Adversarial Examples for Graph Data: Deep Insights into Attack and Defense

Huijun Wu[1,2] , **Chen Wang**[2] , **Yuriy Tyshetskiy** [2] , **Andrew Docherty**[2] ,
Kai Lu[3] , **Liming Zhu**[1,2]
[1]University of New South Wales, Australia
[2]Data61, CSIRO
[3]National University of Defense Technology, China
{first, second}@data61.csiro.au, kailu@nudt.edu.cn

Abstract

Graph deep learning models, such as graph convolutional networks (GCN) achieve state-of-the-art performance for tasks on graph data. However, similar to other deep learning models, graph deep learning models are susceptible to adversarial attacks. However, compared with non-graph data the discrete nature of the graph connections and features provide unique challenges and opportunities for adversarial attacks and defenses. In this paper, we propose techniques for both an adversarial attack and a defense against adversarial attacks. Firstly, we show that the problem of discrete graph connections and the discrete features of common datasets can be handled by using the integrated gradient technique that accurately determines the effect of changing selected features or edges while still benefiting from parallel computations. In addition, we show that an adversarially manipulated graph using a targeted attack statistically differs from un-manipulated graphs. Based on this observation, we propose a defense approach which can detect and recover a potential adversarial perturbation. Our experiments on a number of datasets show the effectiveness of the proposed techniques.

1 Introduction

Graphs are commonly used to model many real-world relationships, such as social networks [Newman *et al.*, 2002], citation networks, transaction networks [Ron and Shamir, 2013], and the control-flow graphs of computer programs [Allen, 1970]. Compared with the traditional machine learning methods [Bhagat *et al.*, 2011; Xu *et al.*, 2013], graph deep learning models have recently pushed forward the state-of-the-art for machine learning tasks on graph data [Kipf and Welling, 2017; Veličković *et al.*, 2018; Cao *et al.*, 2016; Henaff *et al.*, 2015]. In particular, graph convolutional networks (GCN) [Bruna *et al.*, 2013; Edwards and Xie, 2016] and its recent variants [Kipf and Welling, 2017] perform convolution operations in the graph domain by aggregating and combining the information of neighboring nodes. In these techniques, both node features and the graph structure (the edges between nodes) are used by the model.

A common predictive task on graph data is *node classification*: Given a graph, features for all nodes, and labels for a subset of nodes, the goal is to predict the labels for the unlabelled nodes. The labels of the nodes can be, for example, the topics of papers in citation networks, or customer types in e-commerce networks.

Deep learning methods are often criticized for their lack of robustness [Goodfellow *et al.*, 2015]. This has been demonstrated by the ease of crafting adversarial examples that fool the deep neural networks and give incorrect predictions by adding small perturbations to the examples that are unnoticeable to humans. These perturbations are termed adversarial attacks, and they are major obstacles for deep learning applications especially if they are to be used in safety-critical scenarios. Graph neural networks are no exception: a malicious user can manipulate their profile, connect to targeted users, or connect to fake users they have created purposefully in order to mislead a graph deep learning system. Similarly, adding fake comments to specific products can fool the recommender systems of a website.

The key challenge that limits the simple adoption of existing adversarial attack techniques used in non-graph data on graph convolutional networks is that the edges and the features of the graph data are typically discrete. To address this challenge, a recent work uses reinforcement learning or genetic algorithm based methods [Dai *et al.*, 2018] to find the adversarial graph. Zügner et al [Zügner *et al.*, 2018] proposes to attack a simplified surrogate model and exploit the transferability of adversarial attacks to attack the graph convolutional networks. They also show that despite the discreteness of graphs, gradient-based attacks such as fast gradient sign method (FGSM) can also achieve attacks in some cases. Nevertheless, the gradient-based approaches may achieve sub-optimal attack performance due to the inaccurate gradients on discrete data. We note that in the case of weighted graphs with continuous features attacks could be aimed at the weights of the edges and may avoid this problem.

In this paper, we show that using integrated gradients we can calculate the model change caused by flipping a discrete edge or feature accurately. Integrated gradients approximate Shapley values [Hart, 1989; Lundberg and Lee, 2016] by integrating partial gradients with respect to input features from a reference input to the input of interest. Integrated gradients greatly improve the efficiency and accuracy of the node and edge selection in comparison to iterative methods.

Compared with adversarial attacks, the defense against adversarial attacks in graph models has not been well-studied. In this paper, we show that one key reason for the vulnerability of graph deep learning models, in particular GCNs, is that these models use a weighted mean of the features of their neighbors and therefore heavily rely on the aggregate information of their neighborhood when making predictions. We have investigated the perturbations made by the existing attack techniques on GCNs and found that adding edges which connect to nodes with very different features from those in the original neighborhood plays a key role in all of the attack methods. In this paper, we show that statistical analysis of the node features can identify the edges inserted by adversarial attacks. For nodes with bag-of-words (BOW) features we use the Jaccard index to find nodes with features that are highly dissimilar to the other neighbors. We show that by removing such edges we are able to defend against targeted adversarial attacks without decreasing the accuracy of the GCN models. Our results on a number of real-world datasets show the effectiveness and efficiency of the proposed attack and defense.

