MGA: Momentum Gradient Attack on Network

Jinyin Chen, Yixian Chen, Haibin Zheng, Shijing Shen, Shanqing Yu, Dan Zhang, and Qi Xuan, *Member, IEEE*

Abstract—The adversarial attack methods based on gradient information can adequately find the perturbations, that is, the combinations of rewired links, thereby reducing the effectiveness of the deep learning model based graph embedding algorithms, but it is also easy to fall into a local optimum. Therefore, this paper proposes a Momentum Gradient Attack (MGA) against the GCN model, which can achieve more aggressive attacks with fewer rewiring links. Compared with directly updating the original network using gradient information, integrating the momentum term into the iterative process can stabilize the updating direction, which makes the model jump out of poor local optimum and enhance the method with stronger transferability. Experiments on node classification and community detection methods based on three well-known network embedding algorithms show that MGA has a better attack effect and transferability.

Index Terms—Momentum gradient, adversarial attack, graph embedding, node classification, community detection

I. Introduction

NOWADAYS many systems can be modeled as graphs, such as social networks [1]–[3], biological networks [4], [5], traffic networks [6], [7]. With the help of graph embedding, especially some deep learning models, nodes and links in the network can be represented as low-dimensional vectors, based on which downstream tasks can be easily implemented. Such approaches have achieved great success in various applications such as node classification [8]–[10], community detection [11]–[13] and link prediction [14]–[16].

However, it has been found that deep models are vulnerable to adversarial attacks [17]–[21]. In the areas such as computer vision [22]–[25] and recommendation system [26], [27], the adversarial attacks against deep learning models are considered as a severe security problem, because small perturbations may fool the methods and consequently produce completely wrong output. Recently, the adversarial attacks against network algorithms are also carried out to study the robustness of network algorithms [28], [29], by rewiring a small number of links in the original network. Some even study on using the adversarial attack methods to solve privacy problems, that is, to protect certain sensitive information of the network from being leaked by graph mining methods [30], [31].

At present, many adversarial attacks on network algorithms directly based on gradient information, i.e., by calculating the partial derivatives of the target loss function to the element of adjacency matrix [28], [30], [32]. In this way, the generated adversarial network can easily drop into poor local maxima

Corresponding author: Shanqing Yu.

J. Chen, Y. Chen, H. Zheng, S. Shen, S. Yu, D. Zhang and Q. Xuan are with the Institute of Cyberspace Security, College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China. (e-mail: {chenjinyin, 2111803168, 201303080231, 2111803074, yushanqing, danzhang, xuanqi}@zjut.edu.cn).

and thus "overfit" the model, leading to lower transferability across models.

Adopting the momentum method [33], which is a technique to accelerating gradient descent by accumulating a velocity vector in the gradient descent directions, is a widely used method to avoid inadequate local solutions [34]–[36]. In computer vision, many studies on the momentum method have been proposed to improve the gradient method [37]–[40]. Moreover, it was used in adversarial attacks of image classification algorithms to further improve the success rates of black-box attacks [41]. However, momentum has not been used in graph adversarial attack. In order to further improve the attack effectiveness of gradient-based adversarial attack against the Graph Convolutional Network (GCN) model, this paper proposes a momentum gradient attack (MGA) that uses the gradient momentum to update the adversarial network. Specifically, we use momentum to produce a new updating direction by adding velocity vector into the gradient of the loss function, which can stabilize the updating direction and make the attack robust to different models. The main contributions of the proposed MGA are as follows.

- As a new adversarial attack against the network algorithms, MGA is superior to several state-of-the-art adversarial attacks under the same scale of perturbation, including GradArgmax [29], RL-S2V [29], NETTACK [32], and Fast Gradient Attack (FGA) [30].
- MGA has strong transitivity, and adversarial networks generated based on GCN can also be used to effectively attack other deep models, including DeepWalk [42], node2vec [43] and Graph-GAN [44].
- Compared with FGA [30], MGA performs better and is more robust in the absence of some structural information of the original network, so it is more practical.

The rest of paper is organized as follows. In Sec. II, we review the existing adversarial attack methods on graph data. In Sec. III, we briefly introduce the GCN and FGA. In Sec. IV, we introduce MGA method in details, including white-box and black-box adversarial attacks. In Sec. V, we empirically evaluate MGA on multiple tasks, and compare the attack performance on several real-world networks. In Sec. VI, we conclude the paper and highlight future research directions.

II. Related Work

Recently, adversarial attack on networks has attracted a lot of attention and many algorithms have been proposed. In particular, for community detection, Nagaraja [45] first introduced a community hiding algorithm, where they added links based on centrality values to change the community structure. Waniek et al. [46] proposed a heuristic method named Disconnect Internally, Connect Externally (DICE),

which deletes the links within the same communities while adds the links across different communities. Valeria et al. [47] proposed the safeness-based method, which adds and deletes specific links, leading to the maximum increase of safeness. Chen et al. [48] further introduced a community detection attack algorithm based on genetic algorithm, which treated modularity Q as fitness function. For link prediction, Zheleva et al. [49] proposed the link re-identification attack to protect sensitive links from the released data. Link perturbation is a common technique in those early researches that data publisher can randomly modify links on the original network to protect the sensitive links from being identified. Fard et al. [50] introduced a subgraph wise perturbation in directed networks to randomize the destination of a link within subgraphs so as to protect sensitive links. They further proposed a neighborhood randomization mechanism to probabilistically randomize the destination of a link within a local neighborhood [51]. For link prediction, Yu et al. [31] designed evolutionary perturbations based on genetic algorithm and estimation of distribution algorithm to degrade the popular link prediction method Resource Allocation Index (RA). And they found that although the evolutionary perturbations are designed to against RA, it is also effective against other link-prediction methods. For network embedding, Xuan et al. [52] proposed Euclidean Distance Attack (EDA), a genetic algorithm based network embedding attack method, which focuses on disturbing the Euclidean distance between nodes in the embedding space.

Moreover, due the great success of many graph deep learning frameworks, e.g., Graph Convolutional Networks (GCN), several deep learning based attacks have been proposed recently. Bojchevski and Günnemann [28] attacked the embedding model by maximizing the random walk model based loss function, and they demonstrated that the generated adversarial network has strong transferability. Dai et al. [29] proposed a reinforcement learning based attack, which uses a hierarchical Q-function to reduce its time complexity. Zugner et al. [32] proposed NETTACK, which maximizes the difference in the log-probabilities of the target node before and after the attack. This method not only considers rewiring links, but also modifies nodes' features to generate unnoticeable perturbations. Depending on the gradient information, Chen et al. [30] proposed the Fast Gradient Attack (FGA), which directly uses the gradient of adjacency matrix to iteratively update the adversarial network. By using this method, the embedding of target node can be changed significantly through rewiring a small number of links, so as to greatly affect the downstream algorithms based on embedding. However, although it is relatively fast, FGA easily falls into the poor local optimum since it uses the gradient information directly.

