Graphfool: Targeted Label Adversarial Attack on Graph Embedding

Jinyin Chen, Xiang Lin, Dunjie Zhang, Wenrong Jiang, Guohan Huang, Hui Xiong, and Yun Xiang

Abstract—Deep learning is effective in graph analysis. It is widely applied in many related areas, such as link prediction, node classification, community detection, and graph classification etc. Graph embedding, which learns low-dimensional representations for vertices or edges in the graph, usually employs deep models to derive the embedding vector. However, these models are vulnerable. We envision that graph embedding methods based on deep models can be easily attacked using adversarial examples. Thus, in this paper, we propose Graphfool, a novel targeted label adversarial attack on graph embedding. It can generate adversarial graph to attack graph embedding methods via classifying boundary and gradient information in graph convolutional network (GCN). Specifically, we perform the following steps: 1),We first estimate the classification boundaries of different classes. 2), We calculate the minimal perturbation matrix to misclassify the attacked vertex according to the target classification boundary. 3), We modify the adjacency matrix according to the maximal absolute value of the disturbance matrix. This process is implemented iteratively. To the best of our knowledge, this is the first targeted label attack technique. The experiments on real-world graph networks demonstrate that Graphfool can derive better performance than state-of-art techniques. Compared with the second best algorithm, Graphfool can achieve an average improvement of 11.44% in attack success rate.

Index Terms—Graph embedding, targeted label attack, node classification, deep learning

✦

1 INTRODUCTION

G RAPH networks are applied in many real-world scenarios, such as social networks [1], traffic networks [2], communication networks [3], and biological networks [4], etc. The graph embedding, which can learn low-dimensional representations for vertices and edges, provides an effective and efficient way to analyze graphs [5]. It is used in many real-world applications, such as link prediction [6], [7], node classification [8], [9], and community detection [10], [11]. By converting a graph into a set of low-dimensional vectors, downstream graph analysis can be more efficient.

Generally, existing graph embedding techniques transform the graph into a similarity graph and calculate its eigenvectors, e.g., IsoMAP [12], Laplacian eigenmap [13], and local linear embedding [14], etc. As the recent development of machine learning, researchers try to apply the deep learning methods to graph embedding [15]. Graph neural network (GNN) is a semi-supervised graph embedding method It utilizes vertex attributes labels to train the model parameters [16], which extends the existing neural network methods into the graph domain [17]. Recently, many GNN models with excellent performance are proposed, such as GCN [18], GAE [19], and GN [20].

Deep learning techniques are already widely used in security areas, especially in computer vision [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31] are catching our attention recently. Adversarial attacks are triggered by carefully crafted adversarial perturbation added to the original image to fool a convolutional neural network (CNN) model [32]. Similarly, in the network

area, people care about personal privacy protection against graph analysis [33]. To protect personal privacy against excessive graph mining, individuals require the technique to actively manage their connections and hence, fool the graph analysis tools.

Inspired by adversarial attack in computer vision area [21], we propose a novel attack method, Graphfool, on graph embedding. Compared to the existing attack methods, Graphfool can achieve targeted label attack. In essence, we reverse the classification boundary-based optimization procedure of graph convolution network (GCN) model and treat the adjacency matrix as the learning hyperparameters. Our main contributions are summarized as follows:

- To the best of our knowledge, this is the first work targeting label adversarial attack on graph embedding. We propose a novel attack method, Graphfool, to invalidate graph embedding algorithms, such as GCN [18], DeepWalk [6], Node2vec [34], and GraphGAN [35]. The experimental results show Graphfool can trick the well-trained deep model with less average number of modified edges(AME) and better attack success rate(ASR).
- Unlike the existing graph embedding attack methods, Graphfool can construct targeted label attack by modifying certain edges of the original graph. Therefore, the attacked vertex can be misclassified as any specific class.
- Regarding the concealment of Graphfool attack and the processing power of local network, we propose disturbance-limited attack. It controls the perturbation of adjacency matrix to be close to the attacked vertex. Meanwhile, disturbance-limited attack can also reduce the complexity of Graphfool.

The rest of paper is organized as follows. Sec. 2 introduces existing graph attack methods. Sec. 3 discusses the basic theories and techniques of GCN. Sec. 4 describes in detail the Graphfool technique. Sec. 5 evaluates our Graphfool method on several real-world data sets. Sec. 6 concludes this paper and describes the

- J. Chen and Yun Xiang are with the Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou, 310023, China.
- W. Jiang is with the Institute of Big Data and Cyber Security, Zhejiang Police College, Hangzhou, 310053, China.
- J. Chen, X.Lin, D.Zhang, G. Huang, H. Xiong and Y. Xiang are with the College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China.

future works.

2 RELATED WORK

The related work can be generalized into two categories, graph embedding methods and graph attack methods, respectively.

2.1 Graph embedding methods

Recently, studies are focused on embedding graph into a low-dimensional vector space based on word2vec model [36]. They are called shallow graph embedding, such as DeepWalk [6], Node2vec [34], LINE [37], and GraRep [38], etc. DeepWalk [6] is the first model to learn language from a graph, which uses random walk to sample a sequence for each vertex and treats these generated sequences as sentences using the skip-gram mechanism. Tang et al. propose a novel graph embedding method LINE [37], which is a special case of DeepWalk with the window size of contexts set to one. Inspired by DeepWalk, Grover et al. propose an extension of DeepWalk, called Node2vec [34]. Node2vec employs a biased second-order random walk model to provide more flexibility for generating the context vertices. Cao et al. propose an embedding method called GraRep [38], which can preserve the node proximities by constructing different k-step probability transition matrices.

In recent years, many deep embedding methods are proposed, which are generally based on deep learning models, such as convolutional neural network (CNN) [39], [40] and generative adversarial networks model (GAN) [41]. Kipf et al. propose GCN [18] for semi-supervised node classification, which scales linearly in the number of graph vertices and learns hidden layer representations by encoding both local graph structure and features of vertices. Similarly, Pham et al. propose column network (CLN), which is a deep learning model for collective classification [42]. Compared with GCN, this model emphasises on relation learning, which can process multi-relational data. Monti et al. propose a unified framework MoNet [43], which extends the convolution operation to non-Euclidean domains. To strengthen the extraction of critical information, Wang et al. propose non-local neural network [44], which has the ability to capture more detailed information of graph structure. Wang et al. propose GraphGAN [35], which combines two classes of graph representation learning techniques.

