Adversarial Weight Perturbation Improves Generalization in Graph Neural Networks

Yihan Wu[1], **Aleksandar Bojchevski**[2], **Heng Huang**[1]

[1] Electrical and Computer Engineering, University of Pittsburgh, PA, USA
[2] CISPA Helmholtz Center for Information Security
yiw154@pitt.edu, bojchevski@cispa.de, henghuanghh@gmail.com

Abstract

A lot of theoretical and empirical evidence shows that the flatter local minima tend to improve generalization. Adversarial Weight Perturbation (AWP) is an emerging technique to efficiently and effectively find such minima. In AWP we minimize the loss w.r.t. a bounded worst-case perturbation of the model parameters thereby favoring local minima with a small loss in a neighborhood around them. The benefits of AWP, and more generally the connections between flatness and generalization, have been extensively studied for *i.i.d.* data such as images. In this paper, we extensively study this phenomenon for graph data. Along the way, we first derive a generalization bound for *non-i.i.d.* node classification tasks. Then we identify a vanishing-gradient issue with all existing formulations of AWP and we propose a new Weighted Truncated AWP (WT-AWP) to alleviate this issue. We show that regularizing graph neural networks with WT-AWP consistently improves both natural and robust generalization across many different graph learning tasks and models.

1 Introduction

Simply minimizing the standard cross-entropy loss for highly non-convex and non-linear models such as (deep) neural networks is not guaranteed to obtain solutions that generalize well, especially for today's overparamatrized networks. The key underlying issue is that these models have many different local minima which can have wildly different generalization properties despite having nearly the same performance on training and validation data. Naturally, there is a rich literature that studies the properties of well-behaving local minima, as well as the design choices that improve our chances of finding them (Stutz, Hein, and Schiele 2021). The notion of flatness which measure how quickly the loss changes in a neighbourhood around a given local minimum has been empirically shown to correlate with generalization among a variety of different measures (Jiang et al. 2019). In addition, generalization bounds based on the PAC-Bayes framework (McAllester 1999; Foret et al. 2021) provide theoretical insights that corroborate the mounting empirical data. Since the evidence implies that flatter minima tend to generalize better, the obvious question is how to efficiently find them.

Not only do flat minima improve generalization to unseen test data, i.e. the clean accuracy (Foret et al. 2021; Zheng, Zhang, and Mao 2021; Xu and Huang 2022; Kwon et al. 2021; Xu et al. 2022), but they also improve generalization to adversarial examples, i.e. the robust accuracy (Wu, Xia, and Wang 2020; Stutz, Hein, and Schiele 2021; Wu et al. 2022). Improving adversarial robustness is important, especially for models deployed in safety-critical domains, since most standard (undefended) models are vulnerable to adversarial attacks. Attackers can easily craft deliberate and unnoticeable input perturbations that change the prediction of the classifier (Sun et al. 2018).

Flat minima show higher resistance to adversarially perturbed inputs while maintaining good clean accuracy (Stutz, Hein, and Schiele 2021). Among the variety of techniques for finding flat minima Adversarial Weight Perturbation (AWP) (Wu, Xia, and Wang 2020), and the closely-related (adaptive) sharpness-aware minimization (Foret et al. 2021; Kwon et al. 2021) and adversarial model perturbation (Zheng, Zhang, and Mao 2021), seems to be quite effective in practice. The key idea is to minimize the loss w.r.t. a bounded worst-case perturbation of the model parameters, i.e. minimize a local notion of sharpness. The benefits of this approach, and more generally the correlation between flatness and (clean/robust) generalization, have been extensively studied for i.i.d. data such as images. In this paper we study this phenomenon for graph data. Concretely, we analyze and improve the generalization of Graph Neural Networks (GNNs) which are a fundamental building block (in addition to CNNs and RNNs).

Blindly applying existing weight perturbation techniques to GNNs is unfortunately not effective in practice due to a vanishing-gradient issue. Intuitively, the adversarially perturbed weights tend to have a higher norm which in turn leads to a saturation in the last layer where that logits for one class are on a significantly larger scale compared to the rest. Even though this limitation plagues all formulations of AWP, for both GNNs and other models (e.g. ResNets), it has gone unnoticed so far. To address it we propose Weighted Truncated Adversarial Weight Perturbation (WT-AWP) where rather than directly minimizing the (robust) AWP loss we use it as a regularizer in addition to the standard cross-entropy loss. Moreover, we propose to abstain from perturbation in the last layer(s) of the network for a more fine-grained control of the training dynamics. These two modifications are simple, but

necessary and effective. With our resulting formulation the models can obtain useful gradient signals for training even when the perturbed weights have a high norm, mitigating the gradient-vanishing issue. Furthermore, we theoretically study the AWP learning objective and show its invariance for local extrema. We can summarize our contributions as follows:

- We provide a theoretical analysis of AWP on non-i.i.d. tasks and identify a vanishing-gradient issue that plagues all previous AWP variants. Based on this analysis we propose Weighted Truncated Adversarial Weight Perturbation (WT-AWP) that mitigates this issue.
- We study the connections between flatness and generalization for Graph Neural Networks. We show that GNNs trained with our WT-AWP formulation have simultaneously improved natural and robust generalization. The improvement is statistically significant and consistent across tasks (node-level and graph-level classification) and across models (standard and robustness-aware GNNs), at a negligible computational cost.

2 Background and Related Work

Adversarial Weight Perturbation for Images. AWP is motivated by the connection between the flatness of the loss landscape and model generalization. Given a learning objective $L(\cdot)$ and an image classification model with parameters $\boldsymbol{\theta}$, the generalization gap (Wu, Xia, and Wang 2020), also named the sharpness term (Foret et al. 2021), which measures the worst-case flatness of the loss landscape, is defined by $[\max_{||\boldsymbol{\delta}||\leq\rho} L(\boldsymbol{\theta}+\boldsymbol{\delta}) - L(\boldsymbol{\theta})]$. This gap is known to control a PAC-Bayes generalization bound (Neyshabur et al. 2017), with a smaller gap implying better generalization. The AWP objective simultaneously minimizes the loss function and the generalization gap via $\min_{\boldsymbol{\theta}}[L(\boldsymbol{\theta}) + (\max_{||\boldsymbol{\delta}||\leq\rho} L(\boldsymbol{\theta}+\boldsymbol{\delta}) - L(\boldsymbol{\theta}))] = \min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\delta}||\leq\rho} L(\boldsymbol{\theta}+\boldsymbol{\delta})$. Providing further theoretical justification for the effectiveness of the AWP, (Zheng, Zhang, and Mao 2021) prove that this objective favors solutions corresponding to flatter local minima assuming that the loss surface can be approximated as an inverted Gaussian surface. Relatedly, they show that AWP penalizes the gradient-norm.

