Formalizing Generalization and Adversarial Robustness of Neural Networks to Weight Perturbations

Anonymous Author(s)
Affiliation
Address
email

Abstract

Studying the sensitivity of weight perturbation in neural networks and its impacts on model performance, including generalization and robustness, is an active research topic due to its implications on a wide range of machine learning tasks such as model compression, generalization gap assessment, and adversarial attacks. In this paper, we provide the first formal analysis for feed-forward neural networks with non-negative monotone activation functions against norm-bounded weight perturbations, in terms of the robustness in pairwise class margin functions and the Rademacher complexity for generalization. We further design a new theory-driven loss function for training generalizable and robust neural networks against weight perturbations. Empirical experiments are conducted to validate our theoretical analysis. Our results offer fundamental insights for characterizing the generalization and robustness of neural networks against weight perturbations.

1 Introduction

Neural network is currently the state-of-the-art machine learning model in a variety of tasks, including computer vision, natural language processing, and game-playing, to name a few. In particular, feed-forward neural networks consists of layers of trainable model weights and activation functions with the premise of learning informative data representations and the complex mapping between data samples and the associated labels. Albeit attaining superior performance, the need for studying the sensitivity of neural networks to weight perturbations is also intensifying owing to several practical motivations. For instance, in model compression, the robustness to weight quantification is crucial for designing energy-efficient hardware accelerator [1] and for reducing memory storage while retaining model performance [2, 3]. The notion of weight perturbation sensitivity is also used as a metric to evaluate the generalization gap at local minima [4, 5]. In adversarial robustness and security, weight sensitivity can be leveraged as a vulnerability for fault injection and causing erroneous prediction [6, 7]. However, while weight sensitivity plays an important role in many machine learning tasks and problem setups, theoretical characterization of its impacts on generalization and robustness of neural networks remains elusive.

This paper bridges this gap by developing a novel theoretical framework for understanding the generalization gap (through Rademacher complexity) and the robustness (through classification margin) of neural networks against norm-bounded weight perturbations. Specifically, we consider the multi-class classification problem setup and multi-layer feed-forward neural networks with non-negative monotonic activation functions. Our analysis offers fundamental insights into how weight perturbation affects the generalization gap and the pairwise class margin. To the best of our knowledge, this study is the first work that provides a comprehensive theoretical characterization of

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

the interplay between weight perturbation, robustness in classification margin, and generalization gap. Moreover, based on our analysis, we propose a theory-driven loss function for training generalizable and robust neural networks against norm-bounded weight perturbations. We validate its effectiveness via empirical experiments. We summarize our main contributions as follows.

- We study the robustness (worst-case bound) of the pairwise class margin function against weight perturbations in neural networks, including the analysis of single-layer (Theorem 1), all-layer (Theorem 2), and selected-layer (Theorem 3) weight perturbations.
- We characterize the generalization behavior of robust surrogate loss for neural networks under weight perturbations (Section 3.4) through Rademacher complexity (Theorem 4).
- We propose a theory-driven loss design for training generalizable and robust neural networks (Section 3.5). The empirical results in Section 4 validate our theoretical analysis and demonstrate the effectiveness of improving generalization and robustness against weight perturbations. We also show that in our studied setting the associated generalization bounds are non-vacuous.

2 Related Works

In model compression, the robustness to weight quantization is critical to reducing memory size and accesses for low-precision inference and training [2]. [1] showed that neural models that are robust to random bit flipping errors are beneficial for operating in low-voltage regime and thus improving energy efficiency for hardware implementation. [3] showed that incorporating weight perturbation sensitivity into training can better retain model performance (standard accuracy) after quantization. For studying the generalization of neural networks, [4] proposed a metric called sharpness (or weight sensitivity) by perturbing the learned model weights around the local minima of the loss landscape for generalization assessment. [8] introduced weight noise into the training process and concluded that random noise training improves the overall generalization. [9] empirically showed that using adversarial weight perturbation during training can improve generalization against input perturbations. [10] obtained non-vacuous generalization bound on deep (stochastic) neural networks by directly optimizing the PAC-Bayes bound while [5] made a connection between sharpness and PAC-Bayes theory and found that some combination of sharpness and norms on the model weights may capture the generalization behavior of neural networks. Additionally, [11] discovered normalized margin measure to be useful towards quantifying generalization property and a bound was therefore constructed to give an quantitative description on the generalization gap. Moreover, [12] incorporated additional assumptions to offer tighter and size-independent bounds from the setting of [13] and [11] respectively. We note that previous methods use random weight perturbations to evaluate the generalization ability (e.g. sharpness), which we call a standard generalization setting. In contrast, we study a different problem setup: generalization of models trained under worst-case (adversarial) weight perturbation.

Despite development of various generalization bounds, empirical observations in [14] showed that once the size of the training dataset grows, generalization bounds proposed in [5] and [11] will enlarge and thus become vacuous. Discussions on the relation between [14] and our works can be found at Appendix E, where we show that in our studied setting the associated generalization bounds are non-vacuous. On the other hand, [15] and [16] applied several techniques in tandem with a probabilistic method named path sampling to construct a representing set of given neural networks for approximating and studying the generalization property. Another approach considered by [17] consists of segmenting the neural networks into two functions, predictor and feature selection respectively, where two measures (representativeness and feature robustness) concerning these aforementioned functions were later combined to offer a meaningful generalization bound. However, these works only focused on the generalization behavior of the local minima and did not consider the generalization and robustness under weight perturbations. [3] proposed a certification method for weight perturbation retaining consistent model prediction. While the certification bound can be used to train robust models with interval bound propagation [18], it requires additional optimization subroutine and computation costs when comparing to our approach. Moreover, the convoluted nature of certification bound complicates the analysis when studying generalization, whereas our theoretical characterization is an intuitive and tight worst-case analysis.

In adversarial robustness, fault-injection attacks are known to inject errors to model weights at the inference phase and causing erroneous model prediction [6, 7], which can be realized at the hardware

level by changing or flipping the logic values of the corresponding bits and thus modifying the model parameters saved in memory [19, 20]. [21] proposed to use the mode connectivity of the model parameters in the loss landscape for mitigating such weight-perturbation-based adversarial attacks.

