RoGAT: a robust GNN combined revised GAT with adjusted graphs

Xianchen Zhou Hongxia Wang*

Abstract

Graph Neural Networks(GNNs) are useful deep learning models to deal with the non-Euclid data. However, recent works show that GNNs are vulnerable to adversarial attacks. Small perturbations can lead to poor performance in many GNNs, such as Graph attention networks(GATs). Therefore, enhancing the robustness of GNNs is a critical problem. Robust GAT(RoGAT) is proposed to improve the robustness of GNNs in this paper, . Note that the original GAT uses the attention mechanism for different edges but is still sensitive to the perturbation, RoGAT adjusts the edges' weight to adjust the attention scores progressively. Firstly, RoGAT tunes the edges weight based on the assumption that the adjacent nodes should have similar nodes. Secondly, RoGAT further tunes the features to eliminate feature's noises since even for the clean graph, there exists some unreasonable data. Then, we trained the adjusted GAT model to defense the adversarial attacks. Different experiments against targeted and untargeted attacks demonstrate that RoGAT outperforms significantly than most the state-of-the-art defense methods. The implementation of RoGAT based on the DeepRobust repository for adversarial attacks.

1 Introduction

Non-Euclid data occurs widely in our daily life. Graph Neural Networks(GNNs) are proposed to deal with these data represented by graphs. GNNs corresponding to node classification problems are usually tightly connected with two permutation invariant primitives[1][2]: a local aggregation function which updates each nodes representation by combining the features in its neighbors and a readout function that combines all learned node embeddings to obtain a graph representation. The two procedures can be concluded in the framework of GraphSAGE[3]. For example, the Graph Convolution Network(GCN)[4][5][6] are kinds of GrahSAGE. Furthermore, the attention mechanisms, e.g self-attention[7] and soft-attention[8], have significant influence on deep learning, Therefore, the graph attention network(GAT)[9] proposed with the attention mechanism in aggregation function achieved superior results in node classification and link prediction problem. The attention mechanism made the model focus on the most relevant parts. GAT model also computes the hidden representation by the features of connected nodes, and the attention weight coefficients, which are calculated by the self-attention applied to the features. Although GAT is a superior GNN in dealing with graph data, it is also vulnerable to adversarial attacks like ordinary GNNs. One reason is that like GNN, the aggregation function for GAT is tightly connected with graph structure and features[10][11], which are directly disturbed by attacks. Another reason is that GAT only calculates the attention coefficients by the neighbors' features. The attention score cannot distinguish whether the neighbor's is real or generated by attacks. The performance of GAT might be degraded significantly by the adversarial attacks. Therefore, as one type of GNN, how to improve the performance of GAT under the adversarial attacks achieves much attention.

Recently, there are some research about the adversarial attacks and defense on GNNs. The graph adversarial attacks can be divided into targeted attacks and untargeted attacks. The targeted attacks like nettack[12] and RL-S2V[10] tend to let the trained model misclassify small set of test samples, while untargeted attacks like metatack aim to let the trained model have bad overall performance on all test data. Nettack introduced the unnoticeable perturbations on both structures and features. RL-S2V used

*College of Science, National University of Defense Technology, Changsha, Hunan, 410073, P.R.China. Corresponding author. Email: wanghongxia@nudt.edu.cn

reinforcement learning to generate attacks on GNNs. The metattack parametrized the graph structure and used the gradient information to attack GNNs.

As to the method to defense the adversarial attacks, one perspective is establishing a Max-Min problem. Xu[13] used Projected Gradient Descent(PGD) to obtain the smallest loss under the attack. Another perspective to defense the attacks is to eliminate the influence of perturbations such as adding or removing the adversarial edges or clearing up the change of node features. The prior information of graph in specific applications can give the criteria of revision.

For example, Wu[11] applied the Jaccard similarity to eliminate the edges between nodes with low similarity. RGCN in [14] added the penalization of adversarial edge and modeled the hidden layers by Gaussian distributions to reduce the effect of attacks. ProGNN[15] assumed that the graph should be low-rank and sparse and then gave a progressive model for adversarial training, which is efficient but time-consuming. These models used the poisoned graph for training and estimated the clean graph by prior information[16][14]. Besides, GNNguard[17] estimated an importance weight for every edge to reduce the influence of fake edges.

Although these defensive methods improve the robustness of GNN against different attacks, the robustness of GAT has not been discussed thoroughly. As GAT uses the attention mechanisms to mix various features, the intuition is that GAT can adjust the attention scores for fake and edges automatically to resist the attacks during the training procedure. However, the performance of GAT against adversarial attacks degrades obviously.