In some cases we can rescale the weights to achieve arbitrarily sharp minima that also generalize well (Dinh et al. 2017). We can mitigate this issue using a scale-invariant definition of sharpness (Kwon et al. 2021). Since in our experiments such adaptive sharpness was not beneficial we present the non-adaptive case for simplicity but all results can be trivially extended. Keskar et al. (2016) show that large-batch training may reach sharp minima, however, this does not affect GNNs since they tend to use a small batch size.

GNNs, Graph attacks, and Graph defenses. Graph Neural Networks (GNNs) are emerging as a fundamental building block. They have achieved spectacular results on a variety of graph learning tasks across many high-impact domains (see survey (Wu et al. 2020)). Despite their success, it has been demonstrated that GNNs suffer from evasion attacks at test time (Zügner, Akbarnejad, and Günnemann 2018) and poisoning attacks at training time (Zügner and Günnemann 2019). Meanwhile, a series of methods have been developed to improve their robustness. For example, GCNJaccard (Wu et al. 2019) drops dissimilar edges in the graph, as it found that attackers tend to add edges between nodes with different features. GCNSVD (Entezari et al. 2020) replaces the adjacency matrix with its low-rank approximation motivated by the observation that mostly the high frequency spectrum of the graph is affected by the adversarial perturbations. We also have provable defenses that provide robustness certificates (Bojchevski, Klicpera, and Günnemann 2020). Both heuristic defenses (e.g. GCNJaccard and GCNSVD) and certificates are improved with our WT-AWP. For an overview of attacks and defenses see Sun et al. (2018).

3 Adversarial Weight Perturbation on GNNs

To simplify the exposition we focus on the semi-supervised node classification task. Nonetheless, in Sec. **??** we show that AWP also improves graph-level classification. Let $G = (\boldsymbol{A}, \boldsymbol{X})$ be a given (attributed) graph where $\boldsymbol{A}$ is the adjacency matrix and $\boldsymbol{X}$ contains the node attributes. Let $\mathcal{V}$ be the set of all nodes. In semi-supervised node classification problem we have access to the entire graph, the features and neighbors for all nodes $\mathcal{V}$, but we only have labels for a (small) subset of $\mathcal{V}$ (usually 10%). Normally we optimize $\min_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$, where $L_{\text{train}} = \sum_{v\in\mathcal{V}_{\text{train}}} l(f_{\boldsymbol{\theta}}(\boldsymbol{A}, \boldsymbol{X}), y_v)$, f is a GNN parametrized by weights $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_k)$, y_v is the ground-truth label for node v, and l is some loss function (e.g. cross-entropy) applied to each node in the training set $\mathcal{V}_{\text{train}} \subset \mathcal{V}$.

In AWP we first find the worst-case weight perturbation $\boldsymbol{\delta}^*(\boldsymbol{\theta})$ that maximizes the loss. Then we minimize the loss with the perturbed weights. The worst-case perturbation for a given $\boldsymbol{\theta}$ is defined as

$$\boldsymbol{\delta}^*(\boldsymbol{\theta}) := \arg\max_{||\boldsymbol{\delta}||_2\leq\rho} L_{\text{train}}(\boldsymbol{\theta}+\boldsymbol{\delta}; \boldsymbol{A}, \boldsymbol{X}) \tag{1}$$

where ρ is the strength of perturbation. The AWP learning objective is then $\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\delta}||\leq\rho} L_{\text{train}}(\boldsymbol{\theta}+\boldsymbol{\delta}; \boldsymbol{A}, \boldsymbol{X})$, or

$$\min_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}+\boldsymbol{\delta}^*(\boldsymbol{\theta}); \boldsymbol{A}, \boldsymbol{X}). \tag{2}$$

Since the PAC-Bayes bound proposed by McAllester (1999) only holds for i.i.d. data and semi-supervised node classification is a non-i.i.d. task, the analyses in Wu, Xia, and Wang (2020) and Foret et al. (2021) cannot be directly extended to node classification. Thus, we derive a new generalization bound for node classification (with GNNs) based on a recent sub-group generalization bound (Ma, Deng, and Mei 2021).

Theorem 1 (Generalization bound of AWP loss)**.** *Let $L_{\text{all}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ be the loss on all nodes, for any set of training nodes $\mathcal{V}_{\text{train}}$ from $\mathcal{V}$, $\forall m \geq \sqrt{d}$, with probability at least $1-\delta$, we have*

$$\begin{aligned} L_{\text{all}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X}) \leq & \max_{||\boldsymbol{\delta}||_2\leq\rho} [L_{\text{train}}(\boldsymbol{\theta}+\boldsymbol{\delta}; \boldsymbol{A}, \boldsymbol{X})] \\ & + (\frac{m^2}{d} e^{1-\frac{m^2}{d}})^{d/2} + \frac{1}{2\sqrt{N_0}} \left[1 + d\log(1 + \frac{m^2||\boldsymbol{\theta}||_2^2}{d\rho^2})\right] \\ & + \frac{1}{\sqrt{N_0}} \left(\ln\frac{3}{\delta} + \frac{1}{4} + \Theta(K\cdot\epsilon_{\text{all}})\right). \end{aligned} \tag{3}$$

where d is the number of parameters in the GNN, K is the number of groundtruth labels, ϵ_{all} is a fixed constant w.r.t. $\mathcal{V}$, N_0 is the volume of $\mathcal{V}_{\text{train}}$, and ρ is the perturbation strength on the weights.