2.2 Graph attack methods

Adversarial attacks on graph have a wide range of applications. In community detection, Nagaraja propose the first community deception method [45]. Waniek et al. propose disconnect internally, connect externally (DICE) [46], which ensconces community by randomly deleting edges between members and adding edges between members and non-members. Fionda et al. propose a novel community deception method based on the safeness [47]. In link prediction, Waniek et al. propose two strategies called closed-triad-removal and open-triad-creation [48], which solve the privacy problem caused by link prediction methods. Zhou et al. propose a method to attack local similarity and global similarity by deleting edges [49]. Fard et al. introduce a subgraph perturbation method to randomize the destination of an edge within subgraphs to protect sensitive edges [50]. They later propose a neighborhood randomization mechanism to probabilistically randomize the destination of an edge within a local neighborhood [33].

Similar to the adversarial attack in computer vision area, many graph attack techniques are proposed. Dai et al. demonstrate that GNNs are vulnerable by challenging a few edges. They propose reinforcement learning-based method [51] to attack graph embedding methods. Zügner et al. propose NETTACK, which is an adversarial attack on graph [52]. It generates adversarial graph based on GCN iteratively. Dai et al. propose RL-S2V [51]. It can learn to modify the graph structure with the prediction feedback from the target classifier. The modification is implemented by sequentially adding or dropping edges from the graph. Chen et al. propose FGA [53] and IGA [54]. FGA extracts the gradient of pairwise vertices based on the adversarial graph, and then selects the pair of vertices with maximum absolute edge gradient to realize the attack and update the adversarial graph. IGA is designed to mislead the link prediction methods. This method generates adversarial graphs to the target edge. Wang et al. propose Greedy-GAN [55], which inserts some fake vertices with corresponding fake features into the graph. The attacker is a greedy algorithm, which can generate adjacency and feature matrices of fake vertices. Sun et al. propose Opt-attack [56], which is based on projected gradient descent and attacks unsupervised vertex embedding algorithms, such as DeepWalk and LINE.

3 PRELIMINARIES

First, we define notations which are used throughout the paper. A graph can be represented by $G = (V, E)$, where V is the vertex set with $|V| = n$ and E is the edge set. The definition of symbols are listed in the TABLE 1.

TABLE 1: The definitions of symbols.

Symbol	Definition
$G = (V, E)$	input original network with nodes V and edges E
$\bar{G} = (V, \bar{E}, M)$	disturbance network with vertices V, edges $\bar{E}$ and weight M
$\tilde{G} = (V, \tilde{E})$	the adjacency matrix of original network G
$\bar{A}$	the adjacency matrix added self-connections
N	the number of vertices of network G
I_N	identity matrix
Z	the output of the GCN model
σ	$Relu$ active function
W_i	the weight matrices of GCN model
X	the feature matrix of all vertices
C	the number of feature dimensions for each vertex
H	the number of feature maps for hidden layer
F	the number of class for vertices in the network
L	the loss function of the GCN model
V_L	the set of vertices with labels
Y	the real label confidence list
η	learning rate
θ	the sign function

3.1 GCN model

GCN applies the traditional convolutional neural networks to the graph domain. The model uses an efficient layer-wise propagation rule based on the fist-order approximation of spectral convolutions on graphs. The GCN model can achieved good performance in semi-supervised node classification. In this paper, GCN model is used as the adversarial graph generator to trick other node classification algorithms.

Specifically, we employ a two-layer GCN model with softmax classifier. Its forward model is defined as follows.

$$Z = softmax(\bar{A}\sigma(\bar{A}XW_0)W_1), \qquad (1)$$

where $X \in R^{N \times C}$ is the feature matrix of all vertices, $\bar{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$, $\tilde{A} = A + I_N$, $\tilde{D} = \sum_j \tilde{A}_{ij}$, A is the adjacency matrix, and I_N is the identity matrix. Therefore, $\tilde{A}$ is the adjacency matrix of the graph with self-connections, $\tilde{D}$ is a degree matrix of $\tilde{A}$, $W_0 \in R^{C \times H}$ is the input-to-hidden weight matrix with the hidden layer of H feature maps, and $W_1 \in R^{H \times |F|}$ is the hidden-to-output weight matrix. σ denotes the $Relu$ active function. W_0 and W_1 can be derived using gradient descent training.

The loss function is defined as the cross-entropy error over all labeled examples.

$$L = -\sum_{l=1}^{|V_L|} \sum_{k=1}^{|F|} Y_{lk} \ln(Z_{lk}), \qquad (2)$$

where V_L is the set of vertices with labels, $|F|$ is the dimension of the output features which is equal to the number of classes, Z is the output feature.

In training process, the GCN model uses the classical gradient descent to optimize the parameters.

$$W_i^{m+1} = W_i^m - \eta \frac{\partial L}{\partial W_i^m}, \qquad (3)$$

where η is the learning rate. During each iteration, the weights $W_i, i \in \{0, 1\}$ are updated.

The GCN model combines vertex features and graph structures using graph convolution. The features of labeled vertices can be mixed with those of unlabeled vertices according to adjacency matrix. Therefore, GCN model can achieve better performance on certain benchmarks [18].

4 THE GRAPTHFOOL METHOD

Based on two-layered GCN model, Graphfool is designed to achieve targeted label attack by adding or removing few edges of original graph. In this section, we describe the Graphfool in detail.

4.1 The framework of graphfool

GCN model with proper training can achieve good performance in node classification. However, if we modify its relationship with other vertices, the node classification result can be completely different. In that case, Graphfool is a adversarial attack technique targeting graph embedding. It consists of two parts, which are adversarial graph generation and adversarial attack, respectively. The framework of Graphfool is shown in Fig. 1.

- **Adversarial graph generation:** First, we use the original graph and certain labeled vertices to train the GCN model. Then, for each attacked vertex, we use an iterative linearization method to generate minimal adjacency matrix perturbations sufficient to change node classification results. Thus, we derive the final adversarial graph.
- **Adversarial attack:** We use the generated adversarial graph to prevent the attacked vertex from the GCN model. Since GCN has excellent generalization ability, the adversarial attack can be implemented on many other graph embedding methods, i.e., the perturbation generated by GCN is universal and the attack has strong transferability.