Although, to the best of our knowledge, theoretical characterization of generalization and robustness for neural networks against *weight* perturbations remains elusive, recent works have studied these properties under another scenario – *input* perturbations. Both empirical and theoretical evidence suggests the existence of a fundamental trade-off between generalization and robustness against norm-bounded input perturbations [22, 23, 24, 25]. The adversarial training proposed in [26] is a popular training strategy for training robust models against input perturbations, where a min-max optimization principle is used to minimize the worst-case input perturbations of a data batch during model parameter updates. For adversarial training with input perturbations, [27] proved its convergence and [28] derived bounds on its Rademacher complexity for generalization. Different from the case of input perturbation, we note that min-max optimization on neural network training subject to weight perturbation is not straightforward, as the minimization and maximization steps are both taken on the model parameters. In this paper, we disentangle the min-max formulation for weight perturbation by developing bounds for the inner maximization step and provide quantifiable metrics for training generalizable and robust neural networks against weight perturbations.

3 Main Results

We provide an overview of the presentation flow for our main results as follows. First, we introduce the mathematical notations and preliminary information in Section 3.1. In Section 3.2, we establish our weight perturbation analysis on a simplified case of single-layer perturbation. We then use the single-layer analysis as a building block and extend the results to the multi-layer perturbation setting in Section 3.3. Here we would like to note that similar reasoning could be applied to obtain results in convolutional neural networks and we relay the corresponding results and experiments in Appendix A.4, respectively. In Section 3.4, we define the framework of robust training with surrogate loss and study the generalization property using Rademacher complexity. Finally, we propose a theory-driven loss toward training robust and generalizable neural networks in Section 3.5.

3.1 Notation and Preliminaries

Notation We start by introducing the mathematical notations used in this paper. We define the set $[L] := \{1, 2, ..., L\}$. For any two non-empty sets A, B, $\mathbb{F}_{A\mapsto B}$ denotes the set of all functions from A to B. We mark the indicator function of an event E as $\mathbb{1}(E)$, which is 1 if E holds and 0 otherwise. We use $sgn(\cdot)$ to denote element-wise sign function that outputs 1 when input is nonnegative and -1 otherwise. Boldface lowercase letters are used to denote vectors (e.g., $\mathbf{x}$), and the i-th element is denoted as $[\mathbf{x}]_i$. Matrices are presented as boldface uppercase letters, say $\mathbf{W}$. Given a matrix $\mathbf{W} \in \mathbb{R}^{k\times d}$, we write its i-th row, j-th column and (i, j) element as $W_{i,:}$, $W_{:,j}$, and $W_{i,j}$ respectively. Moreover, we write its transpose matrix as $(\mathbf{W})^T$. The matrix (p, q) norm is defined as $\|\mathbf{W}\|_{p,q} := \left\|[\|W_{:,1}\|_p, \|W_{:,2}\|_p, ..., \|W_{:,d}\|_p]\right\|_q$ for any $p, q \geq 1$. For convenience, we have $\|\mathbf{W}\|_p = \|\mathbf{W}\|_{p,p}$ and write the spectral norm and Frobenius norm as $\|\mathbf{W}\|_\sigma$ and $\|\mathbf{W}\|_F$ respectively. We mark one matrix norm commonly used in this paper – the matrix $(1, \infty)$ norm. With a matrix $\mathbf{W}$, we express its matrix $(1, \infty)$ norm as $\|\mathbf{W}\|_{1,\infty}$, which is defined as $\|\mathbf{W}\|_{1,\infty} = \max_j \|W_{:,j}\|_1$ and $\left\|\mathbf{W}^T\right\|_{1,\infty} = \max_i \|W_{i,:}\|_1$. We use $\mathbb{B}^\infty_{\mathbf{W}}(\epsilon)$ to express an element-wise ℓ_∞ norm ball of matrix $\mathbf{W}$ within radius ϵ, i.e., $\mathbb{B}^\infty_{\mathbf{W}}(\epsilon) = \{\hat{\mathbf{W}} |\, |\hat{W}_{i,j} - W_{i,j}| \leq \epsilon, \forall i \in [k], j \in [d]\}$.

Preliminaries In order to formally explain our theoretical results, we introduce the considered learning problem, neural network model, and complexity definition. Let $\mathcal{X}$ and $\mathcal{Y}$ be the feature space and label space, respectively. We place the assumption that all data are drawn from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ and each data point is generated under i.i.d condition. In this paper, we specifically consider the feature space $\mathcal{X}$ as a subset of d-dimensional Euclidean space, i.e., $\mathcal{X} \subseteq \mathbb{R}^d$. We denote the symbol $\mathcal{F} \subseteq \mathbb{F}_{\mathcal{X}\mapsto\mathcal{Y}}$ to be the hypothesis class which we use to make predictions. Furthermore, we consider a loss function $\ell : \mathcal{X} \times \mathcal{Y} \longrightarrow [0, 1]$ and compose it with the hypothesis class to make a function family written as $\ell_{\mathcal{F}} := \{(\mathbf{x}, y) \mapsto \ell(f(\mathbf{x}), y) |\ f \in \mathcal{F}\}$. The optimal solution of this learning problem is a function $f^* \in \mathcal{F}$ such that it minimizes the

population risk $R(f) = E_{(\mathbf{x},y)\sim\mathcal{D}}[\ell(f(\mathbf{x}), y)]$. However, since the underlying data distribution is generally unknown, one typically aims at reducing the empirical risk evaluated by a set of training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, which can be expressed as $R_n(f) = \frac{1}{n}\sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i)$. The generalization error is the gap between population and empirical risk, which could serve as an indicator of model's performance under unseen data from identical distribution $\mathcal{D}$.