The details of the proof are in Sec. A. We can rewrite Eq. 3 into the following simplified version

$$L_{\text{all}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X}) \leq \max_{||\boldsymbol{\delta}||_2 \leq \rho} L_{\text{train}}(\boldsymbol{\theta} + \boldsymbol{\delta}; \boldsymbol{A}, \boldsymbol{X}) + h(||\boldsymbol{\theta}||_2^2/\rho^2) \tag{4}$$

where $h(\cdot)$ is a monotonously increasing function depending on the perturbation strength $\rho(\boldsymbol{\theta})$.

This bound justifies the use of AWP since the perturbed loss on training nodes bounds the standard loss on *all* nodes. Moreover, as $h(||\boldsymbol{\theta}||_2^2/\rho^2)$ is monotonically decreasing with ρ, increasing the perturbation strength ρ can make the bound in Eq. 4 sharper, i.e. the resulting AWP objective should lead to better generalization. In practice we perturb the weights $\boldsymbol{\theta_i}$ of each layer separately, and this bound still holds if we set $\rho = \sum_{i=1}^{k} \rho(\boldsymbol{\theta_i})$ where $\rho(\boldsymbol{\theta_i})$ is the perturbation strength for layer i. We derived a similar result for graph-level tasks in Sec. C.

Since finding the optimal perturbation (Eq. 1) is intractable, we approximate it with a one-step projected gradient descent as in previous work (Wu, Xia, and Wang 2020; Foret et al. 2021; Zheng, Zhang, and Mao 2021),

$$\hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta}) := \Pi_{B(\rho(\boldsymbol{\theta}))}(\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})), \tag{5}$$

where $B(\rho(\boldsymbol{\theta}))$ is an l_2 ball with radius $\rho(\boldsymbol{\theta})$ and $\Pi_{B(\rho(\boldsymbol{\theta}))}(\cdot)$ is a projection operation, which projects the perturbation back to the surface of $B(\rho(\boldsymbol{\theta}))$ when the perturbation is out of the ball. The maximum perturbation norm $\rho(\boldsymbol{\theta})$ could either be a constant (Foret et al. 2021; Zheng, Zhang, and Mao 2021) or layer dependent (Wu, Xia, and Wang 2020). We specify a layer-dependent norm constraint $\rho(\boldsymbol{\theta_i}) := \rho||\boldsymbol{\theta_i}||_2$ because the scales of different layers in a neural network can vary greatly. With the approximation $\hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta})$, the definition of the final AWP learning objective is given by

$$L_{\text{awp}}(\boldsymbol{\theta}) := L_{\text{train}}(\boldsymbol{\theta} + \Pi_{B(\rho(\boldsymbol{\theta}))}(\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X}); \boldsymbol{A}, \boldsymbol{X}), \tag{6}$$

If $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ is smooth enough, $\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X}) = 0$ when $\boldsymbol{\theta}^*$ is a local minimum. In this case $L_{\text{awp}}(\boldsymbol{\theta}) = L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$. A natural question is whether $\boldsymbol{\theta}^*$ will also be the minimum of $L_{\text{awp}}(\boldsymbol{\theta})$? We show that $L_{\text{awp}}(\boldsymbol{\theta})$ keeps the local minimum of $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ unchanged.

Theorem 2. *(Invariant of local minimum) With the AWP learning objective in Eq. 6, and for continuous $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$, $\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$, $\Delta_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$, if $\boldsymbol{\theta}^*$ is a local minimum of $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ and the Hessian matrix $\Delta_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta}*}$ is positive definite, $\boldsymbol{\theta}^*$ is also a local minimum of $L_{\text{awp}}(\boldsymbol{\theta})$.*

The proof is provided in Appendix B. The exact gradient of this new objective is

$$\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta} + \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta}); \boldsymbol{A}, \boldsymbol{X}) = \nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta} + \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta})} + \nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta}) \nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta} + \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta})} \tag{7}$$

Since $\nabla_{\boldsymbol{\theta}} \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta})$ includes second and higher order derivative of $\boldsymbol{\theta}$, which are computationally expensive, they are omitted during training, obtaining the following approximate gradient of the AWP loss

$$\nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta} + \hat{\boldsymbol{\delta}}^*(\boldsymbol{\theta})} \tag{8}$$

Foret et al. (2021) show the models trained with the exact gradient (Eq. 7) have almost the same performance as model trained with the approximate first-order gradient (Eq. 8). Besides, we can also show that the norm of the difference between Eq. 7 and Eq. 8 is proportional to ρ and the second order derivatives of loss the L w.r.t. the weights θ.

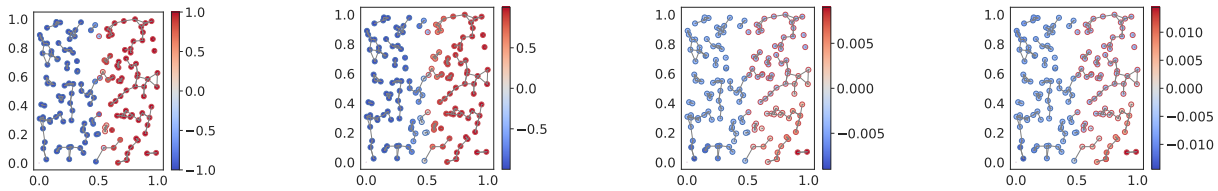
4 Weighted Truncated AWP

In this section we discuss the theoretical limitations of existing AWP methods on GCN, and illustrate them empirically on a toy dataset. We also propose two approaches to improve AWP. Our improved AWP works well on both toy data and on real-world GNN benchmarks across many tasks and models. We also show that similar problems also exist for multi-layer perceptrons (see Appendix D).

