Adaptive Adversarial Attack on Graph Embedding via GAN

Jinyin Chen[1,2(✉)], Dunjie Zhang[2], and Xiang Lin[2]

[1] Institute of Cyberspace Security, Zhejiang University of Technology, Hangzhou 310023, China
chenjinyin@zjut.edu.cn
[2] College of Information Engineering, Zhejiang University of Technology, Hangzhou 310023, China

Abstract. Graph embedding method learns the low-dimensional representation of graph data, which facilitates downstream graph analysis tasks, such as node classification, graph classification, link prediction and community detection. With the in-depth study of graph analysis tasks, the issues of excessive data mining by graph embedding methods have become increasingly prominent, a number of graph embedding attack methods have been put forward. Inspired by promising performance of generative adversarial network, this paper proposes an adaptive graph adversarial attack framework based on generative adversarial network (AGA-GAN). We use the game between a generator and two discriminators with different functions to iteratively generate the adversarial graph. Specifically, AGA-GAN generates the adversarial subgraph according to different attack strategies to rewire the corresponding parts in the original graph, and finally form the whole adversarial graph. To address the scalability problem of existing graph embedding attack methods, we consider the adaptively selected K-hop neighbor subgraph as the attack target instead of the original graph. Experimental study on real graph datasets verifies that the AGA-GAN can achieve state-of-the-art attack performance in most node classifications.

Keywords: Adversarial attack · Generative adversarial network · Graph embedding · Node classification

1 Introduction

Our lives are surrounded by various graph data, which used to represent data in a lot of fields, such as social networks, communication networks, biological networks, transportation networks and so on. Graph embedding methods [1–3] map information of nodes and links in the graph into low-dimensional Euclidean space, enabling the real-world graph analysis tasks such as node classification [4, 5], graph classification [6, 7], link prediction [8, 9], and community detection [10, 11]. The graph embedding methods usually learn the low-dimensional representation of graph structure, which directly determines the performance of downstream tasks, so it has received increasing attention recently.

Y. Xiang et al. (Eds.): SocialSec 2020, CCIS 1298, pp. 72–84, 2020.
https://doi.org/10.1007/978-981-15-9031-3_7

With the widespread application of graph embedding methods in actual graph analysis tasks, many methods for downstream tasks have been proposed. Kipf et al. [12] proposed GCN as a basic graph convolution method for semi-supervised classification. This method is widely used in node classification and graph classification tasks. DIFFPOOL [6] uses a differentiable graph pooling module to adapt to various graph neural network architectures in a hierarchical and end-to-end manner. The DGCNN proposed by Zhang et al. [7] allows traditional neural networks to be trained on graph data. Deepwalk [13], node2vec [15], GCN [12] and other graph embedding methods also show superior performance over traditional algorithms in the task of link prediction. And new methods are constantly emerging in community detection tasks [16, 17].

The rapid emergence of graph embedding methods has also brought about the security problem of graph analysis tasks. In graph analysis task, failing the graph embedding model can also be achieved by faking the graph nodes, rewiring links or attributes modification, so as to protect the graph data from over-explorer. Zugner et al. [18] proposed the first adversarial attack on graph data, since then a number of attacks on node classification [19–21] have been studied. Some studies have focused more on other graph analysis tasks, such as community detection attack[22, 23] and link prediction attack [24, 25].

Generative adversarial network (GAN) [26] has achieved remarkable achievements in computer vision attacks [27, 28], natural language processing, audio recognition etc. GAN has also been used for graph data in recent years [29–31]. Since handling the graph data usually has scalability problem, it takes a lot of time and storage space to generate a full-size adversarial graph using GAN, which makes it difficult for GAN to achieve a fast and efficient attack on graph data. Most of the existing work focuses on how to better learn the embedding representation of graph data, and to our best knowledge, it is the first work on graph attacks via GAN.

In order to generate adversarial graph with minimal perturbation and maximal attack success rate, we propose an adaptive graph attack framework (AGA-GAN). Specifically, we design a multi-strategy attack generator (MAG), Similarity Discriminator (SD) and Attack Discriminator (AD) to form a three-player game. We consider the adaptively selected K-hop neighbor subgraph as the attack target instead of the original graph to reduce the cost of GAN in graph data attack. We attacked several graph embedding methods with node classification as downstream tasks and verified the effectiveness and universality of the proposed AGA-GAN on real graph data.