2 Preliminaries

2.1 Graph Convolutional Network

Given an attributed graph $\mathcal{G} = (\mathcal{A}, \mathcal{X})$, $\mathcal{A} \in [0,1]^{N\times N}$ is the adjacency matrix and $\mathcal{X} \in [0,1]^{N\times D}$ represents the D-dimensional binary node features, the indices for the nodes are $\mathcal{V} = \{1, 2, ..., N\}$ and for the features $\mathcal{F} = \{1, 2, ..., D\}$. We now consider the task of semi-supervised or transductive node classification where a subset of nodes $\mathcal{V}_{\mathcal{L}} \subseteq \mathcal{V}$ are labelled with one class from the set of classes $\mathcal{C} = \{1, 2, ..., c_K\}$. The aim is to predict the class for the unlabelled nodes given the labelled nodes.

In this work, we focus on graph convolutional networks (GCN) [Kipf and Welling, 2017], a well-established method for semi-supervised node classifications. For GCN, initially, $H^0 = X$; the GCN model then applies the following rule to aggregate the neighboring features:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of the graph $\mathcal{G}$ with self connections added, $\hat{D}$ is a diagonal matrix with $\tilde{D}_{i,i} = \Sigma_j \tilde{A}_{ij}$, and σ is the activation function to introduce non-linearity. Each of the above equation corresponds to one graph convolution layer. A fully connected layer with softmax loss is usually used after L layers of graph convolution layers for the classification. A two-layer GCN is commonly used for semi-supervised node classification tasks [Kipf and Welling, 2017]. The model can, therefore, be described as:

$$Z = f(X, A) = \text{softmax}(\hat{A}\sigma(\hat{A}XW^{(0)})W^{(1)}) \tag{2}$$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. $\hat{A}$ is essentially the symmetrically normalized adjacency matrix. $W^{(0)}$ and $W^{(1)}$ are the input-to-hidden and hidden-to-output weights, respectively.

2.2 Gradients Based Adversarial Attacks

Gradients are commonly used to construct adversarial attacks deep learning models [Yuan *et al.*, 2019]. One can either use the gradients of the loss function or the gradients of the model output with respect to (w.r.t.) the input to construct the attacks. Two examples of these methods are Fast Gradient Sign Method (FGSM) attack and Jacobian-based Saliency Map Approach (JSMA) attack. Fast Gradient Sign Method (FGSM) [Ian J. Goodfellow, 2014] generates adversarial examples by perturbing the input along the direction of the sign of gradients of loss function w.r.t. each pixel for image data. The adversarially perturbed input is given by:

$$x^{'} = x + \epsilon\text{sign}(\nabla J_\theta(x, l)) \tag{3}$$

where ϵ is the magnitude of the perturbation. J is the loss of the model, θ are the model parameters, x is the model input, and l is the label of x.

The Jacobian-based Saliency Map Approach (JSMA) attack was first proposed in [Papernot *et al.*, 2016]. It uses perturbations that force the model to mis-classify the example point into a specific *target class*. Specifically, given a feed-forward neural network $\mathbf{F}$ and example point $\mathbf{X}$, the Jacobian is computed by:

$$\nabla F(X) = \frac{\partial F(X)}{\partial X} = \left[\frac{\partial F_j(X)}{\partial x_i}\right]_{i\in 1...N, j\in 1...M} \tag{4}$$

where the dimensions for the model output (number of classes) and the input are M and N, respectively. The aim is to maximize the output for the target class c, $F^{(c)}(X)$, whiles minimizing the output for the other classes $j \neq c$ to decrease. This is accomplished by exploiting the adversarial saliency map which is defined by:

$$S(X, c) = \left\{ \begin{array}{l} 0, \text{ if } \frac{\partial F_c(X)}{\partial X} < 0 \text{ or } \Sigma_{j\neq t}\frac{\partial F_j(X)}{\partial X} > 0 \\ \frac{\partial F_c(X)}{\partial X}|\Sigma_{j\neq t}\frac{\partial F_j(X)}{\partial X}|, \text{ otherwise} \end{array} \right\} \tag{5}$$

The adversarial attack is created by starting from a selected example point and iteratively perturbing the example point in the direction of $S(X, c)$ a small amount until the predicted label changes. For an untargeted attack, the prediction score is minimized for the winning class in a similar fashion.

2.3 Defense for Adversarial Examples

Defense against adversarial attacks have been studied in the context of convolutional neural networks for image classification [Xu *et al.*, 2018; Papernot and McDaniel, 2018]). For images, the feature space is continuous and adversarial examples are crafted with small additive perturbations. Therefore, in some cases, adding some randomization to the images at the testing time can help defend against adversarial attacks [Xie *et al.*, 2018]. Other forms of input pre-processing, such as local smoothing [Xu *et al.*, 2018] and image compression [Shaham *et al.*, 2018] have also been used to defend against attacks. These forms of pre-processing work due to the fact that neighboring pixels of natural images are typically correlated. Adversarial training [Tramèr *et al.*, 2018] introduces generated adversarial examples to the training data to enhance the robustness of the model. For graph convolutional networks, Dai et al. [Dai *et al.*, 2018] briefly introduces the possibility of defending adversarial examples by dropping some edges during the training. However, this method barely improves the robustness of the model.

