Graph Universal Adversarial Attacks: A Few Bad Actors Ruin Graph Learning Models

Xiao Zang [1] **Yi Xie** [1] **Jie Chen** [2] **Bo Yuan** [1]

Abstract

Deep neural networks, while generalize well, are known to be sensitive to small adversarial perturbations. This phenomenon poses severe security threat and calls for in-depth investigation of the robustness of deep learning models. With the emergence of neural networks for graph structured data, similar investigations are urged to understand their robustness. It has been found that adversarially perturbing the graph structure and/or node features may result in a significant degradation of the model performance. In this work, we show from a different angle that such fragility similarly occurs if the graph contains a few bad-actor nodes, which compromise a trained graph neural network through flipping the connections to any targeted victim. Worse, the bad actors found for one graph model severely compromise other models as well. We call the bad actors "anchor nodes" and propose an algorithm, named GUA, to identify them. Thorough empirical investigations suggest an interesting finding that the anchor nodes often belong to the same class; and they also corroborate the intuitive trade-off between the number of anchor nodes and the attack success rate. For the data set Cora which contains 2708 nodes, as few as six anchor nodes will result in an attack success rate higher than 80% for GCN and other three models.

1. Introduction

Graph structured data are ubiquitous with examples ranging from proteins, power grids, traffic networks, to social networks. Deep learning models for graphs, in particular, graph neural networks (GNN) (Scarselli et al., 2008; Bruna et al., 2014; Duvenaud et al., 2015; Defferrard et al., 2016; Li et al., 2016; Gilmer et al., 2017; Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2017) attracted much attention recently and have achieved remarkable success in several tasks, including community detection, link prediction, and node classification. Their success is witnessed by many practical applications, such as content recommendation (Wu et al., 2019b), protein interaction (Tsubaki et al., 2018), and blog analysis (Conover et al., 2011).

Deep learning models are known to be vulnerable and may suffer intentional attack with unnoticeable change of the data (Zügner et al., 2018). This observation originated from early findings by Szegedy et al. (2014) and Goodfellow et al. (2014), who show that images perturbed with adversarially designed noise can be misclassified, while the perturbation is almost imperceptible. This minor but intentional change would result in severe consequences socially and economically. For example, Wikipedia hoaxes lead to disinformation (Kumar et al., 2016). Different from real articles that link to each other coherently, hoax articles usually have few and random connections to real articles. These hoax articles can effectively disguise through modifying their links in a proper manner. For another example, some credit prediction models are based in part on social networks. Bad guys may hide themselves through building plausible friendship that confuses the prediction system.

In this work, we study the vulnerability of GNNs and show that it is indeed possible to attack them if a few graph nodes serve as the bad actors: when their links to a certain node are flipped, the node will likely be misclassified. Such attacks are akin to universal attacks because the bad actors are universal to any targeted node. We propose a graph universal adversarial attack method, GUA, to identify the bad actors.

Our work differs from recent studies on adversarial attack and defense of GNNs (Dai et al., 2018; Zügner et al., 2018; Jin et al., 2019; Xu et al., 2019) in the attack setting. Prior work focus on poisoning attacks (injecting or modifying training data as well as labels to foster a misbehaving model) and evasion attacks (modifying test data to encourage misclassification of a trained model). For graphs, these attacks could modify the graph structure and/or node features in a target-dependent scenario. The setting we consider, on the other hand, is a single and universal modification that applies to all targets. One clear advantage from the attack

[1]Department of Electrical and Computer Engineering, Rutgers University [2]MIT-IBM Watson AI Lab, IBM Research. Correspondence to: Jie Chen <chenjie@us.ibm.com>.