To study the generalization error, one would explore the learning capacity of a certain hypothesis class. In this paper, we adopt the notion of Rademacher complexity as a measure of learning capacity, which is widely used in statistical machine learning literature [29]. The empirical Rademacher complexity of a function class $\mathcal{F}$ given a set of samples $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is $\mathcal{R}_{\mathcal{S}}(\ell_{\mathcal{F}}) = E_\nu[\sup_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \nu_i \ell(f(\mathbf{x}_i), y_i)]$ where $\{\nu_i\}_{i=1}^n$ is a set of i.i.d Rademacher random variables with $\mathbb{P}\{\nu_i = -1\} = \mathbb{P}\{\nu_i = +1\} = \frac{1}{2}$. The empirical Rademacher complexity measures on average how well a function class $\mathcal{F}$ correlates with random noises on dataset $\mathcal{S}$. Thus, a richer or more complex family could better correlate with random noise on average. With Rademacher complexity as a toolkit, one can develop the following relationship between generalization error and complexity measure. Specifically, it is shown in [29] that given a set of training samples $\mathcal{S}$ and assume that the range of loss function $\ell(f(\mathbf{x}), y)$ is $[0, 1]$. Then for any $\delta \in (0, 1)$, with at least probability $1 - \delta$ we have $\forall f \in \mathcal{F}$, $R(f) \leq R_n(f) + 2\mathcal{R}_{\mathcal{S}}(\ell_{\mathcal{F}}) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2n}}$ Note that when the Rademacher complexity is small, it is then viable to learn the hypothesis class $\mathcal{F}$ by minimizing the empirical risk and thus effectively reducing the generalization gap.

Finally, we define the structure of neural networks and introduce a few related quantities. The problem studied in this paper is a multi-class classification task with the number of classes being K. Consider an input vector $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^d$, an L-layer neural network is defined as $f_{\boldsymbol{W}}(\mathbf{x}) = \mathbf{W}^L(...\rho(\mathbf{W}^1\mathbf{x})...) \in \mathbb{F}_{\mathcal{X}\mapsto\mathbb{R}}$ with $\boldsymbol{W}$ being the set containing all weight matrices, i.e., $\boldsymbol{W} := \{\mathbf{W}^k | \forall k \in [L]\}$, and the notation $\rho(\cdot)$ is used to express any non-negative monotone activation function and we further assume that $\rho(\cdot)$ is 1-Lipschitz, which includes popular activation functions such as ReLU applied element-wise on a vector. Moreover, the i-th component of neural networks' output is written as $f^i_{\boldsymbol{W}}(\mathbf{x}) = [f_{\boldsymbol{W}}(\mathbf{x})]_i$ and a pairwise margin between i-th and j-th class, denoted as $f^{ij}_{\boldsymbol{W}}(\mathbf{x}) := f^j_{\boldsymbol{W}}(\mathbf{x}) - f^j_{\boldsymbol{W}}(\mathbf{x})$, is said to be the difference between two classes in output of the neural network. Lastly, we use the notion of $\mathbf{z}^k$ and $\hat{\mathbf{z}}^k$ to represent the output vector of the k-th layer ($k \in [L-1]$) under natural and weight perturbed settings respectively, which are $\mathbf{z}^k = \rho(\mathbf{W}^k(...\rho(\mathbf{W}^1\mathbf{x})...))$ and $\hat{\mathbf{z}}^k = \rho(\hat{\mathbf{W}}^k(...\rho(\hat{\mathbf{W}}^1\mathbf{x})...))$, where $\hat{\mathbf{W}}^i \in \mathbb{B}^\infty_{\mathbf{W}^i}(\epsilon_i)$ denotes the perturbed weight matrix bounded by its element-wise ℓ_∞-norm with radius ϵ_i for some $i \in [k]$. Throughout this paper, we study the weight perturbation setting that each layer has an independent layer-wise perturbation budget, denoted by ϵ_k, $\forall k \in [L]$. In the case of single-layer perturbation, we may omit the layer index and simply use ϵ.

3.2 Building Block: Single-Layer Weight Perturbation

We study the sensitivity of neural network to weight perturbations through the pairwise margin bound $f^{ij}_{\boldsymbol{W}}(\mathbf{x})$. Specifically, when i and j corresponds to the top-1 and the second-top class prediction of $\mathbf{x}$, respectively, the margin can be used as an indicator of robust prediction under weight perturbation to $\boldsymbol{W}$. For ease of understanding, we have provided a simple example of a three-layer neural network and explain the bound through the error propagation incurred by weight perturbation in Appendix A.1. Here we directly introduce and analyze our theorem below.

Theorem 1 **(N-th layer weight perturbation ($N \neq L$))** *Let $f_{\boldsymbol{W}}(\boldsymbol{x}) = \boldsymbol{W}^L(...\rho(\boldsymbol{W}^1\boldsymbol{x})...)$ denote an L-layer neural network and let $f_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) = \boldsymbol{W}^L(..\hat{\boldsymbol{W}}^N...\rho(\boldsymbol{W}^1\boldsymbol{x})...)$ with $\hat{\boldsymbol{W}}^N \in \mathbb{B}^\infty_{\boldsymbol{W}^N}(\epsilon_N), N \neq L$, denote the corresponding network subject to N-th layer perturbation. For any set of perturbed and unperturbed pairwise margin $f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x})$ and $f^{ij}_{\boldsymbol{W}}(\boldsymbol{x})$, we have*

$$f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) \leq f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + \epsilon_N \left\|W^L_{i,:} - W^L_{j,:}\right\|_1 \left\|\boldsymbol{z}^{N-1}\right\|_1 \Pi_{k=1}^{L-N-1} \left\|(\boldsymbol{W}^{L-k})^T\right\|_{1,\infty}$$
$$:= f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + \Delta(\epsilon_N; \boldsymbol{z}^{N-1}; f)$$

where $\boldsymbol{z}^k = \rho(\boldsymbol{W}^k(...\rho(\boldsymbol{W}^1\boldsymbol{x})...))$.

Proof: See Appendix A.2

Since the final layer does not have any activation function, the margin bound on the margin difference when only perturbing the final layer can be simply derived, which is given in the following lemma.

