arXiv:2009.13038v3 [cs.LG] 4 Aug 2022

A Robust graph attention network with dynamic adjusted Graph

Xianchen Zhou[1], Yaoyun Zeng[1], Zepeng Hao[1] and Hongxia Wang[1*]

[1]National University of Defense Technology, Changsha, 410072, Hunan, China.

*Corresponding author(s). E-mail(s): wanghongxia@nudt.edu.cn; Contributing authors: zhouxianchen13@nudt.edu.cn; yaoyun_zeng@nudt.edu.cn; haozepeng@gmail.com;

Abstract

Graph Attention Networks(GATs) are useful deep learning models to deal with the graph data. However, recent works show that the classical GAT is vulnerable to adversarial attacks. It degrades dramatically with slight perturbations. Therefore, how to enhance the robustness of GAT is a critical problem. Robust GAT(RoGAT) is proposed in this paper to improve the robustness of GAT based on the revision of the attention mechanism. Different from the original GAT, which uses the attention mechanism for different edges but is still sensitive to the perturbation, RoGAT adds an extra dynamic attention score progressively and improves the robustness. Firstly, RoGAT revises the edge's weight based on the smoothness assumption which is quite common for ordinary graphs. Secondly, RoGAT further revises the features to suppress features' noise. Then, an extra attention score is generated by the dynamic edge's weight and can be used to reduce the impact of adversarial attacks. Different experiments against targeted and untargeted attacks on citation data on citation data demonstrate that RoGAT outperforms most of the recent defensive methods.

Keywords: Graph Neural Networks, Adversarial attack, Graph Attention Network, Robustness
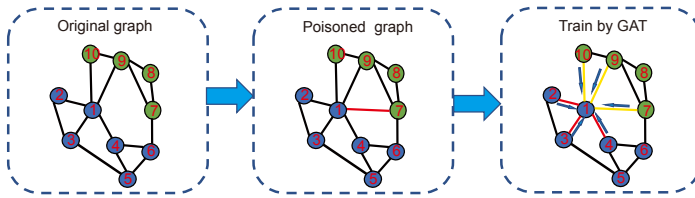
Figure 1: Original graphs can be attacked by the negative edges between node 1 and 7 with different labels. The update function of node 1 aggregates more negative information from nodes 7, 9, 10 than positive information from nodes 2, 3, 4.

1 Introduction

Non-Euclid data occurs widely in our daily life and Graph Attention Network(GAT) [7] achieves remarkable performance in these data represented by graphs. It generates node embedding by using a local aggregation function [1, 3?] with attention mechanism [21, 22], which computes the hidden representation by the features of connected nodes with different attention weight coefficients. The attention mechanism makes GAT focus on the relevant part and perform well. However, GAT is vulnerable to the adversarial attacks, which means subtle perturbations may degrade its performance significantly. The lack of robustness makes GAT practical limited in several fields like military and finance with high requirement of security. For instance, in the social secure field, the criminal can create or hide some social relationship to escape the examination of GAT. Hence, developing the robustness of GAT to resist the different kinds of adversarial attacks [23] is important and urgent.

The vulnerability of GAT is due to its aggregation function. As is shown in Figure 1, the aggregation function aggregates the information from different kinds of neighbors. In intuition, information from the neighbors of similar or same labels(positive edges) make positive effects, while information from dissimilar or distinct neighbors(negative edges) may make negative effects on the iteration of node features. However, adversarial attacks add extra negative edges or delete positive edges, which degrades GAT. In this paper, we assume graphs to be analyzed satisfies feature smoothness assumption [24] which is common in most graphs. It means for most nodes in graph, there often exists more neighbors with positive edges than that with negative edges. Note that there also exists some heterogeneous graphs which don't satisfy smoothness assumption, we do not discuss these graphs here and will address them in the future. We aim to design an improved GAT to defend the adversarial attacks based on the prior information of graphs.

An intuitive idea of defensive technology is using prior information to increase the positive effects and reduce negative effects of neighbors. Two problems are faced here: (1) What kind of methods helps us to distinguish two kinds of neighbors? (2) How to design the attention scores for different edges? This paper proposes a Robust GAT(RoGAT) to solve these problems. RoGAT distinguishes the positive and negative neighbors based on the Laplacian regularization [24] and designs an extra dynamic scores to adjust the attention effects for different edges.

The contribution of RoGAT can be summarized as follows.

(1) The mode defend against adversarial attacks by increasing extra edge attention scores to distinguish two kinds from adversarial edges based on the feature smoothness assumption [24].

(2) The model adjusts the graph structure and feature iteratively during the training procedure, which leads to the ratio of attention scores between real edges and adversarial edges increases.

(3) The experiments on various real-world graphs show that RoGAT can adjust the ratio of attention scores between negative and positive edges iteratively and thus outperform other defensive methods in the node classification task under different types of attacks.

The implementation of RoGAT is based on the DeepRobust [25] repository for adversarial attacks and the experimental settings to reproduce our results can be found in https://github.com/zhouxianchen/robustGAT. The rest of the paper is organized as follows. Section 2 gives the notations and discusses related works of adversarial attacks and defensive methods. Section 3 reviews the original GAT and . In section 4, the relative merits of GAT is discussed and RoGAT is proposed to improve the performance of defending attacks. Section 5 gives some experiments to verify the conclusion. Section 6 gives a further discussion and conclusion of our methods. The last section is the acknowledgement.

