Understanding Universal Adversarial Attack and Defense on Graph

Tianfeng Wang, Command and Control Engineering College, Army Engineering University of PLA, China
Zhison Pan, Command and Control Engineering College, Army Engineering University of PLA, China*
Guyu Hu, Command and Control Engineering College, Army Engineering University of PLA, China
Yexin Duan, Command and Control Engineering College, Army Engineering University of PLA, China
Yu Pan, Command and Control Engineering College, Army Engineering University of PLA, China

ABSTRACT

Compared with traditional machine learning models, graph neural networks (GNNs) have distinct advantages in processing unstructured data. However, the vulnerability of GNNs cannot be ignored. Graph universal adversarial attack is a special type of attack on graph which can attack any targeted victim by flipping edges connected to anchor nodes. In this paper, the authors propose the forward-derivative-based graph universal adversarial attack (FDGUA). Firstly, they point out that one node as training data is sufficient to generate an effective continuous attack vector. Then they discretize the continuous attack vector based on forward derivative. FDGUA can achieve impressive attack performance that three anchor nodes can result in attack success rate higher than 80% for the dataset Cora. Moreover, they propose the first graph universal adversarial training (GUAT) to defend against universal adversarial attack. Experiments show that GUAT can effectively improve the robustness of the GNNs without degrading the accuracy of the model.

KEYWORDS

Class-Discriminative Graph Universal Attack, Graph Adversarial, Graph Neural Networks, Graph Universal Attack

1. INTRODUCTION

All kinds of relationships can be represented by graphs, including social relationships (Gupta et al., 2018), paper citation relationships (Sen et al., 2008), and communication topologies (Leskovec et al., 2007). Compared with traditional machine learning methods, graph neural networks (GNNs) can better model complex relationships. For this reason, GNNs have attracted extensive attention. Many representative models have also been proposed, such as GCN (Kipf et al., 2017), GAT (Veličković et al., 2017) and GraphSAGE (Hamilton et al., 2017). Generally, GNNs realize information transfer between adjacent nodes through well-designed aggregation operation. The obtained graph representations can be applied to various downstream tasks, such as node classification (Wu et al., 2019; Xu et al., 2019), graph classification (Xie & Ying, 2021; Zhang et al., 2019) and community discovery (Chen et al., 2019; Zhang et al., 2020; Zhang et al., 2019). In addition, abstract contents, including images (Nhi
et al., 2022) and documents (Stylianou et al., 2022; Ismail et al., 2022; Urkalan & Geetha, 2020), can be interpreted as nodes in the graph. The graph-based methods help discovering the relationship among contents.

However, GNNs inherit the vulnerability of deep neural networks (DNNs), which may be misguided by unnoticeable perturbation. The research of adversarial attacks on GNNs will clarify the vulnerability of GNNs and improve GNN models. In general, adversarial attacks on GNNs can be categorized into three types according to attack goals. The first type, the global attack on topology, includes CE-PGD (Xu et al., 2019) and Meta-Attack (Zugner et al., 2019). This type aims to degrade the overall performance of GNNs. The second type, the target-dependent attack, includes Nettack (Dai et al., 2018), FGA (Chen et al., 2018), and IG-Attack (Wu et al., 2019). This type attacks a single target node through direct or indirect ways. The third type, the universal attack on graph, aims to achieve the target-independent attack against all nodes. Graph universal attack (GUA) first defines the form of universal attack on the graph that any node can be attacked by flipping edges connected to anchor nodes (Zang et al., 2021). In this paper, we explore graph universal attack and defense on this basis.

Because the graph structure is discrete, universal adversarial attack methods in the image domain cannot be directly transferred to graphs. The current study divides the process of graph universal attack into two steps. First, it generates the continuous attack vector. Second, it discretizes the continuous attack vector. In reviewing GUA, it first generates the continuous attack vector with universal adversarial perturbations (UAP, Moosavi-Dezfooli et al., 2017). Then, it discretizes the continuous attack vector by setting a fixed threshold. UAP is a classical method to calculate universal perturbation in the image domain. UAP first initializes the perturbation with zero vector and then accumulates the perturbation by traversing the training set with DeepFool (Moosavi-Dezfooli et al., 2016) until the targeted attack rate is achieved. Finally, the perturbation generated by UAP can effectively attack different images and be transferable among different models. The DeepFool method calculates minimal attack perturbation. Compared with attack methods in (Goodfellow et al., 2015) (Szegedy et al., 2014), DeepFool can achieve the same attack success rate with smaller perturbation. However, it is debatable to use UAP directly to generate a continuous attack vector. Due to the sparsity of the graph structure, one node as training data is sufficient to generate an effective continuous attack vector with DeepFool. It is not necessary to traverse all the training nodes. In addition, the heuristic discretization of attack vector makes the performance of GUA unstable because the magnitude of perturbation does not determine the effectiveness of attack. Therefore, we propose a new discretization method based on forward derivative. Here, the forward derivative is used to measure the attack contribution of each entry in the attack vector. Specifically, we first calculate the partial derivative of the classified probabilities to each entry in the attack vector and then discretize the vector based on the partial derivative. Compared with the discretization method with fixed threshold, our discretization method can retain more effective components in the continuous attack vector. Based on these observations, we propose the forward-derivative-based graph universal adversarial attack (FDGUA). Moreover, we find that FDGUA can be easily extended to be a class-discriminative graph universal attack (CD-FDGUA). The pure graph universal attack cannot distinguish the classes of attacked nodes, meaning that all nodes will be affected by anchor nodes. In some scenarios with high security requirements, such attack behavior may be detected by monitoring system. In this paper, a class-discriminative graph universal attack is proposed by simply extending FDGUA, which provides ideas for future researches on this problem. Specifically, we combine the anchor nodes with different directions to balance the effects generated by attack. Compared with FDGUA, CD-FDGUA can control the classes of affected nodes in a more flexible way. Meanwhile, results show that CD-FDGUA can achieve a high absolute accuracy drop gap. Moreover, we discuss the defense against FDGUA and propose the first graph universal adversarial training (GUAT). The existing adversarial training methods on graph are aimed at global attack and target-dependent attack, which cannot effectively defend against graph universal attack. Focusing on the particularity of the graph universal attack, we elaborate design the embedding mode of the adversarial samples and the process of adversarial training. By
appropriately adjusting the size of anchor pool and the number of embedded adversarial samples in the adversarial training, the robustness of model will be greatly improved.

