# Sanitizing Backdoored Graph Neural Networks: A Multidimensional Approach

**Rong Zhao**, **Jilian Zhang**, **Yu Wang**, **Yinyan Zhang** and **Jian Weng**
College of Cyber Security, Jinan University, Guangzhou China
zr15625141885@stu.jnu.edu.cn, zhangjilian@jnu.edu.cn, 18236108352@163.com,
yinyan.zhang@connect.polyu.hk, cryptjweng@gmail.com

## Abstract

Graph Neural Networks (GNNs) are known to be prone to adversarial attacks, among which backdoor attack is a major security threat. By injecting backdoor triggers into a graph and assigning a target class label to nodes attached to the triggers, the attacker can mislead the GNN model trained on the poisoned graph to classify test nodes attached with a trigger to the target class. To defend against backdoor attacks, existing defense methods rely on anomaly detection in feature distribution or label transformation. However, these approaches are incapable of detecting in-distribution triggers or clean-label attacks that do not alter the class label of target nodes. To tackle these threats, we empirically analyze triggers from a multidimensional aspect, and our analysis shows that there are clear distinctions between trigger nodes and normal ones in terms of node feature values, node embeddings, and class prediction probabilities. Based on these findings, we propose a Multidimensional Anomaly Detection framework (MAD) that can effectively minimize the impact of triggers by pruning away anomalous nodes and edges. Extensive experiments show that at the cost of slight loss in clean classification accuracy, MAD achieves considerably lower attack success rate as compared to state-of-the-art backdoor defense methods.

## 1 Introduction

Graph Neural Networks (GNNs) [Kipf and Welling, 2016; Veličković *et al.*, 2017] have achieved remarkable success in many fields, such as social network analysis [Sankar *et al.*, 2021], recommender systems [Wu *et al.*, 2022], knowledge graph construction [Wang *et al.*, 2017], and bioinformatics [Zhao *et al.*, 2021]. GNNs effectively capture complex relationship within graph by aggregating information of nodes and their neighbors. Due to the capabilities of learning from graphs, GNNs are widely used for tasks like graph classification [Xu *et al.*, 2018], node classification [Hamilton *et al.*, 2017], and link prediction [Zhang and Chen, 2018].

Despite being a powerful technique for graph processing, GNNs are prone to suffer from various security threats, such as adversarial attacks [Zügner *et al.*, 2018; Li *et al.*, 2021], data privacy leakage [Duddu *et al.*, 2020; Wang *et al.*, 2023], model inversion attacks [Zhang *et al.*, 2022], and backdoor attacks [Zhang *et al.*, 2021]. In a backdoor attack, the attacker implants malicious triggers into the training data, and associates the triggers with some target labels during GNN training. When inferring test nodes adjacent to those triggers, the poisoned GNN model will classify the contaminated test nodes to the target class pre-specified by the attacker, while clean test nodes are predicted as normal.

Graph backdoor attacks have attracted tremendous attention from researchers, with stealthier and more effective attacks having been invented [Xi *et al.*, 2021; Dai *et al.*, 2023; Zhang *et al.*, 2024a]. Meanwhile, backdoor triggers have evolved from traditional Out-of-Distribution (OOD) triggers to In-Distribution (ID) triggers [Zhang *et al.*, 2024a]. Some researchers even propose clean-label attacks, that do not modify the class label of target nodes [Xu and Picek, 2022; Xing *et al.*, 2024]. This poses severe threats to GNNs, for instance, ID triggers are so similar to normal nodes that they can easily bypass traditional anomaly detection methods.

To protect GNNs from backdoor attacks, many defense methods have been invented [Downer *et al.*, 2024; Guan *et al.*, 2023; Jiang and Li, 2022; Yang *et al.*, 2023; Yuan *et al.*, 2024], where most of them resort to either feature anomaly-based detection [Dai *et al.*, 2023; Zhang *et al.*, 2024a] or label transformation-based detection [Zhang *et al.*, 2024b; Sui *et al.*, 2024]. Specifically, feature anomaly-based defense methods identify potential trigger nodes (i.e., the nodes in triggers) by detecting anomalies in node features, while label transformation-based ones rely on pruning trigger edges (i.e., the edges associated with trigger nodes) that lead to the transformation of labels of the target nodes. However, these defense methods are incapable of dealing with ID triggers and clean-label attacks, because it is tricky for them to discover these kinds of triggers whose distribution is similar to that of normal nodes.

In this paper, we conduct an in-depth investigation into trigger nodes and normal nodes from multidimensional aspects, including node features, node embeddings, and class prediction probabilities of nodes. And we find that there are obvious anomalies in trigger nodes, which distinguish them from the normal nodes. Based on these findings, we propose a simple yet effective multidimensional anomaly detection

(MAD) framework that can minimize the impact of triggers through identifying and eliminating anomalous nodes and edges. In summary, the contributions of this paper include:

- We systematically investigate trigger nodes and normal nodes from various aspects, and we found that there are clear distinctions between them, in terms of node features, node embeddings, and class prediction probabilities of nodes.
- By utilizing these findings, we propose a simple yet effective multidimensional anomaly detection framework called MAD, that combines three anomaly detection modules to defend against backdoor attacks for GNNs.
- We conduct extensive experiments on real datasets, and the results show that MAD significantly reduces attack success rate as compared to existing defense methods.

## 2 Related Work

### 2.1 Graph Neural Networks

Graph Neural Network (GNN) is a deep learning model designed specifically for graph data [Kipf and Welling, 2016; Veličković *et al.*, 2017; Hamilton *et al.*, 2017; Xu *et al.*, 2018], which is widely used for various tasks such as node classification [Hamilton *et al.*, 2017; Bojchevski *et al.*, 2020; Sun *et al.*, 2022], graph classification [Xu *et al.*, 2018; Bouritsas *et al.*, 2022; Wang and Ji, 2020], and link prediction [Zhang and Chen, 2018; Yun *et al.*, 2021; Song *et al.*, 2023]. GNNs model graph structure by aggregating information of nodes and their neighbors, thus capturing complex relationship between them. As GNNs keep evolving, many variant models have been proposed to deal with different data types and learning tasks, e.g., GraphSNN [Wijesinghe and Wang, 2022], PathNN [Michel *et al.*, 2023], LRGNN [Wei *et al.*, 2023], and ESC-GNN [Yan *et al.*, 2024].