III. BACKGROUND

A. Double-Layer GCN model

Since the end-to-end node classification model GCN [53] has great performance in node classifications, we generate adversarial networks based on gradient information of GCN same as the work in [30], [32]. Given a network G and the feature of nodes X. Denoting A as the adjacency matrix of the network, the forward propagation of GCN model of a single hidden layer then is defined as follow:

$$Y'(A) = f(\bar{A}\sigma(\bar{A}XW_0)W_1), \qquad (1)$$

where $\bar{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, $\tilde{A} = A + I_N$ is the self-connected form of undirected network G after adding the identity matrix I_N to A, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ is the diagonal degree matrix of $\tilde{A}$; $W_0 \in R^{|V| \times H}$ and $W_1 \in R^{H \times |F|}$ are the input-to-hidden and hidden-to-output weight matrices, respectively, with the node set V, category set F and hidden layer of H feature maps; f and σ are the softmax function and Relu active function, respectively. For semi-supervised multi-label classification on GCN, we define the loss function based on the cross-entropy error over all labeled nodes:

$$L = -\sum_{l=1}^{|V_L|}\sum_{h=1}^{|F|} Y_{lh} \ln(Y'_{lh}(A)), \qquad (2)$$

where V_L is the set of nodes with ground truth labels, Y is the real label matrix with $Y_{lh} = 1$ if node v_l belongs to h-th category and $Y_{lh} = 0$ otherwise, and $Y'(A)$ is the output of the model defined by Eq. (1).

The weight matrices W_0 and W_1 are updated based on gradient descent, so that the GCN model is continuously optimized and the node classification performance is steadily improved. As we can see, in Eq. (2), the adjacency matrix A is another critical variable in the loss function. Therefore, we can implement network attacks, that is, to reduce the accuracy of nodes classification, by modifying the adjacency matrix A based on its gradient information.

B. Fast Gradient Attack method

In this subsection, the FGA method [30], which is a gradient-based adversarial attack method against the GCN model, is briefly introduced. Based on the trained GCN model, the target loss function L_t of target node v_t is defined as Eq. (3). It represents the difference between the predicted label and the real one of the target node v_t.

$$L_t = -\sum_{k=1}^{|F|} Y_{tk} \ln(Y'_{tk}(A)), \qquad (3)$$

which represents the difference between the predicted label and the real one of the target node v_t. The larger value of target loss function corresponds to the worse prediction result. The basic link gradient matrix $\bar{g}$ is represented by Eq. (4).

$$\bar{g}_{ij} = \frac{\partial L_t}{\partial A_{ij}}. \qquad (4)$$

where A_{ij} and $\bar{g}_{ij}$ is the element of adjacency matrix and link gradient matrix respectively.

The primary link gradient matrix $\bar{g}$ is symmetrized by Eq. (5) to obtain the initial Link Gradient Network (LGN) g since the adjacent matrix of an undirected network is symmetry.

$$g_{ij} = g_{ji} = \begin{cases} \frac{\bar{g}_{ij} + \bar{g}_{ji}}{2} & i \neq j \\ 0 & i = j \end{cases}. \qquad (5)$$

TABLE I: The definitions of symbols.

Symbol	Definition
$G = (V, E)$	the original network with nodes V and links E
$\hat{G} = (V, \hat{E})$	adversarial network with nodes V and updated links $\hat{E}$
$A/\hat{A}$	the adjacency matrix of original network/adversarial network
L	the loss function of the GCN model
L_t	the loss function for target node v_t of the GCN model
Y/Y'	the label confidence list before/after the attack
v_t	target node
K	the maximum number of iterations
$\hat{G}^k/\hat{A}^k$	the kth adversarial network/adjacency matrix
$\bar{g}/g$	link gradient matrix/network
$\hat{g}^k$	the kth momentum of link gradient matrix
μ	the decay factor
θ	the sign function

In initial LGN g, each pair of nodes has a positive or negative gradient value. Note that adding/deleting links between pairs of nodes with larger magnitude gradients may cause the target loss function to change rapidly. Therefore, FGA method add links between the pairs of nodes with maximum positive link gradient, and delete links between the pairs of nodes with maximum negative link gradient.

IV. METHOD

In this paper, we design MGA as a new attack method to generate adversarial networks based on the trained GCN model. Due to the ability of momentum term in the iterative process to stabilize the updating direction, MGA can successfully fool white-box model (GCN model) as well as black-box models (i.e. DeepWalk, node2vec, GraphGAN, etc). In the realization of adversarial attack, we first choose the target nodes, and then generate adversarial networks to hide the selected target nodes under certain network analysis methods. The flowchart of MGA is shown in Fig 1. For convenience, the definitions of symbols used in Sec. IV-B are briefly summarized in TABLE I.

A. Problem Definition

We first give the definition of adversarial attack on networks as follows.

DEFINITION 1 (Adversarial attack). *Given the original network $G = (V, E)$, where V and E are the set of nodes and links, and the adversarial network $\hat{G} = (V, \hat{E})$ which is generated by rewiring some edges of original network G. Generally we can use a network analysis model f to get the analysis results $f(G)$ and $f(\hat{G})$, respectively. The purpose of adversarial attack is to fool the network analysis model f by adversarial network $\hat{G}$, which can be formulated as:*

$$\arg\max_{\bar{G}} \phi(f(G), f(\hat{G})), \tag{6}$$

$$s.t. \sum_{i<j} |A_{ij} - \hat{A}_{ij}| \leq \rho \tag{7}$$

where ρ is the size of adversarial perturbation and ϕ is a function measuring the difference between $f(G)$ and $f(\hat{G})$.

B. MGA method

To generate adversarial network for target node v_t from the original network, after the definition of loss function L_t Eq. (3), the network attack methods generate the adversarial network by solving the following constrained optimization problem:

$$\arg\max_{\hat{A}} L_t(\hat{A}), \quad s.t. \sum_{i<j} |A_{ij} - \hat{A}_{ij}| \leq \rho, \tag{8}$$

where ρ is the size of adversarial perturbation.

In MGA method, we modify one link during each iteration, and stop the process when the maximum allowed number of modified links is reached. The kth iteration can be described by the following steps.

Firstly with $\hat{A}^0 = A$, we can generate the kth LGN g^k using the adversarial network adjacency matrix $\hat{A}^{k-1}$ based on Eq. (4) and Eq. (5). Then, we can update the kth Link Momentum Network (LMN) $\hat{g}^k$ by accumulating the velocity vector in the gradient direction using Eq. (9).

$$\hat{g}_{ij}^k = \begin{cases} g_{ij}^k & k = 1 \\ \mu \cdot \hat{g}_{ij}^{k-1} + \frac{g_{ij}^k}{||g_{ij}^k||_1} & else \end{cases} \tag{9}$$

where μ is the decay factor which in the range of $[0, 1]$.