The contributions of this paper are summarized as follows.

We analyze the shortcomings of GUA and propose a simple and effective graph universal attack method titled FDGUA. Compared to GUA, FDGUA utilizes fewer training samples to screen more aggressive anchor nodes. Experiments show that three anchor nodes can result in an attack success rate higher than 80% for Cora.

By extending FDGUA, we propose a new class-discriminative graph universal attack method CD-FDGUA. In CD-FDGUA, we balance the influence of attacks by adjusting the combination proportion of anchor nodes in different attack directions. Experiments show that CD-FDGUA can achieve a high attack success rate of targeted classes and keep the non-targeted classes unaffected, which results in a high absolute accuracy drop gap.

Moreover, we propose the first graph universal adversarial training. Considering the particularity of graph universal attack, we design flexible embedding method of adversarial samples in GUAT. Experiments show that GUAT can effectively improve the robustness of the model without degrading the accuracy.

The following section introduces related works about adversarial attack on graphs, universal attack in the image domain, and adversarial training on graphs. Then, the article introduces and formally describes the settings of the graph universal adversarial attack. FDGUA is then proposed. The article introduces the generation of target-independent continuous attack vectors before explaining how to discretize the continuous attack vector. Next, the article extends the FDGUA and proposes the CD-FDGUA. This is followed by the first GUAT, which can effectively resist the influence of the graph universal attack. Finally, the research verifies the validity of the algorithms via experiments.

2. RELATED WORKS

This section is divided into three parts to introduce the research related to the graph universal adversarial attack. First, we introduce the methods surrounding the graph adversarial attack. Currently, most works about graph adversarial attacks focus on global attack and target-dependent attack. Then, considering that there is limited research on graph universal adversarial attack, we introduce some methods about universal attack in the image domain, which is instructive for graph universal adversarial attack. Finally, we review the methods about graph adversarial training.

According to the purpose of attackers, the methods about graph adversarial attack can be divided into global adversarial attack, target-dependent adversarial attack and universal adversarial attack. Given a graph $\mathcal{G}$ where the adjacency matrix is $A$ and the feature of nodes is $X$, the goal of attackers is to find an imperceptible perturbation to mislead the targeted model. The perturbed graph is denoted as $\hat{\mathcal{G}}$ and the optimization objective is expressed as:

$$\min \mathcal{L}_{atk} \left( f_{\theta} \left( \hat{\mathcal{G}} \right) \right) = \sum_{i \in V_t} l_{atk} \left( f_{\theta} \left( \hat{\mathcal{G}} \right), y_i \right)$$  \hspace{1cm} (1)

$$\text{s.t.} \quad \theta^* = \arg \min_{\theta} \mathcal{L}_{train} \left( f_{\theta} \left( \mathcal{G} \right) \right)$$  \hspace{1cm} (2)

where $\mathcal{L}_{train}$ is the loss function of model training and $\theta^*$ is the parameter of the trained model. $l_{atk}$ is the loss function for training adversarial perturbation which is usually defined as $l_{atk} = -\mathcal{L}_{train}$. In target-dependent graph adversarial attack, the set $V_t$ contains one targeted node. In global graph adversarial attack, the set $V_t$ contains all nodes in the graph. The image domain
would limit the perturbation to a certain range, making the perturbation imperceptible. And the \( l_1 \) norm, \( l_2 \) norm and \( l_\infty \) norm are usually used. However, the connections between nodes and attributes of nodes are discrete. Thus, the study usually uses \( l_0 \) norm to limit the perturbation in the graph. This is expressed as:

\[
\| \hat{A} - A \|_0 + \| \hat{X} - X \|_0 \leq \Lambda
\]  

(3)

where \( \Lambda \) is the upper bound of the perturbation. Chen et al. (2018) propose a target-dependent graph adversarial attack, which achieves the attack by iteratively flipping the edge with the maximum absolute gradient. Considering the adjacency matrix in the undirected graph is symmetric, Chen et al. symmetrize the gradients to enhance stability of attack. Dai et al. (2018) point out that limiting the perturbation budget cannot guarantee the concealment of attack. Thus, Dai et al. propose that the characteristics of the graph should be kept in attack, such as degree distribution and feature co-occurrence. In the eligible set of perturbations, Nettack greedily selects the perturbation with the maximum score to achieve attack. The score function is defined as follows:

\[
\max_{c',w} \ln \left( f(\hat{\mathcal{G}})_{i,c'} \right) - \ln \left( f(\mathcal{G})_{i,c} \right)
\]

(4)

where \( f(\hat{\mathcal{G}})_{i,c} \) represents the probability that node \( i \) is classified into class \( c \). Furthermore, to estimate the effect of the perturbation more exactly, Wu et al. (2018) propose the score function based on the integrated gradient. The global adversarial attacks on the graph can degrade the overall performance of the targeted model by modifying the topology in the whole graph. Zugner and Gunnemann (2019) propose a global adversarial attack based on meta gradient, which regards the graph adjacency matrix as trainable hyperparameters. Meta-attack greedily selects the edge with the largest meta-gradient as perturbation until it reaches the perturbation budget. Specifically, the meta gradient is denoted as follows:

\[
\nabla_{\theta^*} = \nabla_{f_{\theta^*}} \left( f_{\theta^*} (\mathcal{G}) \right) - \nabla \left( f_{\theta^*} (\mathcal{G}) \right) \nabla_{\theta^*} \mathcal{L}_{\text{train}}(f_{\theta^*} (\mathcal{G}))
\]

(5)

where \( \theta^* \) varies with model training. The update rule of \( \theta^* \) is expressed as follows:

\[
\nabla_{\theta_{t+1}} = \nabla_{\theta_t} - \alpha \nabla_{\theta_t} \mathcal{L}_{\text{train}}(f_{\theta_t} (\mathcal{G}))
\]

(6)

Xu et al. (2019) propose a global attack method based on optimization. First, Xu et al. transform the attack problem into an optimization problem of continuous variable. The Projected Gradient Descent (PGD) algorithm is used in the optimization. Then, Xu et al. use randomization sampling (Liu et al., 2016) to generate the near-optimal binary perturbation.