4.1 The Vanishing-gradient Issue of AWP

Consider a GCN $\hat{y} = \sigma_s(\hat{\boldsymbol{A}}(...\sigma(\hat{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_1)...)\boldsymbol{W}_n)$ with a softmax activation σ_s at the output layer and non-linearity σ, where $\hat{\boldsymbol{A}}$ is the graph Laplacian given by $\hat{\boldsymbol{A}} := \boldsymbol{D}^{-1/2}(\boldsymbol{A} + \boldsymbol{I}_N)\boldsymbol{D}^{-1/2}, D_{ii} = \sum_j (\boldsymbol{A} + \boldsymbol{I}_N)_{ij}$. The perturbed model is $\hat{y} = \sigma_s(\hat{\boldsymbol{A}}(\dots\sigma(\hat{\boldsymbol{A}}\boldsymbol{X}(\boldsymbol{W}_1 + \boldsymbol{\delta}_1))\dots)(\boldsymbol{W}_n + \boldsymbol{\delta}_n))$. Since the norm of each perturbation $\boldsymbol{\delta}_i$ could be as large as $\rho||\boldsymbol{W}_i||_2$, in the worst case the norm of each layer is $(\rho + 1)||\boldsymbol{W}_i||_2$, and thus the model will have exploding logit values when ρ is large. If additionaly the logit for one class is significantly larger than the others, the output will approximate a one-hot encoded vector after the softmax. In this case the gradient will be close to 0 and the weights will not be updated. Although in practice the number of GCN layers is often less than 3, we still observe the vanish gradient issue in both toy datasets and GNN benchmarks.

To verify our conclusion, we train a 2-layer GCN network with hidden dimension 64, which is a common setting for GCNs, on a linearly separable dataset. The dataset contains 2 classes $\{-1, 1\}$ and each class has 100 nodes. We apply k-nearest neighbor ($k = 3$) to obtain the adjacency matrix, and use the 2D position of the nodes as the features. The number of training epochs is 200. We use 10% nodes for training, 10% for validating and the rest 80% for testing. In Figure 1 we show the trained classifiers for different ρ values. Models with AWP crash quickly as ρ increases from 0.5 to 2.5. When $\rho = 0.5$, the classification accuracy is 0.97, which is nearly the same as the vanilla model, but when $\rho = 2.5$, the classification accuracy is 0.51, which is the same as a random guess. Besides, when $\rho = 1.5$ and 2.5, the loss of AWP method is almost constant during training (Figure 2) and the prediction score (Figure 1(c) and Figure 1(d)) is around 0. This indicates that the weights are barely updated during training. So with the AWP objective, we cannot select a large ρ. Yet, as we discussed in Sec. 3, we prefer larger values of ρ since they lead to a tighter bound (Eq. 4) and



(a) Vanilla GCN (b) AWP $\rho = 0.5$ (c) AWP $\rho = 1.5$ (d) AWP $\rho = 2.5$

Figure 1: Compare AWP models on a linearly separable dataset with different perturbation strengths ρ. The accuracy of models (a) to (d) is 0.97, 0.97, 0.69, and 0.48 respectively. The face color of each node shows its prediction score and the border color shows its ground-truth label. Grey lines connect the node with its nearest neighbours in the graph. For large values of ρ the model is unable to learn.

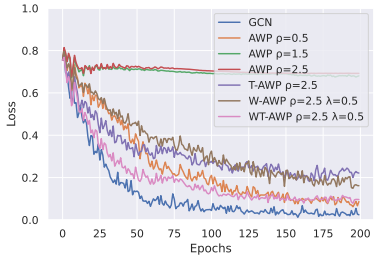


Figure 2: Learning curves for GCN with different losses (and ρ).

are more like to generalize better. As we shown next, our suggested improvements fix this issue.

4.2 Truncated AWP and Weighted AWP

Intuition for WT-AWP. The vanishing gradient is mainly due to the exploding of the logit values, which is caused by perturbing all layers in the model. Thus, a natural idea is to only apply AWP on certain layers to mitigate the issue. This it the truncated AWP. Another idea is to provide a second source of valid gradients which we do by adding the the vanilla loss $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ to the AWP loss. Even when the AWP loss suffers from the vanishing gradient issue, the vanilla loss is not affected.

Definition 1. *(Truncated AWP) We split the model parameters into two parts $\boldsymbol{\theta} = [\boldsymbol{\theta}^{(\text{awp})}, \boldsymbol{\theta}^{(\text{normal})}]$, and we only perform AWP on $\boldsymbol{\theta}^{(\text{awp})}$. The Truncated AWP objective is*

$$\min_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta} + [\hat{\boldsymbol{\delta}}^{(\text{awp})*}(\boldsymbol{\theta}^{(\text{awp})}), 0]; \boldsymbol{A}, \boldsymbol{X}), \tag{9}$$

where $\hat{\boldsymbol{\delta}}^{(\text{awp})*}(\boldsymbol{\theta}^{(\text{awp})})$ is defined as in Eq. 5.

Recall that the AWP objective is the unweighted combination of the regular loss function $L(\boldsymbol{\theta})$ and the sharpness term $\max_{\boldsymbol{\delta} \le \rho}[L(\boldsymbol{\theta}+\boldsymbol{\delta}) - L(\boldsymbol{\theta})]$ (Sec. 2). The weight perturbation in this term can lead to vanishing gradients as we discussed in Sec. 4.1. Therefore, another way to deal with this issue is to assign a smaller weight λ to the sharpness term in the AWP objective. The weighted combination is $[\lambda \max_{\boldsymbol{\delta} \le \rho}[L(\boldsymbol{\theta} + \boldsymbol{\delta}) - L(\boldsymbol{\theta})] + L(\boldsymbol{\theta})] = [\lambda \max_{\boldsymbol{\delta} \le \rho} L(\boldsymbol{\theta} + \boldsymbol{\delta}) + (1-\lambda)L(\boldsymbol{\theta})]$.

Definition 2. *(Weighted AWP) Given a weight $\lambda \in [0, 1]$ the Weighted AWP objective is*

$$\min_{\boldsymbol{\theta}} [\lambda L_{\text{train}}(\boldsymbol{\theta} + \hat{\boldsymbol{\delta}}^{*}(\boldsymbol{\theta}); \boldsymbol{A}, \boldsymbol{X}) + (1-\lambda) L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})] \tag{10}$$

Algorithm 1: WT-AWP: Weighted Truncated Adversarial Weight Perturbation

---

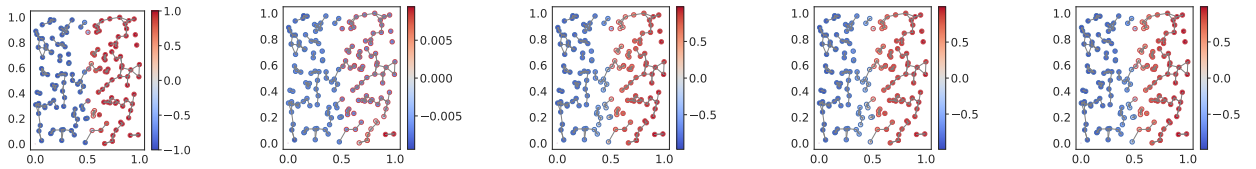
Input: Graph $G = (\boldsymbol{A}, \boldsymbol{X})$; model parameters $\boldsymbol{\theta} = [\boldsymbol{\theta}^{(\text{awp})}; \boldsymbol{\theta}^{(\text{normal})}]$ with and without AWP; number of epochs T; loss function L_{train}; perturbation strength ρ, AWP weight λ; learning rate α.

