

# Robust Graph Neural Networks Against Adversarial Attacks via Jointly Adversarial Training

Hu Tian \* Bowei Ye \*\* Xiaolong Zheng \*\*\*  
Desheng Dash Wu \*\*\*\*

\* *The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China (e-mail: tianhu2018@ia.ac.cn).*

\*\* *Department of Statistics, University of Illinois in Urbana-Champaign, Illinois, USA (e-mail: boweiye2@illinois.edu)*

\*\*\* *The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China; School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China (e-mail: xiaolong.zheng@ia.ac.cn)*

\*\*\*\* *School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China; Stockholm Business School, Stockholm University, SE-106 91 Stockholm, Sweden, (e-mail: dash@risklab.ca)*

**Abstract:** Graph neural networks (GNNs) are powerful tools for analyzing graph-structured data. However, recent studies have shown that GNNs are vulnerable to small but intentional perturbations of input features and graph structures in the node classification task. Existing researches focus on enhancing the robustness of GNNs for a single type of perturbation such as graph structure perturbation or node feature perturbation. An ideal graph neural networks model should be able to resist the two kinds of perturbations. For this purpose, we propose a new adversarial training method for graph-structured data named Graph Jointly Adversarial Training (GJAT) which incorporates Graph Structure Adversarial Training (GSAT) and Graph Feature Adversarial Training (GFAT) two components and can resist perturbations from the topological structure and node attribute. Extensive experimental results demonstrate that our proposed method combining two kinds of adversarial training strategies can effectively improve the robustness of graph convolutional networks (GCNs) which is an important subset of GNNs.

Copyright © 2020 The Authors. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0>)

**Keywords:** Graph neural networks, Robustness, Adversarial Attack, Adversarial Training, Deep Learning.

## 1. INTRODUCTION

Graph structure data plays an important role in many real-world applications. Graph neural networks (GNNs), a powerful tool to implement training and inference on graph data, have shown promising results in various domains, including computer vision (Wang et al., 2018), natural language processing (Marcheggiani and Titov, 2017), applied chemistry and biology (Kearnes et al., 2016), etc.

Despite the remarkable success of GNNs, recent studies show that GNNs are vulnerable to adversarial attacks (Dai et al., 2018; Zügner et al., 2018; Zügner and Günnemann, 2019). These attacks are usually stealthy and only perform subtle perturbations in graph structures or node features to fool the GNNs. State-of-the-art GNNs usually follow a “message-passing” framework (Gilmer et al., 2017) where each node aggregates information from its immediate

neighbors in each layer. We argue that the vulnerability of GNNs could be more serious than the standard neural networks that do not model the graph structure, since the smoothness constraint of GNNs (Li et al., 2018) aggregates the impact of perturbations from nodes connected to the target node and exacerbates the impact of perturbations to the target node. In practical applications, models should guarantee that small perturbations like the updates of node features or edges should not change the predictions much. Therefore, there is a strong need to enhance the robustness of GNNs from the graph structure level and node feature level.

It is worth mentioning that adversarial training has been empirically shown to be able to stabilize neural networks, and enhance their robustness against perturbations in standard classification tasks (Kurakin et al., 2016). An advantage of adversarial training is that it can generate

samples with fatal attack information. Thus, it might be useful to improve the robustness of GNNs by incorporating adversarial training into the graph neural network models.

However, due to the discrete nature of the graph and complex interdependency between nodes, it is non-trivial to directly implement the structure perturbation and feature perturbation. We propose a novel adversarial training method for graph-structure data named Graph Jointly Adversarial Training (GJAT) including Graph Structure Adversarial Training (GSAT) and Graph Feature Adversarial Training (GFAT) two components, which learns to resist perturbations from topological structure and node attribute respectively. To verify the effectiveness of GJAT, we employ it on the Graph Convolutional Network (GCN) (Kipf and Welling, 2016), a state-of-the-art graph neural network model and conduct experiments on two graph datasets (Citeseer, Cora). To the best of our knowledge, we are the first to try to stabilize the graph neural network models from the perspectives of graph structure level and node feature level.

The rest of the paper is organized as follows. In Section 2, we review the related works. In Section 3, we summarize the notations and give some preliminaries. We introduce our proposed method in Section 4 and report experimental results in Section 5. Finally, we conclude the paper and point out future directions.

## 2. RELATED WORK

Our work builds upon three categories of recent researches: adversarial robustness of deep learning, graph adversarial attack and graph adversarial defense.