2 Related works and Notations

2.1 Related Works

Recently, there is some research about adversarial attacks and defense on Graph Neural Networks, which can be used in GAT. The graph adversarial attacks can be divided into targeted attacks and untargeted attacks. The targeted attacks like nettack [8] and RL-S2V [9] tend to let the trained model misclassify a small set of test samples, while untargeted attacks like metattack [10] aimed to let the trained model have bad overall performance on all test data. Nettack introduces the unnoticeable perturbations on both structures and features. RL-S2V uses reinforcement learning to generate attacks on GNNs. The metattack parameterized the graph structure and used the gradient information to attack GAT.

As to the method of defending the adversarial attacks, one perspective to achieve robustness is to eliminate the influence of perturbations such as adding

or removing the adversarial edges or clearing up the change of node features. The criteria of eliminating the influence are mainly based on the prior information of the graph in specific applications. For example, Wu [11] applied the Jaccard similarity to eliminate the edges between nodes with low similarity. GNNguard [12] can detect and quantify the relationship between the graph structure and node features based on the hypothesis that similar nodes are more likely to interact than dissimilar nodes. It estimated an importance weight for every edge to reduce the influence of fake edges. RGCN [13] added the penalization of adversarial edges and modeled the hidden layers by Gaussian distributions to reduce the effect of attacks. PTDNet [14] uses nuclear norm regularization to drop some task-irrelevant edges and improve the robustness. ProGNN [15] assumed that the graph should be low-rank and sparse and then gave a progressive model for adversarial training. These models used the poisoned graph for training and estimated the clean graph by prior information [13, 16]. Besides, another perspective tries to figure out the pattern of adversarial attacks and design defensive methods. Wei [17] found that adversarial attacks can destroy graph structure.They proposed SimPGCN which can effectively and efficiently preserve node similarity while exploiting graph structure. In GCNSVD [18], it can defend the metattack [19] by reducing the rank of the adjacency matrix. PA-GNN [16] designed a meta-optimization algorithm by penalizing the perturbations to restrict the negative impact of adversarial edges. HSC-GAT [**?**] proposes a holistic semantic constraint GAT which approaches the joint modeling of graphs to mitigate the perturbations. These methods can defend the adversarial attacks under the different situations.

2.2 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 $A \in \mathbb{R}^{N \times N}$, where a_{ij}, the (i, j)th elements of A, represents the link weight of node v_i and v_j. In addition, $X = [x_1, x_2, \cdots, x_N]^\top \in \mathbb{R}^{N \times d}$ denotes the feature matrix where $x_i \in \mathbb{R}^d$ is the feature vector of 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 nodes $\mathcal{V}_p = \{v_1, v_2, \cdots, v_m\}, m < N$ are annotated. $\mathcal{Y}_p = \{y_1, y_2, \cdots, y_m\}$, where y_i is the label of v_i. Given graph $G = (V, A, X)$ with partial label $\mathcal{Y}_p$, the goal of node classification is to predict the labels of unlabeled nodes.

3 GAT

GAT generates the new feature of one node by combining the feature vector of each node with attention in its neighbors. The attention mechanisms make GAT more flexible in aggregation. The formulation of L-layer GAT can be denoted by

$$x_v^{(0)} = x_v, v \in \{1, \cdots, N\},$$

$$x_v^{(k)} = \sigma \left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(k-1)} W^{(k-1)} x_u^{(k-1)} \right), k = 1, 2, \cdots, L \tag{1}$$

where

$$\alpha_{uv}^{(k-1)} = \frac{\exp \left(\text{Leaky ReLU} \left(\overrightarrow{\mathbf{a}_k}^T \left[W^{(k-1)} x_v^{(k-1)} \| W^{(k-1)} x_u^{(k-1)} \right] \right) \right)}{\sum_{l \in \mathcal{N}(v)} \exp \left(\text{LeakyReLU} \left(\overrightarrow{\mathbf{a}_k}^T \left[W^{(k-1)} x_v^{(k-1)} \| W^{(k-1)} x_l^{(k-1)} \right] \right) \right)}. \tag{2}$$

$W^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$ are the parameters to learn and σ is the activation function. $d_0 = d$, $d_L = K$ is the number of class. $\|$ represents a concatenation operator, $\overrightarrow{\mathbf{a}_k} \in \mathbb{R}^{2d_k}$ is a weight vector multiplying the concatenation vector, $\mathcal{N}(v)$ represents the neighbors of node v.

The multi-head attention is used to improve the performance of GAT:

$$x_v^{(k)} = \|_{p=1}^M \sigma \left(\sum_{u \in \mathcal{N}(v)} (\alpha_{uv}^{(k-1)} W^{(k-1)})^p x_u^{(k-1)} \right), k = 1, 2, \cdots, L \tag{3}$$

where $x_v^{(0)} = x_v$, $v \in \{1, 2, \cdots, N\}$. The update feature $x_v^{(k)}$ relies on the neighbors' features $x_u^{(k-1)}$ and the weights α_{ij} are computed according to the features.