### 2.2 Graph Backdoor Attack

Graph backdoor attack is a newly emerging attack to GNNs, where malicious triggers are injected into graph training data such that a GNN trained on the backdoored data will make incorrect predictions when encountering trigger data samples. Currently, there are various backdoor attack methods, e.g., [Zhang *et al.*, 2021] proposes to replace original subgraphs with fixed triggers and modify the graph label to the target label. [Xi *et al.*, 2021] generates adaptive triggers based on the features of different samples, making the attack more flexible and effective. [Dai *et al.*, 2023] designs an adaptive method to generate triggers that exhibit high cosine similarity with the target node. [Zhang *et al.*, 2024a] introduces ID triggers that resemble the feature value distribution of normal nodes, thus bypassing traditional anomaly detection methods. To further improve the stealthiness of triggers, clean-label graph backdoor attack methods are brought forward that do not modify the label of target node [Xu and Picek, 2022; Xing *et al.*, 2024; Fan and Dai, 2024].

### 2.3 Defense Against Graph Backdoor Attacks

Current backdoor defense methods can be categorized into detection methods [Downer *et al.*, 2024; Guan *et al.*, 2023] and purification methods [Jiang and Li, 2022; Yang *et al.*, 2023]. Detection methods aim to identify the presence of backdoor attacks on the graph, but cannot eliminate the triggers. Purification methods, however, try to identify and remove the influence of trigger nodes and edges, to ensure the correctness of GNNs. The majority of existing defense methods are purification methods, which can be roughly divided into feature anomaly-based defense methods [Dai *et al.*, 2023; Zhang *et al.*, 2024a] and label transformation-based defense methods [Zhang *et al.*, 2024b; Sui *et al.*, 2024].

Feature anomaly-based defense methods identify and eliminate trigger nodes by analyzing the difference between trigger node features and normal node features. Specifically, [Dai *et al.*, 2023] proposed a method called Prune, which calculates cosine similarity between nodes and prunes the edge between two nodes of low similarity score. As an extension of Prune, Prune+LD not only prunes edges but also discards the label of node incident to trigger edges. Another defense method called Outlier Detection (OD) was brought forward by [Zhang *et al.*, 2024a], which uses autoencoders to reconstruct node features and incident relations between nodes. Based on reconstruction errors, abnormal nodes can be identified. However, when facing ID triggers, the performance of OD will degrade significantly.

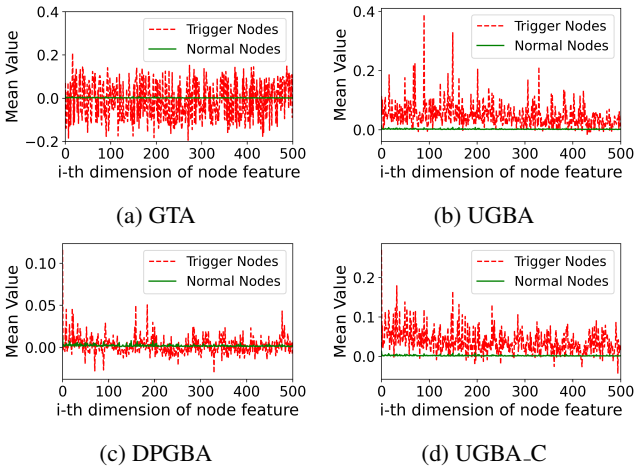
Label transformation-based defense methods utilize the difference between the predicted class label and the true class label to determine the presence of triggers. Specifically, [Zhang *et al.*, 2024b] proposed RIGBD to detect trigger nodes through edge perturbation, which also includes a robust training strategy to reduce the impact of triggers on the target nodes. [Sui *et al.*, 2024] introduced DMGNN that combines counterfactual explanations and generates different levels of perturbed graphs, then employs a denoising model to discover trigger nodes or edges that cause label changes during prediction. Both RIGBD and DMGNN, however, rely on transformation information of class labels. Therefore, they are incapable of dealing with clean-label backdoor attacks.

## 3 Preliminaries

In this paper, we focus on the problem of backdoor attacks for node classification tasks. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges. The node feature matrix is denoted by $\mathbf{X} \in \mathbb{R}^{n \times d}$, where $n$ is the number of nodes and $d$ the dimensionality of node feature. The adjacency matrix of $\mathcal{G}$ is $\mathcal{A} \in \mathbb{R}^{n \times n}$, where $\mathcal{A}_{ij} = 1$ if there is an edge between nodes $v_i$ and $v_j$, and 0 otherwise. The embedding of node $v_i$ at the $l$-th layer of GNN is denoted as $\mathbf{h}_i^l$, where $l$ is the layer number. The predicted class probability vector of node $v_i$ is $\hat{y}_i = (\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^k)$, where $\hat{y}_i^j$ is the predicted probability that $v_i$ belongs to the $j$-th class. The true class label of $v_i$ is represented by $y_i$, and the loss function of GNN model, denoted as $\mathcal{L}(\cdot)$, measures the difference between the model's prediction and the true class label.

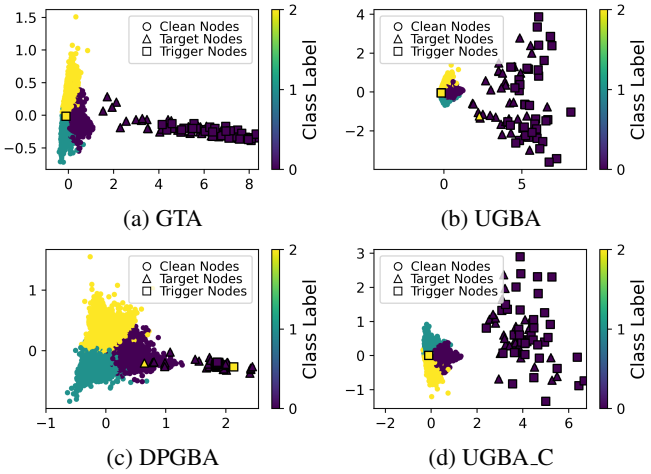
### 3.1 The Threat Model and Defense Model