In this paper, we focus on how to design an effective GAT defending the poisoning attack that a graph is perturbed before training. We aim to design an improved GAT to adjust the attention scores for real edges and adversarial edges. We are faced with two problems: (1) how to distinguish between the fake and real edges in the training procedure. (2) what skills should we take to adjust the attention scores for different edges? Therefore, this paper proposed a Robust GAT(RoGAT) to defend adversarial attacks by increasing extra edge attention scores to distinguish real from adversarial edges. Using feature smoothness assumption, the ratio of attention scores between real edges and adversarial edges increases in the training procedure, which means that the influence of adversarial attacks is eliminated. The experiments on various real-world graphs show that RoGAT can adjust the ratio of attention scores between fake and real edges iteratively and outperform other defense methods in the node classification task under different types of attacks.

The rest of the paper is organized as follows. Section 2 reviews the original GAT and discusses the relative merits of GAT. And RoGAT is proposed to improve the performance of defending attacks in section 3 Section 4 gives some experiments to verify the conclusion. Section 5 gives a further discussion of our methods. The last part gives the acknowledgment.

2 The proposed framework

2.1 Notations

Let $G = (V, E)$ be a graph, where $V = \{v_1, v_2, \cdots, v_N\}$ is the set of nodes and E is the set of edges. Each graph can be represented by the adjacency matrix $A \in \mathbb{R}^{N\times N}$, a_ij where the (i, j)th elements of A represents the link weight of node v_i and v_j. In addition, $X = [x_1, x_2, \cdots, x_N]^\top \in \mathbb{R}^{N\times d}$ is used to denote the node feature matrix where x_i is the feature vector of the node v_i. Hence a simple representation of a graph is $G = (V, A, X)$.

Here we consider the semi-supervised node classification problem. Only parts of node $\mathcal{V}_p = \{v_1, v_2, \cdots, v_m\}$ are part of labels with annotations. $\mathcal{Y}_p = \{y_1, y_2, \cdots, y_m\}$ where y_i is the label of v_i. The goal of node classification is that we use the given graph $G = (V, A, X)$ with partial label $\mathcal{Y}_p$ to predict the labels of unlabeled nodes.

2.2 GAT

As one type of Graph Neural Network(GNN), graph attention network(GAT) is a powerful method to solve machine learning problems on the graph. The basic architecture of GNNs consists of multi-layers that form the new information of one node by combining the feature vector of each node in its neighbors. GNNs can be represented by *aggregation* $f_{\text{aggregation}}$ and *combination* $f_{\text{combination}}$ functions. In formulation, for a GNN with L layer, the update rule of node v is given by:

$$a_v^{(k)} = f_{\text{aggregation}}^{(k)}\left(\left\{x_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right) \tag{1}$$

$$x_v^{(k)} = f_{\text{combination}}^{(k)}\left(x_v^{(k-1)}, a_v^{(k)}\right), \tag{2}$$

where $k = 1, \cdots, L$, $x_v^{(0)}$ is the input feature of x_v and $\mathcal{N}(u)$ represents the neighbors of node u As for GAT, it uses the summation for aggregation and combination. And the weight α is computed by the features of neighbors. The formulation of GAT can be denoted by

$$x_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(k-1)} \mathbf{W} x_u^{(k-1)}\right), \tag{3}$$

where

$$\alpha_{uv}^{(k-1)} = \frac{\exp\left(\text{Leaky ReLU}\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W}x_v^{(k-1)} \| \mathbf{W}x_u^{(k-1)}\right]\right)\right)}{\sum_{k \in \mathcal{N}(v)} \exp\left(\text{LeakyReLU}\left(\overrightarrow{\mathbf{a}}^T\left[\mathbf{W}x_v^{(k-1)} \| \mathbf{W}x_k^{(k-1)}\right]\right)\right)} \tag{4}$$

. $\mathbf{W}$ is the learning parameters. σ is the activation function.

And the multi-head attention is used to improve the performance of GAT.

$$x_v^{(k)} = \|_{m=1}^{M} \sigma\left(\sum_{u \in \mathcal{N}(v)} (\alpha_{uv}^{(k-1)} \mathbf{W})^m x_u^{(k-1)}\right), \tag{5}$$

where $\|$ represents concatenation. The update feature $x_v^{(k)}$ relies on the neighbors' features $x_u^{(k-1)}$ and the weight α_{ij} are computed by the features.