3 Integrated Gradients Guided Attack

Due to their assumption of continuous features in the input pixel space, FGSM and JSMA are not well-studied for graph models. Furthermore, recent explorations into graph adversarial attack techniques show that simply applying these methods may not lead to successful attacks [Zügner *et al.*, 2018; Dai *et al.*, 2018]. These works have addressed this problem by either attacking a simplified surrogate model or reinforcement learning based methods which are typically computationally expensive.

The node features in many graph datasets are often either bag-of-words or categorical features which take on binary values of 0 or 1. Furthermore, the edges in a graph are also represented by binary values of 0 or 1 in the adjacency matrix. When attacking the model, the adversarial perturbations are therefore limited to either changing 1 to 0 or vice versa. The main issue of applying vanilla FGSM and JSMA in graph models is the fact that the gradients are calculated locally at the current example point. In particular, given a target node t, for FGSM attack, $\nabla J_{W^{(1)},W^{(2)}}(t) = \frac{\partial J_{W^{(1)},W^{(2)}}(t)}{\partial X}$ measures the feature importance of all nodes to the loss function value. Here, X is the feature matrix, each row of which describes the features for a node in the graph. For a specific feature j of node n, a larger value of $\nabla J_{W^{(1)},W^{(2)}}{}_{jn}$ indicates increasing the value of feature j locally will decrease the confidence of the predictions for the target node. However, changing the value of feature j to 1 may not help for two reasons: firstly, the feature value might already be 1; secondly, the GCN model is not a linear function and therefore the local gradient does not necessarily predict the result of a large change in value. This argument applies to JSMA. For example, a simple ReLU network $f(x) = ReLU(x - 0.5)$ as an example, when x is increased from 0 to 1, the function value increases by 0.5. However, computing the gradient at $x = 0$ gives 0, which does not predict this increase. To address this, we propose an integrated-gradient based method. Integrated gradients were initially proposed by [Sundararajan *et al.*, 2017] to provide sensitivity and implementation invariance for feature attribution in the deep neural networks, particularly the convolutional neural networks for images.

The integrated gradient is defined as follows: for a given model $\mathbf{F} : R^n \to [0,1]$, where $x \in R^n$ is the input, $x^{'}$ is the baseline input (e.g., a black image for image data). The integrated gradient is the path integral of the gradient along a path from $x^{'}$ to the input x. Namely, for the i^{th} feature of x, the integrated gradients (IG) is given as follows:

$$IG_i(F(x)) ::= (x_i - x_i^{'}) \times \int_{\alpha=0}^{1} \frac{\partial F(x^{'} + \alpha x(x - x^{'}))}{\partial x_i} d\alpha \tag{6}$$

For GCN on graph data, we propose a generic attack framework. Given the adjacency matrix A, feature matrix X, and the target node t, we compute the integrated gradients for function $F_{W^{(1)},W{(2)}}(A, X, t)$ w.r.t. I where I is the input for the attack. $I = A$ indicates edge attacks while $I = X$ indicates feature attacks. When F is the loss function of the GCN model, we call this attack technique IG-FGSM. Similarly, when F is the prediction output of the GCN model we call the attack technique IG-JSMA. For a targeted IG-JSMA or IG-FGSM attack, the optimization goal is to maximize the value of F. Therefore, for the features with a value of 1 or edges that exist in the graph, we select those which have the lowest negative IG scores and either, for features, set their value to 0 or, for edges, remove them. In contrast, the untargeted IG-JSMA attack aims to minimize F for the winning class; therefore for features with a value of 1, we set those which have the highest positive IG scores to 0, and similarly for edges in the graph we remove those which have the highest positive IG scores.

For the baseline input We use either all-zero or all-one feature/adjacency matrices to represent the $1 \to 0$ or $0 \to 1$ perturbations. When removing a specific edge or changing a feature from 1 to 0, we set the adjacency matrix $\mathbf{A}$ or feature matrix $\mathbf{X}$ respectively to all-zeros. On the contrary, to add edges or features, we set either $\mathbf{A}$ or $\mathbf{X}$ respectively to an all-one matrix. To keep the direction of gradients consistent and ensure the computation is tractable, $IG(F(X, A, t))[i, j]$ (for edge attack) is approximated as follows:

$$\begin{cases} \frac{A_{ij}}{m} \times \Sigma_{k=1}^{m} \frac{\partial F(\frac{k}{m} \times (A_{ij} - 0))}{\partial A_{ij}}, \text{removing edges} \\ \frac{1-A_{ij}}{m} \times \Sigma_{k=1}^{m} \frac{\partial F(A_{ij} + \frac{k}{m} \times (1 - A_{ij}))}{\partial A_{ij}}, \text{adding edges} \end{cases} \tag{7}$$

Algorithm 1 shows the pseudo-code for untargeted IG-JSMA attack. We compute the integrated gradients of the prediction score for the winning class c w.r.t. the entries of A and X. The integrated gradients are then used as a measure of the importance of changing specific features or edges in the graph G. Note that we only compute the importance of adding edges if the edge does not exist before and only compute the importance of changing a feature to 1 if it is currently 0 and vice versa. Therefore, for a feature or an edge with high importance, we simply flip the binary value.

