial network attack be defended?," *CoRR*, vol. abs/1903.05994, 2019.

[63] X. Xu, Y. Yu, B. Li, L. Song, C. Liu, and C. Gunter, "Characterizing malicious edges targeting on graph neural networks," 2018.

[64] Y. Zhang, S. Khan, and M. Coates, "Comparing and detecting adversarial attacks for graph deep learning," in *Proc. Representation Learning on Graphs and Manifolds Workshop, Int. Conf. Learning Representations, New Orleans, LA, USA*, 2019.

[65] 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*, pp. 2847–2856, 2018.

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

[67] V. N. Ioannidis, D. Berberidis, and G. B. Giannakis, "Graphsac: Detecting anomalies in large-scale graphs," *arXiv preprint arXiv:1910.09589*, 2019.

[68] X. Xu, Y. Yu, L. Song, C. Liu, B. Kailkhura, C. Gunter, and B. Li, "Edog: Adversarial edge detection for graph neural networks," tech. rep., Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2020.

[69] H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, and L. Zhu, "Adversarial examples on graph data: Deep insights into attack and defense," *arXiv preprint arXiv:1903.01610*, 2019.

[70] A. Said, E. W. De Luca, and S. Albayrak, "How social relationships affect user similarities," in *Proc. of the 2010 workshop on social recommender systems*, pp. 1–4, 2010.

[71] 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*, pp. 1399–1407, 2019.

[72] X. Tang, Y. Li, Y. Sun, H. Yao, P. Mitra, and S. Wang, "Transferring robustness for graph neural network against poisoning attacks," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 600–608, 2020.

[73] D. Zügner and S. Günnemann, "Certifiable robustness of graph convolutional networks under structure perturbations," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1656–1665, 2020.

[74] A. Bojchevski, J. Klicpera, and S. Günnemann, "Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more," in *International Conference on Machine Learning (ICML)*, pp. 11647–11657, 2020.

[75] J. Jia, B. Wang, X. Cao, and N. Z. Gong, "Certified robustness of community detection against adversarial structural perturbation via randomized smoothing," in *Proceedings of The Web Conference 2020*, pp. 2718–2724, 2020.