In adversarial graph generation stage, for a graph dataset, we first derive the node classification results based on adjacency
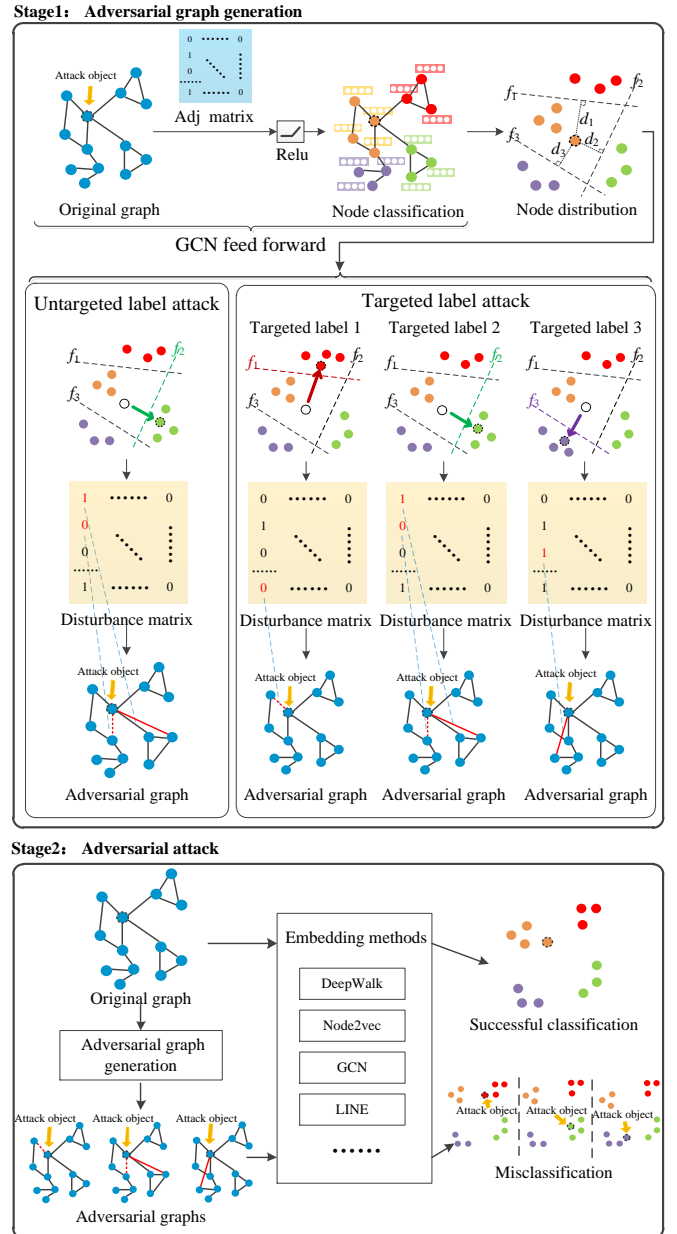

Fig. 1: The frame work of Graphfool.

matrix and the trained GCN model. Then for the attacked vertex, we can calculate the classification boundaries (f_1, f_2 and f_3). We can also derive the minimal distance d_1, d_2 and d_3, where $d_1 > d_3 > d_2$. In untargeted label attack, we select the most vulnerable label whose classification boundary (f_2) is closest to the attacked node. Then the disturbance matrix is calculated and the adversarial graph can be derived by adding/deleting the edges. For the targeted label attack, it is generally the same as the untargeted label attack. The only difference is that in targeted label attack we should assign the corresponding label of the attacked vertex. In adversarial attack stage, we use the adversarial graph to test the attack effect of Graphfool using various network embedding algorithms.

4.2 Adversarial graph generator

In Sec. 3, we introduce the structure and training processing of a two-layered GCN. Based on this model, Graphfool generates the adversarial graph.

4.2.1 Disturbance of adjacent matrix

For the GCN model, the classifier converts the feature of each vertex to an F-dimension vector where F is the number of classes. For each GCN layer, the adjacency matrix is used to capture the structural information of the graph. Therefore, a GCN classifier can be defined as $f : R^C \to R^F$. For vertex i, its classification is implemented using the following mapping equation.

$$\hat{k}(x_i, A) = argmax f_k(x_i, A),\qquad(4)$$

where x_i is the feature vector of vertex i, A is the graph adjacency matrix, $f_k(x_i, A)$ is the output of the k-th class of $f(x_i, A)$.

According to Eq. 4, if we want to influence the node classification result of vertex i, we can change x_i and A. However, in actual graph networks, such as social networks and communication networks, the feature vectors of vertices are predetermined and thus, difficult to change. However, changing the relationship between the target vertex and other vertices is easier and more concealed. Therefore, Graphfool conducts attacks by changing the adjacency matrix of the graph. In that case, Eq. 4 can be modified according to the following equation.

$$\hat{k}(A) = argmax f_k^i(A).\qquad(5)$$

To simplify the attack problem, we first assume an linear classifier $f^i(A)$, i.e., $f^i(A) = W^T A + b$, where W and b are trained classifier parameters. The goal is to make the target vertex misclassified by adding minimal perturbation on A. Therefore, the attack problem can be modeled as follows.

$$argmin\|R\|_2$$
$$s.t. \exists k : w_k^T(A_0 + R) + b_k \geq w_{\hat{k}(A_0)}^T(A_0 + R) + b_{\hat{k}(A_0)},\qquad(6)$$

where A_0 is the original adjacency matrix of the graph, R is perturbation matrix, and w_k is the k-th column of W. Thus, for the k-th class, we can construct class boundary function between $f_k^i(A)$ and $\hat{k}(A)$ using the following equation.

$$f_k^i(A) - \hat{k}(A) = 0(k \neq \hat{k}(A_0)).\qquad(7)$$

For original adjacency matrix A_0, the minimal perturbation $R(A_0)$ corresponds to the minimal distance between A_0 to these boundaries. It can be calculated using the following equation.

$$R(A_0) = \underset{k \neq \hat{k}(A_0)}{argmin} \frac{|f_k^i(A_0) - \hat{k}(A_0)|(w_k^T(A_0) - w_{\hat{k}(A_0)}^T(A_0))}{\|w_k^T(A_0) - w_{\hat{k}(A_0)}^T(A_0)\|_2^2}.\qquad(8)$$

For general non-linear classifiers, we approximate the node classification boundary function using the first-order Taylor expansion of each classifier.

$$f_k^i(A_0) - \hat{k}(A_0) + \triangledown f_k^i(A_0)^T A - \triangledown \hat{k}(A_0)^T A = 0(k \neq \hat{k}(A_0)).\qquad(9)$$

$R(A_0)$ can be derived similarly using Eq. 8.

To guide the change of A_0, we need to calculate the value of $R(A_0)$. Since the adjacent matrix of an undirected graph is symmetric, we symmetrize $R(A_0)$ to obtain $\hat{R}(A_0)$ as shown in the following equation.

$$\hat{R}_{ij}(A_0) = \hat{R}_{ji}(A_0) = \begin{cases} \frac{R_{ij}(A_0) + R_{ji}(A_0)}{2} & i \neq j \\ 0 & i = j. \end{cases}\qquad(10)$$

In Eq. 10, the elements in $\hat{R}(A_0)$ have continuous values. For a specific element $\hat{R}_{ij}(A_0)$ in $\hat{R}(A_0)$, its positive/negative value indicates that we should adding/deleting the edge between the pair of vertices (v_i, v_j). The larger value of $\left|\hat{R}_{ij}(A_0)\right|$ indicates the added/deleted edge can influence the classification result of the target node more significantly.