Lemma 1 (Final-layer weight perturbation) *Consider the case $N = L$ in Theorem 1, we have*

$$f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) \leq f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + 2\epsilon_L \left\|\boldsymbol{z}^{L-1}\right\|_1$$

where $\boldsymbol{z}^{L-1}$ is the output of the $(L-1)$-th layer.

Proof: See Appendix A.2.

3.3 General Setting: Multi-Layer Weight Perturbation

With the developed single-layer analysis in Section 3.2 as a building block, we now extend our analysis to the general setting of multi-layer weight perturbation, which is further divided into two cases: (i) the case of perturbing all L layers; and (ii) the case of perturbing I out of L layers.

3.3.1 Perturbing All Layers

Once equipped with the concept of error propagation over subsequent layers, we consider the scenario where every layer in a neural network is subject to weight perturbation.

For simplicity and understanding, we will introduce an notation previously used but not fully explained in Theorem 1 and Lemma 1. Specifically, the notion $\Delta(\cdot)$ can be described as the error caused by certain layer. Moreover, as we will see later, the function requires three inputs, the perturbation of that certain layer, the previous input and finally the neural network. Explicitly, for an L-layer neural network $f(\cdot)$, a perturbation radius of the k-th layer ϵ_k, and a prior input $\mathbf{z}$, the delta function can be described as the following,

$$\Delta(\epsilon_k; \mathbf{z}; f) = \epsilon_k \left\|W^L_{i,:} - W^L_{j,:}\right\|_1 \|\mathbf{z}\|_1 (\Pi^{L-1}_{m=k+1} \|\mathbf{W}^m\|_{1,\infty})$$

We denote the model under this case as the all-perturbed setting. The following theorem states an upper bound on the pairwise margin between the natural (unperturbed) and all-perturbed settings.

Theorem 2 (all-layer perturbation) *Let $f_{\boldsymbol{W}}(\boldsymbol{x}) = \boldsymbol{W}^L(...\rho(\boldsymbol{W}^1\boldsymbol{x})...)$ denote an L-layer (natural) neural network and let $f_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) = \hat{\boldsymbol{W}}^L(..\hat{\boldsymbol{W}}^N...\rho(\hat{\boldsymbol{W}}^1\boldsymbol{x})...)$ with $\hat{\boldsymbol{W}}^k \in \mathbb{B}^{\infty}_{\boldsymbol{W}^k}(\epsilon_k)$, $\forall k \in [L]$, denote its perturbed version. For any set of pairwise margin $f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x})$ and $f^{ij}_{\boldsymbol{W}}(\boldsymbol{x})$, we have*

$$f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) \leq f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + \underbrace{\Delta(\epsilon_1; \boldsymbol{x}; f)}_{\textit{Input Layer Error}} + \underbrace{\sum_{k=2}^{L-1} \Delta(\epsilon_k; \boldsymbol{z}^{k-1^*}; f)}_{\textit{Intermediate Layer Error}} + \underbrace{2\epsilon_L \left\|\boldsymbol{z}^{L-1^*}\right\|_1}_{\textit{Final Layer Error}}$$

where $\boldsymbol{z}^{k^*} = \rho(\boldsymbol{W}^{k^*}...\rho(\boldsymbol{W}^{1^*}\boldsymbol{x})$ with $\boldsymbol{W}^{k^*}$ defined as

$$\begin{cases} W^{k^*}_{i,j} = W^k_{i,j} + \epsilon_k, \ \forall i,j \textit{ and } \forall k \in [L] \setminus \{1\} \\ W^{1^*}_{i,j} = W^1_{i,j} + sgn([\boldsymbol{x}]_j)\, \epsilon_1, \ \forall i,j \end{cases}$$

Proof: See Appendix A.3.2.

Here, we provide some intuition on deriving the upper bound of the margin in the all-perturbed setting. The scheme behind this all-perturbed scenario can be viewed as an inductive layer-wise error propagation. Specifically, we can choose any perturbed layer as the commencement point of propagation, then fix any other weight matrices' values and further calculate the propagation of error from that layer using the concept in Section 3.2. In such manner, after iterating through all these weight matrices subject to weight perturbation, one could obtain the final change in output value and therefore establish the pairwise margin bound. A close inspection of the bound shows that the propagation of error causes the first term since the input layer and the rest of the terms are errors propagating since the k-th layer in the neural network, where $k \in [L]$.

3.3.2 Perturbing Multiple Layers

The all-perturbed setting is a special case of perturbing layers from an index set I when $I = [L]$. We extend our analysis to the general multi-layer weight perturbation setting with $I \subseteq [L]$, which includes the single-layer ($I = \{N\}$) and all-perturbed ($I = [L]$) settings as special cases.

Theorem 3 (multiple-layer perturbation) *Let $f_{\boldsymbol{W}}(\boldsymbol{x}) = \boldsymbol{W}^L(...\rho(\boldsymbol{W}^1\boldsymbol{x})...)$ denote an L-Layer neural network. Given an index set $I \subseteq [L]$, we define the perturbed neural network as $f_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) = \tilde{\boldsymbol{W}}^L(...\tilde{\boldsymbol{W}}^N...\rho(\tilde{\boldsymbol{W}}^1\boldsymbol{x})...)$ with $\begin{cases} \tilde{\boldsymbol{W}}^k = \boldsymbol{W}^k, \ \forall k \in [L] \setminus I \\ \tilde{\boldsymbol{W}}^k = \hat{\boldsymbol{W}}^k, \ \hat{\boldsymbol{W}}^k \in \mathbb{B}^{\infty}_{\boldsymbol{W}^k}(\epsilon_k), \forall k \in I \end{cases}$*