1: Initialize weight $\boldsymbol{\theta}_0$;.
2: **for** $t \in 1$:T **do**
3: Compute the loss for training nodes: $L_{\text{train}}(\boldsymbol{\theta}_{t-1}; \boldsymbol{A}, \boldsymbol{X})$
4: Compute the approximating weight perturbation for $\boldsymbol{\theta}^{(\text{awp})}_{t-1}$: $\hat{\boldsymbol{\delta}}^{*}(\boldsymbol{\theta}^{(\text{awp})}_{t-1})$ via Eq. 5
5: Compute the approximating gradient for $\boldsymbol{\theta}$:
$\boldsymbol{g} = \lambda \nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta}_{t-1} + [\hat{\boldsymbol{\delta}}^{*}(\boldsymbol{\theta}^{(\text{awp})}_{t-1}), 0]}$
$+ (1-\lambda) \nabla_{\boldsymbol{\theta}} L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})|_{\boldsymbol{\theta}_{t-1}}$
6: Update the weight via $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \alpha \boldsymbol{g}$
7: **end for**
8: **return** $\boldsymbol{\theta}_T$

We compare these two improvements with AWP and natural training on a linearly separable dataset using the same setup as in Sec. 4.1. Figure 3 illustrates the trained models with $\rho = 2.5$. In Figure 3(b) we can see that the model with AWP objective suffers from vanishing gradients and it fails to learn anything useful. The models with Truncated AWP[1] (Figure 3(c)) and Weighted AWP (Figure 3(e)) mitigate this issue, which is also evident in their learning curves (Figure 2), and have relatively good performance (96% and 98% accuracy respectively). Compared to the vanilla model (Figure 3(a)), they both have a significantly smoother decision boundary.

To tackle the vanishing-gradient issue better, we combine Truncated AWP and Weighted AWP, into a Weighted Truncated Adversarial Weight Perturbation (WT-AWP). The details of WT-AWP are shown in Algorithm 1 (description in Sec. D.2). WT-AWP has two important parameters λ and ρ. We study how they influence the model performance in Sec. 5.5.

[1] In Figure 3(c) we perturb only the first-layer, i.e. $\boldsymbol{\theta}^{(\text{awp})} = \boldsymbol{W}_1$ (first layer weights) and $\boldsymbol{\theta}^{(\text{normal})} = \boldsymbol{W}_2$ (last layer weights). Perturbing only the second layer instead performs similarly.



(a) Vanilla model (b) AWP (c) T-AWP (d) W-AWP, λ=0.5 (e) WT-AWP, λ=0.5

Figure 3: Linearly separable dataset. The accuracy of models (a) to (d) is 0.97, 0.51, 0.96, and 0.98 respectively. The face color of each node shows its prediction score and the border color shows the ground-truth label. Grey lines connect the nearest neighbours. Since the perturbation is large ($\rho = 2.5$), AWP fails. The proposed weighting and truncation mitigate the vanishing-gradient issue for the same ρ.

Approachs	Cora	Citeseer	Polblogs
GCN	84.14 ± 0.61	73.44 ± 1.35	95.04 ± 0.66
GCN+WT-AWP	85.16 ± 0.44	74.48 ± 1.04	95.26 ± 0.51
GAT	84.13 ± 0.79	73.71 ± 1.23	94.93 ± 0.51
GAT+WT-AWP	85.13 ± 0.51	74.73 ± 1.07	95.12 ± 0.48
PPNP	85.56 ± 0.46	74.50 ± 1.06	95.18 ± 0.42
PPNP+WT-AWP	**86.13 ± 0.43**	**75.64 ± 0.95**	**95.36 ± 0.37**

Table 1: Clean accuracy comparison. We report the average and the standard deviation across 200 experiments per model (20 random splits $\times$ 10 random initializations). WT-AWP consistently outperform the standard models on all benchmarks. The improvements are statistically significant according to a two-sided t-test at a significance level of $p < 0.001$.

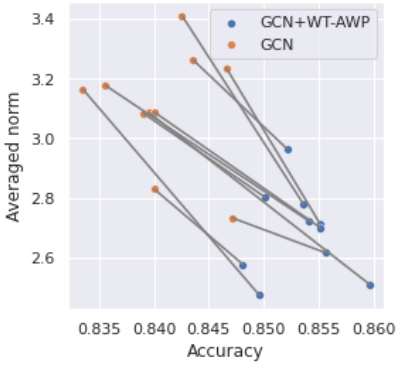


Figure 4: Comparison of the averaged gradient norm w.r.t. the adjacency matrix for GCN models with and without WT-AWP on Cora and Citeseer. Each connected pair of points refers to a GCN and a GCN+WT-AWP model trained with the same data split and initialization.

5 Experimental Results

Setup. We conduct comprehensive experiments to show the effect of WT-AWP on the natural and robustness performance of different GNNs for both node classification and graph classification tasks. We utilize the open-source libraries *Pytorch-Geometric* (Fey and Lenssen 2019) and *Deep-Robust* (Li et al. 2020) to evaluate clean and robust node classification performance respectively. To achieve fair comparison we keep the same training settings for all models. We report the mean and standard deviation over 20 different train/val/test splits and 10 random weight initializations. See Appendix F.4 for further details and hyperparameters.

Datasets. We use three benchmark datasets, including two citation networks, Cora and Citeseer (Sen et al. 2008), and one blog dataset Polblogs (Adamic and Glance 2005). We treat all graphs as undirected and only select the largest connected component (more details and statistics in Appendix F.3).