Denote the learning parameters of GAT by $\theta = \{W^{(0)}, \cdots, W^{(L-1)}\}$ which including all the $W^{(k)}$ in each layer. Then for node classification problem, GAT learns a function $f_\theta^{\text{GAT}} : \mathcal{V} \to \mathcal{Y}$ by applying Boolean classification function to $x_v^{(L)}$ to predict unlabeled nodes. And the objective function is the sum of loss for the labeled nodes,

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

$$\theta^* = \arg \min_\theta \mathcal{L}_{\text{GAT}}(\theta, A, X, \mathcal{Y}_p), \tag{4}$$

where θ is the parameters of GAT, $f_{\theta^*}^{\text{GAT}}(X, A)_i$ is the predicted label of v_i.

4 The proposed method

4.1 Analysis of GAT

To figure out the relationship between graph structure and accuracy of GAT, we established an experiment both on the simulated and real graphs.

As to the simulated data, one thousand nodes are labeled by two classes equally. Any two nodes are connected with probability p_1 if they are in the

same class and probability p_2 while in the distinct class. The number N_1 and N_2 represent the links between the same and distinct labels respectively. 10% percent of nodes are set as the training nodes, while another 10% of nodes are chosen randomly as the testing nodes. We compare the accuracy of GAT on random graphs with different numbers N_1/N_2.

As is shown in Table 1, when N_1 approximates N_2, the accuracy of GAT degrades to 50%. GAT performs better when the ratio increases. And for Cora, Citeseer and Polblogs, the performance of GAT is also related to N_1/N_2.

In fact, the aggregation function for GAT is tightly connected with graph structure and features [9, 11]. Although GAT aggregates the hidden features of its neighbors with attention scores, it cannot distinguish the information came from positive or negative neighbors well. When the labels in neighbors tend to the same, the propagation of GAT preserves the feature well. While the labels in neighbors have multiple labels, the features that came from different labels make considerable and negative effects on the iterated feature. Therefore, N_1/N_2 affects the aggregation process of all nodes averagely.

Table 1: The accuracy of GAT on random graph and three datasets with different link ratio(N_1 and N_2 represents the number of links between the same and different labels, respectively)

Dataset	N_1	N_2	$\frac{N_1}{N_2}$	GAT peformance
Random graph	4982	4864	1	0.5150
	9894	4818	2	0.8125
	14774	5076	3	0.9637
	19714	4944	4	0.9988
Cora	8152	1986	4.10	0.8397
Citeseer	5402	1934	2.79	0.7326
Polblogs	30278	3150	9.61	0.9535

Therefore, some attack methods degrade GAT by affecting N_1/N_2 of graphs based on various technologies. Since GAT uses the attention mechanisms to mix various features, the intuition is that GAT can adjust the attention scores for useful and useless neighbors' information automatically to resist the attacks during the training procedure. However, Table 2 displays the $\frac{N_1}{N_2}$ of Cora and accuracy of GAT under different perturbations rates. The adversarial attack can add or delete the edges to change N_1/N_2 and degrade GAT significantly [13, 15].

Nevertheless, for a given graph with most unlabeled nodes, it is impossible to change N_1/N_2 easily. In this paper, we tends to enhance the impact of N_1 positive edges and reduce that of N_2 negative edges.

Table 2: The accuracy of GAT on Cora dataset with different perturbations rates with mettack.

Perturbation rates	N_1	N_2	$\frac{N_1}{N_2}$	GAT accuracy
0	8152	1986	4.10	0.8397
0.05	8232	2412	3.41	0.8044
0.10	8294	2840	2.92	0.7561
0.15	8286	3332	2.49	0.6978
0.20	8228	3852	2.13	0.5994
0.25	8252	4240	1.95	0.5478

4.2 RoGAT

Note that the original attention scores in GAT are computed based on the node feature and labels of neighbors [26]. And the graph structure only decides the choice of neighbors but cannot help to adjust the attention scores. Adversarial attacks add negative edges or reduce the positive edges to affect the aggregation neighbors.Therefore, we tend to revise the aggregation function and insert an extra attention score to reduce the effect of adversarial edges. Intuitively, it should assign relatively small attention scores to those negative edges though the prior information. Note that the proper attention score is related not only to structure A but also to feature X, we establish a robust model which adjusts the graph structure and feature to generate an extra score, which helps to reduce the negative edges bringing by adversarial attacks.

4.2.1 The optimization model

In most situations especially for homogeneous networks, the connected nodes with the same labels in a graph tend to share similar features. For example, in citation networks, the entities with similar bag-of-words features tend to connect and belong to the same class [6]. And two connected individuals in social graphs may share similar features since they tend to have related hobbies or characters [27]. The characteristic can be represented by the feature smoothness regularization $\mathrm{tr}\left(X^\top L X\right)$, where L is the weighted Laplacian matrix of the graph. Adjust the effect to enlarge the We propose the following optimization model that obtains the revised structure $\bar{A}$, feature $\bar{X}$ and parameters θ, which can be described as:

$$(\bar{A}^*, \bar{X}^*, \theta^*) = \arg \min_{\theta, \bar{A} \in \bar{\mathcal{A}}, \bar{X} \in \mathcal{X}} \mathcal{L}_{\mathrm{re}}(\bar{A}, \bar{X}, \theta) + \lambda \mathcal{L}_{\mathrm{RoGAT}}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_L)$$

with

$$\mathcal{L}_{\mathrm{re}}(\bar{A}, \bar{X}, \theta) = \|A - \bar{A}\|^2 + \beta \|X - \bar{X}\|^2 + \alpha \, \mathrm{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right) \qquad (5)$$

$$= \|A - \bar{A}\|^2 + \beta \|X - \bar{X}\|^2 + \frac{\alpha}{2} \sum_{i,j=1}^{N} \bar{A}_{ij} \left(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\right)^2 ,$$

where $\mathcal{L}_{\mathrm{RoGAT}}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_L)$ is given by