Adversarial robustness of deep learning is already a hot direction. Substantial related researches have proposed various adversarial attack and defense methods (Goodfellow et al., 2014; Samangouei et al., 2018). Benefiting from the adversarial attack and defense, adversarial training has been leveraged to enhance the robustness of the deep learning methods (Goodfellow et al., 2014). From the optimization perspective, adversarial training is an alternative minimax optimization process (Goodfellow et al., 2014).

Recently, a few attempts have been made to study adversarial attacks on GNNs. Dai et al. (2018) proposed a non-target evasion attack on node classification and graph classification. A reinforcement learning method was used to learn the attack policy that applies small-scale modification to the graph structure. Zügner et al. (2018) introduced a poisoning attack on node classification. This work adopted a substitute model attack and formulated the attack problem as a bi-level optimization. Different from Dai et al. (2018), it attacked both the graph structure and node attributes. Furthermore, it considered both direct attacks and influence attacks. Chen et al. (2018a,b) applied adding/deleting edge strategies. Zügner and Günnemann (2019) proposed a simple yet strong global attack with an interpretable strategy (remove internally, insert externally), where it randomly disconnects nodes from the same class and connects nodes from different classes.

So far, there are only a few studies about the adversarial defense for GNNs. Zhu et al. (2019) proposed a robust GCN which, instead of representing nodes as vectors,

adopts Gaussian distributions as the hidden representations of nodes in each convolutional layer and variance-based attention mechanism Jin et al. (2019) highlighted some basic flaws of the fundamental building block of GCN—the Laplacian operator and proposed a new operator based on graph powering to construct an architecture with improved spectral robustness. Overall, it still needs to seek a more general approach to improve the robustness of the GNNs.

## 3. NOTATIONS AND PRELIMINARIES

To demonstrate the utility of GJAT, we will briefly introduce how the Graph Convolutional Network (GCN) (Kipf and Welling, 2016), a representative graph neural network model, works in the semi-supervised node classification and formalize the graph structure perturbation and the node feature perturbation.

### 3.1 Semi-supervised Classification with GCNs

In this paper, a graph is defined as  $G = (\mathbf{V}, \mathbf{E})$ , where  $\mathbf{V} = \{v_1, \dots, v_N\}$  denotes the set of nodes,  $N = |\mathbf{V}|$  is the number of nodes, and  $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$  is the set of edges between nodes with cardinality  $M = |\mathbf{E}|$ . Let  $\mathbf{A}$  represent a binary adjacency matrix.  $A_{ij} = 1$  if there is an edge between node  $i$  and  $j$ , otherwise  $A_{ij} = 0$ .  $\mathbf{X}$  is the node features matrix and  $\mathbf{D} = \text{diag}(d_1, d_2, \dots, d_N)$  denotes the degree matrix of  $\mathbf{A}$ , where  $d_i = \sum_j A_{ij}$  is the degree of vertex  $i$ .

The original form of Kipf’s GCN with two convolutional layers be defined using the following equation:

$$\mathbf{Z} = f(\mathbf{X}, \mathbf{A}) = \text{soft max} \left( \hat{\mathbf{A}} \text{ReLU} \left( \hat{\mathbf{A}} \mathbf{X} \mathbf{W}^{(0)} \right) \mathbf{W}^{(1)} \right), \quad (1)$$

where  $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{\frac{1}{2}}$ ,  $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ .  $\tilde{\mathbf{D}}$  is the degree matrix of  $\tilde{\mathbf{A}}$ .  $\mathbf{I}$  is an identity matrix.  $\mathbf{Z}$  is the predicted labels matrix of nodes.  $\mathbf{W}^{(0)}$  is the input-to-hidden weight matrix and  $\mathbf{W}^{(1)}$  is the hidden-to-output weight matrix. Finally, the loss function is defined as the cross entropy error over all labeled nodes:

$$L_0 = - \sum_{l \in \mathbf{V}_L} \sum_{c=1}^C y_{lc} Z_{lc}, \quad (2)$$

where  $\mathbf{V}_L$  is the set of node indices that have labels and  $C$  is the number of categories.