4.2.2 Adversarial graph generator

In this section, we propose an adversarial graph generator based on our adjacent matrix disturbance generation technique. We modify one edge during each iteration and the generation process runs for K iterations. To avoid excessive perturbation, the total iteration number K should be limited. For a given graph, its average degree represents the sparseness of the graph. Therefore, we limit K based on the graph average degree. The generation process is iterative.

The flows of the generation process is shown in Alg. 1.

1) We construct $\hat{R}(A_{h-1})$ based on A_{h-1}. Using Eq. 7 and Eq. 8, we calculate the classification surface closest to A_{h-1}, its target label l_{h-1}, and the perturbation $\hat{R}(A_{h-1})$ with $A_0 = A$.

2) We select perturbation edges. Based on $\hat{R}(A_{h-1})$, we select a pair of vertices (v_i, v_j) which has maximal absolute value $\hat{R}_{ij}(A_{h-1})$. It should be noted that if $\hat{R}_{ij}(A_{h-1})$ is positive/negative and v_i and v_j are connected/disconnected in A_{h-1}, we cannot further add or delete the edge between this pair of vertices. Hence, we just ignore such pairs of vertices in the process and continue.

3) We update the adjacency matrix A_h. We modify the $(h-1)$-th adjacency matrix with selected pair of vertices and generate a new adversarial graph. The h-th adjacency matrix is calculated using the following equation.

$$A_{h\_ij} = A_{h-1\_ij} + \theta(\hat{R}_{ij}(A_{h-1}))\qquad(11)$$

where A_{h_ij} and A_{h-1_ij} are the elements of A_h and A_{h-1} and $\theta(\hat{R}_{ij}(A_{h-1}))$ is the signed value of the pair of elements with maximal absolute values in perturbation matrix $\hat{R}(A_{h-1})$.

4.3 Targeted label attack

In the previous sections, we introduce our technique to generate adversarial graph based on classification boundary and minimal disturbance of adjacency matrix. Moreover, Graphfool can also perform targeted label attack. Thus, we define a new goal, i.e., for vertex i, we want to misclassify it into class $l(l \neq \hat{k}(A_0))$. Thus, the attack problem can be generalized as follow.

$$argmin\|R\|_2$$
$$s.t. \forall k(k \neq l) : w_l^T(A_0 + R) + b_l \geq w_k^T(A_0 + R) + b_k.\qquad(12)$$

In general, during iteration h, when $f_l^i(A_{h-1}) \neq \hat{k}(A_{h-1})$, the goal is to add the minimal perturbation R to make vertex

Algorithm 1: Adversarial graph generator via GCN

Input: Original graph G, number of iteration K.
Output: The adversarial graph $\tilde{G}$.

1 Train the GCN model on original graph G to obtain
  model parameters W via Eq. 3.
2 Initialize the adjacency matrix of the adversarial graph by
  $A_0 = A$;
3 **for** $h = 1$ *to* K **do**
4   Construct $\hat{R}(A_{h-1})$ based on A_{h-1};
5   Select the perturbation edges which has maximum
      absolute value in $\hat{R}(A_{h-1})$;
6   Update the adjacency matrix A_h according to
7     $A_{h\_ij} = A_{h-1\_ij} + \theta(\hat{R}_{ij}(A_{h-1}))$;
8 **Return** the adversarial graph $\tilde{G}$, with the adversarial
  adjacency matrix A_K

i crossover the classification boundary $f_l^i(A) - \hat{k}(A) = 0$. This minimum perturbation can be derived using the following equation.

$$R(A_{h-1}) = \frac{|f_l^i(A_{h-1}) - \hat{k}(A_{h-1})|(w_l^T(A_{h-1}) - w_{\hat{k}(A_{h-1})}^T(A_{h-1}))}{\|w_l^T(A_{h-1}) - w_{\hat{k}(A_{h-1})}^T(A_{h-1})\|_2^2}. \tag{13}$$

4.4 Transferring adversarial attack

Besides the baseline GCN model, we can also use the modified adjacency matrix to attack other node classification methods. Most node classification algorithms rely upon the connection relationship between vertices. Vertices with strong relationship are typically divided into the same class. Therefore, these algorithms have similar decision boundaries. In that case, the GCN based adversarial attack can also be effective on many other node classification methods. In the experiment section, we use the adversarial graph generated by Graphfool to attack other node classification algorithms. The experimental results show strong transferability of our technique.

5 EXPERIMENTAL RESULTS

To validate our technique, we test it for both untargeted label attack and targeted label attack. Moreover, to demonstrate the concealment of Graphfool, we also perform the single-edge attack and disturbance-limited attack.

5.1 Experimental setup

Our experiments are performed on a machine with i7-7700K 3.5GHzx8 (CPU), TITAN Xp 12GiB (GPU), 16GBx4 memory (DDR4), and Ubuntu 16.04 (OS).

5.1.1 Datasets

In the experiment, we test different techniques on three datasets, which are Cora, Citeseer, and Pol.blogs, respectively. Their statistics are provided in TABLE 2.

• **Cora:** The Cora dataset consists of 2708 scientific publications categorized into seven classes [57]. The citation graph consists of 5429 edges. Each edge represents the citation relationship.

TABLE 2: The basic statistics of three network datasets.

Datasets	Nodes	Links	Classes
Cora	2708	5429	7
Citeseer	3312	4732	6
Pol.blogs	1490	19090	2

• **Citeseer:** The Citeseer dataset consists of 3312 scientific publications categorized into six classes [57]. The citation network consists of 4732 edges.

• **Pol.blogs:** The Pol.blogs dataset [58] shows the political divide of blog. It contains 1490 vertices and 19090 edges. The vertices are divided into two classes.

5.1.2 Evaluation criteria

In this section, we introduce the evaluation criteria for the comparing methods.

• Attack success rate (ASR): We attack a series of vertices in the graph and the ASR is defined as follows.

$$ASR = \frac{N_s}{N} \times 100\%, \tag{14}$$

where N_s is the number of misclassified vertices and N is the total number of nodes being attacked.