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

(4), α, β, γ are non-negative parameters. $\bar{L} = \bar{D} - \bar{A}$ is the laplacian matrix whose diagonal element $\bar{D}_{ii} = \Sigma_j \bar{A}_{ij}$. $\bar{\mathcal{A}}$ represents the domain of adjacency matrix, with element ranging from 0 to 1, while $\bar{\mathcal{X}}$ represents the domain of feature matrix.

Different from the original GAT, $f_\theta^{\mathrm{RoGAT}} : \mathcal{V} \to \mathcal{Y}$ revises the (1) by inserting modified attention score determined by the revised structure $\bar{A}$. The revised aggregation function can be written as

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

where $\bar{\alpha}_{uv}$ is a modified attention combining the feature attention (2) and graph structure attention obtained by the optimization model (5). The revised attention is defined as

$$\bar{\alpha}_{uv} = \bar{A}_{uv} \alpha_{uv}.$$

Here $\bar{A}_{uv}$ is the link weight of the current $\bar{A}$ computed by optimization model.

4.2.2 The optimization algorithm

We update the graph structure $\bar{A}$ and feature $\bar{X}$ and the parameters of GAT alternatively to solve the optimization model (5) as follows. To solve problem (5), firstly we fix $\bar{X}$ and consider the update of $\bar{A}$ by

$$\min_{\bar{A} \in \bar{\mathcal{A}}} \|A - \bar{A}\|^2 + \alpha \operatorname{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right). \tag{8}$$

We initialize $\bar{A} = A$, then update $\bar{A}$ by using projected gradient descent method:

$$\bar{A} \longleftarrow P_{\mathcal{A}}(\bar{A} - \eta_1 \mathcal{L}_s) = \bar{A} - \eta_1 \nabla_{\bar{A}}(\|A - \bar{A}\|^2 + \alpha \operatorname{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right)), \tag{9}$$

where

$$P_{\mathcal{A}}(\bar{A}) = \begin{cases} 0, \bar{A}_{ij} < 0 \\ 1, \bar{A}_{ij} > 1 \\ A, \text{otherwise}. \end{cases} \tag{10}$$

Then we fix $\bar{A}$ and consider the update of $\bar{X}$ by

$$\min_{\bar{X} \in \bar{\mathcal{X}}} \|X - \bar{X}\|^2 + \gamma \operatorname{tr}\left(\bar{X}^\top \bar{L} \bar{X}\right) + \lambda \mathcal{L}_{\mathrm{RoGAT}}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_p). \tag{11}$$

We use gradient descent method to update $\bar{X}$:

$$\bar{X} \longleftarrow P_{\mathcal{X}}(\bar{X} - \eta_2 \nabla_{\bar{X}}(\|X - \bar{X}\|^2 + \gamma \operatorname{tr}\left(\bar{X}^\top L \bar{X}\right) + \lambda \mathcal{L}_{\mathrm{GAT}}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_p))). \tag{12}$$

where $P_{\mathcal{X}}(\cdot)$ is the projection on the feature matrix domain.

The solving procedure of model (5) can be divided into the update of graph structure, feature and parameters of RoGAT alternatively. Algorithm 1 gives the iteration procedure.

Algorithm 1 RoGAT

Input: Graph $G = (V, A, X)$ and part of nodes V_p with labels $\mathcal{Y}_p$.
Parameters:
α, γ, λ: the non-negative parameters
T_1, T_2: outer and inner maximum iteration steps
η_1, η_2, η: learning rates for sub-optimization problems.
Output: the RoGAT model with learned parameters θ.
1: Initialize the RoGAT model with given structure A and set $\bar{A} \longleftarrow A$, $\bar{X} \longleftarrow X$.
2: Randomly initialize the parameter θ of RoGAT.
3: **for** $i = 1$ to T_1 **do:**
4:     $\bar{A} \longleftarrow \bar{A} - \eta_1 \nabla_{\bar{A}}(\|A - \bar{A}\|^2 + \alpha \operatorname{tr}\left(\bar{X}^\top L \bar{X}\right))$,
5:     $\bar{A} \longleftarrow P_{\mathcal{A}}(\bar{A})$,
6:     $\bar{X} \longleftarrow \bar{X} - \eta_2 \nabla_{\bar{X}}(\|X - \bar{X}\|^2 + \gamma \operatorname{tr}\left(\bar{X}^\top L \bar{X}\right) + \lambda \mathcal{L}_{\mathrm{RoGAT}}(\theta, \bar{A}, \bar{X}, \mathcal{Y}_p))$,
7:     $\bar{X} \longleftarrow P_{\mathcal{X}}(\bar{X})$,
8:     **for** $i = 1$ to T_2 **do:**
9:         $\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_{\mathrm{RoGAT}}\left(\theta, \bar{A}, \bar{X}, y_p\right)}{\partial \theta}$.
10:    **end for**
11: **end for**
12: Return θ and RoGAT.