### 3.2 Graph Structure Perturbation and Feature Perturbation

To formalize the edge manipulation, we employ the same method as (Xu et al., 2019) did. We introduce a Boolean symmetric matrix  $\mathbf{B} \in \{0, 1\}^{N \times N}$  to encode whether or not an edge in  $G$  is modified. If the edge connecting nodes  $i$  and  $j$  is perturbed (added or removed), then  $\mathbf{B}_{ij} = \mathbf{B}_{ji} = 1$ . Otherwise,  $\mathbf{B}_{ij} = 0$  if  $i = j$  or the edge  $(i, j)$  is not modified. Then a perturbed graph can be given by

$$\mathbf{A}' = \mathbf{A} + (\mathbf{C} - \mathbf{A}) \odot \mathbf{B}, \quad (3)$$

where  $\mathbf{C} = \mathbf{1}\mathbf{1}^T - \mathbf{A} - \mathbf{I}$  denotes the supplement of existing edges and  $\odot$  denotes the element-wise product. The term  $(\mathbf{C} - \mathbf{A}) \odot \mathbf{B}$  is the graph structure perturbations. To

perturb a graph, we just need to give the encoding matrix  $\mathbf{B}$ . We replace the symmetric matrix variable  $\mathbf{B}$  with its vector form  $\mathbf{b}$  that consists of  $n := N(N-1)/2$  unique perturbation variables in  $\mathbf{B}$ . Compared with the graph structure perturbations, the graph feature perturbations are more easy to depict. We define a node feature perturbation  $\mathbf{r}_i$  for node  $i$ . Term  $\mathbf{x}_i + \mathbf{r}_i$  represents implementing the perturbation  $\mathbf{r}_i$  to the feature vector  $\mathbf{x}_i$  of node  $i$ .

#### 4. GRAPH ADVERSARIAL TRAINING

In this section, we first introduce our proposed Graph Structure Adversarial Training (GSAT) and Graph Feature Adversarial Training (GFAT). Then we present the Graph Jointly Adversarial Training (GJAT).

##### 4.1 Graph Structure Adversarial Training

Applying GSAT would regulate the model parameters to resist the vulnerable local structure in the graph. Therefore, the key challenge is how to define the vulnerable local structure. To solve this problem, in this paper, we regard as those edges as the vulnerable structure, which significantly increase the training loss after modified by  $\mathbf{B}$ . Inspired by the philosophy of standard adversarial training (Goodfellow et al., 2014), we give the following graph adversarial training loss:

$$\begin{aligned} \min_{\mathbf{w}} \max_{\mathbf{b}} F(\mathbf{b}, \mathbf{W}) = & \sum_{i \in \mathbf{V}_L} D(f(\mathbf{x}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \mathbf{y}_i) \\ & + \alpha \sum_{i \in \mathbf{V}_{UL}} D(f(\mathbf{x}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \bar{\mathbf{y}}_i) \quad (4) \\ \text{s.t. } & \mathbf{1}^T \mathbf{b} \leq \varepsilon, \mathbf{b} \in \{0, 1\}^n \end{aligned}$$

where  $D(\cdot, \cdot)$  measures the divergence between two distributions such as cross-entropy function,  $f(\mathbf{x}_i, \mathbf{A}; \mathbf{W}, \mathbf{b})$  is the predicted probability vector of node  $i$ ,  $\mathbf{y}_i$  is the one-hot vector of true label of node  $i$ ,  $\mathbf{V}_L$  is the set of node indices that have labels,  $\mathbf{V}_{UL}$  is the set of unlabeled node indices,  $\alpha$  controls the impact of the loss of the unlabeled nodes,  $\varepsilon$  constraints the number of edge to be perturbed and

$$\bar{\mathbf{y}}_i = f(\mathbf{x}_i, \mathbf{A}; \bar{\mathbf{W}}, \mathbf{b}) \quad (5)$$

is the prediction result vector of unlabeled node  $i$  under the training process, where  $\bar{\mathbf{W}}$  is the initial parameter of GCN model to be trained.

Problem (4) is a combinatorial optimization problem due to the presence of binary variables. To optimize it with a continuous way, we employ the same method as Xu et al. (2019) did. We relax  $\mathbf{b} \in \{0, 1\}^n$  to its convex hull  $\mathbf{b} \in [0, 1]^n$  and solve the following continuous optimization problem,

$$\begin{aligned} \min_{\mathbf{w}} \max_{\mathbf{b}} F(\mathbf{b}, \mathbf{W}) = & \sum_{i \in \mathbf{V}_L} D(f(\mathbf{x}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \mathbf{y}_i) \\ & + \sum_{i \in \mathbf{V}_{UL}} D(f(\mathbf{x}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \bar{\mathbf{y}}_i) \quad (6) \\ \text{s.t. } & \mathbf{b} \in \mathcal{B}, \end{aligned}$$

