# Graph Adversarial Training for More Accurate and Robust GNN

**Difan Zou, Shanxiu He, Ziniu Hu, Zongyue Qin**

## Abstract

Adversarial training, in which neural networks are trained to withstands adversarial attack examples, is widely studied to improve model robustness. Recently, many studies also show that adversarial training can also improve the model generalization to out-of-distribution samples. In this project, we aim to study whether we can improve the robustness and generalization of Graph Neural Networks (GNN) via adversarial training. The difficulty of graph adversarial training is that the graph structure is discrete, so that traditional gradient-based attack methods cannot be directly utilized. We aim to relax the discrete constraint of the graph structure with continuous approximation, and apply standard adversarial training methods on both the structure and node features.

## 1 Introduction

Graph Neural Networks (GNNs) have achieved remarkable success in various graph applications. They leverage the graph structure as the computational flow, and only pass information with connected nodes, which allows the model to integrate both node features and topological structure information.

As GNNs highly rely on the discrete graph structure to conduct message passing, they could be vulnerable to the adversarial attack on the graph. Researchers have proposed several studies to show by changing the graph structure by adding/removing some nodes or edges within a budget, the performance will drop significantly. To enhance the GNN robustness against this adversarial attack, a series of work have been proposed, including GNN-Jaccard [7], Graphdefense [6], GNNGuard [9], etc.

Recently, many researchers have found that adding adversarial training not only enhance the model robustness, but can also improve generalization, especially when the training dataset is scarse and there exist out-of-distribution samples in test set. In graph domain, FLAG [4] tries to utilize PGD on the node features as data augmentation to improve GNN generalization.

Our project aim to apply the state-of-the-art adversarial training techniques to both the node features and graph structures, and seek whether we can improve the GNN robustness as well as generalization performance. In addition, we want to see whether the proposed adversarial augmentation could be benefit to contrastive learning framework.

The key challenge is that graph structure is discrete, so that most existing gradient-based perturbation could not be directly utilized. We seek to relax the discrete property of graph structure and use a continuous approximation to replace it (such as gumbel softmax), and then adding perturbation to both the structure and node features simultaneously.

The plan is that we'll firstly take one week to implement the FLAG baseline and read related papers, then we'll spend 3-4 weeks to implement our approach based on their framework. After that we plan to test its generalization performance to sota GNN models on OGBN dataset for both the node and graph classification tasks. Lastly, we'll use some GNN attack methods in `https://github.com/DSE-MSU/DeepRobust` to see whether the proposed method can also enhance robustness.

Preprint. Under review.

## 2 Related Work

### 2.1 Adversarial Learning of Graph Neural Networks

So far, most of the papers about adversarial training of GNN are for security purposes. And few of them consider perturbing the graph structure and node features together. [1] proposes a reinforcement learning based attack method that learns the generalizable attack policy to maliciously modify the graph structure. [10] uses a meta-gradient based training-time attack method to corrupt GNN's accuracy by treating the graph as a hyperparameter. **GNNGuard** [9] proposes a general algorithm to defend training-time attack towards GNN by learning how to assign higher weights to edges connecting similar nodes, while pruning edges between unrelated nodes. **FLAG** [4] is the first work investigating how to use adversarial training to improve GNN's clean accuracy. However, it only considers perturbation in the input node feature space, but perturbation in graph topological structure could also be important.

### 2.2 Adversarial Learning for Neural Language Models

While pre-trained language models can generalize to a variety of down-stream tasks, these models are still vulnerable to adversarial attacks and could result in a drastic decrease in performance if the researchers test them with altered test set [3]. Some recent papers propose different directions to solve the issues. [2] designs a regularized framework to prevent overfitting given the complexity of the pre-trained model and insufficient tuning samples by smoothness regularization. While [5] apply an algorithm ALUM to modify embedding space and maximize the adversarial loss in training time. Both directions observe substantial improvement on previous large scale language models, signaling the necessity of adversarial training on existing models.

## 3 Method

In this project, we utilize the high-level idea of adversarial training, and aim to design an effective GNN training algorithm that can improve both the robustness and generalization ability compared with the standard training. In particular, different from [4] that considers perturbing the input node features, we will apply the adversarial training in the space of graph structure, i.e., we consider to adversarially perturb the graph structure during the training process, and train the GNN model based on the perturbed graph in each iteration.