Then, for any pairwise margin between $f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x})$ and $f^{ij}_{\boldsymbol{W}}(\boldsymbol{x})$,

$$f^{ij}_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) \leq f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + \underbrace{\sum_{k \in I \setminus \{L\}} \Delta(\epsilon_k; \boldsymbol{z}^{k-1*}; f)}_{\textit{Perturbed Layer Error}} + \underbrace{\mathbb{1}(L \in I) 2\epsilon_L \left\| \boldsymbol{z}^{L-1^*} \right\|_1}_{\textit{Final Layer Error}}$$

$$:= f^{ij}_{\boldsymbol{W}}(\boldsymbol{x}) + \underbrace{\eta^{ij}_{\boldsymbol{W}}(\boldsymbol{x}|I)}_{\textit{Error of Weight Perturbation}}$$

where $\boldsymbol{z}^{k^*} = \rho(\boldsymbol{W}^{k^*}...\rho(\boldsymbol{W}^{1^*}\boldsymbol{x})$ with $\boldsymbol{W}^{k^*}$ defined as $\begin{cases} W^{k^*}_{i,j} = \begin{cases} W^k_{i,j} + \epsilon_k, \ \forall i,j \ \forall k \in [L] \cap I \setminus \{1\} \\ W^k_{i,j}, \ \forall i,j \ \forall k \in [L] \setminus (I \cup \{1\}) \end{cases} \\ W^{1^*}_{i,j} = \begin{cases} W^1_{i,j} + sgn([\boldsymbol{x}]_j)\epsilon_1, \ \forall i,j \ \ if \ 1 \in I \\ W^1_{i,j}, \ \forall i,j \quad otherwise \end{cases} \end{cases}$

and $\boldsymbol{z}^{0^*} = \boldsymbol{x}$.

Proof: See Appendix A.3.3.

3.4 Surrogate Loss and Generalization Bound

3.4.1 Construction of Robust Surrogate Loss on Pairwise Margin

We aim to construct a surrogate loss function based on a standard loss function and study its behavior against weight perturbations. Here we study the single-layer perturbation case and elucidate the multi-layer perturbation case in Appendix B. Specifically, given a perturbation budget ϵ for the k-th layer and the original loss function $\ell(f_{\mathbf{W}}(\mathbf{x}), y)$, robust training aims to minimize the following objective function: $\tilde{\ell}(f_{\boldsymbol{W}}(\mathbf{x}), y) = \max_{\hat{\mathbf{W}}^m \in \mathbb{B}^{\infty}_{\mathbf{W}^m}(\epsilon)} \ell(f_{\widehat{\boldsymbol{W}}}(\mathbf{x}), y)$, which we call it as the robustness (worst-case) loss. Even for a single data point $(\mathbf{x}, y)$, it is hard to assess the exact robustness loss since it requires the maximization of a non-concave function over a norm ball. To make the problem of robust training against weight perturbations more computationally tractable, we aim to design a surrogate loss as an upper bound on the worst-case loss.

We focus on constructing a surrogate loss by means of pairwise margin bounds in Section 3.3. We first define two popular loss functions in the classification problem, ramp loss and cross entropy, and derive their surrogate versions. Define the margin function $M(f_{\boldsymbol{W}}(\mathbf{x}), y)$ as

$$M(f_{\boldsymbol{W}}(\mathbf{x}), y) = \min_{y' \neq y} [f_{\boldsymbol{W}}(\mathbf{x})]_y - [f_{\boldsymbol{W}}(\mathbf{x})]_{y'} = [f_{\boldsymbol{W}}(\mathbf{x})]_y - \max_{y' \neq y} [f_{\boldsymbol{W}}(\mathbf{x})]_{y'} \tag{1}$$

The ramp loss for a given data point $(\mathbf{x}, y)$ and neural network $f_{\boldsymbol{W}}(\cdot)$ is written as $\ell_{\text{ramp}}(f_{\boldsymbol{W}}(\mathbf{x}), y) = \phi_\gamma(M(f_{\boldsymbol{W}}(\mathbf{x}), y))$, where the function $\phi_\gamma : \mathbb{R} \mapsto [0, 1]$ is defined as $\phi_\gamma(t) = 1$ if $t \leq 0$, $\phi_\gamma(t) = 0$ if $t \geq \gamma$, and $\phi_\gamma(t) = 1 - \frac{t}{\gamma}$ if $t \in [0, \gamma]$. Since the ramp loss is a piece-wise linear function, its surrogate loss can be directly obtained with the pairwise margin bound in Section 3.3. The cross entropy is written as $CE(\tilde{f}_{\boldsymbol{W}}(\mathbf{x}), y) = -\ln([\tilde{f}_{\boldsymbol{W}}(\mathbf{x})]_y)$, where $\tilde{f}_{\boldsymbol{W}}(\mathbf{x})$ represents a neural network with its output passing through a softmax layer. That is, $[\tilde{f}_{\boldsymbol{W}}(\mathbf{x})]_i = \frac{\exp^{[f_{\boldsymbol{W}}(\mathbf{x})]_i}}{\sum_{k \in [K]} \exp^{[f_{\boldsymbol{W}}(\mathbf{x})]_k}}$. For ease of demonstration, we will be using ramp loss and its pairwise margin under single-layer perturbation

in the following lemma. The surrogate loss analysis for cross entropy and robust surrogate loss for multiple-layer perturbation is given in Appendix B.

Lemma 2 (robust surrogate ramp loss) *Let $N \in [L]$ denote the perturbed layer index and let*

$$\Psi(f_{\boldsymbol{W}}(\boldsymbol{x})) = 2 \max_{k\in[K]} \epsilon_N \left\|W^L_{k,:}\right\|_1 \Pi_{m=1}^{N-1} \|\boldsymbol{W}^m\|_{1,\infty} \Pi_{k=1}^{L-N-1} \left\|(\boldsymbol{W}^{L-k})^T\right\|_{1,\infty} \|\boldsymbol{x}\|_1$$

be the worst case error and

$$\hat{\ell}(f_{\boldsymbol{W}}(\boldsymbol{x}), y) := \phi_\gamma\big\{ \underbrace{M(f_{\boldsymbol{W}}(\boldsymbol{x}), y)}_{\textit{margin}} - \underbrace{\Psi(f_{\boldsymbol{W}}(\boldsymbol{x}))}_{\textit{worst-case error}} \big\}$$