2 Related Work

2.1 Graph Attack Methods

The development of graph embedding models has given the graph analysis tasks a better theoretical basis. Massive real data are used for graph analysis, coming along security risks caused by excessive graph data analysis. In order to raise the security of graph data such as personal privacy [32], biomolecular structure [33], community structure [34] under the increasingly efficient graph data analysis methods, attack methods on graph data have been proposed to protect privacy from the excessive graph analysis.

Zugner et al. [18.] proposed the first adversarial attack against the graph data to generate the adversarial graph iteratively, namely NETTACK. This method focuses on the attack effect in the node classification, and achieves an effective attack within a limited budget. They further proposed Meta-Self [21], when the classification model and its training weights are unknown, regarding the graph as an optimizable hyper parameter, and using meta-gradients to solve the bi-level problem underlying training-time attacks. The FGA proposed by Chen et al. [19] extracts the gradients of node pairs based on the adversarial graph, and selects the node pairs with the largest absolute gradient to implement a fast gradient attack (FGA). It has strong attack transferability on various graph embedding methods. Chang et al. [20] built graph filters corresponding to the graph embedding models, and realized the attack in the black box environment through attacking the graph filters. Chen et al. [22] regarded community detection attacks as an optimization problem and proposed an attack strategy based on genetic algorithm and Q modularity. Yu et al. further proposed evolutionary perturbation attack (EPA) [23] based on genetic algorithm by rewiring the graph to achieve the attack.

2.2 GANs

Generative adversarial network (GAN) is a deep learning model proposed by Goodfellow et al. [26]. Since then it has become a powerful subclass of generative model [35] widely applied to image generation, text generation, semantic segmentation and other fields. A classical GAN is composed of a generator and a discriminator, using a two-player game idea, it can learn to deal with complex distribution problems through the mutual game between the generator and the discriminanor.

In the field of graph data, the studies of GAN mostly focus on learning the embedding representation of graph data. For instance, GraphGAN [29] uses the generator to learn the potential connectivity distribution in the graph data, and predicts the probability of the existence of a link between a pair of nodes by the discriminator, unifying the generation of the adversarial graph into the GAN's minimax game. Bojchevski et al. [30] proposed NetGAN, applying Wasserstein GAN to the graph field, learns the distribution of biased random walks on graph and generates credible random walks in real graph. Pan [31] further combined the variational graph autoencoder with GAN and proposed a framework ARVGE for learning graph embedding and being able to reconstruct graph data. In conclusion, they all adopt GAN as an efficient embedding representation learning method.

3 Preliminary

In this section, we briefly formulate the graph embedding and node classification attack problem. A graph is represented as $G = \{V, E, X\}$, where $V = \{v_1, \cdots, v_n\}$ is the node set with $|V| = N$, $e_{i,j} = < v_i, v_j > \in E$ denotes that there is a link between nodes v_i and v_j. The node topology of the graph is generally represented by the adjacency matrix $A \in \{0, 1\}^{N \times N}$, $A_{i,j} = 1$ if node v_i directly connected with v_j. $X \in \{0, 1\}^{N \times D}$ is the node attributes matrix, and D denotes the dimension of X. Generally, the adjacency matrix A contains the information of V and E in the graph data, so we use $G = (A, X)$ to represent a graph more concisely.

Graph Embedding. The graph embedding methods map the graph data G into an embedding matrix $Z \in R^{N\times d}$ in a low-dimensional space, while retaining the information of the adjacency matrix A and the node attributes X. The dimension of d is much smaller than N, which allows graph data to use the embedding matrix to design downstream methods to implement graph analysis tasks such as node/graph classification, link prediction and community detection.

Node Classification Attack. Given a graph G and target node v_i. $F = [\tau_1, \cdots, \tau_{|F|}]$ is the category set of nodes, $\tau_{i_{ori}} \in F$ denote the ground true category of the target node v_i. Our goal is to generate the adversarial graph G', which makes the target node v_i can get a prediction category $\tau_{i_{att}}$ with the largest distance from $\tau_{i_{ori}}$ through the target node classifier f_θ^{node}:

$$\arg \max_{\tau_{i_{att}} \neq \tau_{i_{ori}}} \ln Z^*_{v_i,\tau_{i_{att}}} - \ln Z^*_{v_i,\tau_{i_{ori}}} \tag{1}$$

where $Z^* = f_\theta^{node}(G')$, θ denotes the parameter of the target model training with the real graph G.