However, crafting the adversarial perturbation in the graph space is different from that in the input feature space. Existing works on the graph structure based attack are typically focusing on the discrete space: the adversarial perturbation is crafted by adding/removing edges to/from the clean graph. Though this type of adversarial perturbation is effective from the attacker's side, directly generalizing it to the adversarial training may not lead to performance gain in terms of the clean accuracy. This is because the discrete perturbation may greatly change the inherent characteristic of the graph (e.g., in the NLP task if we add a link between a sentence to a word "no", the meaning of this sentence would be entirely altered). Therefore, pushing the GNN model to fit those discretely perturbed graph may not be able to improve the clean accuracy (though this approach can potentially give higher robustness). In order to overcome this problem, we relax the discrete constraint of the graph structure with continuous approximation and generate adversarial perturbations in the continuous space. In particular, let $G$ be a given graph with edge weight being 0 or 1 ($e_{ij} = 1$ means that there's an edge connecting nodes $v_i$ and $v_j$), we will view the $G$ as a fully connected graph and perturb the weights of all edges. Besides, we will control the perturbation level such that the characteristic of the clean graph will not change a lot. Mathematically, let $G = \{e_{ij}\}_{i,j\in[V]}$ be the clean graph and $\mathrm{d}G = \{\mathrm{d}e_{ij}\}_{i,j\in[V]}$ be the adversarial perturbations (which will be generated separately at different training epochs) with $\max_{i,j} |\mathrm{d}e_{ij}| \leq \epsilon$. Then the GNN model will be trained using the perturbed graph $G + \mathrm{d}G$.

In the next we will introduce the proposed graph-structure based adversarial training algorithm. In particular, let $F(\theta; G, x)$ be the output of the GNN model given model parameter $\theta$, input graph $G$ and feature $x$, we follow the similar idea of TRADES [8] that trains the robust models by solving the

**Algorithm 1** Graph structure based adversarial training

**Input:** Input features $\{x_i\}_{1,\ldots,n}$, input graphs $\{G_i\}_{i=1,\ldots,n}$, input labels $\{y_i\}_{i=1\ldots,n}$, learning rate $\eta$
**for** $t = 0, 1, \ldots, T-1$ **do**
  sample a mini-batch of data $B_t$
  **for** $i \in B_t$ **do**
    Initialize $G_i'$ by adding small random noise to $G_i$
    **for** $k = 1, \ldots, K$ **do**
      $\mathrm{d}G = \mathrm{sign}\big(\nabla_{G_i'} \ell\big(F(\theta; G_i, x_i), F(\theta; G_i', x_i)\big)\big)$
      $G_i' = \mathrm{Proj}(G_i' + \eta \cdot \mathrm{d}G)$
    **end for**
  **end for**
  $L(\theta) = B^{-1} \sum_{i \in B_t} \big\{\ell(F(\theta; G_i, x_i), y_i) + \lambda \ell\big(F(\theta; G_i, x_i), F(\theta; G_i', x_i)\big)\big\}$
  Calculate the stochastic gradient $g_t = \nabla L(\theta_t)$
  $\theta_{t+1} = \mathrm{Optimizer}\big(\{\theta_\tau\}_{\tau=0,\ldots,t}, \{g_\tau\}_{\tau=0,\ldots,t}\big)$
**end for**
**Output:** $\theta_T$.

following optimization problem:

$$\theta^* = \min_\theta \frac{1}{n} \sum_{i=1}^n \bigg\{ \ell(F(\theta; G_i, x_i), y_i) + \lambda \max_{\mathrm{d}G: \|\mathrm{d}G\|_\infty \le \epsilon} \ell\big(F(\theta; G_i, x_i), F(\theta; G_i + \mathrm{d}G, x_i)\big) \bigg\},$$

where $\ell(a, b)$ be the loss function that characterizes the difference between $a$ and $b$ (e.g., $\ell_2$ distance or KL divergence), and $\lambda$ is a tunable regularization parameter. Notably, for node classification class we only have one fixed graph, in this case we can simply set $G_1 = \cdots = G_n = G$. In particular, the first term in the training objective is the standard training loss (on clean data) and the second term characterizes the distance between the outputs of GNN using clean or perturbed graph, which is typically referred to as the consistency loss. In practice, we will apply projected sign gradient ascent to solve the inner maximization problem and use standard optimizer (e.g., SGD or Adam) to solve the outer minimization problem. We summarize the entire graph structure based adversarial training procedures in Algorithm 1.