While setting the number of steps m for computing integrated gradients, one size does not fit all. Therefore, we enlarge the number of steps while attacking the nodes with low classification margins until the required accuracy is achieved. Moreover, the calculation can be done in an incremental way if we increase the number of steps by integer multiples.

To ensure the perturbations are unnoticeable, the graph structure and feature statistics should be preserved. The specific properties to preserve depend on the application requirements. For our IG based attacks, we check against these application-level requirements while selecting an edge or a feature for perturbation. In practice, this process can be efficient as many statistics can be pre-computed and updated incrementally [Zügner *et al.*, 2018].

Algorithm 1: IG-JSMA - Integrated Gradient Guided untargeted JSMA attack on GCN

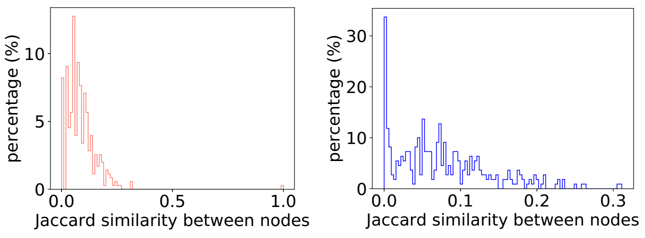
```
Input: Graph $G^{(0)} = (A^{(0)}, X^{(0)})$, target node $v_0$
       $F$: the GCN model trained on $G^{(0)}$
       budget $\Delta$: the maximum number of perturbations.
Output: Modified graph $G^{'} = (A^{'}, X^{'})$.
1  Procedure Attack()
2      //compute the importance scores for edges and features.
3      $s_e \leftarrow$ calculate_edge_importance(A)
4      $s_f \leftarrow$ calculate_feature_importance(X)
5      //sort nodes and edges according to their scores.
6      features $\leftarrow$ sort_by_importance(s_f)
7      edges $\leftarrow$ sort_by_importance(s_e)
8      f $\leftarrow$ features.first, e $\leftarrow$ edges.first
9      while $|A^{'} - A| + |X^{'} - X| < \Delta$ do
10         //decide which to perturb
11         if $s_e[e] > s_f[f]$ then
12             flip feature f
13             f $\leftarrow$ f.next
14         else
15             flip edge e
16             e = $\leftarrow$ e.next
17         end
18     end
19     return $G\prime$
```

4 Defense for Adversarial Graph Attacks

In order to defend against adversarial targeted attacks on GCNs, we first assume that the GCN model is trained on a graph modified by an adversarial attack. Adversarial attacks on deep learning systems are transferable to models with similar architecture and trained on the same dataset, therefore any model trained on the attacked graph will most likely show the same behavior as the model used to craft the attack. A possible defense in this scenario is to make the adjacency matrix trainable. By learning selected edge weights during the training process, there is the potential to change the graph structure enough to make the attack crafted using vanilla GCN on the original graph not effective.

We verify this idea by making the edge weights trainable in GCN models. For the CORA-ML dataset, we select a node that is correctly classified by a trained model. An adversarially modified graph was then constructed by using nettack [Zügner *et al.*, 2018]. Without any defense, the target node is *misclassified* with the confidence of 0.998 after the attack. Our defense technique is then used. Starting with the adversarially modified graph we train the GCN model without making any additional modifications on the loss function, with the exception of adding selected trainable edge weight. Interestingly, with such a simple defense method, the target node is *correctly classified* with the confidence of 0.912 after the attack.

To explain why the defense works, we observe the following: firstly, perturbing edges is more effective than modifying



(a) Clean (b) Attacked

Figure 1: Histograms for the Jaccard similarities between connected nodes before and after FGSM attack.

the features in all the attacks (i.e., FGSM, JSMA, nettack, and IG-JSMA). We have observed that feature-only perturbations generally fail to change the predicted class of the target node. Moreover, the attack approaches tend to favor adding edges over removing edges.

Secondly, nodes with more neighbors are more difficult to attack than those with fewer neighbors. This is also consistent with the observations in [Zügner *et al.*, 2018] that nodes with higher degrees have higher classification accuracy in both the clean and the attacked graphs.