We update the $(k-1)^{th}$ adversarial network by modifying the link (v_i, v_j) of the maximum absolute link momentum in the kth momentum LMN $\hat{g}^k$.

$$\hat{A}_{ij}^k = \hat{A}_{ij}^{k-1} + \theta(\hat{g}_{ij}^k), \tag{10}$$

$$\arg\max_{(v_i, v_j) \in \Omega} |\hat{g}_{ij}^k|, \tag{11}$$

where Ω is the set of all pairs of nodes except $v_i = v_j$, and $\theta(\hat{g}_{ij}^k)$ represents the sign of momentum of the selected pair of nodes (v_i, v_j).

The sign of the momentum is a basic element in LMN which represent the ways of the modifying links, that is, delete or add the links between the selected node pairs. More specially, a positive momentum means to add the links; otherwise, it means to delete the links.

$$\theta(\hat{g}_{ij}^k) = \begin{cases} 1 & \hat{g}_{ij}^k > 0 \\ -1 & else \end{cases} \tag{12}$$

In summary, $\hat{A}^0 = A$ and the iterative number $k = 1$, initially. Next, LGN g^k is calculated based on Eq. (4) and Eq. (5), and LMN $\hat{g}^k$ is generated by Eq. (9). Then, the adversarial network is generated iteratively by Eq. (10) and Eq. (11) and $k = k + 1$. When $k = K$, where K is the maximum number of iterations, the algorithm ends and output the final adversarial network. The whole steps of the adversarial network generator is described by the following steps.

1) Train the GCN model on original network G.
2) Initialize number of iterations $k=1$ and the adjacency matrix of the adversarial network by $\hat{A}^0 = A$
3) Construct the LGN g^k based on the $\hat{A}^{k-1}$ via Eq. (4) and Eq. (5).
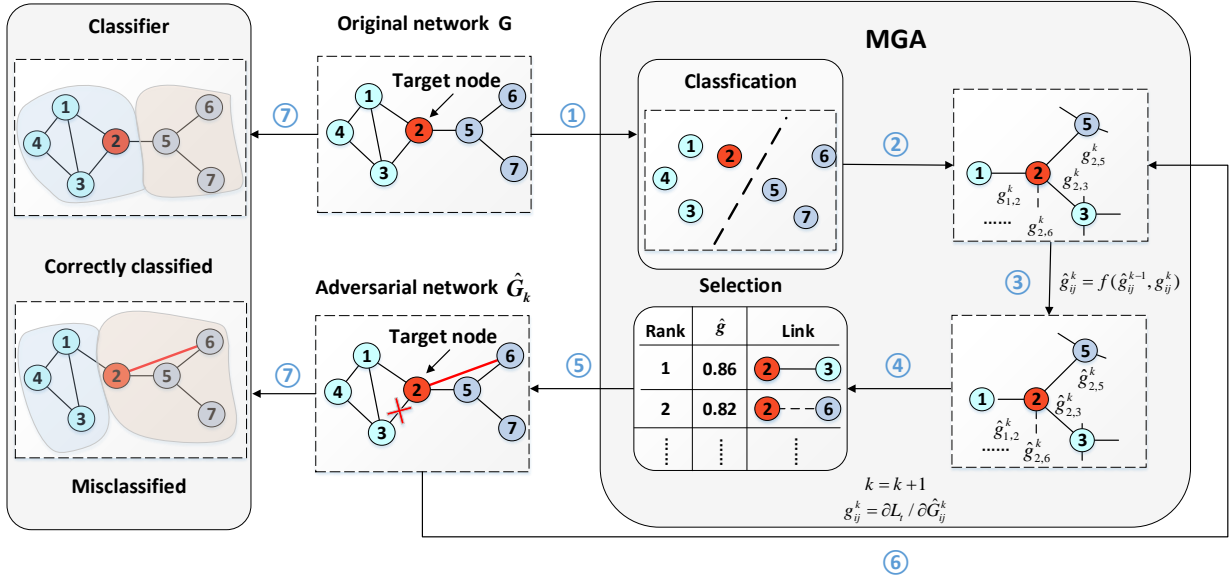4) Construct the LMN $\hat{g}^k$ based on the g^k via Eq. (9).

Fig. 1: The framework of MGA, which consists of seven steps. First, input original network to train GCN model; second, calculate the gradients of each link for the loss function; third, calculate the momentum based on gradient information; fourth, sort the absolute of momentum; fifth, select the largest one for updating the original network; sixth, recalculate the gradient information based on updated network; finally, gain adversarial network after K times of updates and various node classification algorithms are performed on the adversarial network to validate the attack effect of MGA.

5) Select the pair of nodes (v_i, v_j) of the maximum absolute link momentum in $\hat{g}^k$.
6) Update the adjacency matrix $\hat{A}^k$ via Eq. (10) and Eq. (11)
7) If the maximum number of iterations is not reached ($k < K$), $k = k + 1$ and go to step 3); otherwise, go to step 8).
8) Generate the final adversarial network $\hat{G}^K$ based on the adjacency matrix $\hat{A}^K$.

V. EXPERIMENTS

A. Experimental Setting

Here, we will describe the experimental setting. To testify the effectiveness of MGA, we compare it with some state-of-the-art baseline methods by performing several tasks, including node classification attack, community detection attack and anti-defense test on several well-known datasets. The decay factor μ used in the experiment is 0.5.

1) Datasets: The Details of three datasets used in the experiment are presented in TABLE II.

- **Pol.Blogs:** This dataset is compiled by Adamic and Glance [54], which show the political leaning of blog directories. The blogs are divided into two classes and the links between blogs were automatically extracted from the front pages of the blogs. It contains 1,490 blogs and 19,090 links in total.

TABLE II: The basic statistics of the three network datasets.

Dataset	#Nodes	#Links	#Classes
Pol.Blogs	1,490	19,090	2
Cora	2,708	5,429	7
Citeseer	3,312	4,732	6

- **Cora:** This dataset is a typical paper citation network in the computer science domain with the papers divided into seven classes. It contains 2,708 paper and 5,429 links in total [55].
- **Citeseer:** This dataset consists of 3,312 scientific publications classified into six classes. The citation network consists of 4,732 citation links which represents the citation relationships.

2) Baselines: Note that our experimental setting is the same as FGA method [30]. So in the Sec. V-B, we only show the comparison results with FGA algorithm, and FGA algorithm has shown that its effect is better than other algorithms, including NETTACK [32], RL-S2V [29] and GradArgmax [29].

- **GradArgmax** [29]. GradArgmax greedily select the links that are most likely to cause the change to the loss function based on gradient information.
- **RL-S2V** [29]. RL-S2V is a reinforcement learning based attack method which decomposes the quadratic action space by hierarchical method. It use Q-learning to learn the attack procedure which parameterized by S2V [56]. We set the iteration number $\Delta = 200$ for this method.
- **NETTACK** [32]. NETTACK iteratively select the links which can maximize the changes of the surrogate model's results before and after the attack. Note that it only rewires links which can preserve the graph' s degree distribution and feature co-occurrences.
- **FGA** [30]. FGA directly uses the gradient of adjacency matrix to generates adversarial network iteratively. In each iteration, it selects candidate node piars with maximum gradient to update the adversarial network.