Up to now, graph universal attack has not been widely studied. Thus, we introduce research related to universal attack methods in the image domain, which are the source of inspiration for the graph universal attack. Moosavi-Dezfooli et al. propose the first UAP in the image domain. Essentially, UAP is an iterative DeepFool algorithm. First, UAP initializes a global zero perturbation. In each iteration, an image sample is fed to conduct the DeepFool algorithm. The generated perturbation is added into the global perturbation. To conform to the perturbation budget, a projection of \( l_2 \) norm
is conducted after each iteration. Hayes and Danezis (2018) propose a generative universal adversarial attack in which a generative model similarly to GAN is used to craft the perturbation. First, a random vector $z$ is sampled from a Gaussian distribution $\mathcal{N}(0,1)^{100}$, which is used as the input of the generative model. The generated perturbation is denoted as $\delta$ and the scaled perturbation is denoted as $\delta'$. The loss of generator is expressed as follows:

$$
\max \left\{ \log \max_{c' \neq c_0} \log \frac{f(\delta' + x)}{f(x)} - k \right\} + \alpha \|\delta\|_p
$$

(7)

where $c_0$ is the original class. In addition, if the target class of attack is specified, the loss can be expressed as follows:

$$
\max_{c \neq c_0} \log \frac{f(\delta' + x)}{f(x)} - k\right\} + \alpha \|\delta\|_p
$$

(8)

where $c$ is the specified class of attack. Mopuri et al. (2019) first focus on the generation of universal adversarial attack when the data samples are not available. It is proposed that misfiring the features at individual layers can degrade the performance of model. The existing graph universal attack method GUA originates from UAP, which generates the continuous attack vector similar to UAP. Then it discretizes the continuous attack vector by setting a fixed threshold. In this paper, we give a deep analysis about the universal attack on graph and propose a forward-derivative-based graph universal adversarial attack.

Adversarial training is an effective method to resist the adversarial attack, which has been widely used in image domain. The kernel idea of adversarial training is to mix adversarial samples to model training, so the model can classify the adversarial samples correctly in the future. Dai et al. (2018) propose randomly discarding some edges during the training process can slightly improve the robustness of the model. Xu et al. (2019) further improve the robustness against global attack by adding the adversarial perturbation into the training process. Feng et al. (2021) point out that GNNs are more susceptible to the adversarial perturbations than the traditional models, because the smoothness constraint in training will exacerbate the influence of perturbations. And Feng et al. believe it is helpful to take the relations between nodes into account during adversarial training rather than treat nodes as independent individuals. Therefore, Feng et al. propose to maximize the prediction divergence between the target node and its neighbors to generate adversarial perturbations. Besides, an additional constraint is added into the training progress to degrade the prediction divergence between the target node and its neighbors. The adversarial training can also be used to improve the smoothness of the output distribution of graph models. Virtual adversarial training (Miyato et al., 2019) is a classical method to improve the smoothness of output distribution of traditional models. However, it is inefficient to directly apply VAT to GNNs without considering the relations of nodes in the graph. Deng et al. (2019) propose a sample-based batch virtual adversarial training (BVAT) and an optimization-based BVAT to preserve the connection mode between nodes. This can effectively promote the smoothness of GNN classifiers. Chen et al. (2019) believe that degrading the divergence of confidence of model output can improve the robustness of model. Thus, Chen et al. introduce smoothing distillation and smoothing cross-entropy loss function into the training to improve the smoothness of model output. However, the graph universal adversarial training has not been well researched. In this paper, we propose the first graph universal adversarial training on the basis of FDGUA, which is a new attempt of graph adversarial defense.
3. PRELIMINARIES

Given a graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. The number of nodes is $|\mathcal{V}| = N$. The adjacency matrix of the graph is denoted as $A \in \{0,1\}^{N \times N}$, which means if $(i,j) \in \mathcal{E}$, $A_{ij} = 1$, otherwise $A_{ij} = 0$. The features of nodes are denoted as $X \in \{0,1\}^{N \times F}$, where $F$ is the dimension of feature. This paper focuses on the semi-supervised node classification task in which GNNs infer the classes of unlabeled nodes based on $A$, $F$ and part of labels. GCN is used as the benchmark model. The convolution operation in GCN can be regarded as the aggregation of node information, which is expressed as follows:

$$H^{(l+1)} = \hat{A}H^{(l)}W^{(l)}$$

(9)

$\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacency matrix, where $\tilde{A} = A + I^N$ and $\tilde{D}_{ij} = \sum_j \tilde{A}_{ij}$. $H^{(l)}$ is the feature input of layer $l$ and $H^{(0)}$ is the initial feature $X$. Two layers of graph convolution are used in GCN. The output of GCN can be expressed as follows:

$$f(X, A) = \text{softmax} \left( \tilde{A} \sigma \left( \tilde{A} X W^{(0)} \right) W^{(l)} \right)$$

(10)

where $\sigma$ is the nonlinear activation function and $W^{(0)}$, $W^{(l)}$ are weights of full connection layers.

Graph universal attack aims to maximize the attack success rate by target-independent attack. This paper refers to the definition of target-independent attack in GUA, which attacks nodes by flipping the edges connected to anchor nodes. The target-independent attack vector is denoted as $p^{1 \times N}$, where $p_s = 1$ if $s$ is the anchor node, otherwise $p_s = 0$. Before attacking node $i$, we first initialize attack matrix $P = 0^{N \times N}$ and embed attack vector as $P_{si} = p$ and $P_{i,i} = p^T$. Finally, the adjacency matrix can be updated as:

$$A' = A + P \left( I^{N \times N} - I^N - 2A \right)$$

(11)

where $I^N$ is the identity matrix and $\left( I^{N \times N} - I^N - 2A \right)$ is the auxiliary matrix for calculation. The optimization objective of graph universal attack can be denoted as:

$$\arg\max_p \sum_{i \in \mathcal{V}} H(i)$$

(12)

$$H(i) = \begin{cases} 0 & \text{if } \text{argmax}_p \left( X \right)_i = \text{argmax}_p \left( 0^{1 \times N} \right)_i \\ 1 & \text{if } \text{argmax}_p \left( X \right)_i = \text{argmax}_p \left( 0^{1 \times N} \right)_i \end{cases}$$

(13)

where $f(X, A; p)$ is simplified as $f(p)$ and $f(p)$ is equal to original output of GCN if $p = 0^{1 \times N}$.
4. FORWARD-DERIVATIVE-BASED GRAPH UNIVERSAL ADVERSARIAL ATTACK

In this section, we introduce the forward-derivative-based graph universal adversarial attack. First, we present the generation process of continuous attack vector. And then we introduce how to discretize the continuous attack vector based on the forward derivative.