---

5 Experiments

In this section, we firstly empirically evaluate RoGAT on semi-supervised problems with the state of the art defense methods under different kinds of adversarial attacks. Then we analyze the effect of parameters and explain why our method works.

5.1 Experimental settings

5.1.1 Experimental datasets

We choose three benchmark datasets Cora, Citeseer and Polblogs as [8, 19]. The largest component of these datasets [15, 18] are used in this paper in Table 3 .

Table 3: Datasets

	N_{LCC}	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	/

5.1.2 Baselines

Here we compare RoGAT with different Graph Neural Networks and implement the defensive models by the DeepRobust library [25].

- **GCN** [6]: The classical and widely used GCN defines the graph convolution in a spectral domain.
- **GAT** [7]: GAT uses the attention mechanism to learn the representation of nodes.
- **RGCN** [13]: RGCN assumes that all the node representations are defined by Gauss distributions and uses an attention mechanism to reduce the influence of the nodes with high variance.
- **GCN-Jaccard** [11] : As attacks tend to link the nodes with huge feature differences, GCN-Jaccard makes a judgment to eliminate part of edges between nodes with small similarities.
- **GCN-SVD** [18]: Since *nettack* is a high-rank attack, GCN-SVD uses a low-rank approximation of the perturbed graph for further training. This model can also be extended to non-targeted and random attacks.
- **ProGNN** [15]: ProGNN assumes that graph 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.
- **ADA-UGNN** [?]: A general GNN frameworks which is suited for handling varying smoothness properties.
- **HSC-GAT** [?]: A holistic semantic constraint GAT which approaches the joint modeling of graphs to mitigate the perturbations.

5.1.3 Parameter settings

Since RoGAT is based on GAT, we choose the default settings about GAT in [7] 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. The inner and outer iterations T_1 and T_2 are set by 10. The learning rate η_1 and η_2 are set by

0.01.The other hyper-parameters are selected by the accuracy of the validation and manual test. All the experiments are executed 10 times with different random seeds.

5.2 Defensive performance

5.2.1 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 [19]. The Meta-self attack for Cora, CIteseer, and Polblogs is considered the most effective attack. As is shown in Table 4, we compare RoGAT with the other six methods and vary the perturbations rate from 0% to 25%. All the experiments are conducted 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 almost for all the perturbations ratios in Cora and Citeseer datasets and has better performance for larger perturbations in the polblogs dataset. Specifically, the classification accuracy of RoGAT when processing the Cora and Citeseer datasets with 25% disturbance is 13% and 2% higher than other methods, respectively. In addition, for the Polblogs dataset, under 15% to 25% interference, the performance of RoGAT is better than other methods by 2% to 15%.
- Although ProGNN has good performance when dealing with Cora and Citeseer under the larger ratio of perturbations, RoGAT performs best. Compared with ProGNN, RoGAT ignore the regularization of sparsity and low-rank but can still adapt well to the non-targeted adversarial attack. It means that the attention mechanism of RoGAT, which considers 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 costs more than 15 minutes for one script in cora with 2080Ti GPU, while the average time for one RoGAT training is 20 seconds.

5.2.2 Under the targeted adversarial attack

In this part, we evaluate the performance of different methods for node classification problems against the targeted attacks, which aim to attack selected nodes. Here we choose the nettack as the targeted-attack method and use the default parameter in the original paper [8]. The number of perturbations 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 2(a) and 2(b), it shows that our method outperforms most methods and has a similar performance with ProGNN for the Cora and Citeseer. Our approach has 10% and 20% improvement, respectively, in Cora and Citeseer compared with the original GCN. Since the

Table 4: Node classification performance (Accuracy±Std) under non-targeted attacks(metattack)

Dataset	Ptb (%)	GCN [6]	GAT [7]	RGCN [13]	GCN-Jaccard [11]	GCN-SVD [18]	Pro-GNN [15]	ADA-UGNN[?]	**RoGAT**
Cora	0	83.50±0.44	84.57±0.65	83.09±0.44	82.05±0.51	80.63±0.45	82.98±0.23	83.48±0.25	**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**	81.37±0.51	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	**83.68±0.23**	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	75.53±0.83	**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	74.23±0.65	**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	64.74±0.83	**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	**76.29±0.63**	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	**74.13±0.92**	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	71.89±1.04	**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	72.09±1.32	**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	66.09±1.05	**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	67.88±0.98	**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**

dataset polblogs do not have node features, 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 result in Figure 3(a) and 3(b) 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. The results in Figure 3(a) and Figure 3(b) show that our RoGAT performs significantly better than other methods when dealing with Cora and Citeseer datasets, with an accuracy improvement of 15% and 13%, respectively. Since Polblogs do not have exact node features, RoGAT has relatively better performance, only slightly inferior to the gcn-svd method in Figure 3(c). Overall, RoGAT has quite good performance compared with most defensive methods when dealing with different types of adversarial attacks.
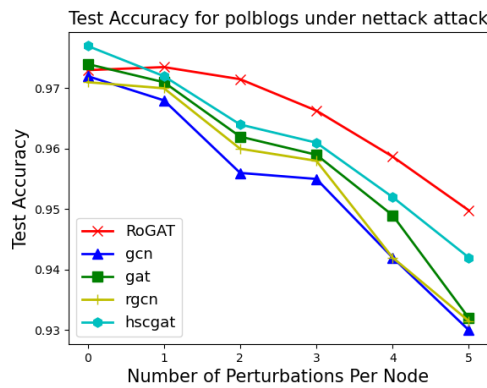
5.3 Ablation study