### 3.1 Theoretical understanding of the training objective

In this part we focus on the node classification task. We first make the following assumption on the observed graph.

**Assumption 1** *There exists a ground truth graph $G^*$, with edge weights being continuous, that precisely characterizes the connections between nodes. The observed graph $G$ is a noisy version of $G^*$ that satisfies $\|G - G^*\|_\infty \le \epsilon$.*

Assumption 1 is kind of strong but it gives some intuitions regarding the connection between ground-truth and observed graphs. This assumption can be further relaxed to that using a high-probability argument: $|e_{ij}^* - e_{ij}| \le \epsilon$ with probability at least $1 - \delta$, which is closer to the practice but requires a more specific design of the adversarial training algorithm.

Now we are going to illustrate the theoretical understanding the training objective. In particular, note that $G^*$ is the ground-truth graph, it is natural to find a model that minimizes $n^{-1} \sum_{i=1}^n \ell\big(F(\theta; G^*, x_i), y_i\big)$, where we use the same $G^*$ for all input features since the graph is fixed in node classification tasks. Then note that the loss function $\ell(a, b)$ can be viewed as the distance between $a$ and $b$. Then under Assumption 1, by triangle inequality, we can get that

$$\begin{aligned}
\ell\big(F(\theta; G^*, x_i), y_i\big) &\lesssim \ell\big(F(\theta; G, x_i), y_i\big) + \ell\big(F(\theta; G, x_i), F(\theta; G^*, x_i)\big) \\
&\lesssim \ell\big(F(\theta; G, x_i), y_i\big) + \max_{\|\mathrm{d}G\|_\infty \le \epsilon} \ell\big(F(\theta; G, x_i), F(\theta; G + \mathrm{d}G, x_i)\big),
\end{aligned}$$

which is consistent with the loss function we used in Algorithm 1 (up to some constant factors). Therefore, the proposed graph-structure based adversarial training algorithm implicitly minimizes an upper bound of the loss on ground-truth graph. If this upper bound is sharp, we can anticipate that Algorithm 1 can effectively improve the clean accuracy compared to standard training.

## 4 Experiments

We evaluate Algorithm 1 for node classification problem. In particular, we compare the clean accuracy and robustness of our algorithm to those achieved by JaccardGCN, SCNSVD, and standard GCN on Cora and Pubmed dataset, which are displayed in Tables 1 and Table 2. Results show that our algorithm can not only achieve the best clean accuracy, but can also give higher robustness under various graph attacks. This demonstrate the superior performance of our adversarial training algorithm.

Table 1: Node classification accuracy on Cora Dataset. Bold numbers mark the best performance

| Attack Method | Perturbation Ratio | JaccardGCN | GCNSVD | GCN | Ours |
|---|---|---|---|---|---|
| Clean Accuracy | / | 0.820 | 0.747 | 0.795 | **0.847** |
| Random | 0.01 | 0.814 | 0.727 | 0.792 | **0.845** |
| Random | 0.02 | 0.806 | 0.708 | 0.791 | **0.847** |
| Random | 0.04 | 0.792 | 0.642 | 0.792 | **0.846** |
| DICE | 0.01 | 0.807 | 0.726 | 0.796 | **0.846** |
| DICE | 0.02 | 0.803 | 0.695 | 0.791 | **0.844** |
| DICE | 0.04 | 0.777 | 0.644 | 0.782 | **0.842** |
| Meta | 0.05 | 0.815 | 0.740 | 0.790 | **0.847** |
| Meta | 0.10 | 0.811 | 0.717 | 0.793 | **0.842** |
| Meta | 0.15 | 0.810 | 0.720 | 0.791 | **0.840** |
| Meta | 0.20 | 0.808 | 0.707 | 0.780 | **0.838** |
| Meta | 0.25 | 0.810 | 0.708 | 0.776 | **0.831** |
| Nettack | 1 | 0.814 | 0.738 | 0.796 | **0.849** |
| Nettack | 2 | 0.814 | 0.727 | 0.792 | **0.849** |
| Nettack | 3 | 0.812 | 0.699 | 0.793 | **0.849** |
| Nettack | 4 | 0.808 | 0.691 | 0.794 | **0.844** |
| Nettack | 5 | 0.807 | 0.687 | 0.791 | **0.844** |