Finally, the attacks tend to connect the target node to nodes with considerably different features and labels. We have observed that this is the most effective way to perform attacks. We verify this observation using the CORA-ML dataset using the Jaccard similarity score. Given two nodes u and v with n binary features, the Jaccard similarity score measures the overlap that u and v share with their features. Each feature of u and v can either be 0 or 1. The Jaccard similarity score is then given by:

$$J_{u,v} = \frac{\sum_{i=1}^{n} u_i v_i}{\sum_{i=1}^{n} u_i + v_i - u_i v_i}. \quad (8)$$

Note that our defense mechanism is generic, while the similarity measures may vary among different datasets, in particular for other types of features we may use different similarity measures such as the cosine similarity or the correlation coefficient.

We train a two-layer GCN on the CORA-ML dataset and study the nodes that are classified correctly with high probability (i.e. ≥ 0.8). For these nodes, Figure 1 shows the histograms for the Jaccard similarity scores between connected nodes before and after the FGSM attack. We note that the adversarial attack significantly increases the number of neighbors which have very low similarity scores to the target nodes. This also holds for nettack [Zügner *et al.*, 2018]. For example, we enable both feature and edge attacks for nettack and attack the node 200 in the GCN model trained on CORA-ML dataset. Given the node degree of 3, the attack removes the edge 200 $\rightarrow$ 1582 because node 1582 and node 200 are similar ($J_{1582,200}$ = 0.113). Meanwhile, the attacks add edge 200 $\rightarrow$ 1762 and 200 $\rightarrow$ 350, and node 200 shares no feature similarity with the two nodes. No features were perturbed in this experiment.

This result is consistent with our observations. Compared with deep convolutional neural networks (for image data) which often have many more layers and parameters than graph neural networks GCN models are relatively shallow. The GCN model essentially aggregates features from the neighborhood of each node at each layer. For a target node, an adversarially crafted graph attempts to connect the nodes with different features and labels to maximally change the aggregated neighborhood features. Correspondingly, while removing edges, the attack tends to remove the edges connecting the nodes that are most similar to the target node. The edge attacks are more effective due to the fact that adding or removing one edge affects all the feature dimensions during the aggregation. In contrast, modifying one feature only affects one dimension in the feature vector and the perturbation can be easily masked by other neighbors of nodes with high degrees.

Based on these observations, we make another hypothesis that the above defense approach works because the model assigns lower weights to the edges that connect the target node to the nodes are dissimilar in terms of features. To verify this, we plot the learned weights and the Jaccard similarity scores of the end nodes for the edges starting from the target node (see Figure 2). Note that for the target node we choose, the Jaccard similarity scores between every neighbor of the target node and itself are larger than 0 in the clean graph. The edges with zero similarity scores are all added by the attack. As expected, the model learns low weights for most of the edges with low similarity scores so that the perturbations have much lower influence to the prediction of the target node.

To make the defense more efficient, we do not even need to use trainable edge weights as the defense. Learning the edge weights inevitably introduces extra parameters to the model, which may affect the scalability and accuracy of GCN models. A simple approach that is potentially as effective can be constructed by noting that in the majority of datasets nodes generally do not connect to nodes that have no feature similarities. In addition, the learning-based defense technique essentially assigns low weights to the edges connecting two dissimilar nodes.

Our simple defense approach is pre-processing based. We perform a pre-processing on a given graph before training. We check the adjacency matrix of the graph and inspect the edges. All the edges that connect nodes with low similarity score (e.g., $J_{u,v} = 0$) are selected as candidates to remove. Although the clean graph may also have a small number of such edges, we find that removing these edges does little harm to the prediction of the target node. On the contrary, the removal of these edges may improve the prediction in some cases. This is intuitive as aggregating features from nodes that differ sharply from the target often over-smooths the node representations. In fact, a recent study [Wu *et al.*, 2019] shows that the non-linearity and multiple weight matrices at different layers do not contribute much to the predictive capabilities of GCN models but introduce unnecessary complexity. For example, [Zügner *et al.*, 2018] uses a simplified surrogate model to achieve the attacks on GCN models for the same reason.

The proposed defense is computationally efficient as it only makes one pass to the existing edges in the graph, thus having the complexity of $O(N)$ where N is the number of edges. For large graphs, calculating the similarity scores can be easily parallelized in implementation.

Finally, we note that this simple pre-processing defense shares similarities with defense techniques in image classification which smooth the images. They both are based on the assumption that the neighborhood of either the pixel in image datasets or the node in graphs datasets are typically highly correlated.

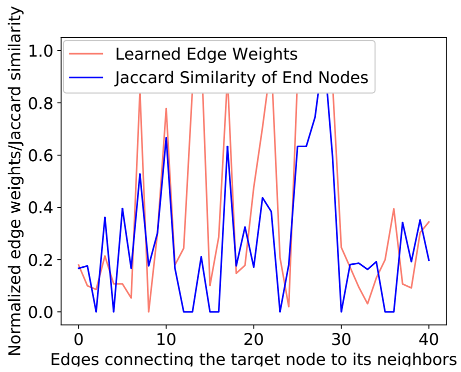


Figure 2: The normalized learned edge weights and the Jaccard similarity scores for the end nodes of the edges. Each value of the x-axis represents an edge in the neighborhood of the target node.