Denote all the learning parameters of GAT by θ. θ includes all the W in each layer. Then for node classification problem, GAT learns a function $f_\theta : \mathcal{V} \to \mathcal{Y}$ by applying Boolean classification function to $x_v^{(L)}$ to predict unlabeled nodes. And the objective function are the sum of loss in labeled nodes,

$$\min_\theta \mathcal{L}_{GNN}(\theta, A, X, \mathcal{Y}_p) = \sum_{v_i \in \mathcal{V}_p} \ell(f_\theta(X, A)_i, y_i), \tag{6}$$

where θ is the parameters of GNN and $(f_\theta(X, A)_i$ is the prediction result of node v_i. $\ell(\cdot, \cdot)$ is a loss function between the prediction and known label of nodes.

2.3 Analysis of GAT

Since GATs learns the new feature of the node by aggregating the neighbors' features to update the representation of node, it faces a problem that if all the features of neighbors make positive effects to obtain the update node feature. And what criteria can be used to revise the neighbors' feature. In intuition, GAT can automatically adjust attention scores to reduce the adversarial edges' negative effect. But it might performs quite terribly with attacks.

Here we give an example for further discussion. As is shown in Figure 1, the first picture is an original graph with 10 nodes, whose labeled are denoted by green and blue colors. Here we assume that node 1's is unknown, and we want to use the nodes 2-10 to infer its color. If the graph is poisoned by the red edge between nodes 1 and 7, now we use the GAT to predict node 1's label. The updated label of node 1 will be affected by the neighbors. As is shown in the third figure, the neighbors of node 1 can be divided into two parts: the neighbors with positive and negative effects. Part of edges with negative effects are colored by yellow. Two edges in original graph are also colored by yellow. Therefore, it may not have the best performance for the supervised problem on the original graph. As shown in [18], for the citation networks like cora, citeseer and pubmed, only a small number of frequency components have useful information to learning. If GAT can distinguish the positive and negative effects of neighbors, the robustness of GAT can be enhanced. To adjust the effects of neighbors, two strategies can be selected. First, we can reduce the attention scores of negative nodes. Second, we can adjust the feature of negative nodes. Based on these thoughts, we proposed Robust GAT.

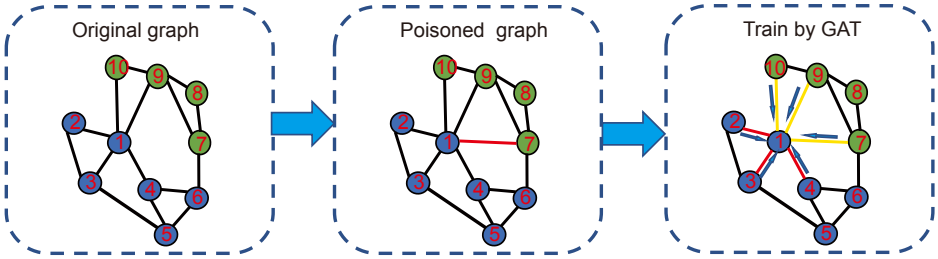


Figure 1. Two graphs with part unreasonable structure and feature

3 The proposed framework

3.1 RoGAT

GAT has an excellent performance on the semi-supervised task with the dynamic weight. However, the performance against adversarial attacks is relatively terrible. We may think that the attention mechanism of GAT can impair the influence of disturbing node features or link since if it could choose relatively small attention scores for those fake or disturbed edges. But it failed on the poisoned graph. It is due to that the attention scores in GAT are computed based on the node features[19]. The graph structure only decides the choice of neighbors. There is no attention difference caused by the graph structure, which is not reasonable[1]. Inspired by the work of [19], which mixed features and structure to obtain attention scores, we choose a new attention score combined the structure and feature attention, which is divided into three parts.

3.1.1 Modify the GAT attention

As shown in GAT, the attention scores are computed by the neighbors' features, which make it hard to recognize fake or adversarial edges. Therefore, we decide to attach another attention score determined by structure, which can be used to distinguish the fake and adversarial edges. Then the RoGAT is revised by

$$x_v^{(k)} = \|_{m=1}^{M} \sigma\left(\sum_{u\in\mathcal{N}(v)} (\bar{\alpha}_{uv}^{(k-1)}\mathbf{W})^m x_u^{(k-1)}\right), \tag{7}$$

where $\bar{\alpha}_{ij}$ is a new attention combined the feature attention (4) and graph structure attention obtained by the feature smoothness assumption defined by

$$\bar{\alpha}_{ij} = a_{ij}\alpha_{ij}.$$

Here a_{ij} is the link weight representing the local structure.