4 Method

Our proposed AGA-GAN attacks node classification by combining different attack strategies. Figure 1 shows the attack process of AGA-GAN, which consists of three parts: multi-strategy attack generator (MAG), similarity discriminator (SD) and attack discriminator (AD). We choose the K-hop neighbor subgraph of the target node in the original graph as the attack target. Through the alternating training of MAG, SD and AD, AGA-GAN chooses to generate adversarial subgraph structure A' or adversarial node attribute X' according to different attack strategies. Then we replace the corresponding part in the original graph with the adversarial subgraph, achieve an effective node classification attack.

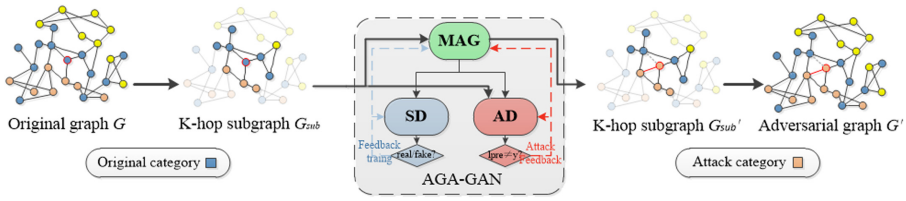

Fig. 1. Process of AGA-GAN adaptive attacking original Graph G. We adaptively set the input size of AGA-GAN according to the K-hop($K = 2$ in here) neighbor subgraph of the target node(normally colored nodes and links). AGA-GAN generates the adversarial subgraph and replaces the corresponding part of the original graph. (Color figure online)

4.1 Multi-strategy Attack Generator

Structure of MAG. The MAG we propose achieves adaptive generation of adversarial graph data through different attack strategies. The MAG contains two parts, feature extractor and a graph reconstructor.

Feature Extractor. In order to learn graph structure A and node attributes X in our proposed AGA-GAN, we consider a two-layer graph convolution network (GCN) as the graph feature extractor. It maps the graph structure and node attribute information to a d-dimensional feature matrix. The low-dimensional features of graph data are defined as:

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

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$, A is the adjacency matrix and $\tilde{A} = A + I_N$ is the adjacency matrix of the real graph G with the added self-connections. I_N is the identity matrix and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ denotes the degree matrix of $\tilde{A}$.$W^{(0)} \in R^{N\times H}$ and $W^{(1)} \in R^{H\times d}$ denote the trainable weight matrix of hidden layer and output layer with H feature maps, N is the number of nodes in the graph, and d denotes the dimension of low-dimensional representation. f and σ are the softmax function and Relu active function.

Graph Reconstructor. After obtaining the low-dimensional representation Z of the graph data through the graph feature extractor, we use a dimension expansion matrix W_{ex} to reconstruct Z into the adversarial graph G':

$$G' = \begin{cases} A' = \bar{\|}[\mathcal{S}((ZW_{ex}^{A} + (ZW_{ex}^{A})^{T})/2)] \\ X' = \bar{\|}[\mathcal{S}(ZW_{ex}^{X})] \end{cases} \tag{3}$$

where $Z \in R^{N\times d}$, $W_{ex}^{A} \in R^{d\times N}$ and $W_{ex}^{X} \in R^{d\times D}$ are the dimension expansion matrix of grpah structure A and node attributes X. Sigmoid function $\mathcal{S}$ maps the element values of generated data between [0–1], then obtains discrete G' by the sign function $\bar{\|}$.

Multiple Training Strategies. In order to satisfy the AGA-GAN's requirements for different graph analysis attack strategies, as shown in Fig. 2, we determine how MAG generates graph structure A or node attributes X based on different attack strategies.

Modify Strategy. In MAG, we implement various attack strategies by modifying graph structure A, node attributes X or a combination of both.

Graph Structure Attack: In a general social network, the structure of the social network represents the interaction relationship between users. Modifying the links between nodes in the graph can effectively hide these relationships.

Node Attributes Attack: In community detection tasks, similar users usually have similar node attributes. By modifying the attributes of the target node itself or its connected nodes, the node information can also be hidden.

Hybrid Attacks: Modify both graph structure A and node attributes X to avoid too much perturbation in the graph structure or node attributes, and ensure that the attack is unnoticeable while playing an effective attack.