Baseline models and attacks. We aim to evaluate the impact of our WT-AWP on natural and robust node classification tasks. We train three vanilla GNNs: GCN (Kipf and Welling 2017), GAT (Veličković et al. 2018), and PPNP (Klicpera, Bojchevski, and Günnemann 2018), and four graph defense methods: RGCN (Zhu et al. 2019)[2], GCNJaccard (Wu et al. 2019), GCNSVD (Entezari et al. 2020), and SimpleGCN (Jin et al. 2021). For detailed baseline descriptions see Appendix F.1.

To generate the adversarial perturbations, we apply three methods including: DICE (Waniek et al. 2018), PGD (Xu et al. 2019), and Metattack (Zügner and Günnemann 2019). For a discussion of the attacks see Appendix F.2.

Settings for WT-AWP. All baseline models have a 2-layer structure. When applying the WT-AWP objective, we only perform weight perturbation on the *first* layer i.e. we assign $\boldsymbol{\theta}^{(\text{awp})} = \boldsymbol{W}_1$ (the first layer) and $\boldsymbol{\theta}^{(\text{normal})} = \boldsymbol{W}_2$ (the last layer). For generating the weight perturbation we use a 1-step PGD as discussed in Sec. 3. In the ablation study Sec. 5.5 we also apply 5-step PGD to generate weight perturbation, in which we utilize an SGD optimizer with learning rate 0.2 and update the perturbation for 5 steps. In the end we project the perturbation on the l_2 ball $B(\rho(\boldsymbol{\theta}))$.

5.1 Clean Accuracy

We evaluate the clean accuracy of node classification tasks for different GNNs and benchmarks. The baselines include GCN, GAT, and PPNP . We use a 2-layer structure (input-hidden-output) for these three models. For GCN and PPNP, the hidden dimensionality is 64; for GAT, we use 8 heads with size 8. We choose $K = 10, \alpha = 0.1$ in PPNP. We also find that the hyperparameters (λ, ρ) of WT-AWP are more related to the dataset than the backbone models. We use $(\lambda = 0.7, \rho = 1)$ for all three baseline models on Cora, $(\lambda =$

[2]Note, we cannot apply WT-AWP to RGCN as the weights are modeled as (Gaussian) distributions.

	Natural Acc		Acc with 5% PGDattack		Acc with 5% Mettack	
Models	Cora	Citeseer	Cora	Citeseer	Cora	Citeseer
GCN	83.73 ± 0.71	73.03 ± 1.19	81.26 ± 1.27	72.04 ± 1.60	78.61 ± 1.66	69.20 ± 1.93
+WT-AWP	**84.66 ± 0.53**	74.01 ± 1.11	82.66 ± 1.07	73.73 ± 1.23	79.05 ± 1.73	70.50 ± 1.65
GCNJaccard	82.42 ± 0.73	73.09 ± 1.20	80.65 ± 1.14	72.05 ± 1.76	78.96 ± 1.54	69.62 ± 1.87
+WT-AWP	83.55 ± 0.60	74.10 ± 1.04	82.12 ± 0.91	73.85 ± 1.38	**80.23 ± 1.38**	71.22 ± 1.44
SimPGCN	82.99 ± 0.68	74.05 ± 1.28	80.71 ± 1.33	73.61 ± 1.39	78.60 ± 1.81	72.52 ± 1.72
+WT-AWP	83.37 ± 0.74	**74.26 ± 1.09**	**83.49 ± 0.78**	**74.43 ± 1.14**	79.76 ± 1.76	**72.95 ± 1.43**
GCNSVD	77.63 ± 0.63	68.57 ± 1.54	76.83 ± 1.42	68.08 ± 1.98	76.28 ± 1.15	67.34 ± 1.93
+WT-AWP	79.05 ± 0.58	71.12 ± 1.42	78.50 ± 0.89	71.43 ± 1.46	77.61 ± 1.08	70.65 ± 1.28

Table 2: Robust accuracy under PGD and Metattack poisoning attacks, with a 5% adversarial budget. We report the average and the standard deviation across 200 experiments per model (20 random splits $\times$ 10 random initializations). Our WT-AWP loss improves over all (vanilla and robust) baselines. All results expect the one marked with * are statistically significant at $p < 0.05$ according to a t-test.

Perturbation strength		5%			10%		
Attacks	Models	Cora	Citeseer	Polblogs	Cora	Citeseer	Polblogs
DICE	GCN	82.83 ± 0.87	71.85 ± 1.31	91.27 ± 0.98	81.87 ± 0.94	71.17 ± 1.50	87.47 ± 1.17
	+WT-AWP	**84.01 ± 0.59**	**73.84 ± 1.10**	**91.45 ± 0.86***	**82.93 ± 0.64**	**73.14 ± 1.25**	**87.70 ± 0.97**
PGD	GCN	79.92 ± 0.62	70.50 ± 1.35	79.41 ± 0.76	77.17 ± 0.74	68.49 ± 1.39	72.90 ± 0.73
	+WT-AWP	**81.00 ± 0.56**	**70.69 ± 1.45***	**80.70 ± 0.90**	**77.87 ± 0.64**	**68.96 ± 1.30**	**75.11 ± 1.03**

Table 3: Robust accuracy under evasion attacks of different strength. We report the average and the standard deviation across 200 experiments per model (20 random splits $\times$ 10 random initializations). Our WT-AWP loss always improves the robustness of the baselines. All results expect the one marked with * are statistically significant at $p < 0.05$ according to a two-sided t-test.

$0.7, \rho = 2.5)$ on Citeseer, and $(\lambda = 0.3, \rho = 1)$ for GCN, $(\lambda = 0.3, \rho = 2)$ for GAT and PPNP on Polblogs. Table 1 shows our results, WT-AWP clearly improves the accuracy of all baseline models, while having smaller standard deviations. Note, we do not claim that these models are state of the art, but rather that WT-AWP provides consistent and statistically significant (two-sided t-test, $p < 0.001$) improvements over the baseline models. These results support our claim that WT-AWP finds local minima with better generalization properties.

