# Detection and Defense of Topological Adversarial Attacks on Graphs

**Yingxue Zhang**[1] **Florence Regol**[2*] **Soumyasundar Pal**[2*] **Sakif Khan**[1]
**Liheng Ma**[2*] **Mark Coates**[2]
Huawei Noah's Ark Lab[1]
Montreal Research Center
McGill University[2]
Department of Electrical and Computer Engineering

## Abstract

Graph neural network (GNN) models achieve superior performance when classifying nodes in graph-structured data. Given that state-of-the-art GNNs share many similarities with their CNN cousins and that CNNs suffer adversarial vulnerabilities, there has also been interest in exploring analogous vulnerabilities in GNNs. Indeed, recent work has demonstrated that node classification performance of several graph models, including the popular graph convolution network (GCN) model, can be severely degraded through adversarial perturbations to the graph structure and the node features. In this work, we take a first step towards detecting adversarial attacks against graph models. We first propose a straightforward single node threshold test for detecting nodes subject to targeted attacks. Subsequently, we describe a kernel-based two-sample test for detecting whether a given subset of nodes within a graph has been maliciously corrupted. The efficacy of our algorithms is established via thorough experiments using commonly used node classification benchmark datasets. We also illustrate the potential practical benefit of our detection method by demonstrating its application to a real-world Bitcoin transaction network.

## 1 Introduction

Graph neural networks (GNNs) have been successful for a wide variety of machine learning tasks for graph-

*Work done as intern at Huawei Noah's Ark Lab, Montreal Research Center. Correspondence to: yingxue.zhang@huawei.com.

Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130. Copyright 2021 by the author(s).

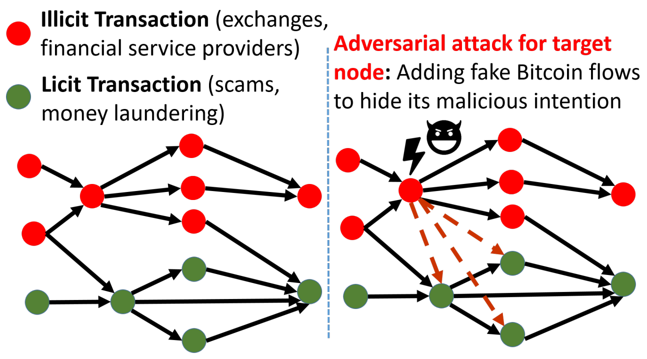
● **Illicit Transaction** (exchanges, financial service providers)
● **Licit Transaction** (scams, money laundering)
**Adversarial attack for target node:** Adding fake Bitcoin flows to hide its malicious intention

**Figure 1:** Bitcoin transaction network. Nodes are transactions; edges indicate flow of bitcoin between transactions.

structured data. With GNNs already being applied to tackle practical problems (Hamilton et al., 2017; Zhang et al., 2019; Ying et al., 2018; Sun et al., 2019; Guo et al., 2019; Xu et al., 2020), there is a concern that GNNs may exhibit the same brittleness displayed by many state-of-the-art CNNs, which can be exploited in adversarial attacks (Carlini and Wagner, 2017). Indeed, researchers have exposed GNN vulnerabilities by crafting adversarial attacks which induce classification errors (Zügner et al., 2018; Zügner and Günnemann, 2019; Dai et al., 2018; Chang et al., 2020). The attacks poison the data by perturbing the graph structure and/or the node features. Such vulnerabilities are a serious concern when deploying models in industrial settings. We are thus motivated to study the adversarial vulnerabilities of GNN models in order to become aware of and insulate models from potential security threats. In this paper, we focus exclusively on structure perturbations as they lead to more severe performance degradation (Zügner et al., 2018) and because the *raison d'être* of GNNs is that they exploit the relational information given by the graph structure.

Adversarial attack detection techniques are important in the analysis of financial, communication, and social networks. For example, fraudulent actors might disguise their financial malfeasance by conducting many transactions with trustworthy agents. We provide a motivating example for financial transaction networks in Figure 1. The goal is fraud detection for a bitcoin transaction network. The model aims to label each Bitcoin transaction as licit (between exchanges, wallet providers, etc.) or illicit (scams, malware, terrorist organization transactions, etc.). Licit transactions tend to cluster together, as do illicit transactions (Weber et al., 2018), so a graph adversarial attack designed to disguise illicit transactions can exploit this by making (adding) fake Bitcoin flows (edges) in the dataset. In this setting, an algorithm that specifically targets the detection of such attacks is important.

Our main contributions are: **(i)** we provide a novel approach for detecting adversarial attacks on GNNs, designing a test to detect perturbations of individual nodes; **(ii)** we design a maximum mean discrepancy (MMD) based test for detecting whether a given subset of nodes has been corrupted; **(iii)** we demonstrate the efficacy of our algorithms via thorough experiments using commonly used node classification benchmark datasets and a real-world Bitcoin transaction network; **(iv)** we design a defense mechanism that leads to more robust classification in the face of attack; and **(v)** we modify existing attacks so that it can avoid detection by our proposed approach (at the cost of reduced impact). The Bitcoin transaction data analysis illustrates the potential practical benefit of our detection method as a mechanism to insulate graph-based models from security threats. We show through empirical analysis that our proposed method significantly outperforms state-of-the-art GNN defense approaches.

## 2 Related Work and Preliminaries

**Detection:** Our detection methods fall into the category of anomaly detection for static, attributed graphs. In contrast to previous work (Sharpnack et al., 2013; Qian and Saligrama, 2014; Li et al., 2017; Akoglu et al., 2015), we do not assume any model and instead learn a suitable representation based on the observations and graph topology. This is more in line with methods that assign an anomaly score to each node based on features and/or graph structure. Such methods are either based on the feature discrepancy within a node's neighbourhood (Perozzi and Akoglu, 2016; Wu et al., 2019b) or measure a reconstruction error (Ding et al., 2019; Ioannidis et al., 2020).

**Defense mechanisms:** Our proposed defense mechanism is related to recent attempts to improve the robustness of GNNs (Xu et al., 2019; Zhou et al., 2019; Zhu et al., 2019; Wu et al., 2019b; Entezari et al., 2020). A recent software library (Li et al., 2020) provides a good framework for investigation and fair comparison.

**Graph attacks:** Most of the prior works on graph adversarial attack focus on designing new attacks rather than detection. Attacks include DICE (Waniek et al., 2018), RL-S2V (Dai et al., 2018), Nettack (Zügner et al., 2018), Meta-attack (Zügner and Günnemann, 2019), and GF-Attack (Chang et al., 2020). These attacks target semi-supervised learning, which is our focus. Some recent work addresses the unsupervised setting (Bojchevski and Günnemann, 2019; Chang et al., 2020). Both DICE and Nettack assume a perturbation budget $\Delta$. DICE deletes $\Delta/2$ randomly chosen edges connected to the attacked node $i$ and then attaches $\Delta/2$ new edges to randomly selected nodes with labels $y$ different from $y_i$. Nettack changes the local topology around node $i$ in order to decrease the classification margin, selecting edges to change using a surrogate model. Meta-attack aims to drive down performance over the entire graph, formulating the selection of edges to modify as a constrained optimization problem. RL-S2V uses reinforcement learning to learn an attack policy. It can be effective, but the computational requirements are significant. GF-Attack is a restricted black-box attack that does not require knowledge of the model.