## 5 Conclusion and Future Work

In this project we proposed a graph-structure based adversarial training method for more accurate and robust GNN models. In particular, the training objective in our algorithm is formulated as a linear combination of the standard loss on clean data and a consistency loss that characterizes the output of GNN models when using the clean graph and adversarially perturbed graph. Experimental results demonstrate that the proposed algorithm cannot only improve the clean accuracy of the standard GNN training algorithm, but also achieve higher robustness than other GNN adversarial training algorithms.

There are a number of future research directions. Currently we apply the same perturbation limit for all edges (i.e., we use a fixed level of $\ell_\infty$ perturbation on edges), which could be potentially improved by exploiting more structure information of the graph. Second, it is also interesting and necessary to see whether applying the adversarial training for both graph and the input feature can further improve the accuracy and robustness.

## References


[1] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial attack on graph structured data. In *International conference on machine learning*, pages 1115–1124. PMLR, 2018.

[2] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[3] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, 2020.

Table 2: Node classification accuracy on Pubmed Dataset. Bold numbers mark the best performance

| Attack Method | Perturbation Ratio | JaccardGCN | GCNSVD | GCN | Ours |
|---|---|---|---|---|---|
| Clean Accuracy | / | 0.770 | 0.784 | 0.841 | **0.848** |
| Random | 0.01 | 0.767 | 0.780 | 0.840 | **0.847** |
| Random | 0.02 | 0.763 | 0.778 | 0.838 | **0.846** |
| Random | 0.04 | 0.758 | 0.771 | 0.834 | **0.846** |
| DICE | 0.01 | 0.767 | 0.781 | 0.838 | **0.847** |
| DICE | 0.02 | 0.764 | 0.776 | 0.836 | **0.844** |
| DICE | 0.04 | 0.756 | 0.771 | 0.831 | **0.841** |
| Meta | 0.05 | 0.702 | 0.710 | 0.815 | **0.838** |
| Meta | 0.10 | 0.663 | 0.668 | 0.801 | **0.831** |
| Meta | 0.15 | 0.629 | 0.635 | 0.780 | **0.822** |
| Meta | 0.20 | 0.606 | 0.612 | 0.771 | **0.821** |
| Meta | 0.25 | 0.585 | 0.588 | 0.756 | **0.817** |
| Nettack | 1 | 0.769 | 0.783 | 0.840 | **0.847** |
| Nettack | 2 | 0.768 | 0.782 | 0.839 | **0.848** |
| Nettack | 3 | 0.766 | 0.779 | 0.839 | **0.847** |
| Nettack | 4 | 0.764 | 0.777 | 0.839 | **0.847** |
| Nettack | 5 | 0.763 | 0.777 | 0.838 | **0.847** |


[4] Kezhi Kong, Guohao Li, Mucong Ding, Zuxuan Wu, Chen Zhu, Bernard Ghanem, Gavin Taylor, and Tom Goldstein. Flag: Adversarial data augmentation for graph neural networks. *arXiv preprint arXiv:2010.09891*, 2020.

[5] Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models, 2020.

[6] Xiaoyun Wang, Xuanqing Liu, and Cho-Jui Hsieh. Graphdefense: Towards robust graph convolutional networks. *CoRR*, abs/1911.04429, 2019.

[7] Huijun Wu, Chen Wang, Yuriy Tyshetskiy, Andrew Docherty, Kai Lu, and Liming Zhu. Adversarial examples for graph data: Deep insights into attack and defense. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 4816–4823. ijcai.org, 2019.

[8] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

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

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


# A Task Distribution Form

Table 3: Task Distribution Form

| Task | People |
|---|---|
| Implementing Algorithm 1 | Ziniu Hu |
| Theoretical Analysis | Difan Zou |
| Evaluating and Comparing Algorithms | Zongyue Qin, Shanxiu He |
| Writing Report | Difan Zou, Ziniu Hu, Zongyue Qin, Shanxiu He |