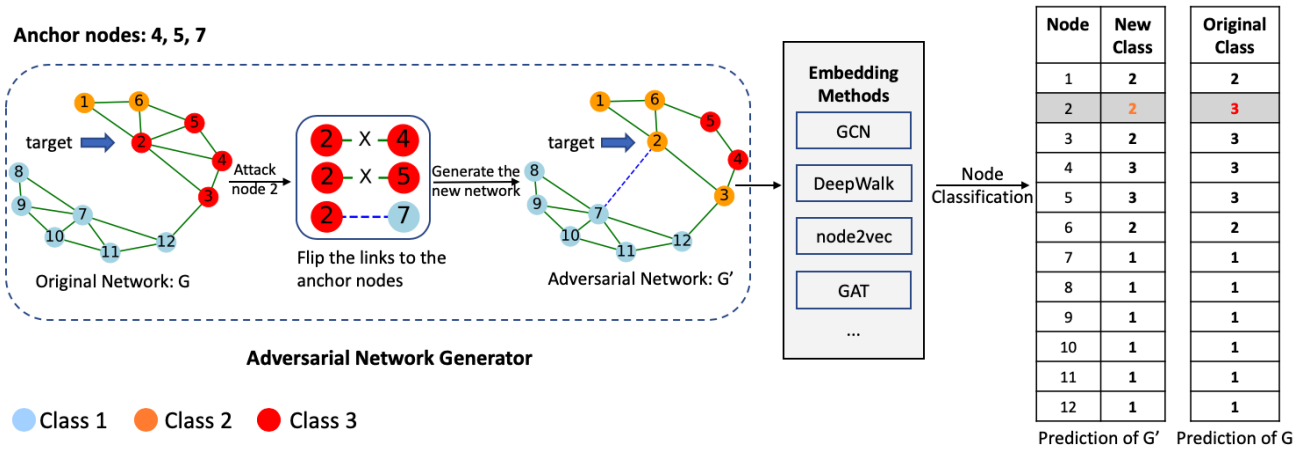


Figure 1. Illustration of GUA. A small number of anchor nodes (4, 5, and 7) is identified. To confuse the classification of a target node (e.g., 2), their connections to this node are flipped.

point of view is that computing the modification incurs a lower cost, as it is done once for all. More advantages will be elaborated later.

While universal attacks were studied earlier (see, e.g., Moosavi-Dezfooli et al. (2017) who compute a single perturbation applied to all images in the data set), graph universal attacks are rarely explored. This work contributes to the literature a setting and a method that may inspire further study on defense mechanisms of deep graph models.

Figure 1 illustrates the universal attack setting we consider. A few bad-actor nodes (4, 5, and 7) are identified; we call them *anchor nodes*. When an adversary attempts to attack the classification of a target node (say, 2), the existing links from the anchor nodes to the target node are removed while non-existing links are created. The identification method we propose, GUA, is conducted on a particular classification model (here, GCN), but the found anchors apply to other models as well (e.g., DeepWalk, node2vec, and GAT).

As a type of attacks, universal attacks may be preferred by the adversary for several reasons. First, the anchor nodes are computed only once and there incurs no extra cost when attacking individual targets. Second, the number of anchors can be very small (it is easier to compromise fewer nodes). Third, attacks are less noticeable when only a limited number of links are flipped.

The contribution of this work is fourfold:

1. We present a novel attack setting for graph structured data, which calls for vigilance when applying graph deep learning models, as well as defense mechanisms to counter such attacks.

2. We propose a novel algorithm for graph universal attack that achieves high success rate and demonstrates vulnerability of graph deep learning models.

3. We demonstrate appealing generalization of the attack algorithm, which finds anchor nodes based on a small training set but successfully attacks a majority of the targets in the graph.

4. We show attractive transferability of the found anchors (based on GCN) through demonstrating similar attack success rates on other graph deep learning models.

2. Notation and Background

A graph is denoted as $G = (V, E)$, where V is the node set and E is the edge set. An unweighted graph is represented by the adjacency matrix $A = \{0, 1\}^{|V|\times|V|}$; a weighted graph replaces the binary values of A by real-valued weights. For undirected graphs A is symmetric. In this work we consider unweighted and undirected graphs. The graph nodes may be accompanied by d-dimensional features, which collectively form the feature matrix X, whose dimension is $|V| \times d$.

More sophisticated feature representations (called embeddings) may be obtained in an unsupervised manner; see e.g., DeepWalk (Perozzi et al., 2014), node2vec (Grover & Leskovec, 2016), and LINE (Tang et al., 2015). Recently, graph neural networks (Scarselli et al., 2008; Bruna et al., 2014; Duvenaud et al., 2015; Defferrard et al., 2016; Li et al., 2016; Gilmer et al., 2017; Kipf & Welling, 2017; Hamilton et al., 2017; Veličković et al., 2017) emerge as a supervised approach for obtaining node embeddings and performing predictions simultaneously. In this work we use GCN (Kipf & Welling, 2017) as an attack example.

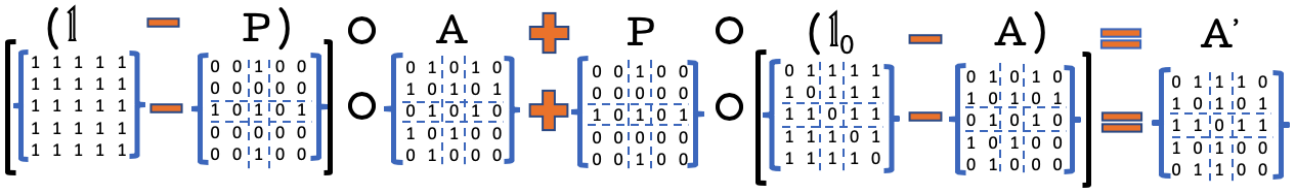


Figure 2. An example of attacking the third node in a five-node graph with the universal attack matrix P.