Then we have upper and lower bounds of $\hat{\ell}$ in terms of 0-1 losses expressed as

$$\max_{\hat{W}^N \in \mathbb{B}^\infty_{W^N}(\epsilon)} \mathbb{1}\{y \neq \arg\max_{y'\in[K]} [f_{\widehat{\boldsymbol{W}}}(\boldsymbol{x})]_{y'}\} \leq \hat{\ell}(f_{\boldsymbol{W}}(\boldsymbol{x}), y) \leq \mathbb{1}\{M(f_{\boldsymbol{W}}(\boldsymbol{x}), y) - \Psi(f_{\boldsymbol{W}}(\boldsymbol{x})) \leq \gamma\}.$$

Proof: Please see Appendix B.1

One could observe in the formula that the margin function $M(f_{\boldsymbol{W}}(\mathbf{x}), y)$ serves as an accuracy objective similar to the standard training process while the latter term could be conceived as the worst-case error caused by weight perturbation that should be suppressed. Therefore, by training under such an objective, we can simulate the scenario of robust training. Another intuition on the surrogate loss function is that the surrogate loss also implies the difficulty of training robust and generalizable models against large weight perturbations. Since error caused by perturbations would be surging rapidly through layers, only small perturbations can be applied in training and practice, permitting the worst-case error term to be smaller than the margin term. However, one follow-up question that naturally arises is whether or not the generalization gap will be widened when training with the robust surrogate loss. The following section investigates the generalization property while conducting robust training and provides some theoretical insights to training toward a generalizable and robust model under weight perturbation.

3.4.2 Generalization Gap for Robust Surrogate Loss

We consider the robust surrogate loss established in Lemma 2 and study its generalization bound via Rademacher complexity in Theorem 4, where $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ denotes the set of i.i.d training samples, $\mathbf{X} := [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ denotes the matrix composed of training data samples, and $d_{\max} = \max\{d, d_1, \ldots, d_L\}$ denotes the maximum dimension among all weight matrices. Firstly, we define the adversarial population risk and empirical risk as

$$R(f) = \mathbb{P}_{(\mathbf{x},y)\sim D}\{\exists \hat{\mathbf{W}}^N \in \mathbb{B}^\infty_{\mathbf{W}^N}(\epsilon)\ s.t.\ y \neq \arg\max_{y'\in[K]} [f_{\hat{\mathbf{W}}}(\mathbf{x})]_{y'}\}$$

$$R_n(f) = \frac{1}{n}\sum_{i=1}^n \mathbb{1}\Big([f_{\boldsymbol{W}}(\mathbf{x}_i)]_{y_i} \leq \gamma + \max_{y'\neq y_i} [f_{\boldsymbol{W}}(\mathbf{x}_i)]_{y'} + \Psi(f_{\boldsymbol{W}}(\mathbf{x}))\Big)$$

Secondly, we denote the following empirical Rademacher complexity of both margin function and worst-case error as

$$\mathcal{R}_{\mathcal{S}}(M_{\mathcal{F}}) = \frac{4}{n^{3/2}} + \frac{60\log(n)\log(2d_{max})}{n} \|\mathbf{X}\|_F \left(\Pi_{h=1}^L s_h\right)\left(\sum_{j=1}^L \left(\frac{b_j}{s_j}\right)^{2/3}\right)^{3/2}$$

$$\mathcal{R}_{\mathcal{S}}(\Psi_{\mathcal{F}}) = \frac{2\epsilon_N \sup_{f\in\mathcal{F}} \Pi_{m=1}^{N-1} \|\mathbf{W}^m\|_{1,\infty} \Pi_{k=0}^{L-N-1} \left\|(\mathbf{W}^{L-k})^T\right\|_{1,\infty}}{n} \|\mathbf{X}\|_{1,2}$$

Theorem 4 (generalization gap for robust surrogate loss) *With Lemma 2, consider the neural network hypothesis class $\hat{\mathcal{F}} = \{f_{\widehat{\boldsymbol{W}}}(\boldsymbol{x}) | \boldsymbol{W} = (\boldsymbol{W}^1, \ldots \hat{\boldsymbol{W}}^N, \ldots, \boldsymbol{W}^L), \hat{\boldsymbol{W}}^N \in \mathbb{B}^\infty_{\boldsymbol{W}^N}(\epsilon_N) \left\|\boldsymbol{W}^h\right\|_\sigma \leq s_h, \left\|(\boldsymbol{W}^h)^T\right\|_{2,1} \leq b_h, h \in [L]\}$. For any $\gamma > 0$, with probability at least $1-\delta$, we have for all $f_{\boldsymbol{W}}(\cdot) \in \mathcal{F}$*

$$R(f) \leq R_n(f) + \frac{1}{\gamma}\Big(\underbrace{\mathcal{R}_{\mathcal{S}}(M_{\mathcal{F}})}_{\textit{standard generalization gap}} + \underbrace{\mathcal{R}_{\mathcal{S}}(\Psi_{\mathcal{F}})}_{\textit{complexity term of robust training}}\Big) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2n}}$$

Proof: Please see Appendix C.1

As highlighted in the bracket term of Theorem 4, if the product of multiple weight norm bounds in the additional complexity term $\mathcal{R}_{\mathcal{S}}(\Psi_{\mathcal{F}})$ caused by robust training is not well confined, the model will suffer from a notable generalization gap. Consequently, our analysis suggests a solution to reduce the generalization gap by imposing norm penalty functions on all weight matrices for training generalizable neural networks subject to weight perturbations.