**The Threat Model.** The primary goal of the attacker is to implant an unnoticeable backdoor trigger into a GNN model. When activated, the target node attacked by the trig-




(a) GTA (b) UGBA (c) DPGBA (d) UGBA_C

Figure 1: Mean of normal node features versus that of trigger node features generated by GTA, UGBA, DPGBA, and UGBA_C




(a) GTA (b) UGBA (c) DPGBA (d) UGBA_C

Figure 2: Embeddings of normal and of target nodes versus that of triggers generated by GTA, UGBA, DPGBA, and UGBA_C

ger will be classified to a pre-specified target class, whereas clean/normal nodes not attacked by the trigger will be classified normally. The attacker has access to the training data and can manipulate nodes, edges, and node features. The attacker may have full control of class labels, i.e., he can modify the target node labels, or he may carry out a clean-label attack, where no label modification occurs during training. However, the attacker has no access to the final GNN model.

**The Defense Model.** The defender's goal is to detect and then eliminate backdoor triggers while preserving the performance of GNN model as much as possible. This involves the discovery of abnormal nodes or edges, and the elimination of those nodes and edges that may belong to triggers. The defender has access to the GNN model, as well as intermediate results generated during model training and testing. Meanwhile, the defender can also access the data, including node features, but he does not know whether there are triggers in the data, nor the target class of the backdoor attack.

## 3.2 Anomaly Analysis of Trigger Nodes

In this section, we investigate anomalous characteristics of trigger nodes that distinguish themselves from normal nodes.

### Anomaly in Node Features

Existing backdoor attack methods wish to craft triggers as similar as possible to normal nodes, e.g., in-distribution (ID) triggers [Zhang *et al.*, 2024a]. Triggers are usually generated by using some artificial mechanisms, like Multi-Layer Perceptron (MLP). This kind of trigger generator, however, will inevitably result in triggers with unnatural feature values, because the linear layers and activation function in MLP prevent the feature values of triggers from being smooth and similar enough to that of the normal nodes.

To verify the difference between node features, we have conducted experiments on the Pubmed dataset, to compare normal nodes with trigger nodes generated by state-of-the-art (SOTA) graph backdoor attack methods, i.e., GTA [Xi *et al.*, 2021], UGBA and UGBA_C [Dai *et al.*, 2023], and DPGBA [Zhang *et al.*, 2024a]. Specifically, we compute the mean of each dimension of the normal node features and that of the trigger node features, respectively. From the results depicted in Figure 1, it is obvious that across feature dimensions, the mean value of the trigger nodes deviates drastically from that of the normal nodes. This confirms that it is possible to tell apart the trigger nodes from the normal nodes by scrutinizing the node features.

### Anomaly in Node Embeddings

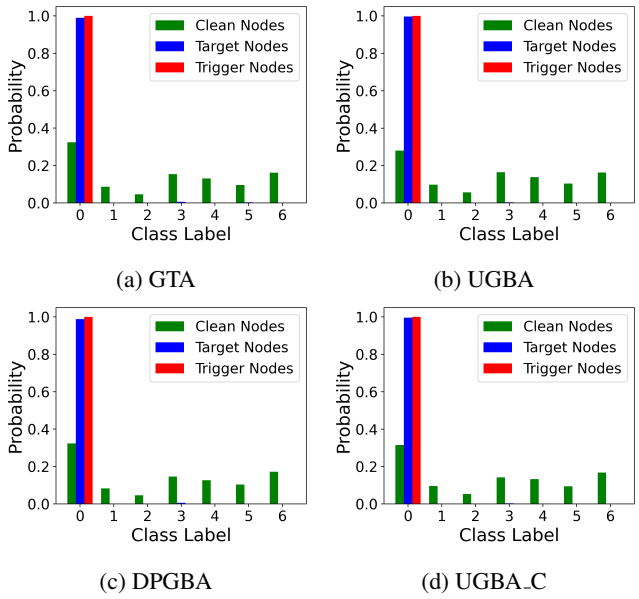
Backdoor triggers also attack target nodes by exploiting the message-passing mechanism of GNNs, where nodes update their embeddings through aggregating the features of neighboring nodes. Since node embedding and predicted class are highly correlated, after being poisoned by trigger nodes, the target node will drift from its original class towards the target class pre-defined by the attacker.

To verify the existence of classification drift, we have extracted outputs from the final layer of a GNN model, i.e., the embedding representation before softmax operation, and visualized the result in a 2-dimensional space. The results are depicted in Figure 2, from which we can see that the embeddings of trigger nodes and of target nodes significantly differ from the embeddings of normal nodes. Hence, this indicates that there is a clear distinction in node embeddings between normal nodes, target nodes, and trigger nodes.

### Anomaly in Class Prediction Probabilities of Nodes

For existing backdoor attack methods, their goal is to successfully induce the GNN model to classify the target node to the target class pre-defined by the attacker with high probability. This means that the trigger nodes and target nodes tend to be classified to the target class with very high probability, whereas most of the normal nodes do not gain abnormally high probability for some specific class.

To demonstrate, we calculate the average prediction probabilities of trigger nodes, target nodes, and normal nodes. From Figure 3 we can see that the average probability of classifying trigger nodes and target nodes to target class 0 is abnormally high, i.e., a significant deviation from that of the




(a) GTA (b) UGBA

(c) DPGBA (d) UGBA_C

Figure 3: Class prediction probabilities of normal, target, and trigger nodes generated by GTA, UGBA, DPGBA, and UGBA_C

normal nodes. This anomaly in class prediction probability is an important clue to trigger detection.

# 4 The Multidimensional Anomaly Detection Framework

Given that trigger nodes generated by existing attack methods incur various anomalies, we may discover trigger nodes by detecting whether there are anomalies in node features, node embeddings, and class prediction probabilities, respectively. In this section, we introduce MAD, a simple yet effective multidimensional anomaly detection framework to defend against graph backdoor attacks.