5.2 Models Trained with WT-AWP are Smoother

To estimate the smoothness of the loss landscape around the adjacency matrix $\boldsymbol{A}$ and the node attributes $\boldsymbol{X}$, we compute the average norm of the gradient of $L_{\text{train}}(\boldsymbol{\theta}; \boldsymbol{A}, \boldsymbol{X})$ *w.r.t.* $\boldsymbol{A}$ and $\boldsymbol{X}$. We compare a vanilla GCN model with GCN+WT-AWP $(\lambda = 0.5, \rho = 1)$ model on Cora. We train 10 models with different random initializations. For each model we randomly sample 100 noisy inputs around $\boldsymbol{A}$ and $\boldsymbol{X}$, and we average the gradient norm for these noisy inputs. When comparing models trained with and without WT-AWP, we keep everything else fixed, including the random initialization, to isolate the effect of WT-AWP. In Figure 4, we can observe that in most cases (37 out of 40) the models trained with WT-AWP have both better accuracy and smaller average gradient norm, *i.e.* are smoother. As we show in Sec. 5.3 and Sec. 5.4 this consequently improves their robustness to adversarial input perturbations.

5.3 Robust Accuracy with Poisoning Attacks

Next we show that our WT-AWP can improve existing defense methods against graph poisoning attacks. We select two poisoning attacks: PGD and Metattack (Zügner and Günnemann 2019), with a 5% adversarial budget. The baseline models are vanilla GCN, and three GCN-based graph-defense models: GCNJaccard, GCNSVD, and SimpleGCN. For all attack and defense methods, we apply the default hyperparameter settings in (Li et al. 2020), which re-implements the corresponding models with the same hyperparameters as the original works. We use Cora, Citeseer, and Polblogs as the benchmark datasets. Note that GCNJaccard does not work on Polblogs as it requires node features. Table 11 in the appendix shows the hyperparameters (λ, ρ) we select for all WT-AWP models.

As we can see in Table 2, none of the defense methods have dominant performance across benchmarks. More importantly, our WT-AWP consistently improves the robust accuracy for both vanilla and robust models. We also evaluate the models against the DICE poisoning attack in Appendix E.2, and again the results demonstrate that WT-AWP adds meaningful improvement over the baselines.

5.4 Robust Accuracy with Evasion Attacks

Next we show that WT-AWP also improves existing defense methods against evasion attacks. We select two evasion attacks, DICE and PGD, with perturbation strengths of 5% and

WT-AWP	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 2.5$	$\rho = 5$
$\lambda = 0.1$	84.15 ± 0.60	84.15 ± 0.61	84.51 ± 0.48	84.58 ± 0.52	84.50 ± 0.51	84.54 ± 0.49
$\lambda = 0.3$	84.10 ± 0.62	84.13 ± 0.58	84.76 ± 0.51	84.91 ± 0.46	84.77 ± 0.46	84.64 ± 0.47
$\lambda = 0.5$	84.11 ± 0.64	84.09 ± 0.61	84.93 ± 0.49	85.06 ± 0.49	84.94 ± 0.45	84.67 ± 0.49
$\lambda = 0.7$	84.13 ± 0.59	84.15 ± 0.64	85.00 ± 0.46	**85.16 ± 0.44**	84.99 ± 0.49	84.66 ± 0.49
$\lambda = 1.0$	84.12 ± 0.69	84.23 ± 0.64	82.45 ± 1.98	60.29 ± 1.94	29.51 ± 0.91	29.19 ± 0.13
AWP	84.16 ± 0.68	84.23 ± 0.68	41.19 ± 1.23	29.18 ± 0.07	29.18 ± 0.02	29.18 ± 0.02
W-AWP	84.12 ± 0.66	84.20 ± 0.66	84.63 ± 0.51	84.32 ± 0.65	83.98 ± 0.93	83.62 ± 1.27

Table 4: Hyperparameter sensitivity study for λ and ρ on the Cora dataset for a GCN base model.

WT-AWP (5 step)	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.5$	$\rho = 1$	$\rho = 2.5$	$\rho = 5$
$\lambda = 0.1$	84.19 ± 0.60	84.17 ± 0.59	84.45 ± 0.51	84.50 ± 0.50	84.39 ± 0.52	84.41 ± 0.54
$\lambda = 0.3$	84.12 ± 0.58	84.15 ± 0.63	84.65 ± 0.54	84.81 ± 0.47	84.70 ± 0.50	84.55 ± 0.55
$\lambda = 0.5$	84.10 ± 0.59	84.11 ± 0.62	84.77 ± 0.53	84.90 ± 0.50	84.82 ± 0.47	84.64 ± 0.52
$\lambda = 0.7$	84.12 ± 0.61	84.11 ± 0.63	84.86 ± 0.49	**84.99 ± 0.48**	84.89 ± 0.51	84.64 ± 0.52
$\lambda = 1.0$	84.11 ± 0.62	84.18 ± 0.63	72.18 ± 1.48	32.55 ± 6.80	29.18 ± 0.03	29.18 ± 0.00

Table 5: Ablation study with λ and ρ on WT-AWP, where we use 5-step PGD weight perturbation. The backbone model is GCN and the benchmark is Cora. We observe no significant improvement compared to the computationally less expensive 1-step PGD.

10%. The baseline model is GCN and we perform experiments on three benchmarks: Cora, Citeseer, and Polblogs. For the PGD attack the hyperparameters (λ, ρ) are (0.5, 0.5) for all datasets. For the DICE attack we use (0.5, 0.5) for Cora, (0.7, 2) for Citeseer, and (0.3, 1) for Polblogs. Table 3 shows the experimental results. WT-AWP again meaningfully improves the robustness of GCN under both PGD and DICE evasion attacks for all perturbation strengths.

5.5 Ablation Study

We compare the performance of GCN+WT-AWP on the Cora dataset for different λ and ρ values. We also compare WT-AWP with AWP under different perturbation strengths ρ. Table 4 lists the results. The accuracy of GCN+WT-AWP first increases with λ and ρ and then slightly decreases. Truncated AWP is a special case for $\lambda = 1$ (since the $(1 - \lambda)$ term disappears in Eq. 10) and it does not perform well, especially for larger ρ. Similarly, WT-AWP outperforms the vanilla AWP that suffers from the vanishing-gradient issue. Weighted but not truncated AWP with $\lambda = 0.5$ (last row) is also worse than WT-AWP, although in general weighting seems to be more important than truncation. These results justify the decision to combine weighting and truncation.