3.5 Theory-driven Loss toward Robustness and Generalization

With our theoretical insights, we now propose a robust and generalizable loss function. Standard neural network classifier training uses a classification loss $\ell_{cls}(f_{\boldsymbol{W}}(\mathbf{x}), y)$ that aims to widen the pairwise margin so as to raise accuracy, but won't necessarily be able to curb the error in output once weight perturbation is imposed. To address this issue, we propose to train under a mixed and regularized objective given a data sample $(\mathbf{x}, y)$, which would, in turn, balance the tradeoff between standard accuracy, robustness, and generalization. The designed loss takes the form:

$$\ell^*(f_{\boldsymbol{W}}(\mathbf{x}), y) = \underbrace{\ell_{cls}(f_{\boldsymbol{W}}(\mathbf{x}), y)}_{\text{standard loss}} + \lambda \cdot \underbrace{\max_{y' \neq y} \{\eta_{\boldsymbol{W}}^{y'y}(\mathbf{x}|I)\}}_{\text{robustness loss from Thm. 3}} + \mu \cdot \underbrace{\sum_{m=1}^{L} \left(\left\| (\boldsymbol{W}^m)^T \right\|_{1,\infty} + \|\boldsymbol{W}^m\|_{1,\infty} \right)}_{\text{generalization gap regularization from Thm. 4}}$$

The first term in (2) originates from standard classification problem for task-specific accuracy, while the second term results from the maximum error on pairwise margin (the term $\eta_{\boldsymbol{W}}^{y'y}(x|I)$ defined in Theorem 3) induced by weight perturbation. Finally, we contribute the last term to the theoretical findings in Theorem 4, where imposing norm constraints on weight matrices could benefit generalization and prevent the generalization gap from widening. Based on our analysis, we also note that the standard technique of L_1 penalty function on weight applies as well.

4 Numerical Validation

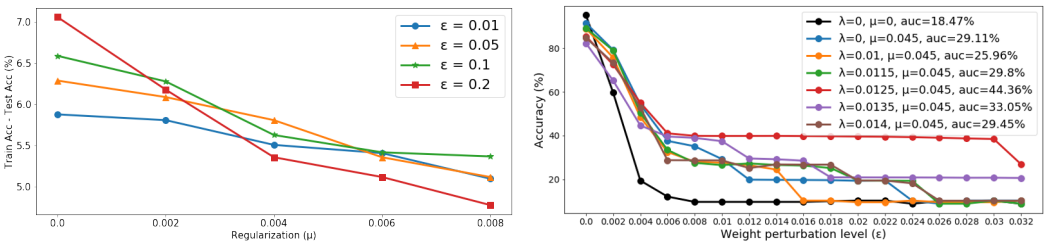
We validate our theoretical results and the designed loss function in (2) through two sets of experiments: *empirical generalization gap with matrix norm regularization* and *robust accuracy against adversarial weight perturbations*.

Experiment setup We used the MNIST dataset comprised of gray-scale images of hand-written digits with ten categories. We trained neural network models as in (**??**) with four dense layers (number of neurons are 128-64-32-10) and the ReLU activation function without the bias term. We used the loss function ℓ^* in (2) with all-layer perturbation bound (i.e., $I = [L]$), identical weight perturbation radius ϵ (or ϵ_{train}), cross entropy as the standard classification loss ℓ_{cls}, and a batch size of 32 with 20 epochs. Stochastic gradient descent with momentum is used for training, with the learning rate set to be 0.01. For the generalization experiment, we follow the same setting as in [28], which uses 1000 data samples to train the neural network. In all settings we assign identical weight perturbation budget ϵ for every layer. The comparison of weight distribution of each layer using standard and our proposed robust training is given in Appendix F. All experiments were conducted using an Intel Xeon E5-2620v4 CPU, 125 GB RAM, and an NVIDIA TITAN Xp GPU with 12 GB RAM. For reproducibility, our codes are given in the supplementary material.

Empirical generalization gap Figure 1 (a) shows the empirical generalization gap (training accuracy - test accuracy) with respect to the matrix norm regularization coefficient μ defined in (2). As indicated in Theorem 4, increasing μ effectively suppresses the Rademacher complexity and thus reduces the generalization gaps, which are consistently observed on neural networks trained with different weight perturbation level ϵ.

Robust accuracy against adversarial weight perturbation To evaluate the robustness against weight perturbations, we modified the projected gradient descent (PGD) attack originally designed for input perturbation [26], which we call as weight PGD attack. Starting from a trained neural network weight $\boldsymbol{W}$, the perturbed weight $\widetilde{\boldsymbol{W}}$ is crafted by iterative gradient ascent using the signed gradient of the standard loss denoted as $\text{sgn}(\nabla_{\boldsymbol{W}} \ell_{cls}(f_{\widetilde{\boldsymbol{W}}}(\mathbf{x}), y))$, followed by an element-wise ϵ clipping centered at $\boldsymbol{W}$. The attack iteration with the step size α is expressed as

$$\widetilde{\boldsymbol{W}}^{(0)} = \boldsymbol{W}, \;\; \widetilde{\boldsymbol{W}}^{(t+1)} = \text{Clip}_{\boldsymbol{W},\epsilon} \left\{ \widetilde{\boldsymbol{W}}^{(t)} + \alpha \; \text{sgn}(\nabla_{\boldsymbol{W}} \ell_{cls}(f_{\widetilde{\boldsymbol{W}}^{(t)}}(\mathbf{x}), y)) \right\}$$



(a) Empirical generalization gap (b) Robust accuracy under adversarial perturbation

Figure 1: (a) Empirical generalization gaps when varying the matrix norm regularization coefficient μ in (2). Consistent with the theoretical results, the gap reduces as μ increases for every ϵ value used for training. The curve of $\epsilon = 0$ is given in Appendix D.1. (b) Comparison of test accuracy of neural networks trained with different λ and μ under weight PGD attack (200 steps) with ϵ perturbation level. AUC refers to the area under curve score. Joint training with the two theory-driven terms in (2) indeed yields more generalizable and robust neural networks against weight perturbations.