As shown in Figure 4, MAD performs trigger detection in three steps, namely, preprocessing, anomaly detection, and trigger pruning. During preprocessing, we use a Graph Convolutional Network (GCN) encoder to generate embedding for each node, which captures the relationship between nodes. Meanwhile, a Shadow GNN is trained to obtain class prediction probabilities for each node. At the anomaly detection step we calculate the standard deviation of node feature values, Euclidean distance between node embeddings, and entropy of class prediction probabilities, respectively. Then, those nodes or edges with abnormal evaluation values are marked as potential trigger nodes or edges. Finally, at the pruning step we eliminate those potential triggers, by removing nodes of the trigger and edges incident to them. In the following, we elaborate on how MAD works.

## 4.1 Feature Value Anomaly Detection

As shown in Section 3.2, feature values of trigger nodes differ significantly from those of normal nodes. To detect anomalies in feature values, we use standard deviation of feature values as an indicator. Specifically, given a feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, where $n$ is the number of nodes and $d$ the dimensionality of node feature, the feature vector of node $v_i$ is $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$, and the standard deviation $\sigma_i$ of $\mathbf{x}_i$ is calculated as:

$$\sigma_i = \sqrt{\frac{1}{d}\sum_{j=1}^{d}(x_{ij} - \mu_i)^2} \tag{1}$$

where $\mu_i$ is the mean of feature vector $\mathbf{x}_i$, computed as:

$$\mu_i = \frac{1}{d}\sum_{j=1}^{d} x_{ij} \tag{2}$$

Typically, the feature values of trigger nodes differ drastically from that of the normal nodes, since triggers are generated by some artificial mechanism, e.g., Multi-Layer Perceptron (MLP), resulting in unnatural feature values after being transformed by linear layers and activation function of MLP. Hence, nodes with suspiciously large standard deviation in feature values are likely to be trigger nodes. By setting an appropriate threshold on the standard deviation of feature values, we may pinpoint those potential triggers. In Section 5, we empirically investigate the setting of the threshold.

## 4.2 Node Embedding Anomaly Detection

To detect anomalies in node embeddings, we first train a Graph Convolutional Network (GCN) encoder to generate embedding for each node, which captures both feature information and structural information of nodes. Specifically, we generate node embedding $\mathbf{E}$ through multiple graph convolution layers, which can be represented as:

$$\mathbf{E} = \mathrm{GCN}(\mathbf{X}, \mathcal{A}) \tag{3}$$

where $\mathbf{X}$ is the node feature matrix, $\mathcal{A}$ the adjacency matrix of the graph. The GCN model captures structural information through $\mathcal{A}$ and obtains feature information through $\mathbf{X}$.

To train the GCN model, we minimize the following loss function $\mathcal{L}_{\mathrm{GCN}}$, which ensures that the predicted label of a node approaches its true label:

$$\mathcal{L}_{\mathrm{GCN}} = \sum_{v_i \in \mathcal{V}} \ell\big(\mathbf{softmax}(\mathbf{w}_i, \mathbf{h}_i),\ y_i\big) \tag{4}$$

where $\mathbf{w}_i$ is a weight for node $v_i$, $\mathbf{h}_i$ the embedding of $v_i$, $\mathbf{softmax}(\mathbf{w}_i, \mathbf{h}_i)$ the predicted label of $v_i$, $y_i$ the true label of $v_i$, and $\ell(\cdot, \cdot)$ the loss function (e.g., cross-entropy loss).

Once the GCN encoder is trained, we can obtain node embeddings using the GCN encoder. After computing similarities between the node embeddings, we are able to identify potential triggers. Specifically, for two neighboring nodes $v_i$ and $v_j$, we compute the Euclidean distance between their embeddings as follows:

$$d(\mathbf{h}_i, \mathbf{h}_j) = \|\mathbf{h}_i - \mathbf{h}_j\|_2 \tag{5}$$

where $\mathbf{h}_i$ and $\mathbf{h}_j$ are the embedding of nodes $v_i$ and $v_j$, respectively, and $\|\cdot\|_2$ the Euclidean distance function.

By computing the Euclidean distance between embeddings of adjacent nodes, we are able to find abnormal edges that connect two nodes far apart in the embedding space. We can mark those edges as trigger edges with high probability, because generally trigger nodes appear farther away from normal nodes in the embedding space.

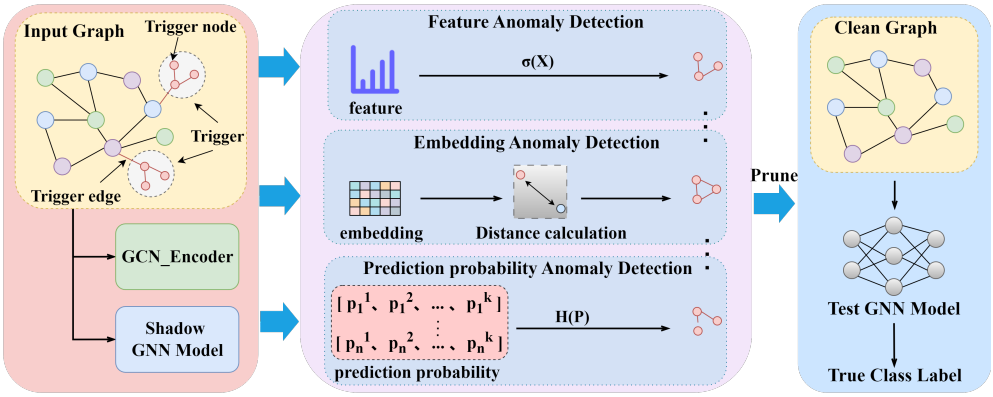



Figure 4: The proposed multidimensional anomaly detection framework (MAD) for graph backdoor attack defense

### 4.3 Class Prediction Probability Anomaly Detection

In the preprocessing step, we train a shadow graph neural network (ShadowGNN) model to obtain class prediction probabilities for each node. The training process of ShadowGNN is similar to that of the GNN model under attack, the difference is that the former serves as a surrogate model to predict class labels for nodes. Specifically, the input to ShadowGNN includes the node feature matrix $\mathbf{X}$ and the adjacency matrix $\mathcal{A}$. Upon finish of training, ShadowGNN outputs class prediction $\hat{\mathbf{Y}}$ for all the nodes, as shown below:

$$\hat{\mathbf{Y}} = \text{ShadowGNN}(\mathbf{X}, \mathcal{A}) \tag{6}$$

Here, an element $\hat{y}_i = (\hat{y}_i^1, \hat{y}_i^2, \ldots, \hat{y}_i^k) \in \hat{\mathbf{Y}}$ is the class prediction probability vector for node $v_i$, with $\hat{y}_i^j$ being the probability that $v_i$ belongs to the $j$-th class.