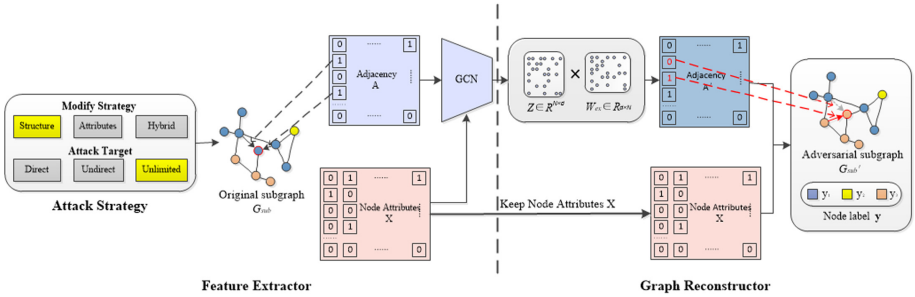

Fig. 2. MAG in node classification attack. The red solid line indicates the added links, and the gray dotted line indicates the deleted links. Here we choose an unlimited attack on the graph structure. We obtain the low-dimensional features of the graph through the feature extractor, and then obtain our adversarial graph structure through the graph reconstructor. (Color figure online)

Attack Scale. In order to efficiently implement the adversarial attack on the graph and reduce the attack cost, we use K-hop graph instead of the original graph in the attack process to achieve an efficient attack. Specifically, we select the target node and its K-hop neighbors from the original network G to form a K-hop subgraph $G_{K-sub}(A_{sub}, X_{sub})$. To prevent the node categories in the subgraph from being too concentrated when K is too small, which leads to poor attack effects, we randomly add nodes with other categories that are 20% of the number of subgraph nodes to the G_{K-sub}, then adaptively generate the adversarial subgraph $G'_{K-sub}(A'_{sub}, X'_{sub})$ corresponding to the size of G_{K-sub} through MAG, when the attack on subgraph is successful, replace the subgraph G_{K-sub} with G'_{K-sub} in the original graph G to obtain the adversarial graph $G' = (A', X')$. We consider the following three different attack scale $k \in N_+$:

Direct Attack($k = 1$): Only delete the existing links of the target node or add a new one, or modify the target node's attributes.
Indirect Attack($k \leq K, k \neq 1$): Delete or add links in the 2-to-K hop node pairs except the target node in subgraph G_{K-sub}, or modify these nodes' attributes.
Unlimited Attack($k \leq K$): Combining the above two attack scales, delete or add links between any pair of nodes in G_{K-sub}, or modify the attributes of any node.

4.2 Similarity Discriminator

SD aims to learn the difference between G_{K-sub} and G'_{K-sub}, and distinguish the two as much as possible. SD also provides feedback to the MAG and guides it to generate the adversarial subgraph that is more similar to the original one.

Structure of SD. We use a classical Multi-layer Perceptron (MLP) with a hidden layer as our SD, where the output layer is set to a one-dimensional sigmoid function. The hidden layer and the output layer in SD can be generally expressed as:

$$h^{(l+1)} = sigmoid(W_{SD}^{(l)} h^{(l)} + b^{(l)}) \tag{4}$$

where $W_{SD}^{(l)}$ and $b^{(l)}$ are the trainable weight matrix and bias term of l layer. We use the subgraph structure A_{sub} or the node attributes X_{sub} as the input $h^{(0)}$ of the SD according to the modification strategy, and calculate the hidden layer's output $h^{(1)}$ as the input of the output layer, then get a one-dimensional value $h^{(out)} \in [0, 1]$ from the sigmoid function.

Training Steps. During the training process, MAG tries to generate a more realistic subgrpah to fool the SD, and the SD needs to maximize the difference between G_{K-sub} and G'_{K-sub}. The optimization objective of alternating training of the SD and the MAG can be defined by:

$$\min_{MAG} \max_{SD} \mathbb{E}_{G \sim p_{real}}[\log SD(G_{K-sub})] + \mathbb{E}_{G' \sim p_{MAG}}[\log(1 - SD(G'_{K-sub}))] \tag{5}$$

where $G_{K-sub} \sim p_{real}$ and $G'_{K-sub} \sim p_{MAG}$ denotes original subgraph and adversarial subgraph generated by MAG.

4.3 Attack Discriminator

In node classification attack, we use the GCN model with the same structure as the MAG as the AD. Through the alternating training of MAG and SD, we obtain the adversarial subgraph that is similar to the real one. AD then provides feedback to MAG and guides it to generate adversarial subgraph which can fail the target model.

Traning Steps. For an effective attack on the target model, AD performs the following two steps in each iteration:

Step1. Freezing the weights of MAG and SD, train the weights of the AD using the real subgrpah G_{K-sub}, and then we optimize the AD by minimizing the cross-entropy loss function to improve the accuracy of AD in classifying nodes in real subgraph:

$$\arg\min L_{AD} = -\sum_{l=1}^{|T_s|} \sum_{k=1}^{|F|} Y_{lk} \mathrm{In}(Z_{lk}(A_{sub}, X_{sub})) \tag{6}$$

where T_s is the set of labled nodes, $F = [\tau_1, \cdots, \tau_{|F|}]$ denotes the category set of nodes, $Y_{lk} = 1$ if node v_l belongs to category τ_k and $Y_{lk} = 0$ otherwise, $Z_{lk}(A, X)$ is the category prediction confidence output calaculated by Eq (2) when $d = |F|$.