## 3 Problem Statement

We are provided with an attributed (possibly directed) graph $\mathcal{G}_{\text{obs}} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{N} = \{1, \ldots, N\}$ is the set of nodes and $\mathcal{E}$ is the set of edges. We represent the graph using the pair $(\mathbf{X}, \mathbf{A})$, where $\mathbf{X} \in \mathbb{R}^{N \times D}$ is a matrix of node features and $\mathbf{A} \in \mathbb{R}^{N \times N}$ is the adjacency matrix. The $i^{th}$ row of $\mathbf{X}$ is a vector $\mathbf{x}_i \in \mathbb{R}^D$ containing features associated with node $i$. We consider the semi-supervised node classification setting; there is a subset of nodes $\mathcal{L} \subset \mathcal{N}$ which have labels $\mathcal{Y}_{\mathcal{L}} = \{y_i \mid i \in \mathcal{L}\}$. The observed graph $\mathcal{G}_{\text{obs}}$ may differ from the "original" graph $\mathcal{G}$ because of adversarial attacks. The original graph has the same node set $\mathcal{N}$ as $\mathcal{G}_{\text{obs}}$ but it may possess a different edge set $\mathcal{E}'$. Consequently, if $\mathbf{A}' \in \mathbb{R}^{N \times N}$ is the adjacency matrix of $\mathcal{G}$, the equation $\mathbf{A} = \mathbf{A}'$ need not hold. However, we assume that the node features $\mathbf{X}$ associated with $\mathcal{G}$ are identical to those of $\mathcal{G}_{\text{obs}}$.

We address four attack detection and defense tasks:
(i) **Single targeted node attack detection**: For a specific node $i$, decide whether its edges have been modified in order to induce poor classification. We formulate this task as a hypothesis test acting on a pertinent statistic.
(ii) **Corrupted subset detection**: We suppose that a specified subset of nodes $\mathcal{S} \subset \mathcal{N}$ consisting of $M$ nodes has potentially been attacked by addition or deletion of edges with at least one endpoint in $\mathcal{S}$. Our task is to decide whether the subset has been attacked.
(iii) **Graph defense mechanism**: For a given GNN model, introduce modifications to the learning procedure, so that classification accuracy for the observed graph $\mathcal{G}_{obs}$ is as close as possible to what would be

achieved using the original graph $\mathcal{G}$.
(iv) **Adaptive attack**: Design a modified attack that can reduce classification accuracy but avoid detection by our proposed test for targeted node attacks.

For the first two tasks, we assume access to a portion of unperturbed data. This is often reasonable; for example, in the Bitcoin transaction network, there are many trusted entities, and their interactions can be assumed to be legitimate. In settings where we do not have access to unperturbed data, we rely on the assumption that the attacker has limited capacity and must focus attention on nodes where a change in outcome is of most interest. In these cases, we can form an "unperturbed" group through random selection; although we may include some attacked nodes, the majority are unperturbed.

**Threat model:** Most existing GNN attacks are white box models that assume full access to the GNN model. Different attacks impose different restrictions on the extent of the perturbation. These include the number of edges that can be changed and the extent of the modification of graph statistics such as the degree distribution. We impose the same constraints as adopted in the original papers that proposed the attacks. Existing attacks do not strive to minimize or restrict the perturbations that we exploit in our detection and defense mechanisms. One contribution of our paper is to show that the majority of existing attacks can be detected. Following the advice of (Carlini et al., 2019), we design an *adaptive adversary*. We show that the adaptive adversary can successfully avoid detection, but its impact on classification performance is reduced substantially compared to the original attacks.

## 4 Detection of Targeted Node Attacks

Most topological attacks strive to drive down the classification margin of a node (and eventually cause an error) by adding edges to nodes that likely belong to other classes. The result is usually a greater discrepancy between a GNN's output softmax probabilities $\{p_j\}_{j\in\mathcal{N}(i)}$ in the closed neighbourhood of an attacked node, as compared to an unperturbed node.

**Test statistic**: Since each $p_j$ can be thought of as a probability mass function over the node classes, it is natural to apply the multi-distribution Jensen-Shannon Divergence (Lin, 1991) to measure the discrepancy between a set of softmax probabilities. The Shannon entropy is $H(p_j) = -\sum_k p_{jk}\log p_{jk}$; let $\mathcal{P}_i = \{p_j\}_{j\in\mathcal{N}(i)}$ be the set of softmax probabilities in the closed neighbourhood of node $i$. $\mathcal{N}(i)$ denotes the local neighbourhood of node $i$. Also please note that the neighbourhood will include the center node itself. For each node

htbp

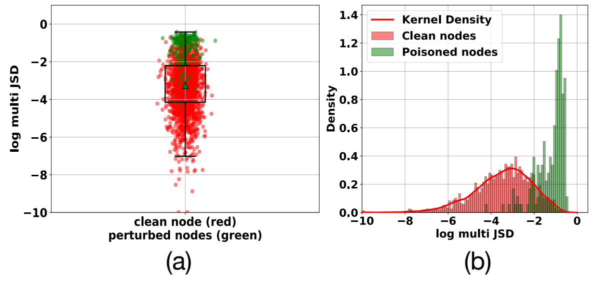

**Figure 2:** Box plots of Multi-JSD statistics after log transform for unperturbed nodes (red) and perturbed nodes (green). (b) using kernel density function to fit the log-transformed statistics from the unperturbed nodes (Citeseer dataset, under Nettack).

$i$, we obtain a discrepancy metric:

$$\mathrm{JSD}\left(\mathcal{P}_i\right) = H\left(\frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} p_j\right) - \frac{1}{|\mathcal{N}(i)|}\sum_{j\in\mathcal{N}(i)} H\left(p_j\right) . \tag{1}$$

Figure 2 shows example distributions of the multi-JSD statistic for clean and perturbed nodes after a log transformation. The difference between the statistics of perturbed versus unperturbed nodes is clear.

We also explore the application of a sharpening technique, as in (Berthelot et al., 2019), prior to computing the JSD statistic. That is, for each $p_j$, we first compute

$$\tilde{p}_j = \mathrm{Sharpen}\left(p_j, T\right) = \left[\frac{\left(p_j\right)_k^{\frac{1}{T}}}{\sum_{k'=1}^{K}\left(p_j\right)_{k'}^{\frac{1}{T}}}\right]_{k=1}^{K}$$

(where $K$ is the number of components of $p_j$. We then calculate the statistics $JSD(\{\tilde{p_j}\}_{j\in\mathcal{N}(i)})$. The scalar $T$ is a "temperature" hyperparameter that controls the amount of sharpening.