4.1 Generation of Target-Independent Continuous Attack Vector

We find that the attack vector generated by DeepFool on the graph are universally effective and strongly directional. Specifically, the attack vector generated on one node can always attack other nodes and the attacked nodes will be classified into the same class. First, we analyze this phenomenon by comparing the magnitude of the node vectors and the attack vector. Graph adjacency matrix is usually sparse, therefore the average degree of nodes is low, such as 3.9 for Cora and 2.8 for Citeseer. This makes the difference between node vectors insignificant compared with the perturbation caused by the attack vector.

Second, we further explain this by analyzing the cosine similarities between attack vectors. We define the direction determined by minimizing perturbation (Moosavi-Dezfooli et al., 2016) as the attack direction. For each attack direction, nodes of various classes are used to generate multiple attack vectors with DeepFool. Then, we calculate the cosine similarities between attack vectors with same direction or different directions, as shown in Table 1. It is noted that the similarities between attack vectors with the same attack direction are high, while the similarities between attack vectors with different attack directions are relatively lower. That means any two attack vectors generated on different nodes \(i, j\) with the same attack direction are highly similar. It is quite possible that the attack vector derived from node \(i\) can effectively attack node \(j\). Further, it infers that all attack vectors with the same attack direction will have similar influence on different nodes with high probability.

Based on these observations, we propose that it is reasonable to generate attack vectors according to attack direction. For each direction, one node as training data is sufficient to generate the continuous attack vector, as shown in lines 1~7 of Algorithm 1. Algorithm 1 removes the selection of attack direction in DeepFool, where \(\Delta f(p)_{i,c} = f(p)_{i,c} - f(p)_{i,r}\), \(\Delta w_c = \nabla f(p)_{i,c} - \nabla f(p)_{i,r}\) and \(r = \text{argmax}_i f(p)_i\). Compared with GUA, it is no longer required to traverse all the training nodes, which greatly simplifies the generation of continuous attack vector.

4.2 Discretization of Continuous Attack Vector

The continuous attack vector needs to be further transformed into a discrete attack vector to achieve effective attack on the graph. GUA uses the heuristic method to discretize the attack vector, where entries with value greater than 0.5 are set to 1 and the others are set to 0. However, the magnitude of

| Class | 0   | 1   | 2   | 3   | 4   | 5   | 6   |
|-------|-----|-----|-----|-----|-----|-----|-----|
| 0     | 0.808 | 0.487 | 0.191 | 0.587 | 0.719 | 0.368 | 0.641 |
| 1     | 0.783 | 0.462 | 0.514 | 0.527 | 0.449 | 0.563 |
| 2     | 0.906 | 0.365 | 0.408 | 0.408 | 0.209 | 0.313 |
| 3     | 0.907 | 0.821 | 0.348 | 0.537 |
| 4     | 0.928 | 0.399 | 0.612 |
| 5     |     | 0.870 | 0.511 |
| 6     |     |     | 0.795 |
perturbation is not directly related to the effectiveness of the attack. Here, we propose to discretize the continuous attack vector based on the forward derivative. In (Papernot et al., 2016), the forward derivative is used to build the adversarial saliency maps, which reflects the influence of input features on outputs. Inspired by this, we measure the influence of each entry of $p$ on classification probabilities based on forward derivative. Assuming the attack direction of $p$ is $c$, the node set contained in the training set which can be successfully attacked by $p$ is denoted as $\mathcal{F}_{c\text{fool}}$. Because the discretized vector $p'$ is supposed to keep the attack performance of $p$, $\mathcal{F}_{c\text{fool}}$ is used to calculate the forward derivative instead of all training nodes. The derivative of classification probability $c$ of node $i$ to $p$ is expressed as $\nabla f(p)_{i,c}$, and the component corresponding to entry $s$ is $\nabla f(p)_{i,c,s}$. The optimization objective of discretization is denoted as follows:

$$\arg\max_{S} \sum_{s \in \mathcal{F}_{c\text{fool}}} \left( \nabla f(p)_{i,c,s} - \sum_{c'\neq c} \nabla f(p)_{i,c',s} \right)$$ \hspace{1cm} (14)

Assuming $M$ is the number of anchor nodes, $S = \{s_1, \ldots, s_M\}$ is the set of entries which will be discretized to 1. The other entries of $p$ will be quantized to 0. $\nabla f(p)_{i,c,s}$ represents the positive effect of entry $s$ to classification probability $c$ of node $i$ and $\sum_{c'\neq c} \nabla f(p)_{i,c',s}$ is regarded as the sum of negative effects. Algorithm 2 shows the detail process of discretization, where $g^{\text{Syn}}$ is the synthetical effect. We prioritize the entries in $p$ based on $g^{\text{Syn}}$ to obtain $S$ and then discretize $p$ based on $S$.

### 4.3 Class-Discriminative Graph Universal Attack

In some scenarios with high requirement of security, attacks without distinguishing target classes will cause suspicion of the monitoring system. However, class-discriminative graph universal attack has not been well studied. The existing graph universal attack cannot distinguish the classes of victims, which means nodes of all classes will be affected by adversarial samples. To further improve the covertness of graph universal attack, we simply extend FDGUA to class-discriminative FDGUA.
by combining the anchors of different attack directions. Obviously, for a single attack direction, the aggressiveness of the adversarial sample is positively correlated with the number of anchors. The classes of anchors are generally same, which are consistent with the attack direction. On the one hand, the adversarial samples with anchors will degrade the classification confidence of nodes of targeted classes, resulting in misclassification. On the other hand, the adversarial samples will improve the classification confidence of nodes of untargeted classes. In pure graph universal attack, only the nodes of the same class as the attack direction will avoid attacks. All the other nodes will be affected by adversarial attack. Therefore, we propose to combine the anchors of different attack directions to achieve the class-discriminative graph universal attack. For example, if the aim is to attack all classes except class “0” and class “1”, it will combine the anchors of class “0” and class “1” to generate adversarial samples. In this example, class “0” and class “1” are untargeted classes. And other classes are targeted classes. As mentioned, the anchors of class “0” are helpful for maintaining the classification results of nodes of class “0” and the anchors of class “1” will degrade the classification confidence of nodes of class “0”. A similar condition happens to class “1”. By adjusting the proportion of anchors of different attack directions, the positive and negative effects on untargeted classes can be offset, which means nodes of untargeted classes will maintain the original classification results. Meanwhile, adversarial samples are still aggressive for nodes of targeted classes. In this paper, we discuss the condition containing two untargeted classes and more complex conditions will be researched in the future.