In GCN, one normalizes the adjacency matrix into $\widehat{A} = \widetilde{D}^{-\frac{1}{2}}\widetilde{A}\widetilde{D}^{-\frac{1}{2}}$, where $\widetilde{A} = A + I$ and $\widetilde{D}$ is the diagonal adjusted degree matrix with diagonal entries $\widetilde{D}_{ii} = \sum_{j=1}^{|V|} \widetilde{A}_{ij}$. Then, the neural network is

$$Z = f(A, X) = \text{softmax}(\hat{A}\sigma(\hat{A}XW^{(0)})W^{(1)}), \quad (1)$$

where σ denotes an activation function and $W^{(0)}$ and $W^{(1)}$ are model parameters. The training of the parameters uses the cross-entropy loss. Let V_L be the set of training nodes and Y be the one-hot label matrix. Then, with C classes, the loss function is

$$L = -\sum_{v \in V_L} \sum_{c=1}^{C} Y_{vc} \ln Z_{vc}. \quad (2)$$

3. Graph Universal Adversarial Attack

Following the notation introduced in the preceding section, given the graph adjacency matrix A and node feature matrix X, we let $f(A, X)$ be the classification model and let $\hat{l}(A, X, i)$ be the predicted label of node i; that is,

$$\hat{l}(A, X, i) = \arg\max f(A, X)_i. \quad (3)$$

Given a trained model f, the goal is for each node i to modify the adjacency matrix A into A' such that

$$\hat{l}(A', X, i) \neq \hat{l}(A, X, i). \quad (4)$$

Note that the modified A' is i-dependent in our attack setting.

3.1. Attack Vector and Matrix

Let the graph have n nodes. We use a length-n binary vector p to denote the attack vector to be determined, where 1 means an anchor node and 0 otherwise. Hence, A' is a function of three quantities: the original adjacency matrix A, the target node i, and the attack vector p.

To derive an explicit form of the function, we extend the vector p to an $n \times n$ matrix P, whose ith row and ith column are the same as p and zero elsewhere. Thus, the (i, j) element of the attack matrix P indicates whether the connection of the node pair (i, j) is flipped: 1 means yes and 0 means no.

It is then not hard to see that one may write the function as

$$A' := g(A, i, p) = (\mathbb{1} - P) \circ A + P \circ (\mathbb{1}_0 - A), e \quad (5)$$

where $\mathbb{1}$ denotes the matrix of all ones and $\mathbb{1}_0$ is similar except that the diagonal is replaced by zero. The term $(\mathbb{1} - P)$ serves as the mask that preserves the connections of all node pairs other than those between the anchors j and the target node i. The term $(\mathbb{1}_0 - A)$ intends to flip the whole A (except diagonal) but the P in the front ensures that only the involved (i, j) pairs are actually flipped. Moreover, one can verify that the diagonal of the new adjacency matrix remains zero.

Figure 2 shows an example for a graph with $n = 5$ nodes. Node $i = 3$ is being attacked. The attack vector $p = [1, 0, 1, 0, 1]$ (that is, the anchor nodes are 1, 3, and 5). The connection between 1 and 3 is flipped from 0 (non-edge) to 1 (edge).

The binary elements of the attack vector p may be relaxed into real values between 0 and 1. In this case, the connections of all node pairs other than those between the anchors j and the target node i remain the same. On the other hand, the connections between the involved (i, j) pairs are fractionally changed. The jth element of p indicates the strength of change. The relaxation opens opportunity for gradient based optimization.

3.2. Outer Procedure: GUA

Recall that V_L is the training set with known node labels. Given an attack success rate threshold δ, we formulate the problem as finding a binary vector p such that

$$Err(V_L) := \frac{1}{|V_L|} \sum_{i=1}^{|V_L|} 1_{\hat{l}(A', X, i) \neq \hat{l}(A, X, i)} \geq \delta. \quad (6)$$

To effectively leverage gradient-based tools for adversarial attacks, we perform a continuous relaxation on p so that it can be iteratively updated. Now elements of p stay in the interval $[0, 1]$. The algorithm proceeds as follows. We initialize p with zero. In each epoch, we begin with a binary p and iteratively visit each training node i. If i is not misclassified by the current p, we seek a minimum continuous

perturbation Δp to misclassify it. In other words,

$$\Delta p = \arg\min_{r} ||r||_2 \quad \text{s.t. } \hat{l}(g(A, i, p+r), X, i) \neq \hat{l}(A, X, i). \tag{7}$$

We will elaborate in the next subsection an algorithm to find such Δp. After all training nodes are visited, we perform a hard threasholding at 0.5 and force back p to be a binary vector. Then, the next epoch begins. We run a maximum number of epochs and terminate when (6) is satisfied.