where  $\mathcal{B} = \{\mathbf{b} | \mathbf{1}^T \mathbf{b} \leq \varepsilon, \mathbf{b} \in [0, 1]^n\}$ . Problem (6) can be solved by first-order alternating optimization (Xu et al., 2019; Lu et al., 2018), where the outer minimization is

handled by gradient descent, and the inner maximization is solved by projected gradient ascent, and (PGA),

$$\mathbf{b}^{(t)} = \Pi_{\mathcal{B}} [\mathbf{b}^{(t-1)} + \eta_t \hat{\mathbf{g}}_t], \quad (7)$$

where  $t$  denotes the iteration index of PGA,  $\eta_t > 0$  is the learning rate at iteration  $t$ ,  $\hat{\mathbf{g}}_t = \nabla F(\mathbf{b}^{(t-1)}, \mathbf{W}^{(t-1)})$  denotes the gradient of the attack loss  $F$  evaluated at  $\mathbf{b}^{(t-1)}$ , and  $\Pi_{\mathcal{B}}(\mathbf{a}) = \arg \min_{\mathbf{b} \in \mathcal{B}} \|\mathbf{b} - \mathbf{a}\|_2^2$  is the projection operator at the constraint set  $\mathcal{B}$ . In fact, under the constraint  $\mathcal{B}$ , the PGA exists a closed-form solution:

$$\Pi_{\mathcal{B}}(\mathbf{a}) = \begin{cases} P_{[0,1]}[\mathbf{a} - \mu \mathbf{1}], & \mu > 0, \mathbf{1}^T P_{[0,1]}[\mathbf{a} - \mu \mathbf{1}] = \varepsilon, \\ P_{[0,1]}[\mathbf{a}], & \mathbf{1}^T P_{[0,1]}[\mathbf{a}] \leq \varepsilon, \end{cases} \quad (8)$$

where  $P_{[0,1]}[x] = x$  if  $x \in [0, 1]$ , 0 if  $x < 0$ , and 1 if  $x > 1$ . The detail proof can be found in (Xu et al., 2019). Due to  $\mathbf{b} \in [0, 1]^n$ , we use a randomization sampling to obtain the binary values (Liu et al., 2016). We summarize the graph structure adversarial training in **Algorithm 1**.

---

##### Algorithm 1 Graph structure adversarial training

---

- 1: Input:  $\mathbf{b}^{(0)}$ ,  $\mathbf{W}^{(0)}$ ,  $\alpha > 0, \varepsilon > 0$ , learning rates  $\gamma_t$  and  $\eta_t$ , iterations  $T_1$  and  $T_2$ , and # of random trials  $K$ ,
  - 2: **for**  $t_1 = 1, 2, \dots, T_1$  **do**
  - 3:   **for**  $t_2 = 1, 2, \dots, T_2$  **do**
  - 4:     gradient ascent:  $\mathbf{a}^{(t_2)} = \mathbf{b}^{(t_2-1)} + \eta_{t_2} \nabla F(\mathbf{b}^{(t_2-1)})$ ,
  - 5:     call projection operation in (7),
  - 6:   **end for**
  - 7:   **for each**  $k = 1, 2, \dots, K$  **do**
  - 8:     draw binary vector  $\mathbf{u}^{(k)}$  following
  - 9:       
$$u_i^{(k)} = \begin{cases} 1 & \text{with probability } b_i \\ 0 & \text{with probability } 1 - b_i \end{cases}, \forall i$$
  - 10:   **end for**
  - 11:   choose a vector  $\mathbf{b}^*$  from  $\mathbf{u}^{(k)}$  which yields the biggest adversarial training loss,  $F(\mathbf{u}^{(k)})$  under  $\mathbf{1}^T \mathbf{b} \leq \varepsilon$ ,
  - 12:   outer minimization over  $\mathbf{W}$ : given  $\mathbf{b}^{(t)}$ , obtain
  - 13:     
$$\mathbf{W}^{(t_1)} = \mathbf{W}^{(t_1-1)} - \gamma_t \nabla_{\mathbf{W}} F(\mathbf{b}^{(t_1)}, \mathbf{W}^{(t_1-1)}),$$
  - 14: **return**  $\hat{\mathbf{W}} = \mathbf{W}^{(T_1)}$
- 