3) Performance metrics: In order to compare the transferability of different adversarial attack methods, we use the

TABLE III: The attack effects, in terms of ASR and AML, obtained by different attack methods on various network embedding methods and multiple datasets. Here, ASR is obtained by rewiring 20 links.

Dataset	NEM	ASR (%)						AML					
		MGA			FGA			MGA			FGA		
		Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect
Pol.Blogs	GCN	**100**	97.87	34.04	87.87	85.74	25.53	**3.23**	3.74	16.09	8.42	8.82	17.61
	DeepWalk	**97.96**	91.67	6.25	84.26	81.66	6.25	**6.76**	7.19	19.11	9.84	10.93	19.20
	node2vec	**97.96**	91.67	5.15	84.34	81.83	0.00	**6.71**	7.08	19.30	9.72	10.16	20.00
	GraphGAN	**98.63**	89.58	5.48	81.21	80.24	0.00	**6.72**	7.15	19.16	9.41	11.02	20.00
	Average	**98.64**	92.70	12.73	84.42	82.37	9.95	**5.86**	6.29	18.42	9.35	10.23	19.20
Cora	GCN	**100**	**100**	84.38	**100**	**100**	88.28	**1.74**	1.85	8.91	2.54	3.21	6.77
	DeepWalk	**100**	**100**	85.98	**100**	97.22	81.55	3.27	**3.03**	9.01	5.61	6.27	10.59
	node2vec	**100**	**100**	85.51	**100**	**100**	84.00	3.14	**3.10**	8.71	5.66	5.58	9.52
	GraphGAN	**100**	**100**	83.80	**100**	96.00	84.62	**3.17**	3.22	8.70	5.65	6.40	11.02
	Average	**100**	**100**	77.75	**100**	98.31	84.61	2.83	**2.80**	7.89	4.87	5.37	9.48
Citeseer	GCN	**100**	**100**	94.23	**100**	**100**	91.36	2.53	**2.40**	7.51	3.52	3.88	7.69
	DeepWalk	**100**	**100**	84.48	**100**	**100**	**100**	5.25	**5.17**	7.64	5.68	6.06	7.76
	node2vec	**100**	**100**	85.34	**100**	**100**	98.21	5.09	**4.97**	7.69	5.62	6.50	7.75
	GraphGAN	**100**	**100**	84.02	**100**	97.89	93.15	**5.04**	5.55	7.98	5.91	6.67	8.18
	Average	**100**	**100**	87.02	**100**	99.47	95.68	**4.48**	4.52	7.71	5.18	5.78	7.85

generated adversarial networks to attack other network embedding approaches including DeepWalk [42], Node2vec [43] and GraphGAN [44]. And we also use the following two metrics to compare the effectiveness of these attack methods.

- **ASR**. The average success rate is the ratio of misclassification of target nodes by changing no more than ρ links. In the experiment, the perturbation size ρ is varied from 1 to 20. The higher ASR corresponds to the better attack effect.
- **AML**. The average number of rewiring links to successfully misclassify a target node. In the experiment, the maximum number of rewiring links is 20, i.e. , when a target node can't be misclassified by rewiring 20 links, we set the number as 20. The lower AML corresponds to the better attack effect.

B. Node Classification Attack

Three node classification attack methods are carried out in this subsection. For each method, we do uniform attack, hub-nodes attack and bridge-nodes attack, respectively.

- **Direct attack:** The attack is directed at the target node; that is, the redistributed link is limited to the neighbor link of the target node in the network.
- **Indirect attack:** The attack is located on a link that is not connected to the target node to make the attack more covert.
- **Unlimited attack:** Generate perturbations without considering whether they are around the target nodes or not.

We randomly choose 10% labeled nodes as training and validation sets, respectively. The rest 80% nodes are used for testing. For all the network embedding methods, the vector dimension is set to 128, the number of walks per node is set to 10, the length of walk is set to 80 and the size of context window is set to 10. The nodes that may be misclassified by the GCN model or logistic regression classifier without attack will not be considered as the target nodes.

1) uniform attack: The uniform attack experiments, which select 20 nodes from each category as the target nodes, were carried out to verify the general attacking effect of the strategies. The results are shown in Tables III and Figs. 2. It

can be seen that, in most cases, whether on ASR or AML, MGA achieves better attack results than FGA. Interestingly, such superiority is even more distinct on Pol.Blogs dataset, which has more links and larger attack solutions space.

Attacking GCN will get better attack performance than attacking other network embedding algorithms, since MGA is developed based on GCN. However, when attacking other network embedding algorithms, the performance of MGA is also better than other attack methods, which proves that MGA has stronger transferability. Even in indirect attacks, MGA performed quite well on the Cora and Citeseer datasets, with ASR exceeding 80% in most cases.

2) Hub-node attack: There are many indicators of concentration that reflect the importance of nodes in the network, including degree centrality, closeness, betweenness [57]. In this subsection, we select 40 nodes with highest degree centrality [58] as hub nodes and consider them as target nodes. The results presented in Tables IV show that, MGA still outperforms other methods, and unlimited MGA outperforms direct or indirect MGA in most cases. Besides, due to the high degree centrality of target nodes, compared to the uniform attack shown in Tables III, the effectiveness of all the attack methods are reduced, especially on the Pol.Blogs dataset whose hub-nodes have higher degree value than other datasets. In addition, we can also find that the transferability of indirected MGA decreases most when attacking hub-node, which may because the target nodes have more neighborhood make their embeddings much more robust and in different models, links far away from hub nodes influence those nodes is different.