The updated $p \leftarrow p + \Delta p$ found through solving (7), if unbounded, may be problematic because (i) it may incur too many anchor nodes and (ii) its elements may be outside $[0, 1]$. We perform an L_2-norm projection to circumvent the first problem and a clipping to circumvent the second. The rationale of L_2-norm projection is to suppress the magnitude of p and encourage that eventually few entries are greater than 0.5. As can be seen from Figure 3, the number of anchor nodes grows quadratically with the projection radius ξ. A small ξ clearly encourages fewer anchors.

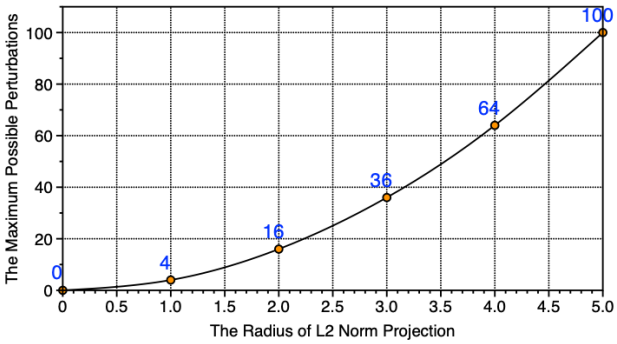


Figure 3. Relationship between the projection radius ξ and the maximum number of possible anchors.

Through experimentation, we find that clipping is crucial to obtaining a stable result. In a later section, we illustrate an experiment to show that the attack success rate may drop to zero in several random trials, if clipping is not performed. See Figure 5.

The procedure presented so far is summarized in Algorithm 1. The algorithm for obtaining Δp is called IMP (iterative minimum perturbation) and will be discussed next.

3.3. Inner Procedure: IMP

To solve (7), we adapt DeepFool (Moosavi-Dezfooli et al., 2016) to find a minimum perturbation that sends the target node i to the decision boundary of another class.

Denote by v be the minimum perturbation. See Figure 4. To find the closest decision boundary other than that of the original class $pred = \hat{l}(A, X, i)$, we first select the closest class k:

$$k = \arg\min_{c \neq pred} \frac{|\Delta f_c|}{\|\Delta w_c\|_2}, \tag{8}$$

where $\Delta f_c = f(A, X)_{i,c} - f(A, X)_{i,pred}$ and $\Delta w_c = \nabla f(A, X)_{i,c} - \nabla f(A, X)_{i,pred}$. Then, we update v by adding to it Δv:

$$\Delta v = \frac{|\Delta f_k|}{||\Delta w_k||_2^2} \Delta w_k. \tag{9}$$

We iteratively update v until $(1 + overshoot) \times v$ successfully attack node i, where $overshoot$ is a small factor that ensures the node passes the decision boundary. We also clip the new A' to ensure stability, in a similar manner to the handling of p in the preceding subsection.

The procedure for computing the minimum perturbation v is summarized in Algorithm 2.

Algorithm 1 Graph Universal Attack (GUA)

Input: max_epoch, max_iter, δ
$p \leftarrow 0$
while $Err(V_L) < \delta$ **and** $epoch < max\_epoch$ **do**
  for $i \in V_L$ **do**
    $A' \leftarrow g(A, i, p)$
    if $\hat{l}(A', X, i) = \hat{l}(A, X, i)$ **then**
      $\Delta p, iter \leftarrow \text{IMP}(A', i)$
      if $iter < max\_iter$ **then**
        $p \leftarrow p + \Delta p$
        $p \leftarrow L_2\text{-norm projection}(p)$
        $p \leftarrow p.clip(0, 1)$
      end if
    end if
  end for
  $p \leftarrow (p > 0.5)\,?\,1 : 0$
  compute $Err(V_L)$
  $epoch \leftarrow epoch + 1$
end while
return p

4. Experiments

In this section, we evaluate thoroughly the proposed attack GUA, through investigation of its design details, comparison with baselines, and validation of transferability from model to model. Code is available at https://github.com/chisam0217/Graph-Universal-Attack

4.1. Details

We compute the anchor set through attacking the standard GCN model (Kipf & Welling, 2017). The parameters of Algorithms 1 and 2 are: $max\_epoch = 100$, $max\_iter =$

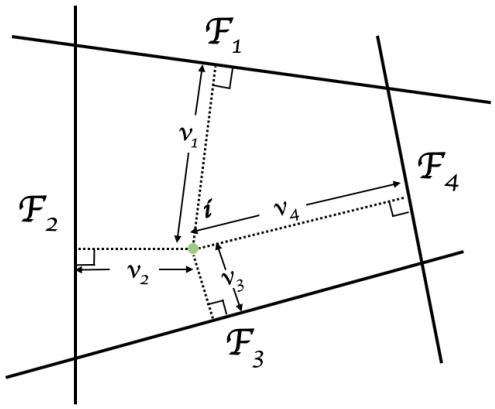


Figure 4. Given a target node i, the F_c's are decision boundaries and the v_c's are the minimum perturbations that send i to each decision boundary.