##### 4.2 Graph Feature Adversarial Training

To prevent node feature perturbation propagation in the “message-passing” framework, we design the GFAT as a regularization term that enforces the model parameters to smooth the output label of model in the local graph structure. Here we formally define the graph feature adversarial training loss in GCNs, where adversarial perturbations are added on features of all nodes:

$$\begin{aligned} \min_{\mathbf{w}} \max_{\mathbf{r}} L_1 = & \sum_{i \in \mathbf{V}_L} D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \mathbf{y}_i) \\ & + \beta \sum_{i \in \mathbf{V}_{UL}} D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}, \mathbf{b}), \bar{\mathbf{y}}_i) \quad (9) \\ \text{s.t. } & \|\mathbf{r}\| \leq \delta \end{aligned}$$

where  $\mathbf{r}_i$  denotes the feature adversarial perturbation, that is the direction which leads to the largest change on the model prediction of  $\mathbf{x}_i$ ,  $\delta$  controls the magnitude of feature perturbation such that the feature distribution of the adversarial examples is close to that of the clean examples.  $\beta$  weights the loss between unlabeled nodes and

label nodes,  $\mathbf{y}_i$  and  $\tilde{\mathbf{y}}_i$  denote the ground truth label and model prediction, respectively. And

$$\tilde{\mathbf{y}}_i = f(\mathbf{x}_i, \mathbf{A}; \hat{\mathbf{W}}, \mathbf{b}), \quad (10)$$

where  $\hat{\mathbf{W}}$  is the return of **Algorithm 1**.

In fact, problem 9 is precisely Virtual Adversarial Training (Miyato et al., 2018) that has a ready-made optimization method. This problem can be optimized by leveraging the approximation method. For labeled nodes,  $\mathbf{r}_i$  can be easily evaluated via linear approximation (Goodfellow et al., 2014), i.e., calculating the gradient of  $D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}), \mathbf{y}_i)$  w.r.t.  $\mathbf{x}_i$ . However, linear approximation is infeasible for unlabeled nodes since the gradient will always be zero because the gradient of  $D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}), \tilde{\mathbf{y}}_i)$  achieves the minimum value (0) at  $\mathbf{x}_i$  ( $\mathbf{r}_i = \mathbf{0}$ ). Although the first-order gradient is always zero, it is feasible to estimate  $\mathbf{r}_i$  from the second-order Taylor approximation of  $D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}), \tilde{\mathbf{y}}_i)$ . That is,  $\mathbf{r}_i \approx \arg\max_{\mathbf{r}_i', \|\mathbf{r}_i'\| \leq \delta} \frac{1}{2} \mathbf{r}_i'^T \mathbf{H} \mathbf{r}_i'$ , where  $\mathbf{H}$  is the Hessian matrix of  $D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}), \tilde{\mathbf{y}}_i)$ . Due to the time cost of calculating Hessian matrix, Miyato et al. (2018) calculate  $\mathbf{r}_i$  via the power iteration approximation:

$$\mathbf{r}_i = \delta \frac{\mathbf{g}}{\|\mathbf{g}_i\|_2}, \quad \mathbf{g} = \nabla_{\mathbf{r}_i} D(f(\mathbf{x}_i + \mathbf{r}_i, \mathbf{A}; \mathbf{W}))|_{\mathbf{r}_i = \xi \mathbf{d}}, \quad (11)$$

where  $\mathbf{d}$  is a random vector and  $\xi$  is the step length of the finite difference. Detailed derivation of the method is referred to Miyato et al. (2018). Finally, we summarize the graph feature adversarial training in **Algorithm 2**.

---

**Algorithm 2** Graph feature adversarial training

---

- 1: Input:  $\mathbf{W}^{(0)}$ ,  $\delta > 0$ ,  $\beta > 0$ ,  $\xi > 0$ , learning rates  $\mu_t$ , iterations  $T_3$
  - 2: **for**  $t_1 = 1, 2, \dots, T_3$  **do**
  - 3: generate a random unit vector:  $\mathbf{d}_i$  using an iid Gaussian distribution,
  - 4: calculate the gradient  $\mathbf{g}_i$  of  $L_1$  with respect to  $\mathbf{r}_i$  on  $\mathbf{r}_i = \xi \mathbf{d}_i$  on each node feature  $\mathbf{x}_i$ ,
  - 5: calculate the feature adversarial perturbation  $\mathbf{r}_i = \delta \frac{\mathbf{g}_i}{\|\mathbf{g}_i\|_2}$ ,
  - 6: update the parameters of the model:  $\mathbf{W}^{(t)} = \mathbf{W}^{(t-1)} + \mu_t \nabla_{\mathbf{W}} L_1$ ,
  - 7: **end for**
  - 8: return  $\bar{\mathbf{W}} = \mathbf{W}^{(T_3)}$
- 