• The average number of modified edges (AME): To attack the target vertex, we add/delete edges between vertices. The modifications should be minor and undetectable. Thus, the method with smaller AME is better. The AME equation is defined as follows.

$$AME = \frac{1}{N} \sum_{i=1}^{N} L_i, \tag{15}$$

where N is the total number of nodes being attacked and L_i is the number of modified edges for vertex i.

5.1.3 Comparing methods

To validate our Graphfool technique, we compre it with five state-of-art graph embedding attack techniques shown as follows.

• **DICE** [46]: For each iteration, DICE removes b edges of the target vertex randomly and then add edges between the target vertex and $K - b$ vertices of different classes.

• **NETTACK** [52]: NETTACK selects key edges based on pivotal data characteristics, e.g., degree distribution. Then it uses two scoring functions to measure the change in the confidence value. After modifying an edge, it uses the feature with highest score to update the adversarial graph according to the confidence value change.

• **FGA** [53]: FGA extracts the gradient of pairwise vertices based on the original graph. Then it selects the pair of vertices with the maximal absolute edge gradient to update the adversarial graph.

• **RL-S2V** [51]: RL-S2V is a hierarchical reinforcement learning based attack method. It learns a Q-function parameterized by S2V to perform a generalized attack.

• **GraArgmax** [51]: GradArgmax calculates the gradient of adjacency matrix based on the output of each hidden layers and the loss function. Then it adopts greedy algorithm to select the pairs of vertices to attack the original graph.

TABLE 3: The attack effects obtained by different attack methods toward varous network embedding methods on multiple datasets.

Datasets	Model	ASR(%)								AME							
		Graphfool			BASELINE					Graphfool			BASELINE				
		unlimited	direct	indirect	FGA	NATTECK	RL-S2V	GradArgmax	DICE	unlimited	direct	indirect	FGA	NATTECK	RL-S2V	GradArgmax	DICE
Cora	GCN	**100**	100	89.66	100	92.87	93.83	90.32	54.95	**1.78**	1.92	5.60	2.54	6.09	6.65	7.02	9.13
	GraRep	**100**	100	87.53	100	97.22	100	95.35	89.09	**5.43**	5.57	9.41	5.56	5.94	5.96	7.29	7.37
	Deepwalk	**100**	100	76.70	100	94.06	95.40	90.95	93.52	5.57	**5.13**	11.07	5.61	7.24	6.92	7.89	7.20
	node2vec	**100**	100	77.57	100	97.29	100	96.24	89.09	**4.94**	4.94	10.21	5.66	6.75	6.14	7.44	7.37
	LINE	**100**	100	85.72	100	96.34	95.51	89.98	88.99	**5.38**	5.47	9.41	5.64	7.02	6.96	8.10	7.66
	GraphGAN	**100**	100	74.77	100	92.26	95.56	88.24	84.55	5.63	**5.05**	10.65	5.65	8.82	6.90	8.02	7.96
	Average	**100**	100	81.99	100	95.01	96.72	91.85	83.37	4.79	**4.68**	9.39	5.11	6.98	6.59	7.63	7.78
Citeseer	GCN	**100**	100	93.55	100	87.50	91.84	88.33	70.37	**1.42**	1.32	4.65	3.52	6.88	5.86	6.62	9.87
	GraRep	**100**	100	97.63	100	94.28	94.44	89.23	93.22	**5.27**	5.89	8.25	5.32	6.51	6.94	6.89	7.56
	Deepwalk	**100**	100	81.03	100	96.96	94.34	90.96	93.44	**5.38**	5.74	5.38	5.68	7.06	6.56	6.90	7.08
	node2vec	**100**	100	79.31	100	93.93	93.88	89.09	91.38	**4.48**	4.75	9.33	5.62	6.34	7.02	6.80	7.13
	LINE	**100**	100	98.36	100	95.82	93.88	86.66	96.72	**5.76**	6.25	7.95	5.88	6.02	6.80	7.26	7.21
	GraphGAN	**100**	100	77.19	100	92.06	94.12	85.32	88.24	**5.54**	5.58	9.25	5.91	7.42	7.04	7.79	8.26
	Average	**100**	100	87.85	100	93.43	93.75	88.27	88.90	**4.64**	4.92	5.16	7.46	6.71	6.70	7.04	7.98
Pol.blogs	GCN	**95.25**	93.62	21.27	87.87	82.97	82.98	78.34	50.27	**4.92**	5.91	17.49	8.42	11.89	9.09	10.21	11.85
	GraRep	**94.87**	94.87	5.28	83.88	79.91	79.17	75.69	61.06	**7.21**	7.56	19.36	9.58	10.48	11.02	11.88	14.22
	Deepwalk	95.25	**97.87**	4.26	84.26	75.41	78.72	76.25	64.52	6.26	**6.01**	18.96	9.84	10.06	10.09	11.04	12.35
	node2vec	**97.87**	97.87	6.38	84.34	78.32	79.17	73.32	67.89	7.13	**7.04**	18.72	9.72	10.58	10.89	11.62	14.86
	LINE	**96.52**	95.87	5.47	85.25	76.35	75.00	70.26	66.74	**6.84**	7.12	19.56	9.90	10.26	11.30	11.73	12.82
	GraphGAN	**95.74**	95.74	8.51	81.21	72.26	79.17	72.02	64.58	**7.61**	7.85	18.85	9.41	11.08	11.55	12.04	12.26
	Average	95.91	95.97	8.52	84.47	77.54	79.02	74.31	62.51	6.66	6.92	18.82	9.48	10.73	10.66	11.42	13.06

5.2 Evaluation results

In this section, we evaluate our Graphfool method on three real-world datasets. The techniques are tested with untargeted label attack, targeted label attack, single-edge attack, and disturbance-limited attack.

5.2.1 Untargeted label attack

First, we randomly select 20 vertices in each category to form the set of attacked vertices. To analyze the relationship between modified edges and attacked vertices, we consider direct, indirect, and unlimited attacks, respectively [53].

- **Direct attack:** This attack method only attacks the edges directly connected to the attacked vertex.
- **Indirect attack:** This attack method attacks the edges not directly connected to the attacked vertex.
- **Unlimited attack:** This attack method can remove or add edges between any pair of vertices.

Without loss of generality, we assume the number of modified edges less than 20 for each attack. The attack results are shown in TABLE 3. For the unlimited attack case, Graphfool outperforms the other attack methods in most of the cases, in terms of higher ASR and lower AML. In Cora and Citeseer, for the unlimited case, both Graphfool and FGA achieve 100% ASR. However, Graphfool has significantly smaller AME, which implies that the adversarial graph generated by Graphfool has much less perturbation. For Pol.blogs, our technique can get 96.03% ASR and 6.48 AME. The performance deteriorates a bit because of the denseness of the network (with average degree close to 25.6). However, our techinique still outperform other any other comparing techniques.