5. GRAPH UNIVERSAL ADVERSARIAL TRAINING

This section introduces an effective adversarial training method against graph universal adversarial attack. Assuming that the direction of the universal attack is known, we hope to improve the robustness without degrading the accuracy through adversarial training. The core idea of adversarial training is adding adversarial samples into the training process, so the model can recognize adversarial samples in the future.

Considering the particularity of graph universal adversarial attack, we carefully design the embedding form of adversarial samples and the process of adversarial training. Here, we denote nodes which attacked by anchors as adversarial samples. First of all, because the combination of anchors will change with model retraining, we need to design flexible adversarial samples to cover possible combinations of anchors in the future. We design the mechanism of anchor pool to generate adversarial samples.

Algorithm 2 Discretization of Continuous Attack Vector

| Input: | A, X, class c, continuous attack vector p |
|---|---|
| Output: | discrete attack vector p |
| 1: | calculate attacked node set $\mathcal{V}_c^{\text{fool}}$, initialize $g^\text{pos}_c, g^\text{neg}_c = 0^{3-N}$ |
| 2: | for $i \in \mathcal{V}_c^{\text{fool}}$ do |
| 3: | $g^\text{pos}_i = g^\text{pos}_i + \nabla f(p)_i^c$ |
| 4: | $g^\text{neg}_i = g^\text{neg}_i + \sum_{\varepsilon \neq c} \nabla f(p)_{i\varepsilon}$ |
| 5: | end for |
| 6: | $g^\text{syn} = g^\text{pos} - g^\text{neg}$ |
| 7: | obtain $S = \{s_0, \ldots, s_M\}$ according to descending order of $g^\text{syn}$ |
| 8: | discretize $p$ based on $S$ and return $p$ |
samples. Specifically, we first select $\varphi$ entries with highest scores calculated by Algorithm 2 to form the anchor pool. Each generated adversarial sample is a random sample of $M$ anchors from the anchor pool to achieve attack.

Next, due to the particularity of graph universal adversarial attack, the following points should be considered when designing how to embed adversarial samples. First, graph universal attack is target-independent. Therefore, it should cover as many nodes as possible instead of one fixed node. Second, to ensure the effectiveness of adversarial training, we need to keep the topology roughly unchanged in adversarial training. This means the number of adversarial samples should not be too large. In the actual process of the graph universal attack, only $M$ edges are flipped and the overall topology remains unchanged. Therefore, we should keep the consistency of the original topology and the modified topology in the adversarial training. We randomly select a certain number of nodes in the training set to constitute adversarial samples in the adversarial training.

Overall, graph universal adversarial training (GUAT) is regarded as an iterative process of attacking and retraining, which is shown in Algorithm 3. GUAT can be divided into three steps as follows. First, we select $\varphi$ entries as the candidate pool of anchor nodes with Algorithm 2. The setting of $\varphi$ should be appropriate. If $\varphi$ is too small, the model will be overfitting and face difficulties in coping with new anchor nodes as they change with the model. In addition, challenges with training will sharply increase if $\varphi$ is too large. Then, we embed $\rho$ adversarial samples into graph before retraining, which is shown in lines 6-11. Appropriately increasing $\rho$ can accelerate the training. However, the embedding graph may severely deviate from the original graph if $\rho$ is too large, which will make the retraining meaningless. Finally, we retrain the model, where the setting of iterations $\varepsilon$ also affects the performance of adversarial training. In experiment 6.3, we will discuss the influence of the above hyperparameters in detail.

Algorithm 3 GUAT

```
Input: $A$, $X$
1: initialize GCN with $X_{train}$, $A$
2: while epoch < max_epoch do
3:     update $\{s_0, \ldots, s_\varphi\}$ with FDGUA
4:     while iter < max_iter do
5:         initialize $P = 0^{l \times N}$ and sample $\{v\}_\rho$ from $V_{train}$
6:         For $i \in \{v\}_\rho$ do
7:             $p = 0^{l \times N}$
8:             sample $\{s\}_{M}$ anchors from $\{s_0, \ldots, s_\varphi\}$
9:             $p_i = 1$
10:            $P_{i,:} = p, P_{:,i} = p^T$
11:        end for
12:     update $A'$ with Equation 3
13:    retrain GCN with $V_{train}, A'$
14: end while
15: end while
```
6. RESULTS

In experiments, the benchmark model is GCN and we evaluate our methods on three well-known datasets: Cora, Citeseer and Pol.Blogs, as shown in Table 2.

6.1 Comparison With Baselines

We set max_epoch=10, max_iters=20 and ε =0.001 in FDGUA. Since there are few methods of graph universal adversarial attack, we expand IG-FGSM and FGA into universal adversarial attacks for comparison. The baselines are summarized as follows:

- **GUA**: GUA generates the continuous attack vector based on UAP and discretizes the attack vector with fixed threshold. To ensure the fairness of running time between methods, we set max_epoch=20 in GUA.
- **FGA**: FGA is a target-dependent attack, which iteratively flips the edge with largest gradient until successfully attacks target. We expand FGA to FGA* as follow steps. First, we select the node with highest classification margin (Wu et al., 2019) in testing set as the victim, which is probably the hardest node to attack. Then, we perform FGA on this victim and specify that only edges connected to the victim can be flipped. Finally, we denote the nodes corresponding to flipped edges as the anchor nodes. And the priorities of anchor nodes follow the order of flipping.
- **IG-FGSM**: IG-FGSM uses cumulative gradient to measure the change of classification confidence caused by perturbation, which performs better than FGA in target-dependent scenario. We expend IG-FGSM in the same way as FGA.