### 4.3 Graph Jointly Adversarial Training

An ideal graph neural network model should be able to resist the two kinds of perturbations. We design a graph jointly adversarial training (GJAT) to enhance the robustness of the GNNs by combining the graph feature adversarial training (GFAT) and the graph structure adversarial training (GSAT). The combination of GSAT and GFAT is straightforward. To gain benefits from two methods, we alternate GFAT and GSAT. We summarize the graph jointly adversarial training in **Algorithm 3**.

## 5. EXPERIMENTS

### 5.1 Compared Methods

To evaluate the effectiveness of our proposed GFAT, GSAT and GJAT, we the following methods as baselines.

---

**Algorithm 3** Graph jointly adversarial training

---

- 1: Input:  $\mathbf{W}^{(0)}$ ,  $\varepsilon, \delta > 0$ ,  $\alpha, \beta > 0$ ,  $\xi > 0$ , learning rates  $\mu_t, \gamma_t$  and  $\eta_t$ , iterations  $T_1, T_2, T_3$  and  $T_4$  and # of random trials  $K$ ,
  - 2: **for**  $t_1 = 1, 2, \dots, T_4$  **do**
  - 3:  $\hat{\mathbf{W}}^{(t)} = \mathbf{W}^{(t-1)}$ ,
  - 4: obtain  $\bar{\mathbf{W}}^{(t)}$  by implementing the **Algorithm 2**,
  - 5: obtain  $\hat{\mathbf{W}}^{(t)}$  by implementing the **Algorithm 1**,
  - 6:  $\mathbf{W}^{(t)} = \hat{\mathbf{W}}^{(t)}$ ,
  - 7: **end for**
  - 8: return  $\mathbf{W} = \mathbf{W}^{(T_4)}$
- 

Table 1. The statistics of the datasets.

Dataset	#Node	#Edge	#Feature	#Class
Citeseer	2,110	3,757	3,703	3
Cora	2,810	7,981	1,433	7

- DeepWalk (Perozzi et al., 2014) is a graph embedding method which applies the technique for learning word representations in multiple random walks from the graph to generate node embedding.
- GCN (Kipf and Welling, 2016) has two graph convolution layers to map the node features and structure to labels and propagates node representations and labels to unlabeled nodes.
- GraphSAGE (Hamilton et al., 2017) is an extended, inductive version of Kipf et al's semi-supervised GCN (Kipf and Welling, 2016).
- GAT (Veličković et al., 2017) enables specifying different weights to different nodes in a neighborhood during stacking the graph convolution layers.

Besides, we also choose the Multilayer Perceptron (MLP) as a baseline.

To evaluate the performances of our proposed methods on defending adversarial attacks, we choose three popular adversarial attacks:

- FGSM (Goodfellow et al., 2014) generates adversarial examples by producing subtle node feature perturbations based on the sign of gradient.
- NETTACK (Zügner et al., 2018) is a state-of-the-art graph structure attack method based on optimization.
- RAND perturbs the node features by randomly adding or deleting some attributes and modifies the graph structure by randomly adding edges to the node that belongs to a different class and/or deleting an edge that connected to the node within the same class.

### 5.2 Learning Settings

For the benchmark datasets, we choose two well-known citation network datasets: Cora and Citeseer (Sen et al., 2008) for node classification tasks. The statistics of two datasets are reported in Table 1. We follow the same experimental settings as in (Xu et al., 2019). During training process, all nodes and features are accessible, but only 40 (20) labels are fed into the model on the dataset Citeseer (Cora). The number of test labeled nodes is 1000 for both datasets. In our paper, we set  $T_1 = 1, T_2 = 10, T_3 = 1, T_4 = 800$  and  $K = 15$  for all datasets.

Table 2. The results of node classification accuracy (%) on the clean datasets.

Methods	Citeseer	Cora
MLP	46.5	55.1
Deepwalk	43.2	67.2
GCN	70.3	81.4
GraphSAGE	70.9	81.2
GAT	<b>72.5</b>	<b>83.3</b>
GFAT	70.7	81.6
GSAT	75.1	81.8
GJAT	70.1	81.4

Table 3. The results of node classification accuracy (%) on the Cora dataset under various attacks.