3.1.2 Modify the graph structure

In most situation, the connected nodes in a graph tends to share similar features. For example, in citation network, the entities with the similar bag-of-words feature tends to connect[6]. And two individuals in

[1] We do not use the sparse and low-rank assumption in [15], which is not suitable for all data and is time-consuming in computation.

social graph may share the similar features since they tend to have the related hobbies or characters[20]. There, we use the $\mathcal{L}_A$ to adjust the graph structure:

$$\begin{aligned}\arg\min_{\bar{A}\in\bar{\mathcal{A}}}\mathcal{L}_S, &= \|A-\bar{A}\|^2+\frac{\alpha}{2}\sum_{i,j=1}^{N}\mathrm{A}_{ij}\left(\mathbf{x}_i-\mathbf{x}_j\right)^2\\ &= \|A-\bar{A}\|^2+\alpha\,\mathrm{tr}\left(X^\top\bar{L}X\right)\end{aligned} \tag{8}$$

where α is a hyper-parameter to balance the constraint of feature smoothness and $\bar{L}$ represents the laplacian matrix defined by $\bar{L}=\bar{D}-\bar{A}$, where $\bar{D}_{ii}=\Sigma_j\bar{A}_{ij}$ is the diagonal element of degree matrix $\bar{D}$.$\mathcal{A}$ represents the domain of adjacency matrix.

3.1.3 Modify the graph feature

After we revised the graph structure, we further make an adjustment to revise the graph feature, which can alleviate the negative effects for the update node features. Since our task is to reduce the $\mathcal{L}_{GNN}$ defined in [15], we use $\mathcal{L}_{GNN}$ as a part of the loss to guide the training of feature.We construct the objective function as:

$$\begin{aligned}\arg\min_{\bar{X}\in\bar{\mathcal{X}}}\mathcal{L}_f &= \|X-\bar{X}\|^2+\gamma\frac{1}{2}\sum_{i,j=1}^{N}\bar{A}_{ij}\left(\bar{\mathbf{x}}_i-\bar{\mathbf{x}}_j\right)^2+\lambda\mathcal{L}_{GNN}(\theta,\bar{A},\bar{X},\mathcal{Y}_p)\\ &= \|X-\bar{X}\|^2+\gamma\,\mathrm{tr}\left(\bar{X}^\top\bar{L}\bar{X}\right)+\lambda\mathcal{L}_{GNN}(\theta,\bar{A},\bar{X},\mathcal{Y}_p),\end{aligned} \tag{9}$$

where $\bar{\mathcal{X}}$ represents the domain of node feature.

3.2 The optimization algorithm

Given the graph $G=(V,A,X)$ with partial nodes $\mathcal{V}_p$ with label $\mathcal{Y}_p$. As is mentioned in section 3.1, our proposed RoGNN model can be described as the following optimization problem:

$$\begin{aligned}&\min_{\theta}\mathcal{L}_{GNN}(\theta,\bar{A},\bar{X},\mathcal{Y}_L)=\sum_{v_i\in\mathcal{V}_L}\ell(f_\theta(\bar{X},\bar{A})_i,y_i),\\ &\text{s.t.}\bar{A}\in\arg\min_{\bar{A}\in\bar{\mathcal{A}}}\mathcal{L}_s \text{ and } \bar{X}\in\arg\min_{\bar{X}\in\bar{\mathcal{X}}}\mathcal{L}_f.\end{aligned} \tag{10}$$

Actually this model is a bilevel programming which is a problem where some variables in the objective function are constrained to be an optimal solution of another optimization problem[21]. Here the parameters $\bar{A}$ and $\bar{X}$ are the solution of inner optimization problems. We use the iterative method to compute this optimization problem.

We establish the iteration algorithm 1 as followed:

4 Experiments

In the experiment part, we firstly empirically evaluate RoGAT on semi-supervised problem with the state of art defense methods under different kinds of adversarial attacks. Then we analysis the parameters and explain why our methods work.

4.1 Experimental settings

4.1.1 Experimental dataset

We choose three benchmark datasets used in [12][22]. To evaluate our proposed method, we choose the largest component of these datasets following [23][15].

	$\mathrm{N_{LCC}}$	$\mathrm{E_{LCC}}$	Classes	Features
Cora	2,485	5,069	7	1,433
Citeseer	2,110	3,668	6	3,703
Polblogs	1,222	16,714	2	/

Algorithm 1 RoGAT

Input: Graph $G = (V, A, X)$ and part nodes V_p with labels $\mathcal{Y}_p$.

Parameters:

α, γ, λ: the adjustment parameter for different terms

T_1,T_2:outer and inner maximum iteration step

η_1, η_2, η:learning rate for different iteration.

Output: the GAT model with learned parameters θ.