We trained neural networks with different combinations of the coefficients λ and μ in (2) using $\epsilon_{\text{train}} = 0.01$. For different optimizers and datasets, we refer to additional experiments in Appendix. Figure 1 (b) shows the test accuracy under different weight perturbation level ϵ (i.e., the robust accuracy) with 200 attack steps. The standard model ($\lambda = \mu = 0$) is fragile to weight PGD attack. On the other hand, neural networks trained only with the robustness loss ($\lambda > 0$ and $\mu = 0$) or the generalization gap regularization ($\lambda = 0$ and $\mu > 0$) can improve the robust accuracy due to improved generalization and classification margin. Moreover, joint training using the proposed loss with proper coefficients can further boost model performance (e.g., $(\lambda, \mu) = (0.0125, 0.045)$), as seen by the significantly improved area under curve (AUC) score of robust accuracy over all tested ϵ values. The AUC of the best model is about $2\times$ larger than that of the standard model. Similar results can be concluded for the attack with 100 steps (see Appendix D.3). In Appendix D.4, we conduct additional experiments on the coefficients λ and μ and discuss their tradeoffs. In summary, the results suggest the effectiveness of our theory-driven loss function for improving generalization against weight perturbation.

Run-time analysis and weight quantization The run-time analysis in Appendix D.6 shows the training costs for robust and standard models are comparable. In Appendix D.7, we also show that neural networks trained with our proposed loss are more robust to weight quantization. Under 2-bit quantization, the decrease in accuracy of the robust model is 3x less than the standard model.

Non-vacuous generalization bound under weight perturbation In the standard generalization setting, [14] showed that when increasing the size of the training data, the error bound in [11] grows rapidly, loosing the ability to describe generalization gap and thus becomes vacuous. In Appendix 3.5, we show that the error bounds of the models trained using our proposed loss function (2) in Section 3.5 would not grow in a polynomial rate and instead shows a decreasing trend, concluding the non-vacuity of the associated generalization bounds in our robust generalization setting.

5 Conclusion

In this paper, we developed a formal analysis of the robustness associated with the pairwise class margin for neural networks against weight perturbations. We also characterized its generalization gap through Rademacher complexity. A theory-driven loss function for robust generalization was proposed, and the empirical results showed significantly improved performance in generalization and robustness. Our analysis offers theoretical insights and informs the principles of training loss design toward generalizable and robust neural networks subject to weight perturbations.

References

[1] Stutz, D., N. Chandramoorthy, M. Hein, et al. Bit error robustness for energy-efficient dnn accelerators. *arXiv preprint arXiv:2006.13977*, 2020.

[2] Hubara, I., M. Courbariaux, D. Soudry, et al. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

[3] Weng, T.-W., P. Zhao, S. Liu, et al. Towards certificated model robustness against weight perturbations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6356–6363. 2020.

[4] Keskar, N. S., D. Mudigere, J. Nocedal, et al. On large-batch training for deep learning: Generalization gap and sharp minima. *International Conference on Learning Representations*, 2017.

[5] Neyshabur, B., S. Bhojanapalli, D. McAllester, et al. Exploring generalization in deep learning. In *Advances in neural information processing systems*, pages 5947–5956. 2017.

[6] Liu, Y., L. Wei, B. Luo, et al. Fault injection attack on deep neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 131–138. 2017.

[7] Zhao, P., S. Wang, C. Gongye, et al. Fault sneaking attack: A stealthy framework for misleading deep neural networks. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. 2019.

[8] An, G. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[9] Wu, D., S.-T. Xia, Y. Wang. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

[10] Dziugaite, G. K., D. M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. *arXiv preprint arXiv:1703.11008*, 2017.

[11] Bartlett, P. L., D. J. Foster, M. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249. 2017.

[12] Golowich, N., A. Rakhlin, O. Shamir. Size-independent sample complexity of neural networks, 2019.

[13] Neyshabur, B., R. Tomioka, N. Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. 2015.

[14] Nagarajan, V., J. Z. Kolter. Uniform convergence may be unable to explain generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 11615–11626. 2019.

[15] Barron, A. R., J. M. Klusowski. Approximation and estimation for high-dimensional deep learning networks. *arXiv preprint arXiv:1809.03090*, 2018.

[16] Theisen, R., J. M. Klusowski, H. Wang, et al. Global capacity measures for deep relu networks via path sampling. *arXiv preprint arXiv:1910.10245*, 2019.

[17] Petzka, H., L. Adilova, M. Kamp, et al. Feature-robustness, flatness and generalization error for deep neural networks. *arXiv preprint arXiv:2001.00939*, 2020.

[18] Gowal, S., K. D. Dvijotham, R. Stanforth, et al. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4842–4851. 2019.

[19] Barenghi, A., L. Breveglieri, I. Koren, et al. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[20] Van Der Veen, V., Y. Fratantonio, M. Lindorfer, et al. Drammer: Deterministic rowhammer attacks on mobile platforms. In *ACM SIGSAC conference on computer and communications security*, pages 1675–1689. 2016.

[21] Zhao, P., P.-Y. Chen, P. Das, et al. Bridging mode connectivity in loss landscapes and adversarial robustness. In *International Conference on Learning Representations*. 2020.

[22] Xu, H., S. Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012.

[23] Su, D., H. Zhang, H. Chen, et al. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648. 2018.

[24] Zhang, H., Y. Yu, J. Jiao, et al. Theoretically principled trade-off between robustness and accuracy. *International Conference on Machine Learning*, 2019.

[25] Tsipras, D., S. Santurkar, L. Engstrom, et al. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*. 2019.

[26] Madry, A., A. Makelov, L. Schmidt, et al. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

[27] Wang, Y., X. Ma, J. Bailey, et al. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, vol. 1, page 2. 2019.

[28] Yin, D., R. Kannan, P. Bartlett. Rademacher complexity for adversarially robust generalization. In *International Conference on Machine Learning*, pages 7085–7094. 2019.

[29] Mohri, M., A. Rostamizadeh, A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section .
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] In this paper, we design a theory-driven loss which theoretically can be used in more applications without any limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [No] The major contribution of our work is to propose a theory-driven loss for training generalizable and robust networks against weight perturbations to improve the model robustness and performances. Namely, it can reduce potential negative impacts.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section3.
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix A,B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We will include the code, data and instructions in the supplementary material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] In our experiments, it does not include the effect on random seeds.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We used publicly available datasets which can be used for educational research purposes such as this work. We have suitably cited all datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] The datasets we used do not include personally identifiable information or offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]