We also generate perturbations as in Eq. 5 but with multi-step PGD. As shown in Table 5, the performance of 5-step WT-AWP is similar to the 1-step WT-AWP, the accuracy of both models first increases with λ and ρ, and then decreases. Since 5-step PGD offers no benefits and 1-step PGD is computationally less expensive, we suggest this as the default setting when applying WT-AWP.

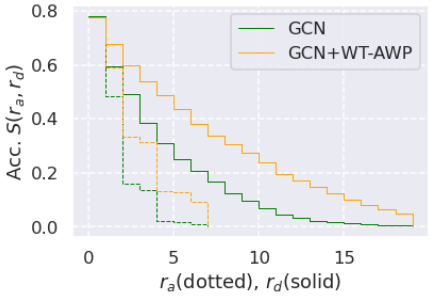


Figure 5: Robustness guarantees on Cora, where r_a is the certified radius – maximum number of adversarial additions (and r_d for deletions). For perturbations to the node features WT-AWP significantly improves the certified accuracy, i.e. the number of nodes guaranteed not to change their prediction, for all certified radii.

5.6 Certified Robustness

In this subsection, we measure the certified robustness of GCN and GCN+WT-AWP on the Cora dataset with sparse randomized smoothing (Bojchevski, Klicpera, and Günnemann 2020). We use $\lambda = 0.5, \rho = 1$ as the hyperparameters for the WT-AWP models. We plot the certified accuracy $S(r_a, r_d)$ for different addition r_a and deletion r_d radii. In Figure 5, we see that compared to vanilla GCN training, our WT-AWP loss increases the certified accuracy *w.r.t.* feature perturbations for all radii. For additional results see Appendix E.3.

6 Conclusion

We proposed a new adversarial weight perturbation method, WT-AWP, and we evaluated it on graph neural networks. We showed that our WT-AWP can improve the regularization of GNNs by finding flat local minima. We conducted extensive experiments to validate our method. In all empirical results, WT-AWP consistently improves the performance of GNNs on a wide range of graph learning tasks including node classification, graph defense, and graph classification. Further exploring the connections between flat minima and generalization in GNNs is a promising research direction.

Acknowledgements

This work was partially supported by NSF IIS 1852606, 1838627, 1837956, 1956002, 2211492, CNS 2213701, CCF 2217003, DBI 2225775.

References

Adamic, L. A.; and Glance, N. 2005. The political blogosphere and the 2004 US election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, 36–43.

Bojchevski, A.; Klicpera, J.; and Günnemann, S. 2020. Efficient robustness certificates for discrete data: Sparsity-aware randomized smoothing for graphs, images and more. In *International Conference on Machine Learning*, 1003–1013. PMLR.

Dinh, L.; Pascanu, R.; Bengio, S.; and Bengio, Y. 2017. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 1019–1028. PMLR.

Entezari, N.; Al-Sayouri, S. A.; Darvishzadeh, A.; and Papalexakis, E. E. 2020. 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*, 169–177.

Fey, M.; and Lenssen, J. E. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware minimization for efficiently improving generalization. *ICLR*.

Jiang, Y.; Neyshabur, B.; Mobahi, H.; Krishnan, D.; and Bengio, S. 2019. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*.

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

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. *ICLR*.

Kwon, J.; Kim, J.; Park, H.; and Choi, I. K. 2021. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. *arXiv preprint arXiv:2102.11600*.

Li, Y.; Jin, W.; Xu, H.; and Tang, J. 2020. Deeprobust: A pytorch library for adversarial attacks and defenses. *arXiv preprint arXiv:2005.06149*.

Ma, J.; Deng, J.; and Mei, Q. 2021. Subgroup Generalization and Fairness of Graph Neural Networks. *arXiv preprint arXiv:2106.15535*.

McAllester, D. A. 1999. PAC-Bayesian model averaging. In *Proceedings of the twelfth annual conference on Computational learning theory*, 164–170.

Neyshabur, B.; Bhojanapalli, S.; McAllester, D.; and Srebro, N. 2017. Exploring generalization in deep learning. *NIPS*.

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.

Stutz, D.; Hein, M.; and Schiele, B. 2021. Relating Adversarially Robust Generalization to Flat Minima. *arXiv preprint arXiv:2104.04448*.

Sun, L.; Dou, Y.; Yang, C.; Wang, J.; Yu, P. S.; He, L.; and Li, B. 2018. Adversarial attack and defense on graph data: A survey. *arXiv preprint arXiv:1812.10528*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *ICLR*.

Waniek, M.; Michalak, T. P.; Wooldridge, M. J.; and Rahwan, T. 2018. Hiding individuals and communities in a social network. *Nature Human Behaviour*, 2(2): 139–147.

Wu, D.; Xia, S.-T.; and Wang, Y. 2020. Adversarial weight perturbation helps robust generalization. *NIPS*.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019. Adversarial examples on graph data: Deep insights into attack and defense. *arXiv preprint arXiv:1903.01610*.

Wu, Y.; Li, X.; Kerschbaum, F.; Huang, H.; and Zhang, H. 2022. Towards robust dataset learning. *arXiv preprint arXiv:2211.10752*.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Xu, A.; and Huang, H. 2022. Detached Error Feedback for Distributed SGD with Random Sparsification. In *International Conference on Machine Learning*, 24550–24575. PMLR.

Xu, A.; Li, W.; Guo, P.; Yang, D.; Roth, H. R.; Hatamizadeh, A.; Zhao, C.; Xu, D.; Huang, H.; and Xu, Z. 2022. Closing the Generalization Gap of Cross-silo Federated Medical Image Segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20866–20875.

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*.

Zheng, Y.; Zhang, R.; and Mao, Y. 2021. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8156–8165.

Zhu, D.; Zhang, Z.; Cui, P.; and Zhu, W. 2019. Robust graph convolutional networks against adversarial attacks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1399–1407.

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.

Zügner, D.; and Günnemann, S. 2019. Adversarial attacks on graph neural networks via meta learning. *ICLR*.