6.1.1 Analysis of Attack Success Rate (ASR)

Figure 1~3 show the performances of attack success rate (ASR) of the discussed methods. Since GUA cannot directly control the number of anchor nodes, we conduct GUA on different projection

| Dataset  | Nodes | Edges | Classes | Accuracy |
|----------|-------|-------|---------|----------|
| Cora     | 2708  | 5278  | 7       | 81.0%    |
| Citeseer | 3327  | 4676  | 6       | 71.1%    |
| Pol.Blogs| 1222  | 16714 | 2       | 94.8%    |

Figure 1. Comparison of ASR on Cora
radius $\xi$ to obtain the average ASR of different number of anchors. For FDGUA, we show the best ASR and average ASR of adversarial attack in different directions, which correspond to two attack scenarios. If the class distribution of target nodes is known before the attack, we will choose the class with the least proportion of targets as the direction of attack. In experiments, we first test the ASR of anchors with different directions in the validation set and then use the anchors with the highest ASR to attack test set, which is shown in FDGUA-Best. In addition, we annotate the best attack direction of FDGUA-Best. The division of datasets in GUA and FDGUA is the same as the training of GCN. However, if the class distribution is unknown, we can only use anchors of random direction to attack the targets. And the expectation of ASR is the average of all cases, which is shown in FDGUA-Avg. Overall, it is shown that the performance of FDGUA is generally better than baselines. When the class distribution is known, only three anchor nodes generated by FDGUA-Best are sufficient to achieve an ASR higher than 80% in Cora and Citeseer, as shown in Figure 1~2. For Cora, the performance of FDGUA-Best is significantly better than other methods. The ASR can even exceed 90% when number of anchors is 10. And the performance of IG-FGSM* is close to FDGUA-Avg and better than GUA and FGA*. We can find that both GUA and FGA* are unstable. In GUA, the discretization of the continuous attack vector is based on a fixed threshold value of 0.5, which causes the anchors may have inconsistent attack influences on victims. Similarly, the consistency of anchors of in FGA* cannot be guaranteed. For Citeseer, FDGUA performs better than other methods in both conditions. In addition, both IG-FGSM* and FGA* can successfully attack the selected target when 7 edges are flipped and no more anchors are chosen. Thus, we fill black blocks in corresponding locations, as shown in Figure 2. We are surprised to find that FGA* outperforms IG-FGSM* when number of anchors is more than 4 for Citeseer. Regarding Pol.Blogs, FGA* even outperforms IG-
FGSM* in all conditions. We suspect that the variation of graph denseness causes the fluctuation of these heuristic methods like FGA* and IG-FGSM*, where the average degree of Pol.Blogs is much higher than Cora and Citeseer. Meanwhile, there are only two classes in Pol.Blogs. The class of selected victim is “1”. IG-FGSM* and FGA* can be regarded as targeted attacks with direction “0” in this case. We find that FDGUA with direction “0” still performs better than IG-FGSM* and FGA*.

6.1.2 Analysis of Classification Margin

To further measure the effectiveness of above attack algorithms, we use violin diagram to visualize the classification margin of victims in the test set. In the violin diagram, the width of shadow of each bar represents the probability of distribution and the solid line in the middle indicates the location of median. In addition, the solid lines at the top and bottom indicate the upper and lower bounds of the classification margin. In the case of Figure 4, it is visualized that FDGUA-Best performs better than other methods because its probability distribution is more concentrated at the bottom. Comparing Figure 4–6, we find that the probability distribution will be more concentrated at the bottom with the increasing of anchors. In Figure 4, the bar shapes of IG-FGSM* and FDGUA-Best are similar to hourglasses. Some hard samples shown in the top of bar limit the ASR of methods. When the number of anchors increases to 10, most of the hard samples are successfully attacked in FDGUA-Best while the problem of hard samples in IG-FGSM* is not well solved. For Citeseer, we display the results of

Figure 4. Classification margin on Cora (number of anchors: 3)

Figure 5. Classification margin on Cora (number of anchors: 6)
the conditions that the number of anchors is 3, 4 and 6. It is observed that GUA performs the worst in all methods where the targeted nodes are not sensitive to anchors generated by GUA. In contract, IG-FGSM* and FGA* can achieve relatively ideal ASR when the number of anchors increases to 6. However, we also find that the classification margin of part of nodes which are successfully attacked by IG-FGSM* and FGA* is close to 0, which means the attacks are not reliable and stable. In the same condition, the performance of FDGUA-Best shown in Figure 9 is better and more reliable. Figure 10–12 show the classification margin results of Pol.Blogs. When the number of anchors is 5 or 15, IG-FGSM* and FGA* perform better than GUA on ASR. However, it is worth mentioned that the median of IG-FGSM* and FGA* are much higher than GUA, which means IG-FGSM* and FGA* are less aggressive on part of nodes compared with GUA. In contrast, the median of FDGUA-Best drops markedly with the increasing of number of anchors, which also reflects the effectiveness of FDGUA.

6.2 Class-Discriminative Graph Universal Attack

FDGUA cannot distinguish between the classes of attacked nodes. This means that all nodes will be attacked except those with the same class as the attack direction. However, in some security-sensitive scenarios, it should be considered that a class-discriminative attack can avoid attack behavior detection. Here, we combine anchors with different attack directions to achieve class-discriminative attack in which only the combination of two directions is considered in this paper. Table 3 shows the results of

Figure 6. Classification margin on Cora (number of anchors: 10)

Figure 7. Classification margin on Citeseer (number of anchors: 3)
Figure 8. Classification margin on Citeseer (number of anchors: 4)

![Classification margin on Citeseer (number of anchors: 4)](image)

Figure 9. Classification of margin on Citeseer (number of anchors: 6)

![Classification of margin on Citeseer (number of anchors: 6)](image)

Figure 10. Classification margin on Pol. Blogs (number of anchors: 5)

![Classification margin on Pol. Blogs (number of anchors: 5)](image)
Table 3. Accuracies of non-targeted classes and ASR of targeted class. The numbers between brackets indicate the proportion of combinations.