Step2. Freezing the weights of SD and AD, and using the AD obtained in *Step1* to fine-turn train the MAG. We get the predicted category confidence of the adversarial subgraph through the two-layer GCN trained in *Step1*, and define the attack loss function:

$$\arg\min L_{MAG} = -\sum_{l=1}^{|T_{tar}|} \sum_{k=1}^{|F|} Y_{lk} \mathrm{In}(1 - Z'_{lk}(A'_{sub}, X'_{sub})) \tag{7}$$

where T_{tar} is the set of attack nodes, $Y_{lk} = 1$ if node v_l belongs to category τ_k and $Y_{lk} = 0$ otherwise.

The optimization objective of alternating training of the AD and the MAG can be defined by:

$$\min_{MAG} \max_{AD} \mathbb{E}_{G_{K-sub} \sim p_{real}}[\log AD^F(G_{K-sub})] + \mathbb{E}_{G'_{K-sub} \sim P_{MAG}}[\log(1 - AD^F(G'_{K-sub}))] \tag{8}$$

where $AD^F(\cdot)$ denotes the AD with F as the node category set.

5 Experiments

In order to testify the effectiveness of our AGA-GAN, we attack the graph embedding models with node classification as the downstream tasks, and compare the results with some baseline attack methods. In each attack, we set the ratio of training times of MAG, SD and AD is 1: 1: 1. For each attacked node, we generate 20 adversarial graphs. Once a confrontation graph can successfully attack the node classification, we consider that the attack was successful our experimental environment consists of i7-7700 K 3.5 GHzx8 (CPU), TITAN Xp 12 GiB (GPU), 16 GB $\times$ 4 memory (DDR4) and Ubuntu 16.04 (OS).

5.1 Dataset and Baseline Methods

Dataset. In the node classification, each node in the graph has a category. We evaluated our method on three real-world datasets: Pol. Blogs [34], Cora [35] and Citeseer [36]. The nodes denote blogs/documents, and the links are blog links/citations. Their basic statistics are shown in Table 1.

Table 1. The basic statistics of the three graph datasets.

Dataset	#Nodes	#Links	#Classes
Pol.Blogs	1490	19090	2
Cora	2708	5427	7
Citeseer	3312	4732	6

Baseline Methods. We compare our AGA-GAN with three graph embedding attack methods:

Dice [36]: DICE randomly disconnect b links of target node, then randomly connect the target node to $M - b$ nodes of different categories.
Nettack [18]: generates adversarial disturbances for graph structure and node attributes, and according to the degree distribution and attributes co-occurrence probability to remain the perturbations are unnoticeable.
GF-Attack [20]: GF-Attack attacks graph embedding models by constructing corresponding graph filters and attacking it in a black box background.

5.2 Attack Performance

For each graph, we randomly select 20 nodes in each category as the target nodes. We give the attack effect of the proposed AGA-GAN method under different attack strategy settings, and compare it with several baseline methods. We use the following three metrics to measure the attack effectiveness.

Attack Success Rate(ASR). ASR [19] is the ratio of targets which will be successfully attacked within a given fixed budget, the ASR is defined as

$$\text{ASR} = \frac{\text{Number of successful attack nodes}}{\text{Number of attack nodes}} \tag{9}$$

Average Modified Links (AML). AML [19] is designed for the structure attack, which indicates the average links perturbation size leading to a successful attack.

$$\text{AML} = \frac{\text{Number of modified links}}{\text{Number of attack nodes}} \tag{10}$$

Average Modified Attributes (AMA). AMA is designed for the attributes attack, which indicates the average attributes perturbation size leading to a successful attack.

$$\text{AMA} = \frac{\text{Number of modified attributes}}{\text{Number of attack nodes}} \tag{11}$$

Selection of K and Attack Scale *k*. In this part, considering that the useful information of the target node mostly exists in its neighborhood, we select $K \in N_+, K \leq 5$ to get proper G_{K-sub}. We compare the average subgraph sizes under different K. We can see from Fig. 3 that when K is less than 3, the subgraph of the Cora and Citeseer dataset only contains less than 2% of the original graph, we consider the information contained in the subgraph is not enough for the subgraph attack to replace the original graph attack.