5 Evaluation

We use the widely used CORA-ML [McCallum *et al.*, 2000], CITESEER [Bojchevski and Günnemann, 2018] and Polblogs [Adamic and Glance, 2005] datasets. The overview of the datasets is listed below.

We split each graph into a labeled (20%) set and an unlabeled set of nodes (80%). Among the labeled nodes, half of them are used for training while the rest are used for validation. For the Polblogs dataset, since there are no feature attributes, we set the attribute matrix to an identity matrix.

5.1 Transductive Attack

As mentioned, due to the semi-supervised or transductive setting, the models are not regarded as fixed while attacking. After perturbing either features or edges, the model is retrained for evaluating the attack effectiveness. To verify the effectiveness of the attack, we select the nodes with different prediction scores. Specifically, we select in total 40 nodes which contain the 10 nodes with top scores, 10 nodes with the lowest scores and 20 randomly selected nodes. We compare the proposed IG-JSMA with several baselines including random attacks, FGSM, and nettack.

To evaluate the effectiveness of the attack, we calculate the classification margin. For a target node v, the classification

Dataset	Nodes	Features	Edges
CORA-ML	2708	1433	5429
Citeseer	3327	3703	4732
Polblogs	1490	-	19025

Table 1: Statistics of the datasets.

Dataset	CORA	Citeseer	Polblogs
JSMA	0.04	0.06	0.04
IG_JSMA	0.00	0.01	0.01

Table 2: The ratio of correctly classified nodes under JSMA and IG-JSMA attacks.

Dataset	no defense	w/ defense
CORA-ML	80.9± 0.6	80.7 ± 0.7
Citeseer	69.5 ± 0.7	69.6 ± 0.8

Table 3: Accuracy (%) of models on clean data with/without the proposed defense. We remove the outliers (i.e., accuracy $\leq 75\%/65\%$ for CORA-ML/Citeseer) due to the high variance.

margin of v is $Z_{v,c} - max_{c' \neq c} Z_{v,c'}$ where c is the ground truth class, $Z_{v,c}$ is the probability of class c given to the node v by the graph model. A lower classification margin indicates better attack performance. Figure 3 shows the classification margins of nodes after re-training the model on the modified graph. We found that IG-JSMA outperforms the baselines. More remarkably, IG-JSMA is quite stable as the classification margins have much less variance. Just as stated in [Zügner *et al.*, 2018], the vanilla gradient-based methods, such as FGSM are not able to capture the actual change of loss for discrete data.

To demonstrate the effectiveness of IG-JSMA, we also compare it with the original JSMA method where the saliency map is computed by the vanilla gradients.

Table 2 compares the ratio of correctly classified nodes after the JSMA and IG-JSMA attacks for 100 random sampled nodes. A lower value is better as this indicates more nodes are misclassified. We can see that IG-JSMA outperforms the JSMA attack. This shows that the saliency map computed by integrated gradients approximates the change patterns of the discrete features/edges better.

Figure 4 gives an intuitive example of this. For a target node in the graph, given a two-layer GCN model, the prediction of the target node only relies on its two-hop ego network. We define the importance of a feature/an edge as follows: For a target node v, The brute-force method to measure the importance of the nodes and edges is to remove one node or one edge at a time in the graph and check the change of prediction score of the target node.

Assume the prediction score for the winning class c is p_c. After removing the edge (i, j) by setting $\mathcal{A}_{ij}$ to 0, p_c changes to $p_c^{'}$. We define the importance of the edge by $\Delta_{p_c} = p_c^{'} - p_c$. To measure the importance of a node, we remove all the edges connected to the node and again see how the prediction scores change. These values can be regarded as the ground truth importance scores.

Both vanilla gradients and integrated gradients are approximations of the ground truth importance scores. The node importance can be approximated by the sum of the gradients of the prediction score w.r.t. all the features of the node as well as the gradients w.r.t. to the entries of the adjacency matrix.

In Figure 4, the node color represents the class of the node. Round nodes indicate positive importance scores while diamond nodes indicate negative importance scores. The node size indicates the value of the positive/negative importance score: a larger node means higher importance. Similarly, red edges are the edges which have positive importance scores while blue edges have negative importance scores and thicker edges correspond to more important edges in the graph. Finally, the pentagon represents the target node in the attack.

Figure 4a, 4b and 4c show the node importance results of brute-force, vanilla gradients and integrated gradients approach respectively (# of steps = 20). The vanilla gradients reveal little information about node/edge importance as almost all the edges have non-zero importance scores and it is difficult to see the relative node/edge influence. However, in the brute-force case, we notice that the majority of edges are not important for the target node. Compared to the brute-force method the vanilla gradients underestimate the importance of the nodes. The integrated gradients, as shown in Figure 4c is consistent with the ground truth produced by brute-force approach shown in Figure 4a. With only 20 steps along the path, integrated gradients provide accurate approximations for the importance scores. This shows the integrated gradients approach as effective as the brute-force technique when used to guide the adversarial attacks on graphs with discrete values.