20, $\delta = 0.9$, and $overshoot = 0.02$. We repeat experiments ten times for each setting.

4.2. Datasets

We experiment with three commonly used node classification benchmark datasets. Their information is summarized in Table 1.

Table 1. Dataset Statistics. Only the largest connected component (LCC) is considered.

DATASET	NODES(LCC)	EDGES(LCC)	CLASSES
CORA	2708	5278	7
CITESEER	3327	4676	6
POL.BLOGS	1222	16714	2

- Cora: It is a citation network of machine learning papers. There are 140 nodes in the training set and 1000 nodes in the test set.
- Citeseer: It is also a citation network. There are 120 nodes in the training set and 1000 nodes in the test set.
- Pol.Blogs: It is a social network of political blogs. There are 121 nodes in the training set and 1101 nodes in the test set.

4.3. Baseline Methods

Because graph universal attacks were barely studied, we design three baseline methods for evaluating the effectiveness of GUA. Additionally, we also compare with Fast Gradient Attack (Chen et al., 2018), a per-node attack method. This attack is not a universal attack; it modifies edges/non-edges connecting to different nodes depending on the target.

Algorithm 2 Iterative Minimum Perturbation (IMP)

Input: adjacency matrix A, node index i, data $overshoot$
$v \leftarrow 0$
$iter \leftarrow 0$
$pred \leftarrow \hat{l}(A, X, i)$
$A' \leftarrow A$
while $\hat{l}(A', X, i) = pred$ **do**
  for $c \neq pred$ **do**
    $\Delta w_c \leftarrow \nabla f(A', X)_{i,c} - \nabla f(A', X)_{i,pred}$
    $\Delta f_c \leftarrow f(A', X)_{i,c} - f(A', X)_{i,pred}$
  end for
  $k \leftarrow \arg\min_{c \neq pred} \frac{|\Delta f_c|}{||\Delta w_c||_2}$
  $\Delta v \leftarrow \frac{|\Delta f_k|}{||\Delta w_k||_2^2} \Delta w_k$
  $v \leftarrow v + \Delta v$
  $A' \leftarrow g(A, i, (1 + overshoot) \times v)$
  $A' \leftarrow A'.clip(0, 1)$
  $iter \leftarrow iter + 1$
end while
$v \leftarrow (1 + overshoot) \times v$
return $v, iter$

- Global Random: Each node has a probability $Prob$ to become an anchor node. In other words, each element of the attack vector p is an independent sample of Bernoulli($Prob$).
- Limited Random: We sample a prescribed number of anchor nodes without replacement from the whole graph.
- Victim-Class Attack: We sample a prescribed number of anchor nodes without replacement from nodes of a particular class. This baseline originates from a finding that the anchor nodes computed by GUA often belong to the same class. More details will come later.
- Fast Gradient Attack (FGA): This method flips the connections between the target node and nodes with the largest absolute gradient values. This baseline does not perform universal attacks.

4.4. Results

The evaluation metric is attack success rate (ASR). Another quantity of interest is the number of modified links (ML). For universal attacks, it is equivalent to the anchor set size.

Importance of clipping. As discussed in the design of GUA, the continuous relaxation of the attack vector p requires clipping throughout optimization. For an empirical supporting evidence, we show in Figure 5 the ASR obtained through executing Algorithm 1 with and without clipping, respectively. Clearly, clipping leads to stabler and superior results. Without clipping, the ASR may drop to zero in

some random trials. The reason is that several entries of p become strongly negative, such that projections result in small values for all positive entries and subsequent hard thresholding zeros out the whole vector p.

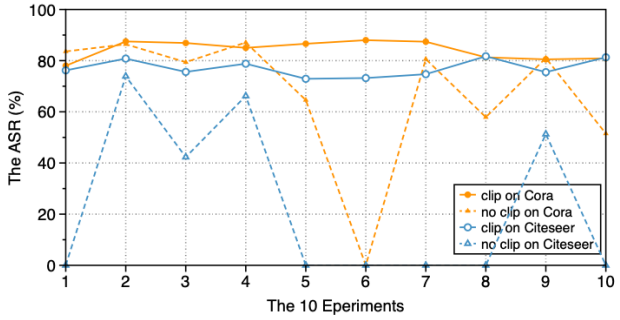


Figure 5. ASR: using clipping versus not. Ten experiments are repeated.

Effect of projection radius. We treat the L_2-norm projection radius ξ as a parameter and study its relationship with the ASR. See Table 2. As expected, the larger ξ the more anchor nodes appear, because the projected vector p likely contains more values greater than 0.5. A larger anchor set also frequently results in higher ASR, because of more changes to the graph.