To measure anomaly in class prediction probability vector $\hat{y}_i$, we calculate the entropy of $\hat{y}_i$, which is a metric used to measure information purity, as computed below:

$$H(\hat{y}_i) = -\sum_{j=1}^{k} \hat{y}_i^j \log(\hat{y}_i^j) \tag{7}$$

where $H(\hat{y}_i)$ is the entropy of the class prediction probability vector of $v_i$. A smaller entropy value of a node indicates higher certainty in the prediction for a node, i.e., the node is predicted to belong to some specific class with remarkably higher probability than to other classes. In contrast, a larger entropy value suggests that the prediction probabilities of that node are relatively even across all classes.

By calculating the entropy of the class prediction probability for each node, we may mark those nodes with very small entropy values as triggers, because the attacker's goal is to induce GNN model to predict a target node to a target class with very high probability through the influence of triggers. In Section 5, we empirically analyze the setting of the threshold for the entropy value of class prediction probability.

### 4.4 Pruning Anomalous Nodes and Edges

Having detected suspicious nodes and edges with anomalies in feature values, embeddings, and class prediction probabilities, MAD employs a simple yet efficient strategy to eliminate those anomalous nodes that may be triggers. Specifically, for the identified anomalous nodes, we delete these nodes and all edges incident to them as well, so as to minimize the impact of trigger nodes on the normal ones. For the detected anomalous edges, on the other hand, we simply remove them from the graph as these edges may be generated by the attacker. In this way, we cut off the bogus link that may be introduced to connect the trigger node and target node, thus diminishing the influence of triggers on the GNN model during training.

| Dataset | #Nodes | #Edges | #Feature | #Classes |
|---|---|---|---|---|
| Cora | 2,708 | 5,429 | 1,443 | 7 |
| Pubmed | 19,717 | 44,338 | 500 | 3 |
| Flickr | 89,250 | 899,756 | 500 | 7 |
| OGB-arxiv | 169,343 | 1,166,243 | 128 | 40 |

Table 1: Dataset Statistics

## 5 Experimental Evaluation

We focus on node classification task and empirically evaluate our MAD framework, and we aim to answer three questions:

- Q1: Is MAD capable of defending against existing SOTA graph backdoor attack methods?
- Q2: How effective is each of the three anomaly detection modules of MAD?
- Q3: How do the thresholds used in the three anomaly detection modules impact the performance of MAD?

### 5.1 Setup

**Datasets.** Four datasets are used, where Cora, Pubmed, and OGB-arxiv are academic citation networks, while Flickr is a large-scale social network. Statistics of these datasets are summarized in Table 1.

**Backdoor Attack Methods for GNN.** We employ four SOTA backdoor attack methods, namely, GTA, UGBA, DPGBA, and UGBA_C. Specifically, GTA generates adaptive triggers; UGBA achieves stealthy backdoor attacks by optimizing the selection of poisoned nodes and generation of triggers. DPGBA uses ID triggers to bypass traditional anomaly detection, while UGBA_C is a clean-label attack method.

| Dataset | Attack method | None | Prune | Prune+LD | OD | RIGBD | DMGNN | MAD |
|---|---|---|---|---|---|---|---|---|
| Cora | GTA | 88.80\|**83.60** | 17.63\|83.06 | 18.35\|80.17 | **00.04**\|83.40 | 05.90\|79.40 | 00.50\|82.10 | 03.76\|83.31 |
| | UGBA | 96.70\|83.58 | 98.89\|82.66 | 95.30\|79.90 | **00.03**\|**83.60** | 07.20\|80.50 | 01.10\|81.70 | 02.07\|83.43 |
| | DPGBA | 97.69\|83.60 | 90.20\|80.20 | 88.30\|79.30 | 93.90\|83.50 | 12.30\|79.90 | 01.40\|81.10 | **00.53**\|**84.10** |
| | UGBA_C | 86.13\|83.38 | 88.18\|83.28 | 91.50\|80.27 | 06.04\|83.62 | – | – | **03.76**\|**83.83** |
| Pubmed | GTA | 90.94\|84.93 | 28.10\|85.05 | 22.00\|83.76 | **00.03**\|84.70 | 06.30\|80.10 | 00.90\|83.60 | 00.70\|**85.06** |
| | UGBA | 88.99\|**85.12** | 89.87\|85.09 | 90.06\|83.75 | **00.01**\|85.00 | 05.50\|78.20 | 01.50\|82.40 | 00.82\|85.10 |
| | DPGBA | 91.91\|85.00 | 89.40\|80.70 | 88.20\|80.10 | 91.80\|**85.10** | 11.80\|78.50 | 02.10\|81.90 | **00.68**\|84.85 |
| | UGBA_C | 82.52\|84.96 | 89.20\|85.03 | 87.76\|83.73 | 03.78\|**85.18** | – | – | **00.34**\|85.08 |
| Flickr | GTA | 88.52\|44.70 | 00.00\|42.71 | 00.00\|44.99 | 00.00\|45.10 | 07.90\|41.90 | 01.30\|**46.20** | **00.00**\|44.84 |
| | UGBA | 95.36\|45.16 | 90.34\|42.99 | 96.81\|42.14 | 00.00\|**45.40** | 08.70\|40.30 | 01.70\|44.90 | **00.00**\|45.16 |
| | DPGBA | 96.43\|44.96 | 87.20\|40.50 | 85.60\|41.10 | 94.80\|**45.80** | 12.80\|40.80 | 02.50\|43.80 | **00.00**\|44.96 |
| | UGBA_C | 99.57\|44.20 | 97.91\|42.14 | 96.60\|**44.41** | 00.00\|42.50 | – | – | **00.00**\|44.34 |
| OGB-arxiv | GTA | 93.51\|64.36 | 00.01\|63.97 | 00.03\|64.30 | 00.01\|64.90 | 06.50\|60.90 | 00.70\|**67.10** | **00.00**\|64.67 |
| | UGBA | 98.38\|65.56 | 93.07\|62.58 | 90.95\|63.19 | 00.01\|64.50 | 08.20\|61.70 | 00.90\|**65.80** | **00.00**\|65.33 |
| | DPGBA | 93.87\|65.35 | 88.90\|60.30 | 89.50\|61.50 | 92.40\|**65.40** | 11.40\|60.30 | 01.30\|64.90 | **00.01**\|64.70 |
| | UGBA_C | 78.64\|**65.39** | 81.11\|63.75 | 79.21\|64.15 | **00.22**\|64.91 | – | – | 00.28\|63.89 |