5.2 Defense

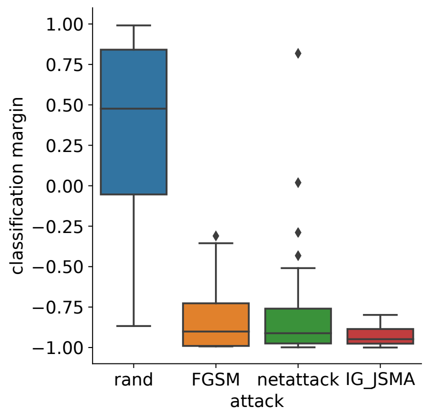
In the following, we study the effectiveness of the proposed defense technique under different settings. We use the CORA-ML and Citeseer datasets that have features for the nodes. We first evaluate whether the proposed defense method affects the performance of the model. Table 4 shows the accuracy of the GCN models with/without the defense.

We find that the proposed defense was cheap to use as the pre-processing of our defense method almost makes no negative impact on the performance of the GCN models. Moreover, the time overhead is negligible. Enabling defense on the GCNs models for the two datasets increases the run time of training by only 7.52s and 3.79s, respectively. Note that run time results are obtained using our non-optimized Python implementation.

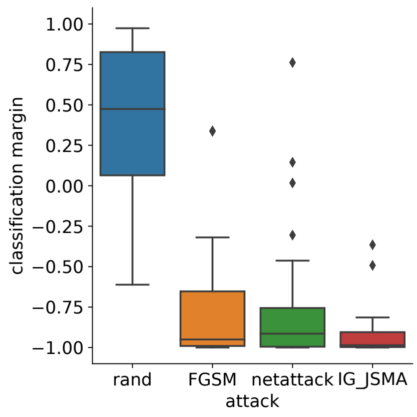
For different attacks, we then evaluate how the classification margins and accuracy of the attacked nodes change with/without the defense. As in the experiments of transductive attack, we select 40 nodes with different prediction scores. The statistics of the selected nodes are the followings: For CORA-ML and Citeseers datasets, we train the

Dataset/Attack	CM (w/ attack)		Accuracy (w/ attack)	
	w/ defense	no defense	w/ defense	no defense
CORA/FGSM	0.299 ± 0.741	-0.833 ± 0.210	0.625	0.025
CORA/JSMA	0.419 ± 0.567	-0.828 ± 0.225	0.775	0
CORA/nettack	0.242 ± 0.728	-0.839 ± 0.343	0.600	0.025
CORA/IG-JSMA	0.397 ± 0.553	-0.897 ± 0.114	0.750	0
Citeseer/FGSM	0.451 ± 0.489	-0.777 ± 0.279	0.825	0.025
Citeseer/JSMA	0.501 ± 0.531	-0.806 ± 0.186	0.775	0.05
Citeseer/nettack	0.421 ± 0.468	-0.787 ± 0.332	0.775	0.025
Citeseer/IG-JSMA	0.495 ± 0.507	-0.876 ± 0.186	0.800	0.025

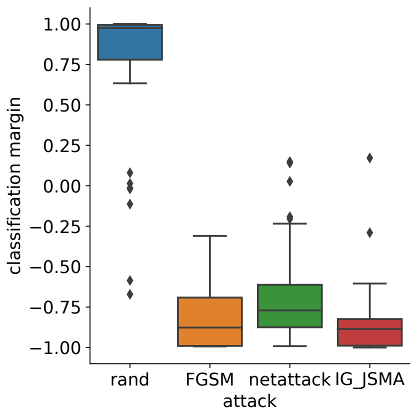
Table 4: Classification margins and error rates (%) for the GCN models with different attacks.



(a) CORA

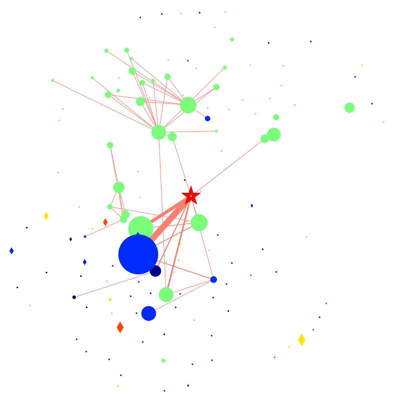


(b) Citeseer

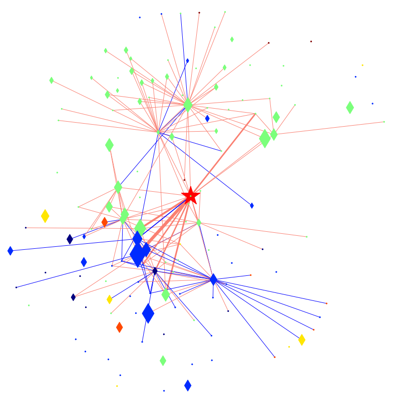


(c) polblogs

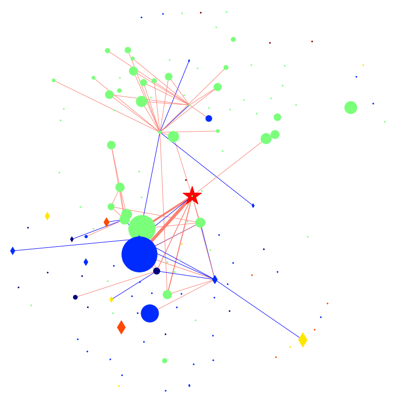
Figure 3: The classification margin under different attack techniques.