3) Bridge-node attack: In each network, 40 nodes with the highest betweenness centrality [59], [60], were selected as bridge nodes. Different from the degree centrality, the betweenness measures the degree to which a node is located on the shortest path between two other nodes [61], [62], and consequently considers the global network structure. The attack results are presented in Table V. Similar to uniform attacks and hub-node attacks, MGA performs better than other methods in the unlimited and directed attacks, and all methods perform best in the unlimited attacks. Meanwhile, it is found that it is easier to attack the bridge nodes than the hub nodes,
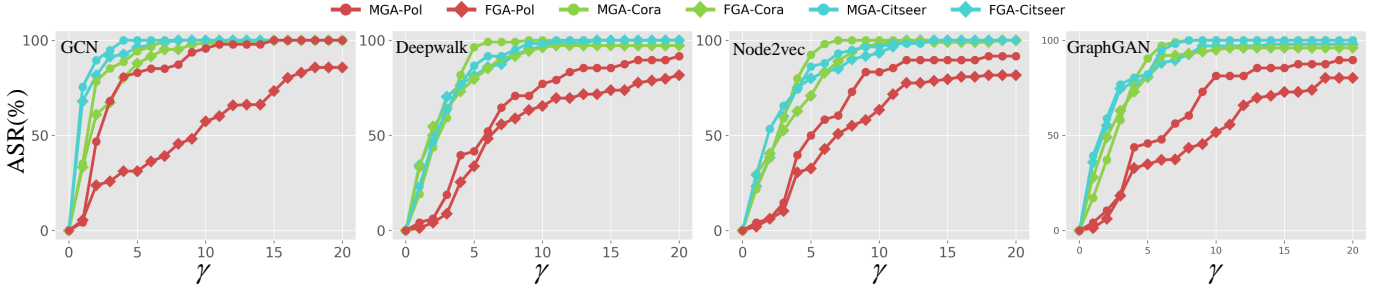
Fig. 2: ASR of different attack methods of perturbation size γ for different dataset.
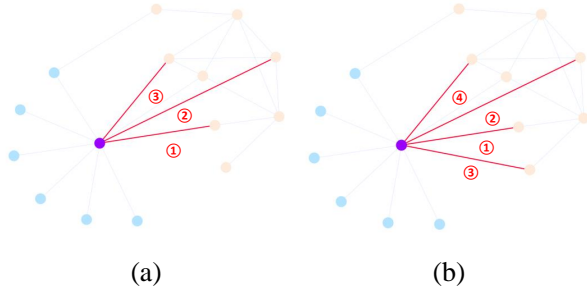

(a)   (b)

Fig. 3: The visualization of target node attack obtained by MGA and FGA. Nodes of the same color belong to the same class and the purple node represent the target node. Note that the red links are the added links and the numbers indicate the order of it. (a) MGA, (b)FGA.

which may be because the degree of the bridge nodes is much smaller than the hub nodes.

Overall, by comparing the results in Table III-Table V, we can find that MGA is useful in attacking different types of nodes, which indicates that integrating momentum terms into the iterative process can enhance the attack effect and transferability. Taking a target node as an example, the visualization results of the MGA and FGA are shown in Fig 3. From Fig 3, we can see that the added links of MGA and FGA are the same in the first two iterations, but results of the third iteration are quite different. MGA can attack successfully after three iterations, while FGA needs four iterations to succeed. It shows that MGA finds the optimal solution faster by using momentum in gradient attack.

C. Selected Links Analysis

In this section, we will explore the reason why the links selected by MGA have a greater impact on node classification compared to others. We calculate the four metrics value to evaluate the selected links. For the γ-th selected links, we calculate the sum of the its related node degree $D(\gamma)$ and its's betweenness $B(\gamma)$. Further more, we also calculate the average distance $A_s(\gamma)$ (resp., $A_d(\gamma)$), which denote the average distance between target node and the labeled nodes which belong to the same class(resp., different classes) after

update the γ links. More specifically, these metrics are defined as follows.

$$D(\gamma) = \sum_{e=(u,v)}^{E_\gamma} \frac{d_u + d_v}{n_t}, \quad (13)$$

where $E_\gamma = \{e_\gamma^1, e_\gamma^2, \cdots, e_\gamma^{n_t}\}$ is the set of γ-th selected links for n_t target nodes. Further more d_u and d_v is the degree of node u and v before update those links respectively.

$$B(\gamma) = \sum_{e \in E_\gamma} \sum_{u,v \in V} \frac{\sigma(u,v|e)/\sigma(u,v)}{n_t}, \quad (14)$$

where V is the node set in the graph, $\sigma(u,v)$ represents the total number of shortest paths between nodes u and v, and $\sigma(u,v|e)$ represents the number of shortest paths through the link e.

$$A_s(\gamma) = \sum_{v_t \in T} \frac{\sum_{u \in C_t} D_\gamma(u,v_t)/n_{C_t}}{n_t}, \quad (15)$$

$$A_d(\gamma) = \sum_{v_t \in T} \frac{\sum_{u \notin C_t} D_\gamma(u,v_t)/\bar{n}_{C_t}}{n_t}, \quad (16)$$

where $D(u, v_t)$ is the distance between target node v_t and node u after update the γ links, and C_t is the class which target node v_t belongs to. Furthermore, the number of nodes which belong or not belong to class C_t is n_{C_t} and $\bar{n}_{C_t}$ respectively.

From Fig. 4, We can find the average betweenness B can't be used to judge the result of node classification, but we can generally conclude that the lower the average degree D, the better the result of classification will be. This may be because modifying the links on the nodes with small degree value has more influence on those nodes than modifying links on nodes with large degree value. The NETTACK algorithm significantly increases A_s and decreases A_d, while MGA, FGA, RL-S2V and GradArgmax don't increase A_s. This is because those methods rarely delete links. We can also find that MGA and FGA both decreases the maximum amplitude A_d as much as possible and ensure that A_s does not decrease much. Considering that the effects of MGA and FGA are better than NETTACK, we believe that the priority of lowering A_d is greater than increasing A_s.

TABLE IV: The hub-node attack effects, in terms of ASR and AML, obtained by different attack methods on various network embedding methods and multiple datasets. Here, ASR is obtained by rewiring 20 links.

| Dataset | NEM | ASR (%) | | | | | | AML | | | | | |
| | | MGA | | | FGA | | | MGA | | | FGA | | |
		Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect
Pol.Blogs	GCN	**57.5**	55.0	5.00	50.00	45.00	5.00	**14.25**	14.45	19.20	14.78	15.28	19.70
	DeepWalk	47.50	50.00	2.50	**53.85**	50.00	2.50	16.12	**15.95**	19.55	16.00	16.82	19.77
	node2vec	**48.75**	47.50	5.00	32.50	45.00	5.00	**17.14**	17.27	19.23	17.62	17.95	19.20
	GraphGAN	**49.17**	40.00	2.50	25.00	17.50	2.50	**17.07**	17.27	19.36	18.68	18.77	19.52
	Average	**50.73**	48.13	3.75	40.34	39.38	3.75	**16.15**	16.24	19.34	16.77	17.21	19.55
Cora	GCN	**96.88**	92.86	55.70	88.90	87.18	54.54	**4.88**	4.90	11.15	6.48	6.62	14.37
	DeepWalk	**93.90**	87.50	47.22	85.63	80.50	43.75	**6.61**	7.75	13.29	7.57	7.73	13.70
	node2vec	**91.67**	91.18	48.39	81.82	80.34	41.96	**6.92**	6.74	12.95	7.22	7.52	13.05
	GraphGAN	**93.75**	87.50	53.13	84.36	82.57	48.74	**7.44**	7.97	12.19	8.00	8.15	12.98
	Average	**94.05**	89.76	58.28	85.18	82.65	47.25	**6.46**	6.84	11.84	7.32	7.51	13.53
Citeseer	GCN	**100**	**100**	74.29	**100**	**100**	65.71	**1.94**	2.49	6.54	5.11	5.23	9.77
	DeepWalk	**94.44**	91.89	18.92	88.89	86.89	8.33	**9.81**	11.62	18.51	10.67	12.14	18.89
	node2vec	**93.15**	90.41	17.81	86.11	88.89	13.51	**9.74**	11.59	18.56	13.08	12.14	18.03
	GraphGAN	**91.74**	89.91	18.02	89.19	88.89	5.56	**10.16**	11.50	18.20	13.47	11.95	19.17
	Average	**94.83**	93.05	32.26	91.05	91.17	23.28	**7.91**	9.30	15.45	10.58	10.37	16.47

TABLE V: The bridge-node attack effects, in terms of ASR and AML, obtained by different attack methods on various network embedding methods and multiple datasets. Here, ASR is obtained by rewiring 20 links.