**Detection test**: This discrepancy measure is appealing given its simplicity and straightforward calculation. We define a detection test by evaluating the JSD statistic for a possibly perturbed node and comparing it to a threshold $\tau$. If it exceeds $\tau$, then the node is deemed to have been attacked. We apply the Neyman-Pearson approach (Neyman and Pearson, 1933) to set the detection threshold for the JSD statistic so that we can maximize the detection probability while targeting a desired false alarm rate. We fit a kernel density estimator (KDE) to unperturbed training data using a Gaussian kernel whose bandwidth is determined via cross-validation. We calculate an appropriate detection threshold $\tau$ by matching the KDE tail probability, estimated with empirical quantiles, to a specific target false positive rate (Silverman, 1986).

The KDE fitting and thresholding procedure requires an i.i.d. assumption on the JSD statistics of different nodes. This assumption is usually not valid for graph data. As a result, we are not guaranteed to achieve the desired false alarm rate. In practice, however, the empirical false alarm rate is very close to the target theoretical value. We report the results of a detailed investigation in the supplementary material.

### 4.1 Defense Mechanism

Our detector can be effectively employed as a preprocessing step to flag adversarial input to a GNN. We now outline a simple potential defense approach to capitalize on the information obtained from applying our detection schemes. Although simple, the experiments below suggest that the proposed mechanism is significantly more effective than state-of-the-art approaches.

Suppose the detector flags a node's neighbourhood as unreliable and probably corrupted. We conduct classification for such a node without using neighbourhood information. This leads to an effective pipeline for integrating detection and training. Given a possibly perturbed graph dataset and a GNN model to train, we do not immediately train the classifier. Instead, we first apply our detection test. If any nodes are identified as perturbed, we simply exclude the neighbourhood data for those nodes. This yields an "ameliorated" dataset that we use to train the classification algorithm.

### 4.2 Adaptive Adversarial Attack

It is important to evaluate the adversarial robustness using adaptive attacks, i.e., attacks that are aware of the detection method and adapt to avoid detection (Carlini et al., 2019). We now propose modifications to Nettack (Zügner et al., 2018) and DICE (Waniek et al., 2018) to construct such adaptive attacks.

Nettack uses a linearized graph convolutional network as a proxy model and searches over candidate edges to delete and add to the neighbourhood of a node. Edges are chosen according to how much they reduce the classification margin of the proxy model. While conducting the search over candidate edges, the algorithm is constrained to preserve the degree distribution. DICE swaps edges in the neighbourhood of an attacked node, removing those that connect to nodes of the same class and adding edges to nodes of different classes. DICE's constraint is how many edges it can change.

Our adaptive attacks adds a constraint that the multi-JSD statistic must not be increased to the extent that the node would be flagged as attacked by the proposed detection test (for a threshold associated with a chosen false-alarm rate). The adaptive attack has the information to calculate this threshold. In adapt-Nettack, we conduct a local greedy search for the node that has the greatest impact on classification, but only accept those that respect the multi-JSD limit. In adapt-DICE, the proposed random edges is evaluated and the modification is rejected if the multi-JSD threshold is exceeded.

## 5 Detection of Corrupted Node Sets

We now switch to the task of detecting whether a subset of nodes $\mathcal{S} \subset \mathcal{N}$, with $|\mathcal{S}| = M \ll N$, has been perturbed. We divide the nodes $\mathcal{N}$ into three sets: $\mathcal{T}$, $\mathcal{R}$, and $\mathcal{S}$. The set $\mathcal{R}$ contains $M$ nodes that are used as a comparison set and are assumed to be unperturbed. The set $\mathcal{T} = \mathcal{N} - (\mathcal{R} \cup \mathcal{S})$, containing $N - 2M$ nodes, is used to learn a suitable embedding of the nodes.

We now provide an overview of the proposed approach. For each node $i$ in $\mathcal{R}$ or $\mathcal{S}$, we create an *extended feature vector* $\tilde{\mathbf{x}}_i \in \mathcal{X}$ (we specify $\mathcal{X}$ below). Briefly, $\tilde{\mathbf{x}}_i$ concatenates the node features $\mathbf{x}_i$ to a binary vector $\mathbf{a}_{i,\mathcal{T}}$ of length $N - 2M$ that encodes the topology in node $i$'s neighbourhood (but only for edges that connect to nodes in $\mathcal{T}$). We expect the difference between $\mathcal{R}$ and $\mathcal{S}$ to manifest as a statistical difference between the extended feature vector sets $\{\tilde{\mathbf{x}}_i\}_{i\in\mathcal{R}}$ and $\{\tilde{\mathbf{x}}_j\}_{j\in\mathcal{S}}$.

Two-sample statistical hypothesis testing is a natural framework for testing for such a discrepancy. Since the space $\mathcal{X}$ is high-dimensional, it can be important to identify a *feature extractor* $z_{\mathcal{T}} : \mathcal{X} \to \mathbb{R}^V$ such that the output $z_{\mathcal{T}}(\tilde{\mathbf{x}}_i) \in \mathbb{R}^V$ succinctly summarizes the graph topology, features and labels in the neighbourhood of node $i$. We then apply a two-sample test to determine whether $\{z_{\mathcal{T}}(\tilde{\mathbf{x}}_i)\}_{i\in\mathcal{R}}$ is drawn from the same distribution as $\{z_{\mathcal{T}}(\tilde{\mathbf{x}}_j)\}_{j\in\mathcal{S}}$. The *null hypothesis*, $H_0$, models $z_{\mathcal{T}}(\tilde{\mathbf{x}}_i) \in \mathbb{R}^V$ for $i \in \mathcal{R} \cup \mathcal{S}$ as i.i.d. random vectors, generated from the distribution $p$. The *alternative hypothesis*, $H_1$, states that the random vectors $z_{\mathcal{T}}(\tilde{\mathbf{x}}_i) \in \mathbb{R}^V$ for $i \in \mathcal{R}$ are i.i.d. and are generated from a distribution $q \neq p$. In the context of detecting topology manipulations, we can write $p(\tilde{\mathbf{x}}) = p_1(\mathbf{x})p_2(\mathbf{a}_{\mathcal{T}}|\mathbf{x})$ and $q(\tilde{\mathbf{x}}) = q_1(\mathbf{x})q_2(\mathbf{a}_{\mathcal{T}}|\mathbf{x})$. Our detection framework assumes $p_1(\mathbf{x}) = q_1(\mathbf{x})$; otherwise, we may detect a discrepancy in the features included in $\mathcal{R}$ versus $\mathcal{S}$. The test thus operates over the conditional distributions $p_2(\mathbf{a}_{\mathcal{T}}|\mathbf{x})$ and $q_2(\mathbf{a}_{\mathcal{T}}|\mathbf{x})$. It essentially tests whether, conditioned on the node features, the probabilistic connectivity is different between sets $\mathcal{S}$ and $\mathcal{R}$.