| Class | 0      | 1      | 2      | 3      | 4      | 5      | 6      |
|-------|--------|--------|--------|--------|--------|--------|--------|
| 0     | 100%/98.6% | 100%/86.1%/93.3% | 82.7%/100%/98.5% | 67.3%/100%/93.3% (4-2) | 99.3%/96.4%/93.7% | 92.1%/100%/86.0% (2-4) | 100%/86.7%/92.2% |
| 1     | 100%/95.4% | 94.5%/100%/96.6% (4-2) | 45.5%/100%/92.8% | 91.1%/100%/90.0% | 100%/94.8%/91.3% | 95.1%/100%/79.5% |
| 2     | 100%/98.8% | 86.9%/100%/96.2% (2-4) | 100%/89.2%/97.7% | 100%/85.4%/86.8% (2-4) | 98.6%/94.7%/92.5% (2-4) |
| 3     | 100%/98.5% | 100%/87.5%/95.4% | 100%/89.6%/87.1% (2-4) | 94.9%/100%/88.1% (2-4) |
| 4     | 100%/99.3% | 92.8%/100%/97.5% (2-4) | 100%/89.8%/87.3% |
| 5     | 100%/93.4% | 83.3%/100%/82.6% |
| 6     | 100%/96.1% |
class-discriminative attack, where the last entry in each cell is the average ASR of targeted class and prior entries are the accuracies of non-targeted classes. In this case, the dataset is Cora and the sum of anchors is 6. First, we find that FDGUA can achieve ASR higher than 90% of all attack directions, as shown in the diagonal. The default combination mode on the non-diagonal is 3-3, which means three anchors with highest priorities of each attack direction are selected. In most cases, the simple combination of 3-3 is sufficient to result in a high absolute accuracy drop gap (Shafahi et al., 2018). In addition, we find the anchors of direction 2 are strongly aggressive and the anchors of direction 5 are weakly aggressive. When the anchors of direction 2 are combined with anchors of another direction in the 3-3 mode, the non-targeted nodes with the class same as another direction will be classified to class 2 with a high probability. So, we adjust the proportion of combination that increase the number of anchors in another direction. For example, when combined with anchors in direction 1,3,5,6, we adjust the combination of 3-3 into 2-4. In contrast, when anchors of direction 5 are combined with other anchors, we increase the number of anchors of direction 5 to improve absolute accuracy drop gap.

6.3 Performance of GUAT

The evaluation of GUAT is conducted on Cora. The attack direction is 6 and the number of anchors is 6. As shown in Figure 1, the original ASR is 88.6%. In the test, we set $\text{max}_\text{epoch}=10$, $\text{max}_\text{iter}=10$. The results are shown in Table 4 and Table 5, where we set $\rho = 20$ in Table 4 and $\varepsilon = 20$ in Table 5. In all cases in Table 4 and Table 5, the accuracy of retrained GCN is within the range of 80.0%~81.3%. In total, we find that moderate $\varphi$ helps improve the robustness of model. When $\varphi = 50$ or 130, the performance is generally worse than other cases. In Table 4, we observe that setting $\varepsilon$ to 20~40 can result in best performance. And the ASR can be degraded to 69.5% when $\varepsilon = 40$ and $\varphi = 70$. In the case that $\varepsilon = 5$, the average ASR is 79.2% which is worse than the other

Table 4. ASRs that change with $\varepsilon$ and $\varphi$

| $\rho$ = 20 | $\varphi$ |
|---|---|
| 5 | 79.0% 77.0% 78.6% 77.1% 81.4% 82.1% |
| 10 | 77.8% 77.9% 76.1% 76.6% 77.1% 73.6% |
| 20 | 74.0% 73.2% 74.6% 74.8% 72.7% 75.3% |
| 40 | 73.6% 69.5% 70.6% 74.6% 73.9% 75.7% |
| 60 | 73.4% 75.5% 74.9% 75.5% 73.8% 74.2% |

Table 5. ASRs that change with $\rho$ and $\varphi$

| $\varepsilon$ = 20 | $\varphi$ |
|---|---|
| $\rho$ | 50 | 70 | 90 | 110 | 130 | 150 |
| 5 | 79.1% 79.1% 80.2% 80.9% 80.9% 81.9% |
| 10 | 74.5% 76.2% 78.3% 74.4% 77.5% 80.2% |
| 30 | 78.6% 73.5% 74.1% 76.7% 72.6% 74.0% |
| 50 | 82.9% 79.4% 75.5% 72.1% 75.1% 75.1% |
| 70 | 86.2% 82.4% 80.5% 79.6% 78.8% 77.0% |
settings. The ASR will be improved to 82.1% if $\varphi = 150$. Some exceptions are found in Table 4, such as ASR is 73.6% when $\varepsilon = 10$ and $\varphi = 150$. It may be caused by the randomness of the anchor set and embedding position. Results in Table 5 suggest that setting $\rho$ to 30~50 is optimal to improve robustness of model. First, too small $\rho$ leads to insufficient exploration of anchor pool. Meanwhile, too large $\rho$ will make the embedding graph deviate from the original graph, which degrades the effectiveness of adversarial training.

6.4 Visualization Analysis

To better understand the process of graph universal attack and defense, we visualize the output of the middle layer of the GCN model based on T-SNE (Maaten & Hinton, 2008). The settings are the same as 6.3, where the attack direction is 6. Figure 13 shows the attack on the pure model and Figure 14 shows the attack on the robust model. The size of the node indicates whether the node is under attack, where small nodes are non-attacked and large nodes are attacked. In addition, the shape represents the true class and the color represents the classification result of the model. As shown in

Figure 13. Attack on pure GCN model

![Figure 13. Attack on pure GCN model](image1)

Figure 14. Attack on robust GCN model

![Figure 14. Attack on robust GCN model](image2)
Figure 13, the representations of attacked nodes are far from original representations and close to the representations of class 6. However, adversarial training can effectively restrain this trend. We find that the representations of attacked nodes in Figure 14 are relatively closer to the original representation. In addition, the nodes have higher probabilities to be immune from the attack.