Initialize the GAT model with given structure A and set $\bar{A} \longleftarrow A$, $\bar{X} \longleftarrow X$.
Randomly initialize the parameter θ of GAT.
for $i = 1$ to T_1 **do**:
    $\bar{A} \longleftarrow \bar{A} - \eta_1 \nabla_{\bar{A}}(\|A - \bar{A}\|^2 + \alpha \operatorname{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right))$
    $\bar{X} \longleftarrow \bar{X} - \eta_2 \nabla_{\bar{X}}(\|X - \bar{X}\|^2 + \gamma \operatorname{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right) + \lambda \mathcal{L}_{GNN}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_p))$
    for $i = 1$ to T_2 **do**:
        $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_{GNN}\left(\theta, \bar{A}, \bar{X}, y_p\right)}{\partial \theta}$
Return θ and GAT.

4.1.2 Baselines

Here we compare RoGAT with state of the art GNN and implement the defensive models by the DeepRobust library. [24]

- **GCN**[6]: The classical and widely used GCN give by Kipf.
- **GAT**[9]: GAT uses the attention mechanisms to learn the representation of nodes.
- **RGCN**[14]: RGCN assumes that all the node representations are defined by Gauss distributions and use attention mechanism to reduce the influence the nodes with high variance.
- **GCN-Jaccard**[11] : As attacks tend to link the nodes with huge features' difference, GCN-Jaccard makes a judgment to eliminate part of edges between nodes with small similarity. The method can work when the features of the nodes are available.
- **GCN-SVD**[23]: Since *nettack* is a high-rank attack, GCN-SVD uses an low-rank approximation of the perturbed graph for further training. This model can also be extended to a non-targeted attacks and random-attacks.
- **ProGNN**[15]: ProGNN assumes that data in reality is low-rank and sparse. It uses the progressive procedure to adjust the structure and parameters of GCN. This method performs robustly under three kinds of attacks but is time-consuming.

4.1.3 Parameter settings

Since RoGAT is based on GAT, we choose the default settings about GAT in [9] with a two-layer model. Here the dropout parameter $p = 0.6$ is applied to both layers' input. The learning rate for the training feature and adjacency matrix for SGD is set by 0.01. For GCN, we use the default settings in [6]. For RGCN, we use the same settings as the experiments in [15] with $\{16, 32, 64, 128\}$ hidden units. For GCN-Jaccard, $\{0.01, 0.02, 0.03, 0.04, 0.05, 0.1\}$ are set as the threshold of similarity for removing the edges for different perturbations ratios. For GCN-SVD, $\{5, 10, 15, 50, 100, 200\}$ are used as the reduced rank.

For all the tested graphs, we randomly choose 10% of nodes as the training datasets and 10% of nodes as the validation datasets. The remaining 80% of nodes are used for testing for the non-targeted attack. All the experiments are executed 10 times with different random seeds. The inner and outer iterations are set by 10. The other hyper-parameters are selected by the accuracy of the validation and manual test.

Table 1. Node classification performance (Accuracy±Std) under non-targeted attacks

Dataset	Ptb (%)	GCN[6]	GAT[9]	RGCN[14]	GCN-Jaccard[11]	GCN-SVD[23]	Pro-GNN[15]	**RoGAT**
Cora	0	83.50±0.44	83.97±0.65	83.09±0.44	82.05±0.51	80.63±0.45	82.98±0.23	**84.59 ±0.55**
	5	76.55±0.79	80.44±0.74	77.42±0.39	79.13±0.59	78.39±0.54	**82.27±0.45**	81.22 ±1.09
	10	70.39±1.28	75.61±0.59	72.22±0.38	75.16±0.76	71.47±0.83	79.03±0.59	**79.53 ±1.60**
	15	65.10±0.71	69.78±1.28	66.82±0.39	71.03±0.64	66.69±1.18	76.40±1.27	**80.47 ±0.71**
	20	59.56±2.72	59.94±0.92	59.27±0.37	65.71±0.89	58.94±1.13	73.32±1.56	**78.40 ±2.18**
	25	47.53±1.96	54.78±0.74	50.51±0.78	60.82±1.08	52.06±1.19	69.72±1.69	**78.99 ±0.96**
Citeseer	0	71.96±0.55	73.26±0.83	71.20±0.83	72.10±0.63	70.65±0.32	73.28±0.69	**73.49 ±1.96**
	5	70.88±0.62	72.89±0.83	70.50±0.43	70.51±0.97	68.84±0.72	72.93±0.57	**73.64 ±1.33**
	10	67.55±0.89	70.63±0.48	67.71±0.30	69.54±0.56	68.87±0.62	72.51±0.75	**72.73 ±0.69**
	15	64.52±1.11	69.02±1.09	65.69±0.37	65.95±0.94	63.26±0.96	72.03±1.11	**73.02 ±1.16**
	20	62.03±3.49	61.04±1.52	62.49±1.22	59.30±1.40	58.55±1.09	70.02±2.28	**72.43 ±1.48**
	25	56.94±2.09	61.85±1.12	55.35±0.66	59.89±1.47	57.18±1.87	68.95±2.78	**73.19 ±0.49**
Polblogs	0	95.69±0.38	95.35±0.20	95.22±0.14	-	95.31±0.18	-	**95.67 ±0.36**
	5	73.07±0.80	83.69±1.45	74.34±0.19	-	**89.09±0.22**	-	79.18 ±1.12
	10	70.72±1.13	76.32±0.85	71.04±0.34	-	**81.24±0.49**	-	74.95 ±1.08
	15	64.96±1.91	68.80±1.14	67.28±0.38	-	68.10±3.73	-	**70.14 ±1.45**
	20	51.27±1.23	51.50±1.63	59.89±0.34	-	57.33±3.15	-	**65.85 ±1.38**
	25	49.23±1.36	51.19±1.49	56.02±0.56	-	48.66±9.93	-	**63.37 ±2.03**