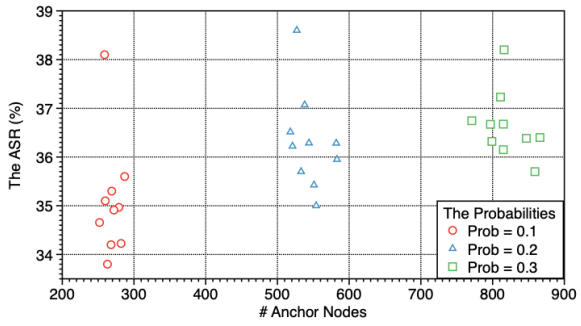
Table 2. Average ML and average ASR under different projection radius ξ.

DATASET	ξ	AVG. ML	AVG. ASR(%)
CORA	3	4.4	75.60
	4	7.9	84.21
	5	10.8	83.56
CITESEER	3	3.3	63.67
	4	7.7	77.07
	5	13.9	80.01
POL.BLOGS	3	2.6	17.68
	4	5.3	22.65
	5	10	30.84

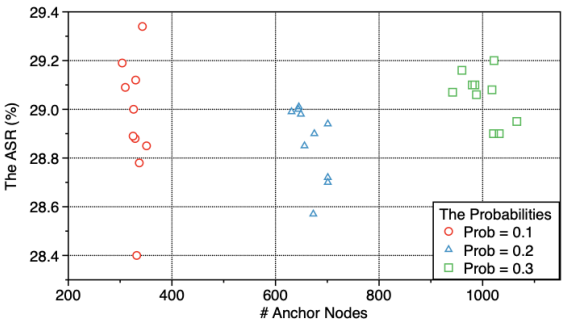
One sees from the table that attacks on Cora and Citeseer are quite effective. In fact, the individual result for each trial may suggest even more attractive findings. For example, for the case of Cora and $\xi = 4$, the MLs for the ten trials are $\{5, 9, 10, 7, 8, 9, 9, 9, 6, 7\}$ and the corresponding ASRs are $\{0.780, 0.875, 0.869, 0.850, 0.866, 0.880, 0.874, 0.813, 0.805, 0.809\}$. This result means that as few as six anchor nodes are sufficient to achieve 80% ASR.

On the other hand, one also sees that the attacks on Pol.Blogs are less effective. The reason is that the graph has a large average degree, which makes it relatively robust to universal attacks. As observed by Zügner et al. (2018) and Wu et al. (2019a), nodes with more neighbors are harder to attack than those with fewer neighbors. The higher density of the graph requires a larger anchor set to achieve higher ASR.

Blindly using more anchors does not work. Now that we have a sophisticated method to compute the anchor nodes, we investigate whether randomly chosen anchor nodes are similarly effective. In Figure 6, we plot the ASR results of the baseline Global Random. One sees that all ASRs for Cora are below 40% and for Citeseer below 30%. Such results are way inferior to those of the proposed GUA. Moreover, using hundreds and even a thousand anchors does not improve the ASR. The random method is not effective.



(a) Cora.



(b) Citeseer.

Figure 6. Performance of global random attack, repeated ten times.

Anchors often belong to the same class. With analysis of the anchors, an interesting finding is that one class dominates. In Figure 7, we plot the entropy of the class distribution of the anchors for each random trial. One sees that a majority of the entropies is zero, which indicates that in these cases only one class appears.

Wrong classifications often coincide with the anchor class. A natural conjecture following the above finding is that a target node will be misclassified to the (majority) class

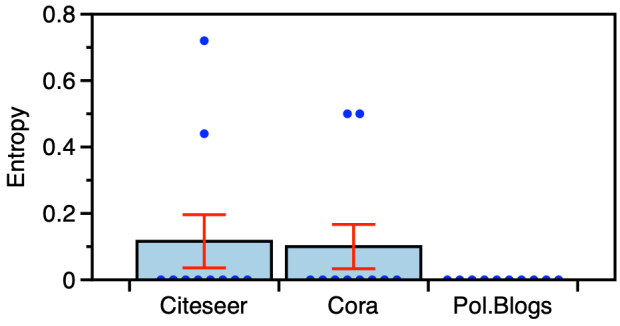


Figure 7. Class distribution entropy of the anchor nodes. Ten experiments are repeated.

of the anchors. Table 3 (on the data set Cora) corroborates this conjecture. The table indicates that 96% of the test nodes are misclassified to class 6 when all the anchor nodes belong to this class. An analysis of the data set shows that each node has two neighbors on average. Hence, flipping the connections to the anchor nodes possibly makes the anchor class dominate among the new set of neighbors. Then, classifying into the anchor class becomes more likely. This result echoes one mentioned by Nandanwar & Murty (2016), who conclude that classification of a node is strongly influenced by the classes of its neighbors; it tends to coincide with the majority class of the neighbors.