(a) Ground Truth



(b) Vanilla Gradients



(c) Integrated Gradients

Figure 4: The approximations of node/edge importance.

GCN models on the clean graphs. The selected nodes have classification margins of 0.693 ± 0.340 and 0.636 ± 0.419, respectively.

The results are given in Table 4. First of all, without defenses, most of the selected nodes are misclassified as the accuracy is always under 0.05 for any attacks. By enabling the defense approach, the accuracy can be significantly improved regardless of the attack methods. This, to some degree, shows that all the attack methods seek similar edges to attack and the proposed defense approach is attack-independent. Although a few nodes were still misclassified with the defense, the prediction confidence for their winning class is much lower since the classification margins increase. Therefore, it becomes harder to fool the users because manual checks are generally involved in predictions with low confidence. Overall, the proposed defense is effective even though we only remove the edges that connect nodes with Jaccard similarity score of 0.

6 Conclusions and Discussion

Graph neural networks (GNN) significantly improved the analytic performance on many types of graph data. However, like deep neural networks in other types of data, GNN suffers from robustness problems. In this paper, we gave insight into the robustness problem in graph convolutional networks (GCN). We proposed an integrated gradients based attack method that outperformed existing iterative and gradient-based techniques in terms of attack performance. We also analyzed attacks on GCN and revealed the robustness issue was rooted in the local aggregation in GCN. We give an effective defense method to improve the robustness of GCN models. We demonstrated the effectiveness and efficiency of our methods on benchmark data. Although we use the GCN model as a case in this paper, both the attack and defense principles are applicable to other variations of GNNs due to the fact that these models are also aggregation-based.

References

[Adamic and Glance, 2005] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd International Workshop on Link Discovery*, pages 36–43. ACM, 2005.

[Allen, 1970] Frances E Allen. Control flow analysis. In *ACM Sigplan Notices*, volume 5, pages 1–19. ACM, 1970.

[Bhagat *et al.*, 2011] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[Bojchevski and Günnemann, 2018] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of attributed graphs: Unsupervised inductive learning via ranking. *In ICLR'18*, 2018.

[Bruna *et al.*, 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Cao *et al.*, 2016] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *In AAAI'16*, 2016.

[Dai *et al.*, 2018] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *In ICML'18*, 2018.

[Edwards and Xie, 2016] Michael Edwards and Xianghua Xie. Graph based convolutional neural network. *In BMVC'16*, 2016.

[Goodfellow *et al.*, 2015] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *In ICLR'15*, 2015.

[Hart, 1989] Sergiu Hart. Shapley value. In *Game Theory*, pages 210–216. Springer, 1989.

[Henaff *et al.*, 2015] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *In NeurIPS'15*, 2015.

[Ian J. Goodfellow, 2014] Christian Szegedy Ian J. Goodfellow, Jonathon Shlens. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.06572*, 2014.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *In ICLR'17*, 2017.

[Lundberg and Lee, 2016] Scott Lundberg and Su-In Lee. An unexpected unity among methods for interpreting model predictions. *In NeurIPS'16*, 2016.

[McCallum *et al.*, 2000] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[Newman *et al.*, 2002] Mark EJ Newman, Duncan J Watts, and Steven H Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.

[Papernot and McDaniel, 2018] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*, 2018.

[Papernot *et al.*, 2016] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[Ron and Shamir, 2013] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[Shaham *et al.*, 2018] Uri Shaham, James Garritano, Yutaro Yamada, Ethan Weinberger, Alex Cloninger, Xiuyuan Cheng, Kelly Stanton, and Yuval Kluger. Defending against adversarial images using basis functions transformations. *arXiv preprint arXiv:1803.10840*, 2018.

[Sundararajan *et al.*, 2017] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *In ICML'17*, 2017.

[Tramèr *et al.*, 2018] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. *In ICLR'18*, 2018.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *In ICLR'18*, 2018.

[Wu *et al.*, 2019] Felix Wu, Tianyi Zhang, Amauri Holanda Souza Jr., Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153*, 2019.

[Xie *et al.*, 2018] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *In ICLR'18*, 2018.

[Xu *et al.*, 2013] Huan Xu, Yujiu Yang, Liangwei Wang, and Wenhuang Liu. Node classification in social network via a factor graph model. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 213–224. Springer, 2013.

[Xu *et al.*, 2018] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *In NDSS'18*, 2018.

[Yuan *et al.*, 2019] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2019.

[Zügner *et al.*, 2018] Daniel Zügner, Amir Akbarnejad, and Stephan Günnemann. Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2847–2856. ACM, 2018.