Methods	Cora			
	CLEAN	FGSM	NETTACK	RAND
GCN	81.4	73.5	60.7	61.8
GraphSAGE	81.2	71.3	63.3	60.5
GAT	<b>83.3</b>	75.2	64.1	62.3
GFAT	81.6	<b>78.7</b>	66.4	67.6
GSAT	81.8	74.5	70.5	66.9
GJAT	81.4	77.6	<b>71.2</b>	<b>68.8</b>

### 5.3 Against Adversarial Attacks

Firstly, we conduct experiments on the clean datasets. The experimental results are reported in Table 2. We can observe that our proposed methods can achieve comparable results on Citeseer and Cora, which indicates that GFAT, GSAT and GJAT are adequate for traditional node classification task. Next, we mainly investigate the prediction accuracy of our three defense methods under three types of attack methods. The results are shown in Table 3 and 4. CLEAN represents the performance of corresponding methods on the clean datasets.

From the Tables 3 and 4, we can find GJAT can improve the prediction accuracy significantly except the NETTACK on the Citeseer dataset. Quantitatively, GJAT can improve the prediction accuracy upon the best baseline by a margin from 3.19% to 21.98%. Especially, we find GFAT and GSAT can resist the corresponding attack methods to some extent. In most cases, GJAT the combination of GFAT and GSAT can achieve better results than the individual method. This observation implies that jointly adversarial training considering the node feature and graph structure is a promising direction to enhance the robustness of graph neural networks.

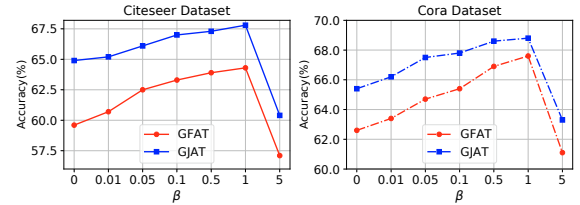
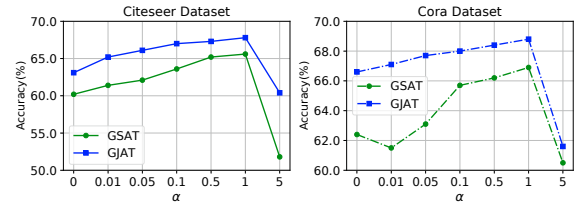
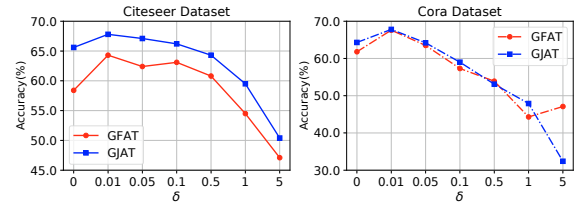
### 5.4 Parameters Analysis

Finally, we conduct the sensitive analysis to investigate the effects of hyperparameters ( $\beta, \alpha, \delta, \varepsilon$ ) for the performance of our proposed methods. Each experiment is implemented by trying the different target hyperparameter and fixing the other hyperparameters. The results are reported in Figures 1, 2, 3 and 4.

From these figures, we can observe the following results:

Table 4. The results of node classification accuracy (%) on Citeseer dataset under various attacks.

Methods	Citeseer			
	CLEAN	FGSM	NETTACK	RAND
GCN	70.3	49.1	51.2	58.4
GraphSAGE	70.9	50.5	54.7	59.1
GAT	<b>72.5</b>	49.9	52.4	62.9
GFAT	70.7	60.6	53.8	64.3
GSAT	70.5	53.2	<b>55.6</b>	65.6
GJAT	70.1	<b>61.6</b>	55.0	<b>67.8</b>

Fig. 1. Evaluation accuracy of GFAT and GJAT with different values of hyperparameter  $\beta$  on Citeseer and Cora datasets, which is the weight of unlabeled nodes loss of GFAT.Fig. 2. Evaluation accuracy of GSAT and GJAT with different values of hyperparameter  $\alpha$  on Citeseer and Cora datasets, which is the weight of unlabeled nodes loss of GSAT.Fig. 3. Evaluation accuracy of GFAT and GJAT with different values of hyperparameter  $\delta$  on Citeseer and Cora datasets, which controls the perturbation magnitude of node feature.