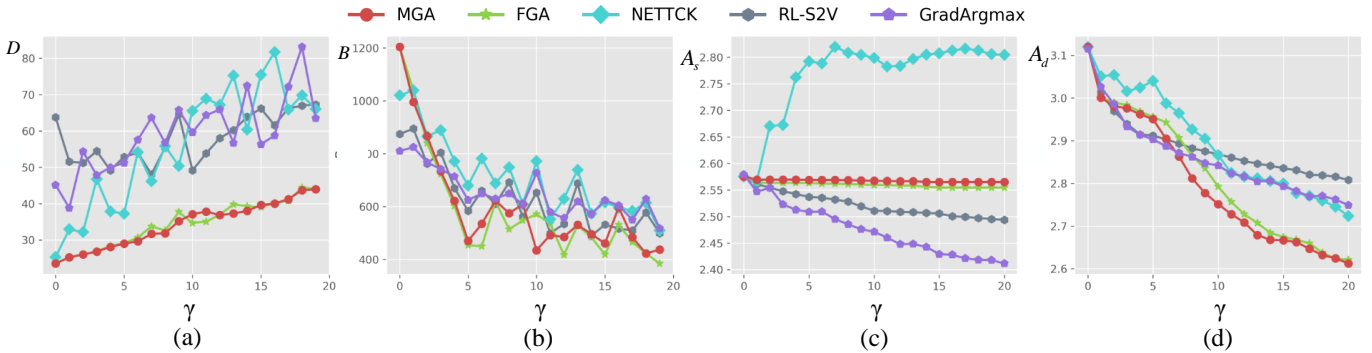| Dataset | NEM | ASR (%) | | | | | | AML | | | | | |
| | | MGA | | | FGA | | | MGA | | | FGA | | |
		Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect
Pol.Blogs	GCN	**63.16**	57.9	18.42	62.16	54.05	5.26	**13.71**	13.92	17.97	14.65	14.95	19.29
	DeepWalk	41.54	**44.74**	7.84	36.84	36.84	7.89	16.97	**16.84**	18.83	17.08	17.61	18.82
	node2vec	**56.76**	52.11	5.41	50.00	52.63	5.26	**15.89**	16.95	19.06	16.21	16.03	19.13
	GraphGAN	**39.47**	36.84	5.26	30.77	25.79	5.26	**14.13**	17.16	19.37	17.92	18.25	19.37
	Average	**50.23**	47.90	9.23	44.94	42.33	5.92	**15.18**	16.22	18.81	16.47	16.71	19.15
Cora	GCN	**96.97**	93.33	78.57	92.59	88.89	55.56	**3.25**	4.11	9.14	4.63	4.78	10.93
	DeepWalk	**93.75**	90.32	41.16	**93.75**	90.32	28.12	**7.31**	8.55	14.52	7.91	8.03	16.03
	node2vec	**93.75**	90.91	45.31	**93.75**	86.67	40.00	**6.84**	8.18	14.14	8.25	9.03	14.46
	GraphGAN	**93.25**	90.32	42.71	90.62	90.00	33.33	**7.93**	8.48	14.14	8.69	8.87	15.97
	Average	**94.43**	91.22	51.94	92.68	88.97	39.25	**6.33**	7.33	12.99	7.37	7.68	14.35
Citeseer	GCN	**100**	**100**	93.75	96.97	96.97	66.67	**2.03**	**2.03**	5.50	4.94	5.03	11.61
	DeepWalk	**96.77**	93.33	45.16	93.33	93.33	20.69	**6.90**	8.10	15.65	8.80	9.33	18.76
	node2vec	**95.00**	93.55	42.62	93.33	90.32	26.67	**6.97**	7.87	16.39	9.63	9.06	16.27
	GraphGAN	**93.41**	90.62	41.76	93.55	93.33	19.35	**6.80**	7.38	16.62	8.97	8.33	17.87
	Average	**96.30**	94.38	55.82	94.30	93.49	33.35	**5.68**	6.35	13.54	8.09	7.94	16.13

Fig. 4: The information of selected links. (a) Node degree; (b) Betweenness; (c) The average distance of target nodes to the same kind of nodes; (d) The average distance of target nodes to the different kinds of nodes.

D. Community Deception

Community detection is aim to find community structure by maximizing cluster quality measures such as modularity [63], [64]. Similar to node classification, community detection also labels each node, but most of them are unsupervised. In this subsection, experiments are performed to study the effect of MGA on community detection methods. Community deception [45]–[47] is strive to hide target community from being discovered by community detection algorithms. We first use attack methods to generate the adversarial networks and then use two community detection algorithms and a combination of network embedding methods and K-means to detect the communities. The two community detection algorithms used in this subsection are introduced as follows

- **LPA [65]**. Firstly each vertex has its own label and then update its label by choosing the most frequent label of its neighbor until the labels don't change. This algorithm
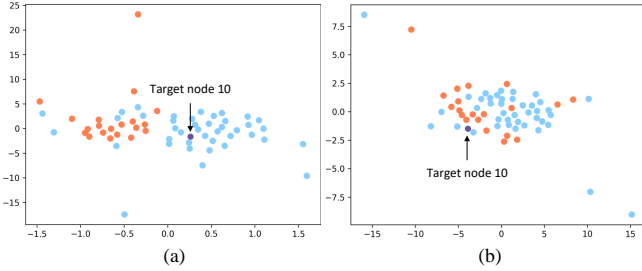
Fig. 5: The visualization of target node attack obtained by MGA, utilizing t-SNE algorithm, on Dolphin network. The purple node represent the target node 10, which originally belongs to the blue community. After the attack, it belongs to the red community. Except the target node, the nodes of same color still belongs to the same community detected by the fast greedy algorithm [70]. (a) before attack; (b) after attack.

has randomness which runs in time $O(E)$.

- **Louvain [66]**. A modularity based algorithm which can generate a hierarchical community structure by compressing the communities as new nodes and its runs in time $O(|V|\log|V|)$.

The datasets we used is described as follow.