7. CONCLUSION

In this paper, we propose FDGUA, a new graph universal attack method. It is verified that FDGUA performs better than baselines. In addition, we show that FDGUA can transfer into a CD-FDGUA by simply combining anchors with different directions. To defend FDGUA, we propose the first GUAT, which can improve the robustness of model without degrading the accuracy. In the future, we consider to research direction-independent defense against graph universal attack.
REFERENCES

Chen, J., Wu, Y., Lin, X., Xuan, Q. (2019). Can adversarial network attack be defended?

Chen, J., Wu, Y., Xu, C., Chen, Y., Zheng, H., Xuan, Q. (2018). Fast gradient attack on network embedding.

Chen, Z., Bruna, J., & Li, L. (2019). Supervised community detection with line graph neural networks. 7th International Conference on Learning Representations.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L. (2018). Adversarial attack on graph structured data.

Deng, Z., Dong, Y., & Zhu, J. (2019). Batch virtual adversarial training for graph convolutional networks. 36th International Conference on Machine Learning.

Feng, F., He, X., Tang, J., & Chua, T. S. (2021). Graph adversarial training: Dynamically regularizing based on graph structure. IEEE Transactions on Knowledge and Data Engineering, 33(6), 2493–2504. doi:10.1109/TKDE.2019.2957786

Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. 3rd International Conference on Learning Representations.

Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., & Alahi, A. (2018). Social gan: Socially acceptable trajectories with generative adversarial networks. In 31st IEEE Conference on Computer Vision and Pattern Recognition (pp. 2255-2264). doi:10.1109/CVPR.2018.00240

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In 30th Advances in Neural Information Processing Systems. Academic Press.

Hayes, J., & Danezis, G. (2018). Learning universal adversarial perturbations with generative models. 2018 IEEE Security and Privacy Workshops.

Ismail, S., Shishtawy, T. E., & Alsammak, A. K. (2022). A new alignment word-space approach for measuring semantic similarity for Arabic text. International Journal on Semantic Web and Information Systems, 18(1), 1–18. doi:10.4018/IJSWIS.297036

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. 5th International Conference on Learning Representations.

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. ACM Transactions on Knowledge Discovery from Data, 1(1). 10.1145/1217299.1217301

Liu, S., Chepuri, S. P., Fardad, M., Masazade, E., Leus, G., & Varshney, P. K. (2016). Sensor selection for estimation with correlated measurement noise. IEEE Transactions on Signal Processing, 64(13), 3509–3522. doi:10.1109/TSP.2016.2550005

Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9(86), 2579–2605.

Miyato, T., Maeda, S. I., Koyama, M., & Ishii, S. (2019). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(8), 1979–1993. doi:10.1109/TPAMI.2018.2858821 PMID:30040630

Moosavi-Dezfooli, S. M., Fawzi, A., Fawzi, O., & Frossard, P. (2017). Universal adversarial perturbations. In 30th IEEE Conference on Computer Vision and Pattern Recognition (pp. 1765-1773). IEEE.

Moosavi-Dezfooli, S. M., Fawzi, A., & Frossard, P. (2016). DeepFool: A simple and accurate method to fool deep neural networks. In 29th IEEE Conference on Computer Vision and Pattern Recognition (pp. 2574-2582). doi:10.1109/CVPR.2016.282

Mopuri, K. R., Ganeshan, A., & Babu, R. V. (2019). Generalizable data-free objective for crafting universal adversarial perturbations. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(10), 2452–2465. doi:10.1109/TPAMI.2018.2861800 PMID:30072314
Nhi, N. T., Le, T. M., & Thanh, T. V. (2022). A model of semantic-based image retrieval using C-Tree and Neighbor Graph. *International Journal on Semantic Web and Information Systems, 18*(1), 1–23. doi:10.4018/IJSWIS.295551

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy* (pp. 372-387). doi:10.1109/EuroSP.2016.36

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine, 29*(3), 93–106. doi:10.1609/aimag.v29i3.2157

Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L. S., Goldstein, T. (2018). Universal adversarial training.

Stylianou, N., Vlachava, D., Konstantinidis, I., Bassiliades, N., & Peristeras, V. (2022). Doc2KG: Transforming document repositories to knowledge graphs. *International Journal on Semantic Web and Information Systems, 18*(1), 1–20. doi:10.4018/IJSWIS.295552

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., & Fergus, R. (2014). Intriguing properties of neural networks. *2nd International Conference on Learning Representations.*

Urkalan, K., & Geetha, T. V. (2020). Concept map information content enhancement using joint word embedding and latent document structure. *International Journal on Semantic Web and Information Systems, 16*(4), 45–60. doi:10.4018/IJSWIS.2020100103

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2017). Graph attention networks.

Wu, F., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In *36th International Conference on Machine Learning* (pp. 6861-6871). Academic Press.

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., & Zhu, L. (2019). Adversarial examples for graph data: Deep insights into attack and defense. In *28th International Joint Conference on Artificial Intelligence* (pp. 4816-4823). doi:10.24963/ijcai.2019/669

Xie, J., Ying, R. (2021). Fea2Fea: Exploring structural feature correlations via graph neural networks. . 10.1007/978-3-030-93736-2_19

Xu, B., Shen, H., Cao, Q., Qiu, Y., & Cheng, X. (2019). Graph wavelet neural network. In *7th International Conference on Learning Representations* (pp. 6861-6871). Academic Press.

Zugner, D., Gunemann, S. (2019). Adversarial attacks on graph neural networks via meta learning.

Zang, X., Xie, Y., Chen, J., & Yuan, B. (2021). Graph universal adversarial attacks: A few bad actors ruin graph learning models. In *30th International Joint Conference on Artificial Intelligence* (pp. 3328-3334). doi:10.24963/ijcai.2021/458

Zhang, T., Xiong, Y., Zhang, J., Zhang, Y., Jiao, Y., & Zhu, Y. (2020). CommDGI: Community detection oriented deep graph infomax. In *29th ACM International Conference on Information and Knowledge Management* (pp. 1843-1852). doi:10.1145/3340531.3412042

Zhang, X., Liu, H., Li, Q., & Wu, X. M. (2019). Attributed graph clustering via adaptive graph convolution. In *28th International Joint Conference on Artificial Intelligence* (pp. 4327-4333). doi:10.24963/ijcai.2019/601

Zhang, Z., Bu, J., Ester, M., Zhang, J., Yao, C., Yu, Z., Wang, C. (2019). Hierarchical graph pooling with structure learning.