We employ the non-parametric two-sample kernel test from (Gretton et al., 2012). Let $k$ be a (measurable and bounded) kernel function; this kernel has associated with it a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_k$ of functions on the set $\mathcal{X}$. Given two distributions $p$ and $q$, the kernel $k$ induces a metric which compares the two distributions. This metric is the *Maximum Mean Discrepancy (MMD)*:

$$MMD_k^2 := \mathbb{E}_{r,r'}[k(r,r')] + \mathbb{E}_{s,s'}[k(s,s')] - 2\mathbb{E}_{r,s}[k(r,s)]. \tag{2}$$

Here $r, r' \overset{i.i.d.}{\sim} p$ and $s, s' \overset{i.i.d.}{\sim} q$. We reject $H_0$ when the MMD statistic exceeds a threshold $c_\alpha$, where $\alpha$ is the desired $p$-value of our test.

We approximate the MMD using samples to form an unbiased estimator $\widehat{MMD}^2_{U,k}$. Under $H_1$, $\widehat{MMD}^2_{U,k}$ is asymptotically normal (Gretton et al., 2012) but under $H_0$, it converges to a distribution that depends on the unknown distribution $p$. The threshold $c_\alpha$ thus cannot be computed analytically. Fortunately, (Sutherland et al., 2017) provide a method for estimating a data-dependent threshold $\hat{c}_a$ via permutation.

### 5.1 Details of the test

For $i \in \mathcal{N} - \mathcal{T}$, let $\mathbf{a}_{i,\mathcal{T}}$ denote the binary vector with $N - 2M$ elements, such that entry $a_{i,j,\mathcal{T}}$ indicates whether there is an edge between node $i$ and node $j \in \mathcal{T}$. As mentioned, $\tilde{\mathbf{x}}_i$ is the concatenation of $\mathbf{x}_i$ and $\mathbf{a}_{i,\mathcal{T}}$. The vectors thus constructed lie in $\mathcal{X} = \mathbb{R}^D \times \{0,1\}^{N-2M}$. We need to identify a suitable feature extractor $z_\mathcal{T}(\tilde{\mathbf{x}}) : \mathcal{X} \mapsto \mathbb{R}^V$. The subscript $\mathcal{T}$ indicates that the function is dependent on the training set nodes and the associated graph topology.

The (sharpened) Jensen-Shannon divergence can act as a valuable feature for subset detection. We thus use the function $z'_\mathcal{T}(\tilde{\mathbf{x}}_i) = JSD(\{\tilde{p}_{j,\mathcal{T}}\}_{j \in \mathcal{N}(i) \cap \mathcal{T}})$. The notation $\tilde{p}_{j,\mathcal{T}}$ indicates that the softmax probabilities are derived using only the nodes in $\mathcal{T}$. The multi-JSD is evaluated over the nodes in $\mathcal{N}(i) \cap \mathcal{T}$ to avoid the introduction of undesirable strong dependencies. In order to ensure asymptotic consistency of the test (see discussion below) we prefer to construct a function $z_\mathcal{T}$ that concatenates $\gamma_1 \mathbf{x}_i$, $\gamma_2 \mathbf{a}_{i,\mathcal{T}}$ and $z'_\mathcal{T}(\tilde{\mathbf{x}}_i)$. Here $\gamma_1$ and $\gamma_2$ are scalars that can be selected to control the impact of the respective terms.

### 5.2 Consistency of the test

The performance of the test can be measured by the *risk*: $R^{(N)}_M = P(H_1|H_0) + P(H_0|H_1)$. A detection test is *(asymptotically) consistent* if $R^{(N)}_M \to 0$ as $M \to \infty$.

The MMD test employing $\widehat{MMD}^2_{U,k}$ is asymptotically consistent if the kernel $k$ is *characteristic* (Gretton et al., 2012). A bounded, measurable kernel on a measurable space, $(\mathcal{X}, \mathcal{B})$, that defines an RKHS $\mathcal{H}$ is called *characteristic* if the mapping $P \mapsto m_P = E_{X \sim P}[k(\cdot, X)]$ is injective for probability distributions $P$ on the space (Fukumizu et al., 2004, 2008). As discussed by (Sutherland et al., 2017), the composition of a kernel $k : \mathcal{X}_2 \times \mathcal{X}_2 \to \mathbb{R}$ and a function $z : \mathcal{X}_1 \to \mathcal{X}_2$ defines a composite kernel $k \circ z : \mathcal{X}_1 \times \mathcal{X}_1 \to \mathbb{R}$. If $z$ is an injective function and $k$ is a characteristic kernel on $\mathcal{X}_2$, then $k \circ z$ is a characteristic kernel on $\mathcal{X}_1$.

The function $z_\mathcal{T} : \mathcal{X} \mapsto \mathbb{R}^{N-2M+D+V}$ is clearly injective, because we can recover $\tilde{\mathbf{x}}$ simply by dividing each of the first $N - 2M$ elements by $\gamma_1$ or $\gamma_2$. The Gaussian radial basis function kernel is characteristic on $\mathbb{R}^d$. We can then construct a characteristic kernel $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ by forming the composition of the function $z_\mathcal{T}$ and a Gaussian RBF kernel $k'$ defined over $\mathbb{R}^{N-2M+D+Q} \times \mathbb{R}^{N-2M+D+Q}$.

### 5.3 Independence considerations

We now address a somewhat subtle point concerning the i.i.d. assumptions required by the MMD test. It may appear at first that the samples in $\{z'_\mathcal{T}(\tilde{\mathbf{x}}_i)\}$ are not i.i.d. because they contain entries from the adjacency matrix. However, note that $\mathbf{a}_{i,\mathcal{T}}$ and $\mathbf{a}_{j,\mathcal{T}}$ do not have any elements in common. The node $i$ belongs to either $\mathcal{R}$ or $\mathcal{S}$ and we are considering potential edges between it and nodes in a disjoint subset $\mathcal{T}$. The entire adjacency matrix can have a strong dependency structure and yet the vectors $\mathbf{a}_{i,\mathcal{T}}$ can be independent. Consider, for example, a generative model parametrized by $\mathbf{c}_\mathcal{T}$, a vector indicating the class memberships of the nodes in $\mathcal{T}$, and a matrix $\boldsymbol{\beta}$ that indicates the probability that a node in class $r$ has an edge with a node in class $s$. The joint distribution for nodes $i$ and $j$ in $\mathcal{S}$ is:

$$p(\mathbf{a}_{i,\mathcal{T}}, \mathbf{a}_{j,\mathcal{T}} | \mathbf{c}_\mathcal{T}, \boldsymbol{\beta}) = \prod_{k \in \mathcal{T}} p(a_{i,k}|c_i, c_k, \boldsymbol{\beta}) p(a_{j,k}|c_j, c_k, \boldsymbol{\beta})$$

The vectors are thus conditionally independent even though the generative model can achieve strong dependencies in the adjacency matrix through the vector $\mathbf{c}$. The key point is that it is these conditional distributions that we test with the MMD. We do not assume that the graphs are samples from a generative model with the structure above. Rather, we can perform a test under an independence assumption while allowing for intricate dependencies in the adjacency matrix.