We further combine different modification scales k to observe the attack effect. We choose $K = 3$ as the initial setting of the three attack scales. When K increases, we will also increase k to perform the corresponding three-scale attacks. Figure 3 also shows the ASR under different attack scales. The solid line represents our unrestricted attack, and the dashed line is an indirect attack. Since the initial value of K is 3, when k = 1,

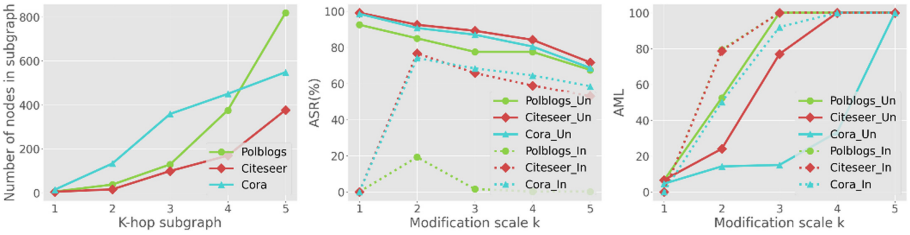

Fig. 3. The subgraph size under different K and the ASR/AML under different modificantion scales k. Here we do not limit the number of modifications to get ASR, when the number of modifications is greater than 100, AML is set to 100.

Table 2. The 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	Model	ASR(%)				AML			
		Ours	Baseline			Ours	Baseline		
		AGA-GAN	DICE	NETTACK	GF-Attack	AGA-GAN	DICE	NETTACK	GF-Attack
Pol. Blogs	GCN	**92.50**	50.27	82.97	19.89	**6.47**	11.85	11.89	20
	Deepwalk	**85.50**	64.52	75.41	12.82	**7.21**	12.35	10.06	20
	LINE	**85.00**	66.74	76.35	23.48	**7.98**	12.82	10.26	20
	Average	**87.67**	60.51	78.25	18.73	**7.22**	12.34	10.74	20
Cora	GCN	**98.57**	54.95	92.87	82.55	6.62	9.13	**6.09**	20
	Deepwalk	**96.43**	93.52	94.06	63.47	**6.71**	7.20	7.24	20
	LINE	95.71	88.99	**96.34**	83.19	**6.64**	7.66	7.02	20
	Average	**96.90**	79.15	94.42	76.40	6.66	7.99	6.78	20
Citeseer	GCN	**99.17**	70.37	87.50	61.78	**4.53**	9.87	6.88	20
	Deepwalk	**98.33**	93.44	96.96	50.87	**6.18**	7.08	7.06	20
	LINE	**99.17**	96.72	95.82	60.41	6.42	7.21	**6.02**	20
	Average	**98.89**	86.84	93.42	57.69	**5.71**	8.05	6.65	20

we are actually conducting a direct attack, at this time we set the ASR and AML of the indirect attack to 0.

We can see that our unlimited attack has better results than indirect attack. When $k = 1$, i., e, when performing a direct attack, the highest ASR and the lowest AML are obtained. Interestingly, we found that as the scale of the attack increases, while AML increased, ASR decreased significantly in Fig. 3. We consider this because in G_{K-sub}, the effect of indirect links on the target node is different from that in G. Modifying the indirect links in the G_{K-sub} can misclassify the target node, but the impact of these modifications may not be that effective in G.

Compared with the Baseline Attack Methods. According to the experimental results above, we set $K = 3$ in AGA-GAN and use direct attack, i.e., $k = 1$. We compare with several other baseline attack methods using the most effective direct attack AGA-GAN, and the attack results are shown in Table 2, we can see that AGA-GAN outperforms all the other attack methods in all the cases, in terms of higher ASR and lower AML. However, when attacking other graph embedding methods, the ASR of AGA-GAN may slightly lower than that of attacking GCN, which is different from other baseline attack methods. This may be because the subgraph is missing part of the original graph information, which is further expanded when attacking other graph embedding methods.