4.2 Defensive performance

4.2.1 Under the non-targeted adversarial attack

First, we evaluate the performance of RoGAT against the non-targeted adversarial attack, which aims to degrade the performance on all nodes. Here we use the metattack as the non-targeted attack and adopt the same parameter settings as [22]. The Meta-self attack for Cora, CIteseer, and Polblogs is considered as it is the most effective attack. As is shown in Table 1, we compare RoGAT with the other six methods and vary the perturbations rate from 0% to 25%. All the experiments are conducted for 10 times, and then the average accuracy and standard deviation are recorded. RoGAT performs the best under the meta attack for all the tested datasets.

- RoGAT outperforms other methods for Cora and Citeseer almost for all the f perturbations ratios and has better performance with larger perturbations for the polblogs dataset. Specifically, RoGAT improves the accuracy of classification over 13% and 2% than others at the rate of 25% for Cora and Citeseer, respectively. In addition, under 15% to 25% perturbations for polblogs, RoGAT performs better than other methods varying from 2% to 15%.

- Although ProGNN has good performance when dealing with Cora and Pubmed under the larger ratio of perturbations, RoGAT performs best. Compared with ProGNN, RoGAT ignored the regularization of sparsity and low-rank but can adapt well to the non-targeted adversarial attack. It means that the attention mechanism of RoGAT considering the feature smoothing can adjust the weight of fake and real edges.

- Compared with ProGNN, RoGAT has a lower computation time without the regularization of sparsity. ProGNN cost more than 15 minutes for one script in cora with 2080Ti GPU, while the average time for one RoGAT training is 20s.

4.2.2 Under the targeted adversarial attack

In this part, we evaluate the performance of different methods about node classification problems against the targeted attacks, which aims to attack selected nodes. Here we choose the nettack as the targeted-attack method and use the default parameter in the original paper[12]. The number of perturbations of per node varies from 1 to 5. And similar to [15], all the nodes with a degree larger than 10 are chosen as the targeted nodes. We display the performance of node classification for different methods. In figure