## 6 Experimental Results

### 6.1 Experimental settings

**Datasets**: We use the datasets analyzed in (Shchur et al., 2018; Zügner et al., 2018). For targeted node detection, we use citation datasets Cora, Citeseer, Pubmed, and political blogs network Polblogs; for subset detection, we also include collaboration networks Coauthor CS, Coauthor Physics and co-purchase graphs Amazon Computers and Amazon Photo. We follow the pre-processing steps of (Shchur et al., 2018). For full dataset descriptions and statistics, please refer to the supplementary material.

**Graph attack models**: We evaluate over 5 major existing adversarial attack approaches including both targeted and global attacks. We test the targeted node attacks DICE (Waniek et al., 2018), Nettack (Zügner et al., 2018), and GF-Attack (Chang et al., 2020). For

**Table 1:** Detection Area-under-Curve (AUC) % comparison for Nettack (Zügner et al., 2018)

| Dataset | Ours | GraphSAC | GAE | Amen | Radar | OCSVM raw | OCSVM emb | Jaccard |
|---|---|---|---|---|---|---|---|---|
| **Cora** | **86.4** | 80.0 | 50.2 | 75.0 | 77.0 | 50.3 | 72.7 | 69.9 |
| **Citeseer** | **80.1** | 75.0 | 64.4 | 73.0 | 67.0 | 36.8 | 67.8 | 69.3 |
| **Polblogs** | 85.4 | **98.0** | 51.2 | 89.0 | 76.0 | - | 59.9 | - |
| **Pubmed** | **87.8** | 82.0 | 69.2 | 62.0 | 44.0 | 58.5 | 57.9 | 82.4 |

**Table 2:** Detection Area-under-Curve (AUC) % under various graph attacks.

| Model | Ours | GAE | OCSVM raw | OCSVM emb | Jaccard | Ours | GAE | OCSVM raw | OCSVM emb | Jaccard |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Cora | | | | | Polblogs | | |
| **Nettack** | **86.4** | 58.2 | 33.9 | 59.4 | 71.0 | **88.9** | 57.8 | - | 53.1 | - |
| **DICE** | **77.5** | 64.6 | 50.0 | 50.0 | 61.2 | **74.2** | 48.5 | - | 33.0 | - |
| **GF-Attack** | **87.1** | 60.7 | 28.7 | 46.0 | 61.2 | **59.7** | 46.8 | - | 49.9 | - |
| **DICE-global** | **81.8** | 56.8 | 41.2 | 43.0 | 59.4 | **67.3** | 48.3 | - | 49.2 | - |
| **Mettack** | **66.1** | 56.4 | 43.8 | 61.0 | 57.3 | **87.0** | 50.4 | - | 51.1 | - |
| | | | Citeseer | | | | | Pubmed | | |
| **Nettack** | **83.2** | 63.5 | 27.9 | 56.3 | 67.5 | **91.0** | 70.2 | 72.6 | 82.3 | 82.0 |
| **DICE** | **76.8** | 62.6 | 44.2 | 47.1 | 62.9 | **68.0** | 54.4 | 60.8 | 57.0 | 57.9 |
| **GF-Attack** | **82.3** | 69.0 | 27.7 | 43.6 | 58.2 | **95.5** | 58.7 | 71.8 | 61.9 | 73.5 |
| **DICE-global** | **84.3** | 57.2 | 37.1 | 43.0 | 61.5 | **76.8** | 53.1 | 42.9 | 40.9 | 63.4 |
| **Mettack** | 66.3 | 57.5 | 37.7 | 66.2 | 59.2 | **83.1** | 54.0 | 35.8 | 37.3 | 65.5 |

global attacks, we use DICE-global (Waniek et al., 2018) and Mettack (Zügner and Günnemann, 2019).

**Detection Baselines**: We compare our single node detection method with state-of-the-art alternatives: (i) two methods based on local feature homophily, *Amen* (Perozzi and Akoglu, 2016) and *Jaccard similarity* (Wu et al., 2019b). The latter was designed to identify anomalous edges; we extend it to node detection by computing the average Jaccard index between the center node and each neighbor; (ii) two methods based on anomaly scores, *GAE* (Ding et al., 2019) and *Radar* (Li et al., 2017); (iii) *GraphSAC* (Ioannidis et al., 2020), which uses random sampling and consensus; and (iv) two baselines that use One-class SVM (OCSVM) (Manevitz and Yousef, 2001) — *OCSVM-raw* uses the average of the raw neighbourhood features and *OCSVM-emb* uses GNN node representations.

**Defense comparison baselines**: We compare with three state-of-the-art robust GCN training schemes (defense mechanisms): *RGCN* (Zhu et al., 2019), *GCN-Jaccard* (Wu et al., 2019b) and *GCN-SVD* (Entezari et al., 2020). We also compare to three different GNNs: GCN (Kipf and Welling, 2017), SGCN (Wu et al., 2019a) and GAT (Velickovic et al., 2018). GAT (Graph Attention Network) can alleviate the impact of spurious edges using the attention mechanism.

## 6.2 Results for single node detection

**Attack configuration:** We adopt the attack setup in (Zügner et al., 2018), including the target node selection procedure and perturbation budget. We use GCN (Kipf and Welling, 2017) as the GNN under attack. For Nettack (Zügner et al., 2018), GF-Attack (Chang et al., 2020), and Mettack (Zügner and Günnemann, 2019), we use default settings and specified hyperparameters. For targeted DICE, we apply a light attack with perturbation budget $\Delta = \frac{1}{4}d$, where $d$ is the degree of the targeted node. For the global DICE attack and Mettack (Zügner and Günnemann, 2019), we set the perturbation budget $\Delta$ to 5% of the total number of edges in the graph. We evaluate the performance of our proposed model under other settings and report results in the supplementary material. The behavior is consistent across settings. Experimental results are obtained by averaging over 10 runs.

**Experiment 1:** For the Nettack attack, we report the average area-under-the-curve (AUC) values across 10 runs for $K$=10 targeted nodes in Table 1 (results for GraphSAC and Amen are taken from (Ioannidis et al., 2020)). This experiment allows us to compare to the reported results for the state-of-the-art GraphSAC detector (code is currently unavailable and we could not replicate the performance).

**Experiment 2:** This more comprehensive experiment investigates detection capabilities for 5 representative graph adversarial attacks. For the targeted attacks, Nettack, DICE, and GF-Attack, we choose $K = 40$ nodes to attack. Table 2 reports the obtained AUCs.