For our Graphfool technique, the unlimited and direct attacks have relatively close performance. However, the indirect attack has relatively worse performance. This demonstrates that direct attacks are typically more effective than indirect ones. In graphfool, we generate a adversarial graph through the GCN model. The generated adversarial graph is then used as input to attack other node classification algorithms. TABLE 3 shows the attacking results. Although the adversarial graph is generated based on GCN model, it can also achieve excellent attack performance for other node classification algorithms. This demonstrated the strong transferability of our technique. Moreover, our adversarial graph achieves less AME on GCN model. This phenomenon indicates that the adversarial graph can capture the vulnerability of GCN model more accurately.

Moreover, for the datasets of Cora and Citeseer, where the graphs are relatively sparse, indirect attacks can achieve relatively high attack performance, which is similar to DICE of direct attack. This implies that we may be able to change the edges far away from the target nodes to perform the attack. In other words, the local structure of these vertices is not necessarily destroyed, making the attack harder to detect. For the dataset of Pol.Blogs, since it is very dense, the performance of graphfool is limited, especially for indirect attack.

5.2.2 Targeted label attack

In this section, we perform targeted label attack for the Cora and Citeseer dataset, which have more than 2 labels. For each dataset, we also randomly select 20 vertices in each category to form the set of attacked vertices. The specific attack strategy of Graphfool is the unlimited attack.

The results of the targeted label attack are shown in TABLE 4 and TABLE 5. Compared with untargeted label attack, the ASR of targeted label attack has a significant decline, while its AME also increases. It implies that the targeted label attack is not as effective as untargeted label attack. The reason is that due to the explicit directionality of targeted label attack, its generated disturbance has more conditional constraints. These constraints can increase the cost of ASR and AME. ***(what does these sentences mean? Your method has random performance on different labels?? Why??)In addition, in the Cora dataset, when the targeted label is 3, the method of targeted label attack could achieve higher ASR and lower AME, which means that other vertices are more likely to disguise as the vertices with label 3. And the same phenomenon also appears on Citeseer dataset. We can conclude that the vertices with different labels have different properties, which leads to different ASR and AML under different targeted labels on targeted label attack.

5.2.3 Single-edge attack

In computer vision area, other than the success rate of adversarial attack, minimizing its disturbance is also an important goal [21]. Similarly, in graph based attacks, if an attack method could get close performance with fewer modified edges, it has a better attack concealment. In this section, to evaluate the concealment of Graphfool, we design a single-edge attack experiment for these three datasets. In this experiment, each attack method could only change one edge of the original graph to generate adversarial

TABLE 4: Targeted label attack on Cora.

Metris	Model	Targeted Label						
		0	1	2	3	4	5	6
ASR(%)	GCN	73.96	70.79	66.67	98.86	84.27	79.12	66.30
	GraRep	72.92	68.54	67.82	96.59	82.02	79.12	65.22
	DeepWalk	73.96	67.42	66.67	95.45	84.27	78.02	65.22
	Node2vec	72.92	70.79	65.52	94.32	83.15	76.92	61.53
	LINE	70.83	68.54	65.52	96.59	85.39	76.92	64.13
	GraphGAN	72.92	71.91	66.67	94.32	85.39	79.12	61.53
	Average	72.92	69.67	66.48	96.02	84.08	78.20	63.99
AME	GCN	6.28	6.91	7.56	2.00	4.57	5.47	8.03
	GraRep	7.03	7.41	7.58	3.25	5.03	6.31	8.46
	DeepWalk	6.95	7.28	7.69	3.14	5.08	6.16	8.53
	Node2vec	7.03	7.37	7.63	3.26	5.11	6.28	8.23
	LINE	7.26	7.37	7.73	3.23	4.93	6.34	8.37
	GraphGAN	6.96	7.45	7.60	2.99	4.72	6.42	8.23
	Average	6.75	7.30	7.63	2.98	4.91	6.16	8.31

TABLE 5: Targeted label attack on Citeseer.

Metris	Model	Targeted Label					
		0	1	2	3	4	5
ASR(%)	GCN	83.78	78.46	74.19	92.19	85.16	88.52
	GraRep	85.14	78.46	79.03	90.63	83.33	90.16
	DeepWalk	82.43	78.46	80.65	90.63	87.04	86.89
	Node2vec	85.14	76.92	77.41	89.06	81.48	85.25
	LINE	82.43	80.00	75.81	93.75	87.04	86.89
	GraphGAN	81.08	76.92	72.58	89.06	81.48	83.60
	Average	83.33	78.20	76.61	90.89	84.26	86.89
AME	GCN	4.55	5.72	5.56	3.11	4.39	3.77
	GraRep	6.44	7.13	6.92	4.78	5.59	4.33
	DeepWalk	6.64	6.92	6.69	4.63	5.61	4.30
	Node2vec	6.70	7.02	6.95	4.46	5.76	4.33
	LINE	6.64	6.88	6.71	4.48	5.63	4.18
	GraphGAN	6.53	7.12	6.92	4.28	5.69	4.26
	Average	6.25	6.80	6.62	4.29	5.45	4.20

graph. In other words, we set the AME of all attack methods to 1. The set of attacked vertices is the same as the one in the untargeted label attack experiment. We also experiment with direct, indirect, and unlimited attacks, respectively.

The results of single-edge attack are shown in TABLE 6. In general, single-edge attack is a special case of untargeted label attack. Therefore, the results in TABLE 6 are consistent with those in TABLE 3. For the unlimited and direct cases, Graphfool still outperforms most of the other attack methods. FGA is the closest algorithm, with 2% to 5% lower ASR on average. Moreover, for Cora and Citeseer, both unlimited Graphfool and direct Graphfool achieve approximately 50% ASR, but in Pol.blogs, which is much denser than the other two graphs, these two methods only get 18.16% ASR.

In addition, the ASRs of Graphfool, FGA, and NETTECK in GCN model are higher than those in other node classification algorithms. This is because of the following reasons. First, Graphfool, FGA, and NETTECK are GCN-based graph attack methods. Second, for other node classification algorithms, they all have certain randomness. Single-edge attack only changes one edge in original graph. It may affect more significantly during random process, thus reducing the ASR for these algorithms.