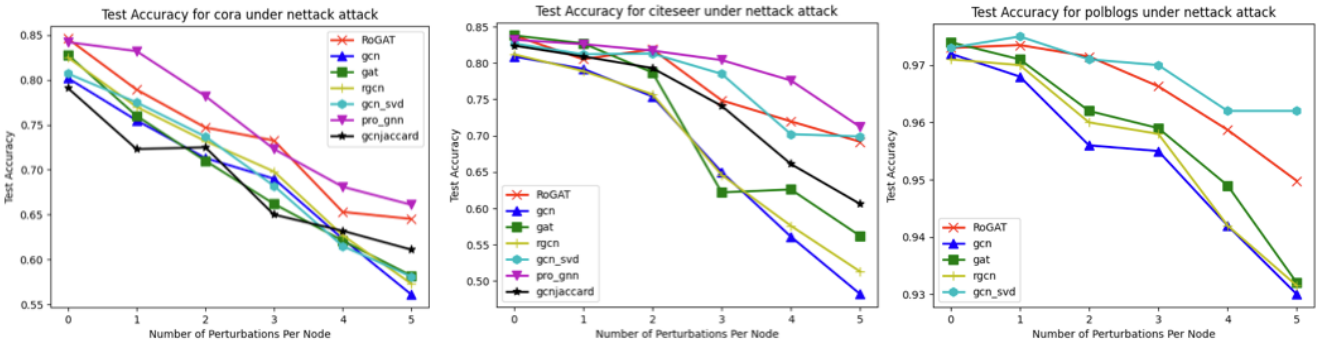


Figure 2. Results of different models under nettack

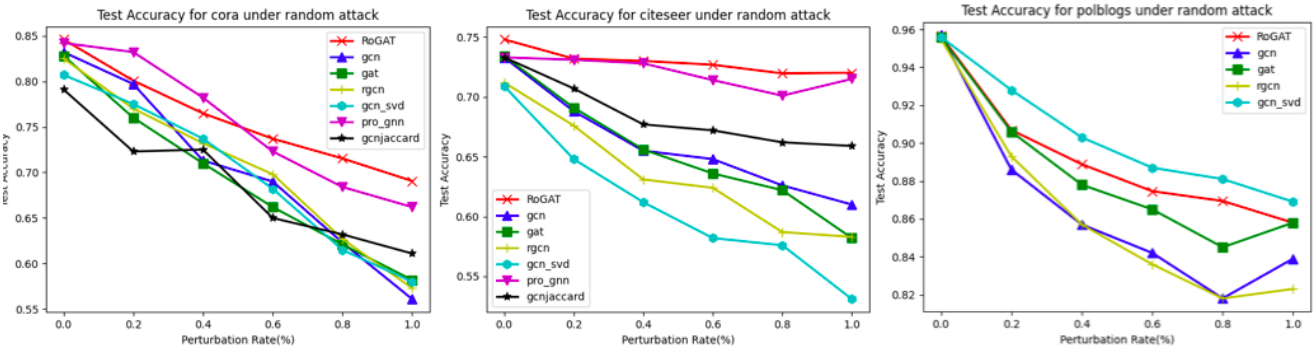


Figure 3. Results of different models under random attack

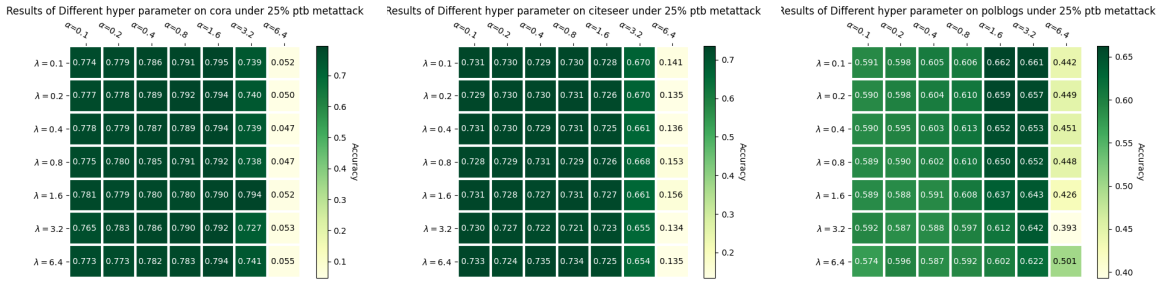
2, it shows that our method outperforms most methods and has similar performance with Pro-GNN under the Cora and Citeseer. Our approach has 10% and 20% improvement, respectively, in Cora and Citeseer compared with the original GCN. Since the dataset polblogs do not have the feature, our method performs better than other methods except for GCN-SVD.

We also do some experiments to evaluate our methods when dealing with the random attack, which adds the perturbation on nodes randomly. Different ratios of perturbations varying from 0% to 100% are adopted to disturb the graph structure. The results in Figure 3 shows that our RoGAT outperforms other methods in dealing with Cora and Citeseer and has relatively better performance with the dataset Polblogs. RoGAT has more than 15% and 13% improvement with Cora and Citeseer. It means that RoGAT can successfully resist the random attack.

Overall, RoGAT has quite good performance compared with most defensive methods when dealing with different types of adversarial attacks.