**Discussion:** The results clearly demonstrate the effectiveness of our proposed detection technique. It achieves a relatively high AUC for most attacks and significantly outperforms comparison methods, both for local and global attacks, on all datasets except for Polblogs. Polblogs has no node attributes and has very dense connectivity. In the absence of attributes, GraphSAC's label propagation model outperforms the GCN used to derive the logits. Our method's AUC is lower for Mettack, which is a relatively ineffective attack (Table 5 shows that Mettack reduces accuracy by only 5-20% compared to 65-85% for Nettack) so detection is harder. Besides, we evaluate the effectiveness for other GNN models. Detection performance is similar for the Graph Markov Neural Network (GMNN) (Qu et al., 2019), simplified GCN (SGCN) (Wu et al., 2019a), and Graph Attention Network (GAT) (Velickovic et al., 2018). Results are included in the supplementary material.

## 6.3 Adaptive graph adversarial attacks

In this experiment, we provide the results for the adaptive attacks described in Section 4.2. We select 40 target nodes. Proposed perturbations are only permitted if the resultant multi-JSD is below a threshold $\tau$, which is selected to correspond to the 5% false positive rate on training data. More details of the experiment setting are provided in the supplementary material.

**Table 3:** Accuracy (%) comparison for the target nodes between the original attacks and the adaptive attacks with the multi-JSD constraint.

| | Clean Nettack | Nettack | Nettack mJSD | Clean Dice | DICE | DICE mJSD |
|---|---|---|---|---|---|---|
| **Cora** | 82.2 | 25.0 | 49.3 | 89.8 | 64.1 | 63.2 |
| **Citeseer** | 73.6 | 28.6 | 45.0 | 87.5 | 62.6 | 61.8 |
| **Polblogs** | 95.0 | 37.8 | 84.3 | 87.0 | 27.7 | 64.8 |
| **Pubmed** | 89.2 | 9.7 | 52.4 | 86.7 | 58.4 | 61.1 |

**Table 4:** Comparison of detection rates (%) at 5% false-alarm threshold between the original attacks and the adaptive attacks multi-JSD constraint.

| | Nettack | Nettack-mJSD | DICE | DICE-mJSD |
|---|---|---|---|---|
| **Cora** | 43.2 | 21.1 | 50.0 | 42.5 |
| **Citeseer** | 54.1 | 26.6 | 43.6 | 2.1 |
| **Polblogs** | 60.2 | 9.1 | 38.9 | 4.3 |
| **Pubmed** | 52.5 | 22.5 | 52.5 | 20.1 |

Table 3 shows the accuracy reduction achieved by the adaptive attacks and in Table 4, we report the detection rate of the adaptive attacks before and after adding the local smoothness constraint (we report detection rate instead of AUC to more clearly demonstrate the effectiveness of the adaptation. AUC results are reported in the supplementary material). The adaptive attack greatly reduces the probability of detection (Table 4), but this comes at the price of potential less impactful attacks. Note that the detection rates for the adaptive attacks are non-zero because the attacks evaluate the multi-JSD constraint using a proxy linearized model and there is a mismatch between this proxy and the the true GNN used during detection and classification.

## 6.4 Results for Sets of Corrupted Nodes

In order to simulate a poisoned subset of nodes $\mathcal{S}$, we use DICE to sequentially corrupt edges of a set of $M$ nodes chosen uniformly at random. The set $\mathcal{R}$ consists of another $M$ uncorrupted nodes that are not in $\mathcal{S}$. To test the effectiveness of our detection scheme, we also use a "control" dataset where the nodes are partitioned into three sets $\mathcal{R}_{\text{control}}$, $\mathcal{S}_{\text{control}}$ and $\mathcal{T}_{\text{control}}$ but where $\mathcal{S}_{\text{control}}$ has *not* been subjected to any perturbations. This provides a way to validate the false alarm rate.

We expect that it is harder to detect suspicious sets of nodes for small $M$ and we therefore carry out our detection experiments for a range of values of $M$. For each value of $M$, we generate 100 different random partitions $\{\mathcal{R}, \mathcal{S}, \mathcal{T}\}$. Our detector employs an RBF kernel $k'$ with a fixed bandwidth, a hyperparameter that must be tuned. We apply the kernel optimization procedure from (Sutherland et al., 2017) wherein the sets of samples are divided into a "training set" and a "test set" for tuning kernel parameters. The other hyperparameters $\gamma_1$ and $\gamma_2$ are positive scalars that are small relative to the kernel bandwidth and the choice of $\gamma_1$, $\gamma_2$ is thus folded into the bandwidth tuning. The softmax probabilities used in the MMD test are derived from a GCN trained using the graph structure and node features incident on $\mathcal{T}$. This allows us to obtain embedding vectors for all nodes in $\mathcal{S}$ and $\mathcal{R}$. Finally, we used a false positive threshold $\alpha = 0.05$ to derive the data-dependent threshold $\hat{c}_\alpha$ for the hypothesis test.

Figure 3 shows the detection rate for a range of $M$ values. The thin lines at the bottom of each graph are the corresponding false alarm rates; we achieve to stay close to the target 5%. We compare three different kernel mapping functions: Multi-JSD, sharpened Multi-JSD ($T = 0.01$) and hard maximum ($T \to 0$)). From the experimental results of all nine datasets, we demonstrate the effectiveness of our proposed detectors on the node subset detection task. We observe that the sharpened Multi-JSD consistently outperforms the other two kernel mapping functions. The performance of the MMD subset detection approach is much better than repeated application of single node detection; due to the multiple hypothesis testing effect, the threshold of the single node detector must be set very high to avoid too many false alarms and as a result the detec-

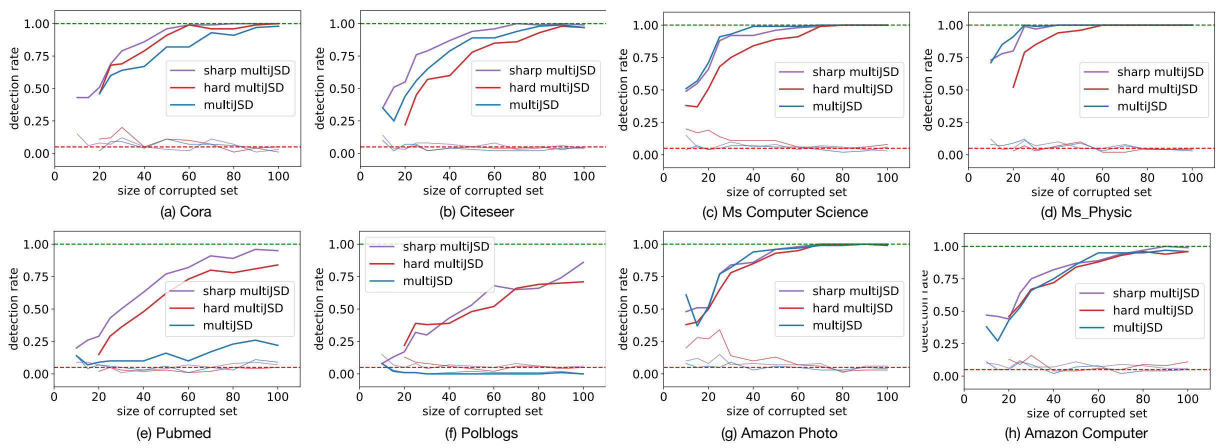

**Figure 3:** Detection performance for subgraph attack detection for 8 datasets under DICE attack.

**Table 5:** Accuracy (%) comparison with GCN defense mechanism under targeted attack and global attack. The uncertainty represents the 95/5 confidence interval. Bold values denote cases where a Wilcoxon signed rank test indicates a statistically significant difference between the best and second-best (underline) algorithms.