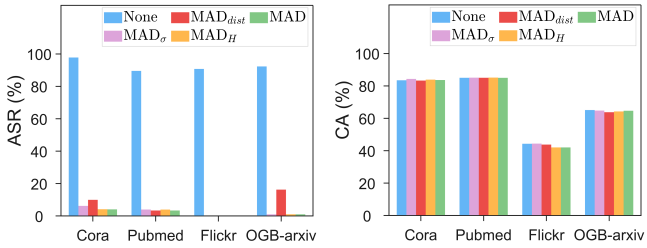
Table 2: Comparison between MAD and state-of-the-art backdoor defense methods (ASR (%) | Clean Accuracy (%))

**Baseline Defense Methods.** We compare MAD against five SOTA backdoor defense methods, i.e., Prune [Dai *et al.*, 2023], Prune+LD, OD [Zhang *et al.*, 2024a], RIGBD [Zhang *et al.*, 2024b], and DMGNN [Sui *et al.*, 2024].

**Evaluation Metrics.** We use Attack Success Rate (ASR) and Clean Accuracy (CA) to evaluate the effectiveness of backdoor defense methods. Specifically, ASR is defined as the proportion of target nodes that are successfully classified as the target class specified by the attacker, while CA is the prediction accuracy of GNN on normal nodes.

**Implementation Details.** Each of the experiment datasets is divided into a training set and a test set, where the former and the latter contain 80% and 20% of data samples, respectively. We inject backdoors to graph data strictly following the same way as in SOTA backdoor attack methods. Meanwhile, we define the attack budget as the number of nodes poisoned by triggers, which is set to 10, 40, 80, and 160 for Cora, PubMed, Flickr, and OGB-arxiv, respectively.

As discussed in Section 4, for the computed standard deviation of feature values, Euclidean distance between node embeddings, and entropy of class prediction probabilities, we denote their corresponding threshold as $t_\sigma$, $t_{dist}$, and $t_H$, respectively. Specifically, we set $t_\sigma$ to be the cut-off value of nodes with top 3% highest deviations of node features for Cora and Pubmed, while $t_\sigma$ is set to be that of the nodes with top 1% highest deviations of node features for Flickr and OGB-arxiv, because the two datasets are larger in size. Similarly, we set $t_{dist}$ in the same way as $t_\sigma$, i.e., the cut-off value of node pairs with top 3%, 3%, 1%, and 1% highest Euclidean distance between node embeddings for the four datasets, respectively. As for $t_H$, however, it is defined as the cut-off value of nodes with the top 3%, 3%, 1%, and 1% smallest entropy for the four datasets, respectively. Nodes that exceed the three thresholds are regarded as potential trigger nodes or edges, and they must be removed from the graph. We employ three GNN models, i.e., GCN, GAT, and GraphSAGE when testing, and we report the average ASR and CA.




(a) ASR (b) CA

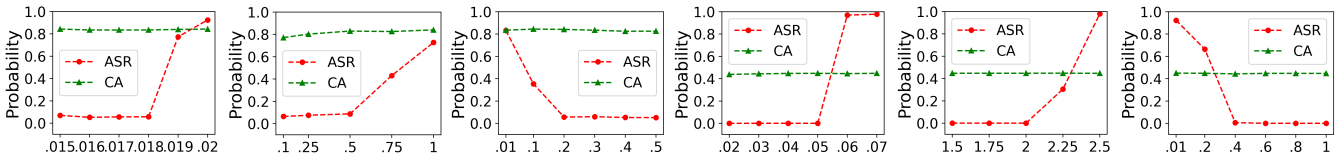
Figure 5: Performance of each anomaly detection module of MAD

## 5.2 Performance of MAD (Q1)

To answer Q1, we compare MAD with five SOTA baseline defense methods, namely, Prune, Prune+LD, OD, RIGBD, and DMGNN, when defending against four existing attack methods, i.e., GTA, UGBA, DPGBA, and UGBA_C. The results are shown in Table 2, and the column 'None' stands for the case where no backdoor defense method is used in GNN.

From Table 2 we can see that MAD overwhelmingly outperforms Prune, Prune+LD, RIGBD, and DMGNN, except that MAD incurs slight loss in CA on Flickr and OGB-arxiv, as compared to DMGNN under GTA and UGBA attacks. Note that Prune and Prune+LD result in very high ASR in most cases, meaning that they are incapable of effectively eliminating triggers generated by SOTA attack methods. The rationale is that trigger nodes generated by UGBA, DPGBA, and UGBA_C exhibit high cosine similarity with their neighboring nodes, making it difficult for Prune and Prune+LD to differentiate between trigger nodes and normal nodes.

Note that although OD achieves very low ASR on all the datasets under the attacks of GTA, UGBA, and UGBA_C, the ASR of OD remains at as high as 91.8% when defending against DPGBA attack. This demonstrates that OD is effective in defending against OOD triggers (generated by GTA, UGBA, and UGBA_C), yet it cannot deal with ID trig-




(a) $t_\sigma$ (Cora) (b) $t_{dist}$ (Cora) (c) $t_H$ (Cora) (d) $t_\sigma$ (Flickr) (e) $t_{dist}$ (Flickr) (f) $t_H$ (Flickr)

Figure 6: Impact of thresholds on performance of GNN model when employing MAD to defend against DPGBA attack

gers (generated by DPGBA). In contrast, the maximal ASR of MAD is only 0.68% when defending against DPGBA. Therefore, MAD is capable of defending both OOD triggers and ID triggers, while maintaining high CA for GNN models.