5.2.4 Disturbance-limited attack

To improve the attack concealment, one may want to limit the perturbation to a certain range. This is called disturbance-limited attack. In disturbance-limited attack, attackers only change the edges of the subgraph which is composed by the attacked vertex and its neighbors. For each attacked vertex, we first calculate its k-order neighbors (vertices whose link distance to the attacked vertex are less than k) and construct the subgraph accordingly. Then we perform graph attack in this subgraph using unlimited

Graphfool attack method. In this experiment, we set the range of k from 1 to 5. The value k is corresponding to the size of the modifiable subgraph. To quantify the size of these subgraphs in the original graph, for each dataset, we calculate the average ratio of the number of vertices in the k-th order subgraph of each attacked vertex to the corresponding number of vertices in original graph. The results are shown in TABLE 7.

TABLE 8, TABLE 9, and TABLE 10 show the results of disturbance-limited attack on three datasets. With the increase of neighbor order, ASR in every node classification algorithm increase significantly, while AME is monotonously decreasing. This is consistent with the general idea that when the size of the constructed subgraph gets larger, the Graphfool attack is more likely to succeed. Moreover, for the sparse dataset Cora and Citeseer, when the neighbor order of each attacked vertex is 5, the average sizes of subgraphs are only 14.22% and 2.87%. However, Graphfool attack also could get 67.69% and 49.18% average ASR, 6.61 and 9.04 average AME, respectively. This implies that even if we limit the disturbance to a small local subgraph of the attacked vertex, we can still perform effective attacks. However, For the dense graph Pol.blogs, when the neighbor order reaches 5, the average size of subgraphs becomes 81.88%. It covers most of the original graph and the results are close to the unlimited case in TABLE 3.

6 CONCLUSION

In this paper, we propose Graphfool, a graph attack technique to mislead the node classification algorithms. This method uses two-layer GCN model as the attack model, constructs the decision boundary according to the classification results, and implements attack by modifying edge relationship of adjacency matrix. The experiments suggest that Graphfool can generate smaller perturbation and obtain the higher attack success rate simultaneously, which demonstrate the vulnerability of current graph embedding algorithms.

REFERENCES

[1] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, no. 5916, pp. 892–895, 2009.
[2] V. Latora and M. Marchiori, "Is the boston subway a small-world network?" *Physica A Statistical Mechanics & Its Applications*, vol. 314, no. 1, pp. 109–113, 2002.
[3] M. Kistler, M. Perrone, and F. Petrini, "Cell multiprocessor communication network: Built for speed," *IEEE Micro*, vol. 26, no. 3, pp. 10–23, 2006.
[4] M. JM and S. RV, "Small world patterns in food webs." *Journal of Theoretical Biology*, vol. 214, no. 3, pp. 405–412, 2002.
[5] R. Hong, Y. He, L. Wu, Y. Ge, and X. Wu, "Deep attributed network embedding by preserving structure and attribute information," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2019.
[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.
[7] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media," in *SDM*, 2017.
[8] J. Tang, M. Qu, and Q. Mei, "Pte: Predictive text embedding through large-scale heterogeneous text networks," pp. 1165–1174, 2015.
[9] S. Wang, J. Tang, C. Aggarwal, and H. Liu, "Linked document embedding for classification," pp. 115–124, 2016.
[10] F. Tian, B. Gao, Q. Cui, E. Chen, and T. Y. Liu, "Learning deep representations for graph clustering," in *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014, pp. 1293–1299.
[11] K. Allab, L. Labiod, and M. Nadif, "A semi-nmf-pca unified framework for data clustering," *IEEE Transactions on Knowledge & Data Engineering*, vol. 29, no. 1, pp. 2–16, 2016.

TABLE 6: Single-edge attack on three datasets.

Datasets	Model	ASR(%)							
		Graphfool			BASELINE				
		unlimited	direct	indirect	FGA	NATTECK	RL-S2V	GradArgmax	DICE
Cora	GCN	70.63	**71.71**	39.35	65.84	68.78	20.38	17.36	10.58
	GraRep	**43.26**	43.26	5.08	38.48	32.35	18.86	16.27	8.08
	Deepwalk	**45.71**	45.71	4.27	41.54	29.36	15.42	14.06	10.58
	node2vec	**43.26**	43.26	4.27	42.73	31.54	16.03	14.06	10.58
	LINE	42.73	**45.71**	5.08	42.73	28.45	17.65	16.27	9.27
	GraphGAN	**44.56**	43.26	3.58	39.67	26.76	16.03	15.75	10.58
	Average	48.36	48.82	10.27	45.17	36.21	17.40	15.63	9.95
Citeseer	GCN	**85.74**	84.62	39.33	75.77	66.37	29.62	23.58	12.46
	GraRep	**49.56**	49.56	7.23	45.32	36.53	25.76	21.57	10.37
	Deepwalk	**47.32**	47.32	6.59	42.58	34.87	22.43	20.64	12.46
	node2vec	**46.59**	46.59	5.48	44.76	32.64	21.57	23.58	10.37
	LINE	48.23	**49.56**	5.93	43.41	35.29	26.38	24.61	11.48
	GraphGAN	**46.59**	46.59	6.59	42.58	34.02	23.58	21.57	9.75
	Average	54.01	**54.04**	11.86	49.07	39.95	24.89	22.59	11.15
Pol.blogs	GCN	**47.05**	47.05	12.43	44.91	33.61	6.53	7.20	5.80
	GraRep	8.96	8.96	2.50	**9.89**	6.53	7.20	5.80	4.06
	Deepwalk	**10.53**	10.53	0.00	9.89	8.96	5.80	0.00	6.53
	node2vec	12.42	12.42	0.00	**13.16**	12.42	3.81	0.00	5.80
	LINE	**15.21**	15.21	2.50	8.96	13.16	0.00	3.81	2.50
	GraphGAN	**14.81**	14.81	0.00	11.72	14.02	3.81	0.00	2.50
	Average	**18.16**	18.16	2.91	16.42	14.78	4.53	2.80	4.53

TABLE 7: Attacked vertices' average ratio of different neighbor orders(%).

Datasets	Neighbor order				
	1	2	3	4	5
Cora	0.12	0.36	1.79	5.22	14.22
Citeseer	0.01	0.21	0.59	1.48	2.87
Pol.blogs	1.41	29.46	66.38	80.20	81.88

TABLE 8: Disturbance-limited attack on Cora.