Table 3. Number of test nodes predicted into a certain class, when the anchor nodes belong to class 6. Data set: Cora.

DATASET	CLASS 6	OTHER CLASSES
# TEST NODES	959	41

To generalize the above result, in Table 4 we list the entropy of the class distribution before and after attack. For all data sets, the entropy decreases, indicating stronger dominance of one class after attack. The decrease is more substantial for Cora and Citeseer than for Pol.Blogs, which is expected, because the latter has denser and more varied connections, which eclipse the dominance of the anchor class.

Table 4. Class entropy before and after attack.

ENTROPY	CORA	CITESEER	POL.BLOGS
BEFORE ATTACK	1.863	1.784	0.689
AFTER ATTACK	0.203	0.297	0.623

Comparison with baselines. We compare the proposed GUA with four baseline methods explained in Section 4.3. Since we cannot choose the number of anchor nodes for GUA, we obtain this value based on the results in Table 1 when $\xi = 4$. In this case, the average ML for Cora, Citeseer, and Pol.Blogs is respectively 7.9, 7.7, and 5.3. Therefore, we set the number of anchor nodes for Limited Random and Victim-Class Attack, as well as the average number of anchor nodes for FGA, to be the ceiling of these values. For Global Random, $Prob$ is set such that the expected number of anchor nodes is these values.

Table 5. Average ASR. For a fair comparison, all methods except FGA use the same number of anchor nodes. FGA is not a universal attack and we set the average number of modified links per node to be the same as the number of anchor nodes.

ATTACK METHOD	CORA	CITESEER	POL.BLOGS
GUA	84.21%	77.07%	22.65%
GLOBAL RANDOM	22.00%	26.58%	8.61%
LIMITED RANDOM	17.26%	29.95%	11.13%
VICTIM ATTACK	62.64%	54.47%	19.28%
FGA (NOT UNIV.)	89.70%	84.82%	59.80%

From Table 5, one sees that GUA significantly outperforms other universal attack methods. Among them, Victim-Class Attack is the most effective, but it is still inferior to GUA. This result suggests that GUA leverages more information (in this case, node features) than the class labels, although we have seen strong evidences that anchor nodes computed by GUA mostly belong to the same class.

From the table, one also sees that GUA is inferior to FGA if only ASR is concerned. FGA is not a universal attack method; it finds different anchors for each target node. Thus, it is possible to optimize the number of anchors (possibly different for each target) to aim at a certain ASR, or equivalently, to achieve a better ASR given a certain number of anchors. However, it is also because FGA is not a universal attack, that the total number of anchor nodes for all targets soars. For example, FGA modifies links with 1406 anchors on Cora and 1359 anchors on Citeseer in total.

Effect of removing anchor nodes. Once a set of anchor nodes is identified, a natural question asks if the set contains redundancy. We perform the following experiment: we randomly remove a number of anchor nodes and recompute the ASR. Because on Cora and Citeseer, the average number anchor nodes for $\xi = 4$ is 7.9 and 7.7 respectively, we use an anchor set of size eight to conduct the experiment. For each case, we randomly remove 1–7 nodes from the anchor set and report the corresponding average ASR. The results are shown in Figure 8.

From the figure, one sees that the average ASR gradually decreases to zero as more and more anchor nodes are removed. This result indicates that there exists no redundancy in the anchor set. The decease is faster when more nodes

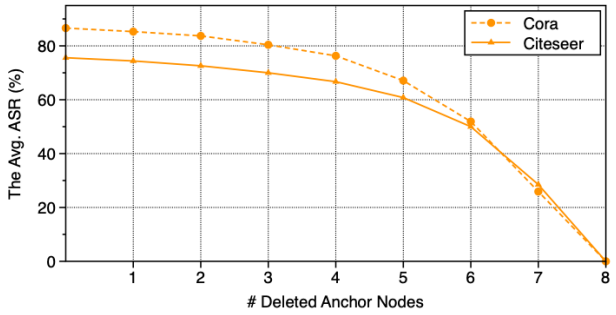


Figure 8. Average ASR after deleting nodes from the anchor set.

are removed, but the average ASR is still quite high even when removing half of the nodes. This finding is the second evidence that supports the trade-off between anchor set size and ASR, in additional to the prior Table 2.

Transferability. We have already seen that GUA is quite effective in attacking GCN, based on the results obtained so far. Such an attack belongs to the white-box family, because knowledge of the model to be attacked is assumed. Such knowledge includes the model form as well as the model parameters. In reality, however, the model parameters may not be known at all, not even the model form. Attacks under this scenario is called black-box. One approach to conducting black-box attack is to use a surrogate model. In our case, if one is interested in attacking graph deep learning models other than GCN, GCN may serve as the surrogate. The important question is whether anchors found by attacking the surrogate can effectively attack other models as well.