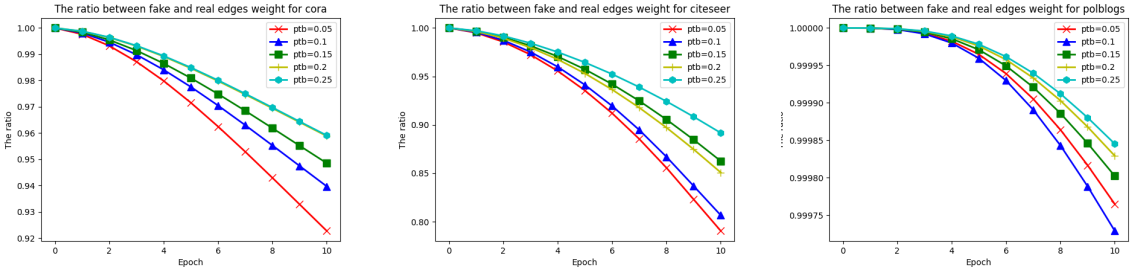
4.2.3 Hyperparameter analysis

In this section, we discuss the influence of hyper-parameters for RoGAT. Here we set $\alpha = \gamma$ and only consider the impact of α and λ on the Cora dataset with perturbation rates of 25% metattack. We vary α and λ from 0.1 to 6.4 in a log scale base 2 on the Cora dataset, respectively. Figure 4 shows RoGAT with different λ and α. The introduction of λ and α contributes to the robusty of GAT. Compared with λ, appropriate value of α has more influences on the performance of RoGAT. The performance of RoGAT is not sensitive to α and λ with not too large α. It means that the feature smoothness is tightly connected with the performance of RoGAT. For different kinds of datasets, α decides the ratio between two parts loss, which needs to be selected carefully. Therefore for RoGAT, using feature smoothness to revise the structure's attention is effective in defending adversarial attacks.



Cora Citeseer Polblogs

Figure 4. The performance for different λ and γ under 25% ptb metattack



Cora Citeseer Polblogs

Figure 5. The ratio of the weight of fake and real edges during training procedure for cora, citeseer and Polblogs

4.3 Attention weight analysis

Since we assume that the revised weight $\bar{\alpha}$ gives more attentions to real edges and reduce the influence of fake edges. Here we compute the ratio of weight between the fake edges and real edges during the training procedure. Here we set $\alpha = \gamma = 1$ and compute the average weight $\bar{\alpha}$ during the training procedure under the metattack. For different rate of perturbations and dataset, the weight ratio between fake edges and real edges decreases from the initial value 1 to smaller value. And almost for all three datasets, the ratio decreases faster for the smaller perturbations, which leads to better performance. As is shown in Figure 1, real and fake edges make equivalent contributions to the update of node information. The RoGAT can adjust the Therefore, it is consistent with the assumptions that less attention will be given to the fake edges to reduce the influence of fake information during the aggregation procedure.

5 Conclusion

Graph neural networks, including graph convolutional networks and graph attention networks are easily disturbed by graph adversarial attacks. This paper adjusts the attention mechanism and propose the robust GAT called RoGAT, which revises the structure and feature of the poisoned graph iteratively. The results of experiments show that RoGAT can reduce the influence of fake edges and performs better than most of the state-of-the-art baselines, especially in defending the metattack. In further research, we want to use more prior information to improve RoGAT.

Acknowlegements

This research work is supported by the National Science Foundation of China(NSFC) under 61977065.

References

[1] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61, 2009.

[2] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 12631272. JMLR.org, 2017.

[3] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1024–1034. Curran Associates, Inc., 2017.

[4] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. Spectral networks and locally connected networks on graphs. *arXiv: Learning*, 2013.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[6] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv: Learning*, 2016.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[8] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv: Computation and Language*, 2014.

[9] Petar Velikovi, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Li, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[10] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*, 2018.

[11] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv: Learning*, 2019.

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

[13] Kaidi Xu, Hongge Chen, Sijia Liu, Pin-Yu Chen, Tsui-Wei Weng, Mingyi Hong, and Xue Lin. Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*, 2019.

[14] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '19, page 13991407, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. *arXiv preprint arXiv:2005.10203*, 2020.

[16] Xianfeng Tang, Yandong Li, Yiwei Sun, Huaxiu Yao, Prasenjit Mitra, and Suhang Wang. Transferring robustness for graph neural network against poisoning attacks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 600–608, 2020.

[17] Xiang Zhang and Marinka Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. *arXiv preprint arXiv:2006.08149*, 2020.

[18] N T Hoang and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv: Machine Learning*, 2019.

[19] Kai Zhang, Yaokang Zhu, Jun Wang, and Jie Zhang. Adaptive structural fingerprints for graph attention networks. In *International Conference on Learning Representations*, 2020.

[20] Miller Mcpherson, Lynn Smithlovin, and James M Cook. Birds of a feather: Homophily in social networks. *Review of Sociology*, 27(1):415–444, 2001.

[21] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153(1):235–256, 2007.

[22] Daniel Zügner and Stephan Günnemann. Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*, 2019.

[23] Negin Entezari, Saba A Al-Sayouri, Amirali Darvishzadeh, and Evangelos E Papalexakis. All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 169–177, 2020.

[24] Yaxin Li, Wei Jin, Han Xu, and Jiliang Tang. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*, 2020.