- GFAT, GSAT and GJAT are relatively insensitive to the regularization weights  $\alpha$  and  $\beta$ . When  $\alpha$  and  $\beta$  are in the interval  $[0.1, 1]$ , they can achieve the strong performance in most cases.
- GFAT, GSAT and GJAT are relatively insensitive to the feature and structure perturbations when  $\delta$  and  $\varepsilon$  are in a very small interval.
- Under most cases, GJAT has better performance than GFAT and GSAT. This phenomenon demonstrates the advantage of GJAT.

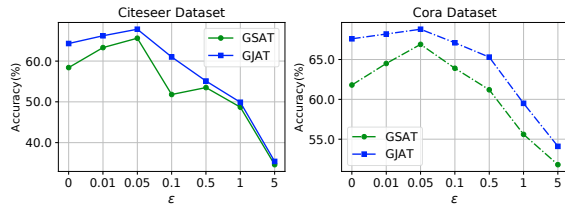


Fig. 4. Evaluation accuracy of GSAT and GJAT with different values of hyperparameter  $\varepsilon$  on Citseer and Cora datasets, which controls the perturbation magnitude of graph structure.

## 6. CONCLUSION

In this paper, we proposed a jointly adversarial training method to enhance the robustness of the graph neural networks by combining the graph feature adversarial training and graph structure adversarial training. Our extensive experiments show that our graph jointly adversarial training can resist the attacks from the node feature attack and graph structure attack effectively. Our proposed graph feature adversarial training and graph structure adversarial training can be used to deal with different application settings. However, the graph structure adversarial training involves the a discrete optimization, which limits the possibilities for the large scale scenarios. Scaling up the robust graph neural networks is still an open problem.

## ACKNOWLEDGEMENTS

This work is supported by the Ministry of Health of China under Grant No. 2017ZX10303401-002 and 2017YFC1200302, and the Natural Science Foundation of China under Grant No. 71472175, 71602184 and 71621002, and the National Key Research and Development Program of China under 2016QY02D0305.

## REFERENCES

- Chen, J., Shi, Z., Wu, Y., Xu, X., and Zheng, H. (2018a). Link prediction adversarial attack. *arXiv preprint arXiv:1810.01110*.
- Chen, J., Wu, Y., Xu, X., Chen, Y., Zheng, H., and Xuan, Q. (2018b). Fast gradient attack on network embedding. *arXiv preprint arXiv:1809.02797*.
- Dai, H., Li, H., Tian, T., Huang, X., Wang, L., Zhu, J., and Song, L. (2018). Adversarial attack on graph structured data. *arXiv preprint arXiv:1806.02371*.
- Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., and Dahl, G.E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 1263–1272. JMLR. org.
- Goodfellow, I.J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.
- Jin, M., Chang, H., Zhu, W., and Sojoudi, S. (2019). Power up! robust graph convolutional network against evasion attacks based on graph powering. *arXiv preprint arXiv:1905.10029*.
- Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. (2016). Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8), 595–608.
- Kipf, T.N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.
- Li, Q., Han, Z., and Wu, X.M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- Liu, S., Chepuri, S.P., Fardad, M., Maşazade, E., Leus, G., and Varshney, P.K. (2016). Sensor selection for estimation with correlated measurement noise. *IEEE Transactions on Signal Processing*, 64(13), 3509–3522.
- Lu, S., Singh, R., Chen, X., Chen, Y., and Hong, M. (2018). Understand the dynamics of gans via primal-dual optimization.
- Marcheggiani, D. and Titov, I. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*.
- Miyato, T., Maeda, S.i., Ishii, S., and Koyama, M. (2018). Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*.
- Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 701–710. ACM.
- Samangouei, P., Kabkab, M., and Chellappa, R. (2018). Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*.
- Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3), 93–93.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.
- Wang, X., Ye, Y., and Gupta, A. (2018). Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6857–6866.
- Xu, K., Chen, H., Liu, S., Chen, P.Y., Weng, T.W., Hong, M., and Lin, X. (2019). Topology attack and defense for graph neural networks: An optimization perspective. *arXiv preprint arXiv:1906.04214*.
- Zhu, D., Zhang, Z., Cui, P., and Zhu, W. (2019). Robust graph convolutional networks against adversarial attacks. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19)*, <https://doi.org/10.1145/3292500.3330851>.
- Zügner, D., Akbarnejad, A., and Günnemann, S. (2018). Adversarial attacks on neural networks for graph data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2847–2856. ACM.
- Zügner, D. and Günnemann, S. (2019). Adversarial attacks on graph neural networks via meta learning. *arXiv preprint arXiv:1902.08412*.