In this part, we build ablation studies to figure out how different components affect the model. As is shown in our model, two main pre-process procedures occur before training RoGAT. To understand the different impacts of each
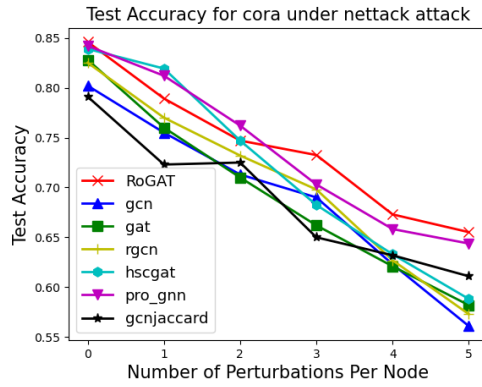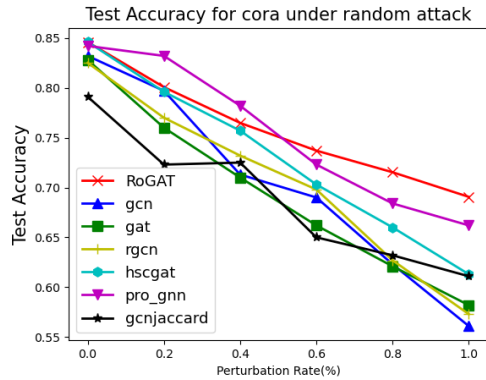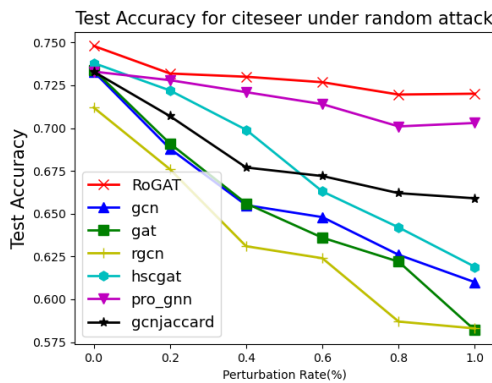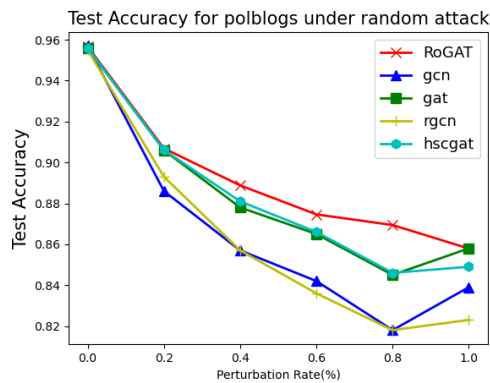
(a)
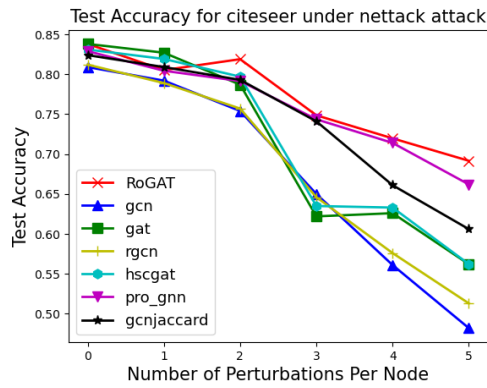
(b)

(c)

Figure 2: Results of Cora, Citeseer, Polblogs under netttack

(a)

(b)

(c)

Figure 3: Results of Cora, Citeseer, Polblogs under random attack

procedure, we eliminate the modification of graph structure and feature respectively to check the performance changes. Here we only report results on Cora, since other datasets have a similar performance. We use RoGAT(no structure) and RoGAT(no feature) to represent RoGAT without modification structure and feature respectively. In Table 5, we observe that when the attack is in low-level, RoGAT with no modification of structure has good performance, while RoGAT with no modification of feature has relatively better performance when the graph is heavily poisoned.