Table 6. Average ASR after applying the anchor nodes found by GUA when $\xi = 4$ on other node classification models.

METHODS	CORA	CITESEER
GCN	84.21%	77.07%
DEEPWALK	85.80%	81.71%
NODE2VEC	80.84%	74.07%
GAT	85.15%	77.02%

We perform an experiment with three such models: DeepWalk (Perozzi et al., 2014), node2vec (Grover & Leskovec, 2016), and GAT (Veličković et al., 2017). The first two compute, in an unsupervised manner, node embeddings that are used for downstream classification, whereas the last one is a graph neural network that directly performs classification. In Table 6, we list the ASR for these models. One sees that the ASRs are similarly high as that for GCN; sometimes even surpass. This finding concludes that the results of GUA are well transferable.

5. Related Work

Since the seminal work by Szegedy et al. (2014), various types of methods have been proposed to generate adversarial examples. For instance, Goodfellow et al. (2014) introduce the fast gradient sign method and Carlini & Wagner (2017) develop a powerful attack through iterative optimization. This work is related to recent advances in adversarial attacks on GNNs and general universal attacks.

Adversarial attack on GNN. Zügner et al. (2018) propose NETTACK that uses a greedy mechanism to attack the graph embedding model by changing the entry that maximizes the loss change. Dai et al. (2018) introduce a reinforcement learning based method that modifies the graph structure and significantly lowers the test accuracy. Wang et al. (2018) propose Greedy-GAN that poisons the training nodes through adding fake nodes indistinguishable by a discriminator. Chen et al. (2018) recursively compute connection-based gradients and flip the connection value based on the maximum gradient. Zügner & Günnemann (2019) use meta-gradients to solve a bi-level problem formulated for training-time attacks.

Universal attack. Moosavi-Dezfooli et al. (2017) train a quasi-imperceptible universal perturbation that successfully attacks most of the images in the same dataset. Brown et al. (2017) generate a small universal patch to misclassify any image into any target class. Wu & Fu (2019) search for a universal perturbation that can be well transferred to several models.

6. Conclusion

In this work, we consider universal adversarial attacks on graphs and propose the first algorithm, named GUA, to effectively conduct these attacks. GUA finds a set of anchor nodes to mislead the classification of all nodes in the graph through flipping the connections between the anchors and the target node. GUA achieves the highest ASR compared to several universal attack baselines. There exists a trade-off between ASR and the anchor set size and we find that a very small size is sufficient to achieve remarkable attack success. Additionally, we find that the computed anchor nodes often belong to the same class. We also find that the anchor nodes used to attack one model equally well attack other models. In the future, we plan to develop defense mechanisms to effectively counter these attacks.

Acknowledgements

J. Chen is supported in part by DOE Award de-oe0000910.

References

Brown, T., Mane, D., Roy, A., Abadi, M., and Gilmer, J. Adversarial patch. 2017. URL https://arxiv.org/pdf/1712.09665.pdf.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *ICLR*, 2014.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., and Xuan, Q. Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*, 2018.

Conover, M. D., Ratkiewicz, J., Francisco, M., Gonçalves, B., Menczer, F., and Flammini, A. Political polarization on twitter. In *Fifth international AAAI conference on weblogs and social media*, 2011.

Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

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

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

Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *NIPS*, 2017.

Jin, M., Chang, H., Zhu, W., and Sojoudi, S. Power up! robust graph convolutional network against evasion attacks based on graph powering. *arXiv preprint arXiv:1905.10029*, 2019.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kumar, S., West, R., and Leskovec, J. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web*, pp. 591–602. International World Wide Web Conferences Steering Committee, 2016.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. In *ICLR*, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Nandanwar, S. and Murty, M. N. Structural neighborhood based classification of nodes in a network. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1085–1094. ACM, 2016.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR*, 2014.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

Tsubaki, M., Tomii, K., and Sese, J. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35 (2):309–318, 2018.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Wang, X., Eaton, J., Hsieh, C.-J., and Wu, F. Attack graph convolutional networks by adding fake nodes. *arXiv preprint arXiv:1810.10751*, 2018.

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

Wu, J. and Fu, R. Universal, transferable and targeted adversarial attacks. *arXiv preprint arXiv:1908.11332*, 2019.

Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., and Tan, T. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 346–353, 2019b.

Xu, K., Chen, H., Liu, S., Chen, P.-Y., Weng, T.-W., Hong, M., and Lin, X. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*, 2019.

Zügner, D. and Günnemann, S. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

Zügner, D., Akbarnejad, A., and Günnemann, S. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2847–2856. ACM, 2018.