| **Dataset** | **GCN (clean)** | **GCN** | **SGCN** | **GAT** | **GCN-SVD** | **Jaccard** | **RGCN** | **Ours** |
|---|---|---|---|---|---|---|---|---|
| | | | | Targeted attack: **Nettack** | | | | |
| **Cora** | 89.6±1.8 | 17.2±3.8 | 23.7±0.1 | 37.5±1.1 | <u>43.6±1.4</u> | 35.7±1.5 | 18.6± 0.9 | **50.3±2.4** |
| **Citeseer** | 90.0±0.8 | 18.0±1.0 | 20.0±0.0 | 31.0±2.2 | <u>41.7±1.5</u> | 25.3±1.6 | 17.5±1.7 | **63.3±1.7** |
| **Polblogs** | 93.8±1.2 | 14.1±2.5 | - | <u>24.0±8.7</u> | 19.0±1.9 | - | 9.4±0.9 | **36.7±0.5** |
| **Pubmed** | 83.3±1.2 | 28.8±0.1 | 26.6±2.1 | 44.5±1.3 | <u>49.1±1.0</u> | 30.1±3.1 | 29.4±0.6 | **67.9±0.6** |
| | | | | Global attack: **Mettack** | | | | |
| **Cora** | 82.8±0.2 | 75.1±0.1 | 76.2±0.0 | 77.2±0.1 | 77.8±0.4 | <u>78.5±0.4</u> | 77.1±0.4 | **79.0±0.5** |
| **Citeseer** | 71.9±0.2 | 65.9±0.3 | 69.6±0.1 | <u>71.8±0.1</u> | 68.9±0.8 | 70.2 ±1.1 | 71.2±0.1 | 72.1±0.3 |
| **Polblogs** | 95.3±0.3 | 76.4±0.6 | - | <u>86.9±1.5</u> | **92.3±0.4** | - | 77.3±0.6 | 77.5±0.4 |
| **Pubmed** | 85.9±0.1 | 81.2±0.0 | 74.4±0.0 | 79.6±0.3 | <u>82.0±0.1</u> | 81.3±0.1 | 80.8±0.1 | **82.4±0.1** |

tion rate is very low. See supplementary material for detailed results.

## 6.5 Defense Mechanism

We compare our proposed defense strategy with the existing GNN defense approaches including GCN-Jaccard (Wu et al., 2019b), Robust GNN (Zhu et al., 2019) and GCN-SVD (Entezari et al., 2020). We conduct experiments on Cora, citeseer, Polblogs, and Pubmed under one target attack, Nettack, and one global attack Meta-attack. For the targeted attack, we report the classification accuracy for the $K = 80$ selected nodes (with target selection policy as in (Zügner et al., 2018)) before and after the attack. For the global attack, we apply Meta-attack with 5% perturbed edges to poison the underlying data topology. We then report the classification accuracy for the entire test set. We use our proposed detector to flag any nodes with multi-JSD metrics above the detection threshold $\tau$ (computed by setting a 5% false positive rate on training data). Then we exclude the neighborhood data around those nodes and yield the "ameliorated" adjacency matrix.

As shown in Table 5 when we train GCN (Kipf and Welling, 2017) or SGCN (Wu et al., 2019a) on the poisoned dataset, we observe a significant degradation in accuracy. GAT (Velickovic et al., 2018) is more robust, which could be explained by the learned attention scores can mitigate the impact of spurious edges. Compared with other robust GNN training alternatives, our approach is more robust against targeted node attacks, with an average 15% improvement over the second-best baselines. For the defense against the global meta-attack, our method also has an advantage, with the exception of the Polblogs dataset. As mentioned previously, Polblogs does not have node attributes and its topology information dominates the representation of nodes. The low-rank approximation of the graph used by GCN-SVD can effectively filter out the noisy edges.

We further explore the performance of our detector and defense strategy under different perturbation budget (severities of attack). We conduct a global attack (Meta-attack) under different perturbation budgets. Figure 4 shows the example results for Citeseer. The right panel compares defense strategies and the left compares de-

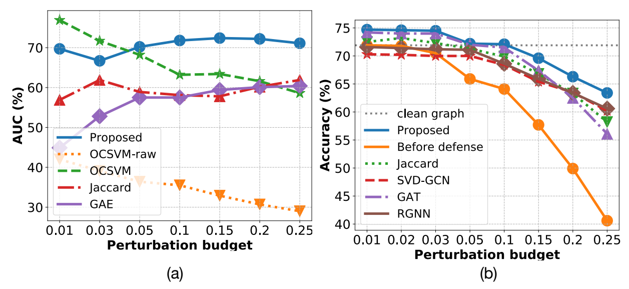

**Figure 4:** Detection and defense under different perturbation budget (Citeseer)

tection strategies. We observed similar trends for other datasets. The results are consistent with the results reported in table 5 and table 2. Our proposed detector and defense outperforms alternatives in the majority of the cases.

Finally, we compare the time complexity across different defense techniques. In Table 6, we can observe that our proposed defense strategy is computationally efficient and scales much better for larger graphs compared to the GCN-SVD and R-GCN approaches. In addition, our defense strategy does not introduce extra space complexity compared to the original GCN model.

**Table 6:** Training time complexity (s)

| | **GCN** | **Jaccard** | **GCN-SVD** | **R-GCN** | **Ours** |
|---|---|---|---|---|---|
| **Cora** | 7.9 | 12.7 | 25.2 | 61.2 | 14.6 |
| **Citeseer** | 8.5 | 12.8 | 22.8 | 73.4 | 16.8 |
| **Polblogs** | 6.4 | 7.8 | 7.1 | 25.2 | 12.3 |
| **Pubmed** | 45.8 | 88.4 | 1070.2 | 1999.8 | 96.9 |

**Table 7:** Left: Prediction accuracy (%) for the illicit transaction before and after the adversarial graph perturbations and after using our defense strategy. Right: Detection Area-under-Curve (AUC) under Nettack attacks using GCN as the prediction model.

| (Acc %) | **Clean** | **Under Nettack** | **Defense Nettack** | **Detection Nettack** (AUC %) |
|---|---|---|---|---|
| **TS 1** | 92.5 | 85 | **87.5** | 83.5 |
| **TS 2** | 72.5 | 50.0 | **52.5** | 78.0 |
| **TS 3** | 90.0 | 62.5 | **82.5** | 86.3 |
| **TS 4** | 97.5 | 67.5 | **82.5** | 87.1 |
| **TS 5** | 95 | 55.0 | **87.5** | 75.9 |

### 6.6 Real world application: Adversarial attack detection in Bitcoin Networks

We evaluate our proposed detection scheme on real-world Bitcoin transaction networks provided in the Elliptic Bitcoin Dataset (Weber et al., 2018). From the raw Bitcoin data, the graph is constructed such that nodes represent transactions and edges represent the flow of Bitcoin currency from one transaction to the next. The goal is to label each Bitcoin transaction as licit or illicit (see Figure 1). The dataset statistics are provided in the supplementary material.