### 5.3 Module Performance Analysis (Q2)

To systematically evaluate the performance of each of the three anomaly detection modules of MAD, we conduct experiments separately for each module. Specifically, we modify the second step of MAD by keeping the feature anomaly detection module while removing the rest two, and we denote this modified MAD framework as $\text{MAD}_\sigma$. Similarly, we keep the node embedding anomaly detection module and the class prediction probability anomaly detection module in turn while discarding the rest two modules, and we denote these two modifications as $\text{MAD}_{dist}$ and $\text{MAD}_H$, respectively.

Figure 5 depicts the results of five defense strategies, i.e., MAD, $\text{MAD}_\sigma$, $\text{MAD}_{dist}$, $\text{MAD}_H$, and 'None' that denotes the case where no defense method is employed in GNN, when facing DPGBA attack. As shown in Figure 5(a), it is obvious that for None, DPGBA incurs a remarkably high ASR ($\geq$ 90%) on the GNN model over all datasets, whereas after adopting one of our defense methods, i.e., MAD, $\text{MAD}_\sigma$, $\text{MAD}_{dist}$ and $\text{MAD}_H$, ASR of DPGBA drops tremendously over all datasets. Note that for OGB-arxiv, $\text{MAD}_{dist}$ results in about 15% ASR of DPGBA. The reason might be that OGB-arxiv has a larger number of classes, leading to less distinguishable embedding distance between nodes.

Meanwhile, as shown in Figure 5(b), no matter which of the five defense methods is adopted by GNN model, the classification accuracy (CA) remains relatively stable when facing DPGBA attack. That is, our four defense methods successfully defend GNN model against DPGBA while introducing negligible degradation of CA. It is worth noting that although achieving slightly lower CA on the Flickr dataset, MAD gains the lowest ASR over all the four datasets, as compared to $\text{MAD}_\sigma$, $\text{MAD}_{dist}$, and $\text{MAD}_H$.

### 5.4 Impact of Thresholds on MAD (Q3)

In this section, we investigate the impact of thresholds $t_\sigma$, $t_{dist}$, and $t_H$ of MAD on the GNN model when confronting DPGBA attacks. We vary the thresholds in such a way that they increase from the cut-off value of top 0.1% nodes to that of top 3% nodes. Due to space limitations, we only report results on Cora and Flickr. As shown in Figure 6, CA of the GNN model remains relatively stable with the three thresholds. For $t_\sigma$, ASR remains very low before 0.018 and 0.05 for Cora and Flickr, respectively, after which ASR increases significantly. A similar trend can be observed for $t_{dist}$. On the contrary, ASR drops steadily from more than 80% to less than 1.0% when $t_H$ decreases to 0.2 and 0.4 for Cora and Flickr, respectively. Hence, to achieve the best performance of MAD to defend against backdoor attacks, one may need to tune the thresholds to find corresponding optimal values according to the dataset at hand.

| Trigger Size | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| None | 86.1 | 90.3 | 91.5 | 91.7 | 91.3 |
| MAD | 0.9 | 1.2 | 1.1 | 1.2 | 1.3 |

Table 3: ASR(%) under different sized triggers

### 5.5 Impact of Trigger Configuration on MAD

We conducted experiments with different trigger configurations, i.e., varying trigger size (the number of nodes in a trigger, which is normally set to 3 in literature that employs MLP to generate triggers), and attack budget (the number of nodes poisoned by triggers), against the latest attack method DPGBA on Pubmed. From Table 3 and 4 we can see that under different configurations, the ASR obtained by our MAD is below 1.3%, as compared to more than 86% when there is no defense (Note) against the latest attack method DPGBA.

| ATK Budget | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|
| None | 89.8 | 90.1 | 91.5 | 92.3 | 91.9 |
| MAD | 0.7 | 0.8 | 1.1 | 1.2 | 1.2 |

Table 4: ASR(%) under different attack (ATK) budgets

## 6 Conclusion

In this paper, we studied the problem of defending graph neural network (GNN) against backdoor attacks. We found that there are anomalies in node features, node embeddings, and class prediction probabilities of the triggers generated by state-of-the-art (SOTA) backdoor attack methods, which distinguish the trigger nodes from the normal nodes. We designed a simple yet effective multidimensional anomaly detection framework (MAD) for GNN models to defend against potential triggers through the identification and elimination of anomalous nodes and edges. Extensive experiments on real datasets confirmed that at the cost of slight loss in clean accuracy, MAD achieves considerably lower attack success rate as compared to SOTA defense methods. This is especially so when defending against the latest attack method DPGBA.

## Acknowledgements

This paper was supported by National Key R&D Program of China (2022YFB3103100), NSFC (62020106013 and 62472197), National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong-Hong Kong Joint Laboratory for Data Security and Privacy Preserving, and the Engineering Research Center of Trustworthy Al, Ministry of Education, Jinan University. Jian Weng was supported in part by NSFC (No. 62332007 and U22B2028), the Science and Technology Major Project of Tibetan Autonomous Region of China (No. XZ202201ZD0006G), and the Open Research Fund of Machine Learning and Cyber Security Interdiscipline Research Engineering Center of Jiangsu Province (No. SDGC2131). Jilian Zhang is the corresponding author.

## References

[Bojchevski *et al.*, 2020] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2464–2473, 2020.

[Bouritsas *et al.*, 2022] Giorgos Bouritsas, Fabrizio Frasca, Stefanos Zafeiriou, and Michael M Bronstein. Improving graph neural network expressivity via subgraph isomorphism counting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(1):657–668, 2022.

[Dai *et al.*, 2023] Enyan Dai, Minhua Lin, Xiang Zhang, and Suhang Wang. Unnoticeable backdoor attacks on graph neural networks. In *Proceedings of the ACM Web Conference 2023*, pages 2263–2273, 2023.

[Downer *et al.*, 2024] Jane Downer, Ren Wang, and Binghui Wang. Securing gnns: Explanation-based identification of backdoored training graphs. *arXiv preprint arXiv:2403.18136*, 2024.

[Duddu *et al.*, 2020] Vasisht Duddu, Antoine Boutet, and Virat Shejwalkar. Quantifying privacy leakage in graph embedding. In *MobiQuitous 2020-17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 76–85, 2020.