Table 5: Classfication accuracy performance of RoGAT variants under metattack

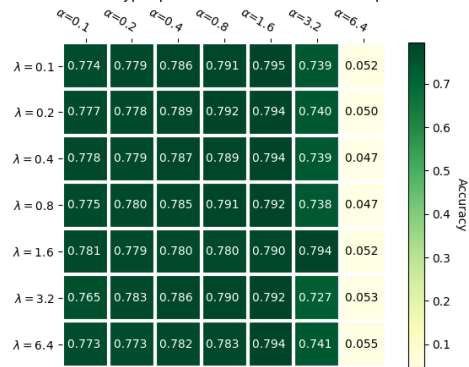| ptb	0	0.05	0.10	0.15	0.20	0.25
RoGAT(no structure)	82.95	78.92	76.86	68.31	63.22	60.40
RoGAT(no feature)	77.41	76.11	75.40	75.35	74.95	74.60
RoGAT	**84.59**	**81.22**	**79.53**	**80.47**	**78.40**	**78.90**

5.3.1 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 the test accuracy of RoGAT with different λ and α. The introduction of λ and α can contribute to the robustness of GAT. And compared with λ, the appropriate value of α has more influence 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 thus needs to be selected carefully. Therefore for RoGAT, using feature smoothness to revise the structure's attention is effective in defending adversarial attacks.

5.4 Attention weight analysis

Since we assume that the revised weight $\bar{\alpha}$ gives more attention to positive edges and reduces the influence of negative edges, we compute the ratio of weight between the negative edges and positive 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. The fake edges represent the edges generated by adversarial attacks, while the real edges represent the edges in the original graph. For different rates of perturbations and datasets, the weight ratio between fake edges and real edges decreases from the initial value 1 to a smaller value. And almost for all three datasets, the ratio decreases faster for the smaller perturbations, which leads to better performance. RoGAT can

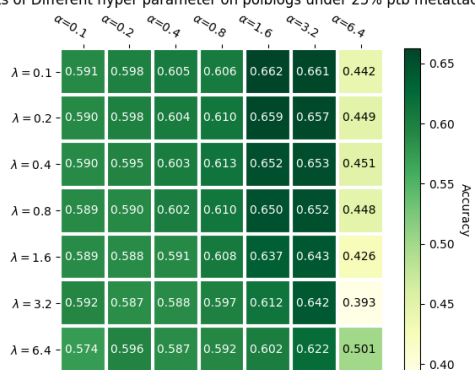Results of Different hyper parameter on cora under 25% ptb metattack

	$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.4$	$\alpha=0.8$	$\alpha=1.6$	$\alpha=3.2$	$\alpha=6.4$
$\lambda=0.1$	0.774	0.779	0.786	0.791	0.795	0.739	0.052
$\lambda=0.2$	0.777	0.778	0.789	0.792	0.794	0.740	0.050
$\lambda=0.4$	0.778	0.779	0.787	0.789	0.794	0.739	0.047
$\lambda=0.8$	0.775	0.780	0.785	0.791	0.792	0.738	0.047
$\lambda=1.6$	0.781	0.779	0.780	0.780	0.790	0.794	0.052
$\lambda=3.2$	0.765	0.783	0.786	0.790	0.792	0.727	0.053
$\lambda=6.4$	0.773	0.773	0.782	0.783	0.794	0.741	0.055

Cora

Results of Different hyper parameter on citeseer under 25% ptb metattack

	$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.4$	$\alpha=0.8$	$\alpha=1.6$	$\alpha=3.2$	$\alpha=6.4$
$\lambda=0.1$	0.731	0.730	0.729	0.730	0.728	0.670	0.141
$\lambda=0.2$	0.729	0.730	0.730	0.731	0.726	0.670	0.135
$\lambda=0.4$	0.731	0.730	0.729	0.731	0.725	0.661	0.136
$\lambda=0.8$	0.728	0.729	0.731	0.729	0.726	0.668	0.153
$\lambda=1.6$	0.731	0.728	0.727	0.731	0.727	0.661	0.156
$\lambda=3.2$	0.730	0.727	0.722	0.721	0.723	0.655	0.134
$\lambda=6.4$	0.733	0.724	0.735	0.734	0.725	0.654	0.135

Citeseer

Results of Different hyper parameter on polblogs under 25% ptb metattack

	$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.4$	$\alpha=0.8$	$\alpha=1.6$	$\alpha=3.2$	$\alpha=6.4$
$\lambda=0.1$	0.591	0.598	0.605	0.606	0.662	0.661	0.442
$\lambda=0.2$	0.590	0.598	0.604	0.610	0.659	0.657	0.449
$\lambda=0.4$	0.590	0.595	0.603	0.613	0.652	0.653	0.451
$\lambda=0.8$	0.589	0.590	0.602	0.610	0.650	0.652	0.448
$\lambda=1.6$	0.589	0.588	0.591	0.608	0.637	0.643	0.426
$\lambda=3.2$	0.592	0.587	0.588	0.597	0.612	0.642	0.393
$\lambda=6.4$	0.574	0.596	0.587	0.592	0.602	0.622	0.501

Polblogs

Figure 4: The performance for different λ and γ under 25% ptb metattack(Green color represents the performance)

adjust the ratio of contributions for fake and real edges. Therefore, it is consistent with the assumption that less attention will be given to the fake edges to reduce the influence of negative information during the aggregation procedure.

6 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 then proposes 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 recent baselines, especially in defending the metattack. Therefore, the prior information can help us to revise the attention score for fake and real edges progressively. Different graphs including homogeneous and heterogeneous have different prior information, which deserves further research in designing robust algorithms.