The dataset consists of multiple transaction graphs associated with different times. We select for analysis the 5 time steps with the most illicit transactions. We consider targeted attacks that have the goal of disguising illicit transactions. We apply Nettack and DICE to simulate this real work attack mechanism (DICE results can be found in the supplementary), with attack budget and constraints set as in Section 6.2. After obtaining the perturbed graph, we conduct the target node detection procedure.

Table 7 shows how effective the attacks are in deceiving a GCN classifier and disguising illicit transactions. In the right column, we report the detection AUC for our single node detector. The proposed detector achieves high AUC scores, successfully detecting most attacked transactions without generating many false alarms.

## 7 Conclusion

We have presented methods for detecting adversarial attacks against graph data in two distinct regimes. In both cases, we assume that the data has been poisoned by modification of edges in the graph in order to mislead a node classification algorithm and degrade its performance. The existence of adversarial vulnerabilities for the task of node classification was seen to be tied to the assumption of local smoothness of the graph data. As practical motivation, we illustrated using Bitcoin transaction network data how graph adversarial attacks could be used to disguise illicit transactions. We then demonstrated that the proposed detection procedure could reveal such attacks. Building on the detection procedures, we designed a defense mechanism that results in a much more robust training procedure. Finally, we proposed an adaptive attack which significantly reduces the detection rate by using our designed smoothness metric as an unnoticeable criteria for limiting the search space for the perturbation edge.

## References

Akoglu, L., Tong, H., and Koutra, D. (2015). Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. (2019). Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*.

Bojchevski, A. and Günnemann, S. (2019). Adversarial attacks on node embeddings via graph poisoning. In *Proc. Int. Conf. Machine Learning*.

Carlini, N., Athalye, A., Papernot, N., Brendel, W., Rauber, J., Tsipras, D., Goodfellow, I., Madry, A., and Kurakin, A. (2019). On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

Carlini, N. and Wagner, D. A. (2017). Towards evaluating the robustness of neural networks. In *Proc. IEEE Symposium on Security and Privacy*, pages 39–57.

Chang, H., Rong, Y., Xu, T., Huang, W., Zhang, H., Cui, P., Zhu, W., and Huang, J. (2020). A restricted black-box adversarial framework towards attacking graph embedding models. In *Proc. AAAI Int. Conf. Artificial Intelligence*.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial attack on graph structured data. In *Proc. Int. Conf. Machine Learning*, pages 1115–1124, Stockholmsmässan, Stockholm Sweden.

Ding, K., Li, J., Bhanushali, R., and Liu, H. (2019). Deep anomaly detection on attributed networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 594–602. SIAM.

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., and Papalexakis, E. E. (2020). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177.

Fukumizu, K., Bach, F. R., and Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *J. Machine Learning Research*, 5(Jan):73–99.

Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. (2008). Kernel measures of conditional dependence. In *Advances in Neural Information Processing Systems*, pages 489–496.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proc. AAAI Conf. Artificial Intell.*

Hamilton, W. L., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1025–1035.

Ioannidis, V. N., Berberidis, D., and Giannakis, G. B. (2020). GraphSAC: Detecting anomalies in large-scale graphs. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learning Representations*.

Li, J., Dani, H., Hu, X., and Liu, H. (2017). Radar: Residual analysis for anomaly detection in attributed networks. In *IJCAI*, pages 2152–2158.

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

Lin, J. (1991). Divergence measures based on the shannon entropy. *IEEE Trans. Information Theory*, 37(1):145–151.

Manevitz, L. M. and Yousef, M. (2001). One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154.

Neyman, J. and Pearson, E. S. (1933). On the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 231(694-706):289–337.

Perozzi, B. and Akoglu, L. (2016). Scalable anomaly ranking of attributed neighborhoods. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 207–215. SIAM.

Qian, J. and Saligrama, V. (2014). Efficient minimax signal detection on graphs. In *Advances in Neural Information Processing Systems*, pages 2708–2716.

Qu, M., Bengio, Y., and Tang, J. (2019). GMNN: Graph Markov neural networks. In *Proc. Int. Conf. Machine Learning*.

Sharpnack, J. L., Krishnamurthy, A., and Singh, A. (2013). Near-optimal anomaly detection in graphs using lovasz extended scan statistic. In *Advances in Neural Information Processing Systems*, pages 1959–1967.

Shchur, O., Mumme, M., Bojchevski, A., and Günnemann, S. (2018). Pitfalls of graph neural network evaluation. *CoRR*.

Silverman, B. W. (1986). *Density estimation for statistics and data analysis*. Routledge.

Sun, J., Zhang, Y., Ma, C., Coates, M., Guo, H., Tang, R., and He, X. (2019). Multi-graph convolution collaborative filtering. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1306–1311. IEEE.

Sutherland, D. J., Tung, H.-Y., Strathmann, H., De, S., Ramdas, A., Smola, A., and Gretton, A. (2017). Generative models and model criticism via optimized maximum mean discrepancy. In *Proc. Int. Conf. Learning Representations*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. In *Proc. Int. Conf. Learning Representations*.

Waniek, M., Michalak, T. P., Wooldridge, M. J., and Rahwan, T. (2018). Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2):139.

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., and Leiserson, C. E. (2018). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In *2019 KDD Deep Learning on Graphs: Methods and Applications workshop (DLG'19)*.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019a). Simplifying graph convolutional networks. In *Proc. Int. Conf. Machine Learning*, pages 6861–6871, Long Beach, California, USA.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., and Zhu, L. (2019b). Adversarial examples for graph data: Deep insights into attack and defense. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 4816–4823.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. (2019). Topology attack and defense for graph neural networks: An optimization perspective. In *International Joint Conference on Artificial Intelligence, IJCAI*.

Xu, Y., Zhang, Y., Guo, W., Guo, H., Tang, R., and Coates, M. (2020). Graphsail: Graph structure aware incremental learning for recommender systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., and Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, pages 974–983.

Zhang, Y., Pal, S., Coates, M., and Ustebay, D. (2019). Bayesian graph convolutional neural networks for semi-supervised classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Zhou, K., Michalak, T. P., and Vorobeychik, Y. (2019). Adversarial robustness of similarity-based link prediction. In *The 19th IEEE International Conference on Data Mining*.

Zhu, D., Zhang, Z., Cui, P., and Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1399–1407.

Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proc. ACM Int. Conf. Knowledge Discovery & Data Mining*, pages 2847–2856.

Zügner, D. and Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. *CoRR*, abs/1902.08412.