[Fan and Dai, 2024] Xuanhao Fan and Enyan Dai. Effective clean-label backdoor attacks on graph neural networks. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 3752–3756, 2024.

[Guan *et al.*, 2023] Zihan Guan, Mengnan Du, and Ninghao Liu. Xgbd: Explanation-guided graph backdoor detection. *arXiv preprint arXiv:2308.04406*, 2023.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[Jiang and Li, 2022] Bingchen Jiang and Zhao Li. Defending against backdoor attack on graph nerual network by explainability. *arXiv preprint arXiv:2209.02902*, 2022.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Li *et al.*, 2021] Jintang Li, Tao Xie, Liang Chen, Fenfang Xie, Xiangnan He, and Zibin Zheng. Adversarial attack on large scale graph. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):82–95, 2021.

[Michel *et al.*, 2023] Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *International Conference on Machine Learning*, pages 24737–24755. PMLR, 2023.

[Sankar *et al.*, 2021] Aravind Sankar, Yozen Liu, Jun Yu, and Neil Shah. Graph neural networks for friend ranking in large-scale social platforms. In *Proceedings of the Web Conference 2021*, pages 2535–2546, 2021.

[Song *et al.*, 2023] Xiran Song, Jianxun Lian, Hong Huang, Zihan Luo, Wei Zhou, Xue Lin, Mingqi Wu, Chaozhuo Li, Xing Xie, and Hai Jin. xgcn: An extreme graph convolutional network for large-scale social link prediction. In *Proceedings of the ACM Web Conference 2023*, pages 349–359, 2023.

[Sui *et al.*, 2024] Hao Sui, Bing Chen, Jiale Zhang, Chengcheng Zhu, Di Wu, Qinghua Lu, and Guodong Long. Dmgnn: Detecting and mitigating backdoor attacks in graph neural networks. *arXiv preprint arXiv:2410.14105*, 2024.

[Sun *et al.*, 2022] Mingchen Sun, Kaixiong Zhou, Xin He, Ying Wang, and Xin Wang. Gppt: Graph pre-training and prompt tuning to generalize graph neural networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1717–1727, 2022.

[Veličković *et al.*, 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[Wang and Ji, 2020] Zhengyang Wang and Shuiwang Ji. Second-order pooling for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(6):6870–6880, 2020.

[Wang *et al.*, 2017] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE transactions on knowledge and data engineering*, 29(12):2724–2743, 2017.

[Wang *et al.*, 2023] Huanran Wang, Wu Yang, Dapeng Man, Wei Wang, and Jiguang Lv. Anchor link prediction for privacy leakage via de-anonymization in multiple social networks. *IEEE Transactions on Dependable and Secure Computing*, 20(6):5197–5213, 2023.

[Wei *et al.*, 2023] Lanning Wei, Zhiqiang He, Huan Zhao, and Quanming Yao. Search to capture long-range dependency with stacking gnns for graph classification. In *Pro-*

*ceedings of the ACM Web Conference 2023*, pages 588–598, 2023.

[Wijesinghe and Wang, 2022] Asiri Wijesinghe and Qing Wang. A new perspective on" how graph neural networks go beyond weisfeiler-lehman?". In *International Conference on Learning Representations*, 2022.

[Wu *et al.*, 2022] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[Xi *et al.*, 2021] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1523–1540, 2021.

[Xing *et al.*, 2024] Xiaogang Xing, Ming Xu, Yujing Bai, and Dongdong Yang. A clean-label graph backdoor attack method in node classification task. *Knowledge-Based Systems*, 304:112433, 2024.

[Xu and Picek, 2022] Jing Xu and Stjepan Picek. Poster: clean-label backdoor attack on graph neural networks. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 3491–3493, 2022.

[Xu *et al.*, 2018] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[Yan *et al.*, 2024] Zuoyu Yan, Junru Zhou, Liangcai Gao, Zhi Tang, and Muhan Zhang. An efficient subgraph gnn with provable substructure counting power. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3702–3713, 2024.

[Yang *et al.*, 2023] Xiao Yang, Gaolei Li, Xiaoyi Tao, Chaofeng Zhang, and Jianhua Li. Black-box graph backdoor defense. In *International Conference on Algorithms and Architectures for Parallel Processing*, pages 163–180. Springer, 2023.

[Yuan *et al.*, 2024] Dingqiang Yuan, Xiaohua Xu, Lei Yu, Tongchang Han, Rongchang Li, and Meng Han. E-sage: Explainability-based defense against backdoor attacks on graph neural networks. In *International Conference on Wireless Artificial Intelligent Computing Systems and Applications*, pages 402–414. Springer, 2024.

[Yun *et al.*, 2021] Seongjun Yun, Seoyoon Kim, Junhyun Lee, Jaewoo Kang, and Hyunwoo J Kim. Neo-gnns: Neighborhood overlap-aware graph neural networks for link prediction. *Advances in Neural Information Processing Systems*, 34:13683–13694, 2021.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31, 2018.

[Zhang *et al.*, 2021] Zaixi Zhang, Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Backdoor attacks to graph neural networks. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, pages 15–26, 2021.

[Zhang *et al.*, 2022] Zaixi Zhang, Qi Liu, Zhenya Huang, Hao Wang, Chee-Kong Lee, and Enhong Chen. Model inversion attacks against graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):8729–8741, 2022.

[Zhang *et al.*, 2024a] Zhiwei Zhang, Minhua Lin, Enyan Dai, and Suhang Wang. Rethinking graph backdoor attacks: A distribution-preserving perspective. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 4386–4397, 2024.

[Zhang *et al.*, 2024b] Zhiwei Zhang, Minhua Lin, Junjie Xu, Zongyu Wu, Enyan Dai, and Suhang Wang. Robustness-inspired defense against backdoor attacks on graph neural networks. *arXiv preprint arXiv:2406.09836*, 2024.

[Zhao *et al.*, 2021] Chengshuai Zhao, Shuai Liu, Feng Huang, Shichao Liu, and Wen Zhang. Csgnn: Contrastive self-supervised graph neural network for molecular interaction prediction. In *IJCAI*, pages 3756–3763, 2021.

[Zügner *et al.*, 2018] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2847–2856, 2018.