- **The USA political books [67]:** The network represents co-purchasing of US politics books around the time of the 2004 presidential election. The network has 105 books and 441 links in total, and the books are classified as conservative, liberal, and neutral.
- **Dolphin social network [68]:** The social network represents the common interactions of 62 dolphins which live off Doubtful Sound, New Zealand. The network has 159 links and can be divided into two parts.

In this subsection, we randomly choose 10% nodes from each community as training and validation sets respectively. The rest 80% nodes are used for testing. For all the network embedding methods, the dimension of embedding vector is set to 20 and other settings are consistent with Sec. V-B. As an example, we visualize the attack effect of MGA on the Dolphin network using node2vec and t-SNE [69], as shown in Fig. 5. From Fig. 5 we can see that after the attack the target nodes which originally belong to the blue community are more closer to the red community than blue community, while the other nodes still show a more distinct community distribution.

Table VI shows the results of each algorithm on community deception, from which we can see that MGA perform better than other attack methods, suggesting that MGA is effective for attacking community detection algorithms. Besides, the effect of each algorithm on Pol.Book is slightly lower than that on dolphins data set, suggesting that community deception is still affected by the degree of target nodes.
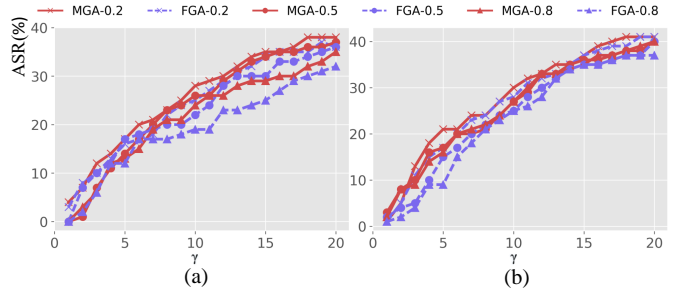

Fig. 6: The attack results of MGA and FGA when knowledge is limited. In each missing case, the probability of missing links is set to 0.2, 0.5, and 0.8. (a) Ensuring the 1-hop neighbors; (b) Randomly missing.

E. Limited Knowledge

In the previous experiments, we have assumed full knowledge of the input datasets, which may be unrealistic. In practice, only part information of the dataset is available. This raises the question of how MGA performs when knowledge is limited. In this experiment, we consider two cases of limited knowledge: the first is ensuring the 1-hop neighbors of the target node and the rest links are randomly missing, and the second is that all links are randomly. Due to it's easy to attack low-degree nodes, we randomly select 10% nodes as the target nodes whose degrees are greater than the average value. Taking the Pol.Blogs dataset as an example, the effect of MGA is shown in Fig. 6.

From Fig. 6, we can see that MGA performs better than FGA when knowledge is limited, especially in the case of randomly missing. Besides, in the second case, the attack effect of both two methods is significantly better than that of the first case, suggesting that the target node is greatly affected by its neighbors. Moreover, we can find that as the missing probability increases, the attack effect of both methods decreases but is not significant. This may be because low-degree nodes are very vulnerable to attack, and only the target nodes with moderate degree values may have different results in different missing probability.

VI. CONCLUSION

In this paper, we propose a network adversarial attack algorithm MGA based on the momentum of gradient, which can fool network analysis methods and make the target nodes mis-analyzed, including node embedding, node classification and community detection. In this method, we first use the original network to train GCN model and extract the gradient of pairwise nodes based on the adversarial network. Then we calculate the momentum of gradient based on the gradients of previous iterations. By finding the the pair of nodes with maximum absolute momentum, we update the original network iteratively and generate the adversarial network. In the experiment we not only do uniform attack, hub-node attack and bridge-node attack on three network embedding methods, but also do community deception to to verify that MGA is

TABLE VI: The attack effects on community detection, in terms of ASR and AML, obtained by different attack methods on various network embedding methods and multiple datasets. Here, ASR is obtained by changing 20 links.

Dataset	NEM	ASR (%)						AML					
		MGA			FGA			MGA			FGA		
		Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect	Unlimited	Direct	Indirect
PloBook	DeepWalk	**100**	98.55	56.06	93.55	88.64	22.95	**5.33**	7.28	13.83	7.84	8.70	18.33
	node2vec	**100**	97.06	69.23	92.06	90.26	19.05	**5.12**	7.41	12.55	8.68	9.01	18.41
	GraphGAN	**100**	95.65	66.67	93.65	89.21	17.46	**5.31**	7.54	13.20	8.32	8.34	18.63
	LPA	**100**	97.37	60.53	96.37	95.36	24.35	**5.24**	5.31	13.42	5.94	6.03	18.21
	Louvain	**100**	94.74	55.26	98.38	94.12	21.10	**5.86**	6.05	13.62	5.91	6.39	18.38
	Average	**100**	96.67	61.55	94.80	91.52	20.98	**5.37**	6.72	13.32	7.34	7.69	18.39
Dolphin	DeepWalk	**100**	**100**	72.22	100	100	68.18	**3.52**	4.22	13.21	3.64	4.39	13.45
	node2vec	**100**	**100**	68.42	100	100	63.64	**3.41**	4.09	13.83	3.41	4.25	14.00
	GraphGAN	**100**	**100**	57.89	100	96.57	50.00	**3.94**	4.31	14.26	4.00	4.61	14.50
	LPA	**100**	**100**	57.89	100	100	55.78	**3.75**	3.90	13.21	3.82	3.98	13.42
	Louvain	**100**	**100**	63.16	100	98.54	58.86	**4.21**	**4.21**	12.53	4.02	4.32	13.86
	Average	**100**	**100**	63.92	100	99.02	59.29	**100**	**100**	63.92	**100**	99.02	59.29

not only useful for node classification attacks. Besides, We also consider the case of limited knowledge, and MGA is still successful.

Acknowledgment

References

[1] W. Chen and S.-H. Teng, "Interplay between social influence and network centrality: A comparative study on shapley centrality and single-node-influence centrality," in *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2017, pp. 967–976.

[2] C. Ding, F. Xia, G. Gopalakrishnan, W. Qian, and A. Zhou, "Teamgen: An interactive team formation system based on professional social network," in *International Conference on World Wide Web Companion*, 2017, pp. 195–199.

[3] Q. Xuan, Z.-Y. Zhang, C. Fu, H.-X. Hu, and V. Filkov, "Social synchrony on complex networks," *IEEE transactions on cybernetics*, vol. 48, no. 5, pp. 1420–1431, 2018.

[4] H. Huang, X. Shao, Y. Xie, T. Wang, Y. Zhang, X. Wang, and X. Deng, "An integrated genomic regulatory network of virulence-related transcriptional factors in pseudomonas aeruginosa," *Nature communications*, vol. 10, no. 1, p. 2931, 2019.

[5] K. B. Muchowska, S. J. Varma, and J. Moran, "Synthesis and breakdown of universal metabolic precursors promoted by iron," *Nature*, vol. 569, no. 7754, p. 104, 2019.