Different Modification Strategies. We also have node attributes attack and hybrid attack and compare the AGA-GAN-ori without adding random nodes to the k-hop subgraph. Here we also set $K = 3$ and use direct attack. When attacking node attributes, we limit the modification of up to 100 attributes to obtain ASR. In each epoch of the hybrid attack, the MAG generates the adversarial subgraph structure A'_{sub}, and then generates the adversarial node attributes X'_{sub} based on A'_{sub}. From Table 3, we can find that the attack on node attributes can only obtain less than 50% ASR. This may be that the node attributes of the citation datasets are relatively sparse, and the node categories are more

determined by the graph structure. When we use a hybrid attack, we can get the highest ASR while reducing the AML by modifying the node attributes. Similar results can be observed in AGA-GAN-ori, however, the ASR of AGA-GAN-ori is much lower than that of AGA-GAN. This is because the category distribution of the nodes is relatively concentrated in the original neighbor subgraph of the target node, which makes the attack very difficult. This proves that our strategy of randomly adding nodes to the subgraph is effective.

Table 3. The ASR, AML and AMA obtained by AGA-GAN and AGA-GAN-Orisub attacking GCN model with different attack strategies. Here, ASR is obtained by changing 20 links or 100 attributes.

Attack method	Dataset	ASR(%)			AML		AMA	
		A	X	Hybrid	A	Hybrid	X	Hybrid
AGA-GAN	Pol. Blogs	92.50	67.50	**95.00**	3.34	**2.21**	5.63	**4.30**
	Cora	98.57	49.29	**99.29**	6.62	**5.74**	**12.88**	13.93
	Citeseer	99.17	61.67	**100**	4.53	**2.99**	**33.48**	34.97
AGA-GAN-ori	Pol. Blogs	67.50	47.50	**57.50**	18.23	**17.12**	**9.14**	10.53
	Cora	66.43	27.86	**35.00**	**3.58**	3.72	7.48	**7.04**
	Citeseer	69.17	45.83	**47.50**	5.52	**4.97**	21.97	**19.47**

Time Efficiency of Attack. Most existing methods iteratively generate adversarial perturbations on the original graph one by one to obtain the adversarial graph, which is essentially different from AGA-GAN directly generating a complete adversarial graph. In Fig. 4, we compare the training time spent by each iteration when the AGA-GAN attacks graph data of different sizes. As the graph data size increases, the training time also increases greatly. Since our K-hop neighbor subgraph size is smaller than 20% of the original graph, which means that our adaptive strategy of attacking K-hop neighbor subgraph can effectively reduce the attack cost.

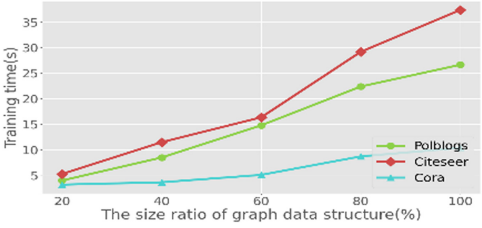

Fig. 4. The time spend in each iteration when attacking different graph sizes.

6 Conclusion

In this paper, we propose an adaptive graph adversarial attack framework based on generative adversarial network(AGA-GAN). We designed MAG, SD and AD. According to different modification strategies and attack scales, MAG trains alternately with SD and AD respectively, and generates the adversarial graph similar to the real graph, which can successfully attack graph embedding models. In order to reduce the attack cost of GAN on graph data, we consider the adaptively selected K-hop neighbor subgraph as the attack target instead of the original graph. We compared the attack effects of AGA-GAN under different attack strategies, and attacked several graph embedding methods with node classification as downstream task. The experimental results show that AGA-GAN can achieve state-of-the-art attack performance in node classification when only using a small part of the original graph's structure and node attribute information.

References

1. Cai, H., Zheng, V.W., Chang, K.: A comprehensive survey of graph embedding: problems, techniques and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
2. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: a survey of approaches and applications. IEEE Trans. Knowl. Data Eng. **29**(12), 2724–2743 (2017)
3. Choi, E., Bahadori, M.T. Song, L., Stewart, W.F., Sun, J.: Gram: graph-based attention model for healthcare representation learning. In: The ACM SIGKDD International Conference, pp. 787–795 (2017)
4. Tang, J., Qu, M., Mei, Q.: Pte: predictive text embedding through large-scale heterogeneous text networks, pp. 1165–1174 (2015)
5. Wang, S., Tang, J., Aggarwal, C., Liu, H.: Linked document embedding for classification. In: Proceedings of the 25th ACM international on conference on information and knowledge management pp. 115–124 (2016)
6. Ying R., et al. Hierarchical graph representation learning with differentiable pooling. In: Advances in Neural Information Processing Systems, pp. 4800–4810 (2018)
7. Gibert, D., Mateu, C., Planes, J.: An end-to-end deep learning architecture for classification of malware's binary content. In: Kůrková, V., Manolopoulos, Y., Hammer, B., Iliadis, L., Maglogiannis, Ilias (eds.) ICANN 2018. LNCS, vol. 11141, pp. 383–391. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01424-7_38
8. Wang, S., Tang, J. Aggarwal, C. Chang, Y. Liu, H.: Signed network embedding in social media. In: SDM, pp. 327–335 (2017)
9. Tian, F., Gao, B., Cui, Q., Chen, E., Liu, T.Y.: Learning deep representations for graph clustering. In: Twenty-Eighth AAAI Conference on Artificial Intelligence, pp. 1293–1299 (2014)
10. Allab, K., Labiod, L., Nadif, M.: A semi-nmf-pca unified framework for data clustering. IEEE Trans. Knowl. Data Eng. **29**(1), 2–16 (2017)
11. Liu, L., Cheung, W.K., Li, X., Liao, L.: Aligning users across social networks using network embedding. In IJCAI, pp. 1774–1780 (2016)
12. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
13. Masashi, T., Kentaro, T., Jun, S.: Compound-protein Interaction Prediction with End-to-end learning of neural networks for graphs and sequences. Bioinformatics **35**(2), 309–318 (2018)

14. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710 (2014)
15. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: SIGKDD, pp. 855–864 (2016)
16. Newman, M.E.J.: Fast algorithm for detecting community structure in networks. Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top **69**(6), 066133 (2004)
17. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. J. Stat. Mech., Theory Exp. **2008**(10), 10008 (2008)
18. Zügner, D., Akbarnejad, A., Günnemann, S.: Adversarial attacks on neural networks for graph data. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, 19–23 August 2018, pp. 2847–2856 (2018)
19. Chen J, Wu Y, Xu X, et al. Fast Gradient Attack on Network Embedding (2018)
20. Chang H., et al. A restricted black-box adversarial framework towards attacking graph embedding models. (2019)
21. Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning (2019)
22. Chen, J., et al.: GA based Q-attack on community detection. IEEE Transactions on Computational Social Systems **6**(3), 491–503 (2018)
23. Chen J, Chen Y, Chen L, et al.: Multiscale evolutionary perturbation attack on community detection 2019
24. Milani Fard, A., Wang, K.: Neighborhood randomization for link privacy in social network analysis. World Wide Web **18**(1), 9–32 (2013). https://doi.org/10.1007/s11280-013-0240-6
25. Chen J, Shi Z, Wu Y, et al. Link Prediction Adversarial Attack (2018)
26. Goodfellow, I.J, Pouget-Aabadie, J., Mirza, M., et al.: Generative adversarial networks. In: Proceedings of International Conference on Neural Information Processing Systems Kuching:, pp. 2672–2680 (2014)
27. Mangla, P., Jandial, S., Varshney, S., et al. AdvGAN ++: Harnessing latent layers for adversary generation. In: Proceedings of the IEEE International Conference on Computer Vision Workshops 2019
28. Zhu, Z.A., Lu Y.Z., Chiang C.K.: Generating adversarial examples by makeup attacks on face recognition. In: 2019 IEEE International Conference on Image Processing (ICIP). IEEE (2019)
29. Wang, H., Wang, J., Wang, J., et al. GraphGAN: Graph representation learning with generative adversarial nets. IEEE Trans. Knowl. Data Eng. (2017)
30. Bojchevski, A., Shchur, O., Zügner, D., et al.: NetGAN: generating graphs via random walks (2018)
31. Pan, S., Hu, R., Long, G., et al.: Adversarially regularized graph autoencoder for graph embedding (2018)
32. Razavi, M.N., Iverson, L.: Improving personal privacy in social systems with people-tagging. In: Proceedings of the 2009 International ACM SIGGROUP Conference on Supporting Group Work, GROUP 2009, Sanibel Island, Florida, USA, May 10–13, 2009. ACM (2009)
33. Garcia, J.O., Ashourvan, A., Muldoon, S.F., et al.: Applications of community detection techniques to brain graphs: algorithmic considerations and implications for neural function. Proc. IEEE **106**, 1–22 (2018)
34. Nagaraja, S.: The Impact of Unlinkability on Adversarial Community Detection: Effects and Countermeasures Privacy Enhancing Technologies. Springer, Berlin Heidelberg (2010)
35. Lucic, M., Kurach, K., Michalski, M., Gelly, S., Bousquet, O.: Are gans created equal? a large-scale study. In: Advances in Neural Information Processing Systems, pp. 700–709 (2018)
36. Waniek, M., Michalak, T., Rahwan, T.: et al. Hiding Individuals and Communities in a Social Network. Nat. Hum. Behav. (2016)