Acknowlegements

References

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

[2] Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. ICML'17, pp. 1263–1272. JMLR.org, ??? (2017)

[3] Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 1024–1034. Curran Associates, Inc., ??? (2017)

[4] Bruna, J., Zaremba, W., Szlam, A., Lecun, Y.: Spectral networks and locally connected networks on graphs. In: International Conference on Learning Representations (ICLR2014), CBLS, April 2014 (2014)

[5] Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in Neural Information Processing Systems, pp. 3844–3852 (2016)
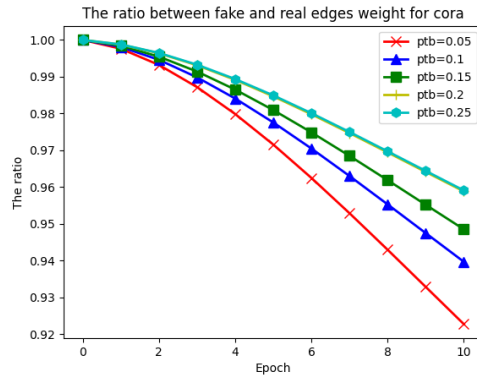
(a)
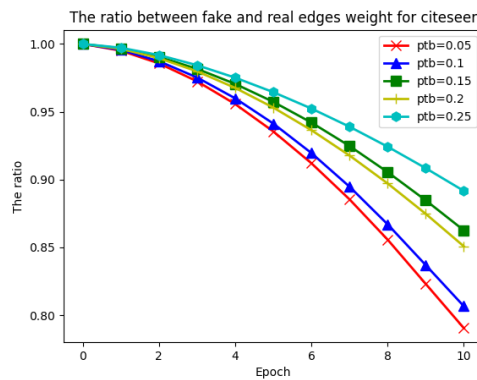
(b)
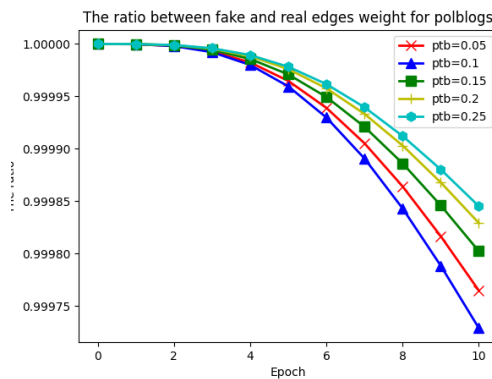
(c)

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

[6] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (ICLR) (2017)

[7] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (2018)

[8] 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, pp. 2847–2856 (2018)

[9] Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., Song, L.: Adversarial attack on graph structured data. In: International Conference on Machine Learning, pp. 1115–1124 (2018). PMLR

[10] Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. In: International Conference on Learning Representations (ICLR) (2019)

[11] Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., Zhu, L.: Adversarial examples for graph data: Deep insights into attack and defense. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, pp. 4816–4823. International Joint Conferences on Artificial Intelligence Organization, ??? (2019). https://doi.org/10.24963/ijcai.2019/669. https://doi.org/10.24963/ijcai.2019/669

[12] Zhang, X., Zitnik, M.: Gnnguard: Defending graph neural networks against adversarial attacks. Advances in Neural Information Processing Systems **33** (2020)

[13] Zhu, D., Zhang, Z., Cui, P., Zhu, W.: Robust graph convolutional networks against adversarial attacks. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '19, pp. 1399–1407. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3292500.3330851

[14] Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., Zhang, X.: Learning to drop: Robust graph neural network via topological denoising. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 779–787 (2021)

[15] Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., Tang, J.: Graph structure learning for robust graph neural networks. In: 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2020, pp. 66–74 (2020). Association for Computing Machinery

[16] Tang, X., Li, Y., Sun, Y., Yao, H., Mitra, P., Wang, S.: Transferring robustness for graph neural network against poisoning attacks. In: Proceedings of the 13th International Conference on Web Search and Data Mining, pp. 600–608 (2020)

[17] Jin, W., Derr, T., Wang, Y., Ma, Y., Liu, Z., Tang, J.: Node similarity preserving graph convolutional networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 148–156 (2021)

[18] Entezari, N., Al-Sayouri, S.A., Darvishzadeh, A., Papalexakis, E.E.: 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. WSDM '20, pp. 169–177. Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3336191.3371789

[19] Zügner, D., Günnemann, S.: Adversarial attacks on graph neural networks via meta learning. arXiv preprint arXiv:1902.08412 (2019)

[20] Kim, D., Oh, A.: How to find your friendly neighborhood: Graph attention design with self-supervision. In: International Conference on Learning Representations (2021)

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)

[22] Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv: Computation and Language (2014)

[23] Jin, W., Li, Y., Xu, H., Wang, Y., Tang, J.: Adversarial attacks and defenses on graphs: A review and empirical study. arXiv e-prints, 2003 (2020)

[24] Wang, H., Zhang, F., Zhang, M., Leskovec, J., Zhao, M., Li, W., Wang, Z.: Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 968–977 (2019)

[25] Li, Y., Jin, W., Xu, H., Tang, J.: Deeprobust: A pytorch library for adversarial attacks and defenses. arXiv preprint arXiv:2005.06149 (2020)

[26] Zhang, K., Zhu, Y., Wang, J., Zhang, J.: Adaptive structural fingerprints for graph attention networks. In: International Conference on Learning Representations (2019)

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