[6] K. An, Y.-C. Chiu, X. Hu, and X. Chen, "A network partitioning algorithmic approach for macroscopic fundamental diagram-based hierarchical traffic network management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1130–1139, 2017.

[7] X. Zhang and S. Mahadevan, "A bio-inspired approach to traffic network equilibrium assignment problem," *IEEE transactions on cybernetics*, vol. 48, no. 4, pp. 1304–1315, 2017.

[8] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han, "Graph-based consensus maximization among multiple supervised and unsupervised models," in *Advances in Neural Information Processing Systems*, 2009, pp. 585–593.

[9] L. Jian, J. Li, and H. Liu, "Toward online node classification on streaming networks," *Data Mining and Knowledge Discovery*, vol. 32, no. 1, pp. 231–257, 2018.

[10] S. Ping, D. Liu, B. Yang, Y. Zhu, H. Chen, and Z. Wang, "Batch mode active learning for node classification in assortative and disassortative networks," *IEEE Access*, vol. 6, pp. 4750–4758, 2018.

[11] J. Cheng, X. Wu, M. Zhou, S. Gao, Z. Huang, and C. Liu, "A novel method for detecting new overlapping community in complex evolving networks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–13, 2018.

[12] P. Govindan, C. Wang, C. Xu, H. Duan, and S. Soundarajan, "The k-peak decomposition: Mapping the global structure of graphs," in *Proceedings of the 26th International Conference on World Wide Web.* International World Wide Web Conferences Steering Committee, 2017, pp. 1441–1450.

[13] J. Cheng, W. Xiao, M. Zhou, S. Gao, and L. Cong, "A novel method for detecting new overlapping community in complex evolving networks," *IEEE Transactions on Systems Man and Cybernetics Systems*, vol. PP, no. 99, pp. 1–13, 2018.

[14] C. Fu, M. Zhao, L. Fan, X. Chen, J. Chen, Z. Wu, Y. Xia, and Q. Xuan, "Link weight prediction using supervised learning methods and its application to yelp layered network," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[15] A. Pecli, M. C. Cavalcanti, and R. Goldschmidt, "Automatic feature selection for supervised learning in link prediction applications: a comparative study," *Knowledge and Information Systems*, vol. 56, no. 1, pp. 85–121, 2018.

[16] Y. Cen, J. Zhang, G. Wang, Y. Qian, C. Meng, Z. Dai, H. Yang, and J. Tang, "Trust relationship prediction in alibaba e-commerce platform," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

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

[18] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[19] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[20] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "The space of transferable adversarial examples," *arXiv preprint arXiv:1704.03453*, 2017.

[21] E. Yang, T. Liu, C. Deng, and D. Tao, "Adversarial examples for hamming space search," *IEEE transactions on cybernetics*, 2018.

[22] 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.

[23] 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.

[24] Q.-Z. Cai, M. Du, C. Liu, and D. Song, "Curriculum adversarial training," *arXiv preprint arXiv:1805.04807*, 2018.

[25] Y. Duan, Z. Zhao, L. Bu, and F. Song, "Things you may not know about adversarial example: A black-box adversarial image attack," *arXiv preprint arXiv:1905.07672*, 2019.

[26] J. Tang, X. Du, X. He, F. Yuan, Q. Tian, and T.-S. Chua, "Adversarial training towards robust multimedia recommender system," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[27] M. Fang, G. Yang, N. Z. Gong, and J. Liu, "Poisoning attacks to graph-based recommender systems," in *Proceedings of the 34th Annual Computer Security Applications Conference.* ACM, 2018, pp. 381–392.

[28] A. Bojcheski and S. Günnemann, "Adversarial attacks on node embeddings," *arXiv preprint arXiv:1809.01093*, 2018.

[29] 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.

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

[31] S. Yu, M. Zhao, C. Fu, H. Huang, X. Shu, Q. Xuan, and G. Chen, "Target defense against link-prediction-based attacks via evolutionary perturbations," *arXiv preprint arXiv:1809.05912*, 2018.

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

[33] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[34] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995.

[35] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[36] B. D. Ripley and N. Hjort, *Pattern recognition and neural networks*. Cambridge university press, 1996.

[37] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[38] A. A. Gorodetsky and J. D. Jakeman, "Gradient-based optimization for regression in the functional tensor-train format," *Journal of Computational Physics*, vol. 374, pp. 1219–1238, 2018.

[39] Y. Qiao, B. P. Lelieveldt, and M. Staring, "An efficient preconditioner for stochastic gradient descent optimization of image registration," *IEEE transactions on medical imaging*, 2019.

[40] Y. Zhang, A. M. Saxe, M. S. Advani, and A. A. Lee, "Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning," *Molecular Physics*, vol. 116, no. 21-22, pp. 3214–3223, 2018.

[41] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9185–9193.

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

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

[44] 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," *arXiv preprint arXiv:1711.08267*, 2017.

[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] J. Chen, L. Chen, Y. Chen, M. Zhao, S. Yu, Q. Xuan, and X. Yang, "Ga-based q-attack on community detection," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 3, pp. 491–503, 2019.

[49] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," in *Privacy, security, and trust in KDD*. Springer, 2008, pp. 153–171.

[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] 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.

[52] Q. Xuan, J. Zheng, L. Chen, S. Yu, J. Chen, D. Zhang, and Q. Z. Member, "Unsupervised euclidean distance attack on network embedding," *arXiv preprint arXiv:1905.11015*, 2019.

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

[54] 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.

[55] 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.

[56] H. Dai, B. Dai, and L. Song, "Discriminative embeddings of latent variable models for structured data," in *International conference on machine learning*, 2016, pp. 2702–2711.

[57] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.

[58] Y. Yustiawan, W. Maharani, and A. A. Gozali, "Degree centrality for social network with opsahl method," *Procedia Computer Science*, vol. 59, pp. 419–426, 2015.

[59] A. Shimbel, "Structural parameters of communication networks," *The bulletin of mathematical biophysics*, vol. 15, no. 4, pp. 501–507, 1953.

[60] A. Bavelas, "A mathematical model for group structures," *Applied anthropology*, vol. 7, no. 3, pp. 16–30, 1948.

[61] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social networks*, vol. 32, no. 3, pp. 245–251, 2010.

[62] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.

[63] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[64] M. Newman, "Community detection in networks: Modularity optimization and maximum likelihood are equivalent," *arXiv preprint arXiv:1606.02319*, 2016.

[65] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical review E*, vol. 76, no. 3, p. 036106, 2007.

[66] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.

[67] M. E. Newman, "Modularity and community structure in networks," *Proceedings of the national academy of sciences*, vol. 103, no. 23, pp. 8577–8582, 2006.

[68] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[69] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[70] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Physical review E*, vol. 69, no. 6, p. 066133, 2004.