Metris	Model	Neighbor order				
		1	2	3	4	5
ASR(%)	GCN	8.14	17.86	48.14	63.75	72.41
	GraRep	3.33	12.50	44.17	55.84	66.67
	DeepWalk	6.67	14.39	47.37	58.61	70.37
	Node2vec	5.83	13.33	46.67	56.67	65.83
	LINE	7.50	15.01	49.17	52.17	63.33
	GraphGAN	5.83	15.83	42.50	53.33	67.50
	Average	6.22	14.82	46.34	56.73	67.69
AME	GCN	18.73	16.86	12.63	8.41	5.32
	GraRep	19.76	17.93	15.04	8.57	6.48
	DeepWalk	19.23	18.69	14.39	9.01	6.21
	Node2vec	19.54	18.63	13.58	9.25	7.01
	LINE	19.06	17.59	14.69	9.46	7.74
	GraphGAN	19.38	18.06	15.43	10.39	6.87
	Average	19.28	17.96	14.29	9.18	6.61

TABLE 9: Disturbance-limited attack on Citeseer.

Metris	Model	Neighbor order				
		1	2	3	4	5
ASR(%)	GCN	3.96	17.82	27.66	44.55	52.50
	GraRep	1.98	14.85	23.76	39.60	46.53
	DeepWalk	2.97	12.87	21.78	40.59	47.52
	Node2vec	3.96	15.84	19.80	36.63	47.52
	LINE	2.97	11.88	22.77	41.58	51.48
	GraphGAN	2.97	13.56	22.77	37.62	49.50
	Average	3.14	14.47	23.09	40.10	49.18
AME	GCN	19.26	17.96	14.87	9.87	7.95
	GraRep	19.94	18.79	16.04	10.93	10.01
	DeepWalk	19.76	19.04	16.88	11.57	9.36
	Node2vec	19.53	18.54	17.06	12.03	9.51
	LINE	19.87	19.02	15.96	11.42	8.54
	GraphGAN	19.87	18.33	15.83	12.21	8.87
	Average	19.71	18.61	16.11	11.34	9.04

TABLE 10: Disturbance-limited attack on Pol.blogs.

Metris	Model	Neighbor order				
		1	2	3	4	5
ASR(%)	GCN	10.64	74.47	89.36	91.30	93.30
	GraRep	8.53	72.25	85.36	88.36	91.30
	DeepWalk	7.50	67.53	86.27	87.87	89.47
	Node2vec	5.63	65.48	84.26	86.49	87.87
	LINE	7.50	68.83	85.25	87.87	87.87
	GraphGAN	4.87	62.57	82.03	86.49	89.47
	Average	7.45	68.52	85.42	88.06	89.88
AME	GCN	15.85	10.68	6.38	5.84	5.74
	GraRep	17.53	12.84	9.07	6.98	6.73
	DeepWalk	16.45	12.25	8.83	7.59	7.53
	Node2vec	16.88	13.47	8.52	6.26	6.37
	LINE	16.27	13.09	8.04	7.23	6.81
	GraphGAN	17.26	13.62	9.21	8.14	7.15
	Average	16.71	12.66	8.34	7.01	6.72

[12] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[13] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[14] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[15] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[16] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: A review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[17] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] ——, "Variational graph auto-encoders," *arXiv preprint arXiv:1611.07308*, 2016.

[20] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[21] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the*

IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2574–2582.

[22] S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners." in *AAAI*, 2015, pp. 2871–2877.

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

[24] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 4, pp. 984–996, 2014.

[25] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM, 2017, pp. 506–519.

[26] G. F. Elsayed, S. Shankar, B. Cheung, N. Papernot, A. Kurakin, I. Goodfellow, and J. Sohl-Dickstein, "Adversarial examples that fool both human and computer vision," *arXiv preprint arXiv:1802.08195*, 2018.

[27] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," *arXiv preprint*, 2017.

[28] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*. IEEE, 2017, pp. 39–57.

[29] A. Kurakin, I. Goodfellow, S. Bengio, Y. Dong, F. Liao, M. Liang, T. Pang, J. Zhu, X. Hu, C. Xie *et al.*, "Adversarial attacks and defences competition," *arXiv preprint arXiv:1804.00097*, 2018.

[30] Y. Shang, "Subgraph robustness of complex networks under attacks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–12, 2017.

[31] L. Faramondi, G. Oliva, S. Panzieri, F. Pascucci, M. Schlueter, M. Munetomo, and R. Setola, "Network structural vulnerability: a multiobjective attacker perspective," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–14, 2018.

[32] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE transactions on neural networks and learning systems*, 2019.

[33] A. M. Fard and K. Wang, "Neighborhood randomization for link privacy in social network analysis," *World Wide Web*, vol. 18, no. 1, pp. 9–32, 2015.

[34] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *SIGKDD*, 2016, pp. 855–864.

[35] H. Wang, J. Wang, J. Wang, M. Zhao, W. Zhang, F. Zhang, X. Xie, and M. Guo, "Graphgan: Graph representation learning with generative adversarial nets," 2017.

[36] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[37] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[38] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

[42] T. Pham, T. Tran, D. Q. Phung, and S. Venkatesh, "Column networks for collective classification." in *AAAI*, 2017, pp. 2485–2491.

[43] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5115–5124.

[44] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7794–7803.

[45] S. Nagaraja, "The impact of unlinkability on adversarial community detection: effects and countermeasures," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2010, pp. 253–272.

[46] M. Waniek, T. P. Michalak, M. J. Wooldridge, and T. Rahwan, "Hiding individuals and communities in a social network," *Nature Human Behaviour*, vol. 2, no. 2, p. 139, 2018.

[47] V. Fionda and G. Pirro, "Community deception or: How to stop fearing community detection algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 4, pp. 660–673, 2018.

[48] M. Waniek, K. Zhou, Y. Vorobeychik, E. Moro, T. P. Michalak, and T. Rahwan, "Attack tolerance of link prediction algorithms: How to hide your relations in a social network," *arXiv preprint arXiv:1809.00152*, 2018.

[49] K. Zhou, T. P. Michalak, T. Rahwan, M. Waniek, and Y. Vorobeychik, "Adversarial link prediction in social networks," *arXiv preprint arXiv:1809.08368*, 2018.

[50] A. M. Fard, K. Wang, and P. S. Yu, "Limiting link disclosure in social network analysis through subgraph-wise perturbation," in *Proceedings of the 15th International Conference on Extending Database Technology*. ACM, 2012, pp. 109–119.

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

[52] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on classification models for graphs," *arXiv preprint arXiv:1805.07984*, 2018.

[53] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," 2018.

[54] J. Chen, Z. Shi, Y. Wu, X. Xu, and H. Zheng, "Link prediction adversarial attack," *arXiv preprint arXiv:1810.01110*, 2018.

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

[56] M. Sun, J. Tang, H. Li, B. Li, C. Xiao, Y. Chen, and D. Song, "Data poisoning attack against unsupervised node embedding methods," 2018.

[57] A. K. Mccallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[58] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 36–43.