Large Norms of CNN Layers Do Not Hurt Adversarial Robustness

Youwei Liang[1], **Dong Huang**[1,2*]
[1] College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China
[2] Pazhou Lab, Guangzhou, China
liangyouwei1@gmail.com, huangdonghere@gmail.com

Abstract

Since the Lipschitz properties of convolutional neural networks (CNNs) are widely considered to be related to adversarial robustness, we theoretically characterize the ℓ_1 norm and ℓ_∞ norm of 2D multi-channel convolutional layers and provide efficient methods to compute the exact ℓ_1 norm and ℓ_∞ norm. Based on our theorem, we propose a novel regularization method termed norm decay, which can effectively reduce the norms of convolutional layers and fully-connected layers. Experiments show that norm-regularization methods, including norm decay, weight decay, and singular value clipping, can improve generalization of CNNs. However, they can slightly hurt adversarial robustness. Observing this unexpected phenomenon, we compute the norms of layers in the CNNs trained with three different adversarial training frameworks and surprisingly find that adversarially robust CNNs have comparable or even larger layer norms than their non-adversarially robust counterparts. Furthermore, we prove that under a mild assumption, adversarially robust classifiers can be achieved using neural networks, and an adversarially robust neural network can have an arbitrarily large Lipschitz constant. For this reason, enforcing small norms on CNN layers may be neither necessary nor effective in achieving adversarial robustness. The code is available at https://github.com/youweiliang/norm_robustness.

Introduction

Convolutional neural networks (CNNs) have enjoyed great success in computer vision (LeCun, Bengio, and Hinton 2015; Goodfellow, Bengio, and Courville 2016). However, many have found that CNNs are vulnerable to adversarial attack (Akhtar and Mian 2018; Eykholt et al. 2018; Huang et al. 2017; Moosavi-Dezfooli, Fawzi, and Frossard 2016; Moosavi-Dezfooli et al. 2017). For example, changing one pixel in an image may change the prediction of a CNN (Su, Vargas, and Sakurai 2019). Many researchers link the vulnerability of CNNs to their Lipschitz properties and the common belief is that CNNs with small Lipschitz constants are more robust against adversarial attack (Szegedy et al. 2014; Cisse et al. 2017; Bietti et al. 2019; Anil, Lucas, and Grosse 2019; Virmaux and Scaman 2018; Fazlyab et al. 2019). Since computing the Lipschitz constants of CNNs is intractable (Virmaux and Scaman 2018), existing approaches seek to regularize the norms of individual CNN layers. For example, Cisse et al. (2017) proposed Parseval Network where the ℓ_2 norms of linear and convolutional layers are constrained to be orthogonal. However, from Table 1 in their paper, we can see Parseval Network only slightly improves adversarial robustness in most cases and even reduces robustness in some cases. Anil, Lucas, and Grosse (2019) combined GroupSort, which is a gradient norm preserving activation function, with norm-constrained weight matrices regularization to enforce Lipschitzness in fully-connected networks while maintaining the expressive power of the models. Li et al. (2019) further extended GroupSort to CNNs by proposing Block Convolution Orthogonal Parameterization (BCOP), which restricts the linear transformation matrix of a convolutional kernel to be orthogonal and thus its ℓ_2 norm is bounded by 1. Again, we find that the improvement of adversarial robustness is typically small while the standard accuracy drops considerably. For example, we use the state-of-the-art adversarial "Auto Attack" (Croce and Hein 2020) to test the checkpoint from the authors[1] and find that, the robust accuracy of their best model on CIFAR-10 is 8.4% (under standard ℓ_∞ attack with $\epsilon = 8/255$), which is much smaller than the state of the art (59.5%[2]) such as the methods of (Carmon et al. 2019; Wang et al. 2019; Pang et al. 2020), while the standard accuracy drops to 72.2%. Besides, since GroupSort and BCOP have virtually changed the forward computation and/or architecture of the network, it is *unclear* whether their improvement in adversarial robustness is due to regularization of norms or the change in computation/architecture. These issues raise concerns over the effectiveness of regularization of norms.

The approaches of regularization of norms are motivated by the idea that reducing norms of individual layers can reduce global Lipschitz constant and reducing global Lipschitz constant can ensure smaller local Lipschitz constants and thus improve robustness. In this paper, we carefully investigate the connections and distinctions between the norms of layers, local Lipschitz constants, and global Lipschitz constants. And our findings, both theoretically and empirically, do not support the prevailing idea that large norms are bad for adversarial robustness.

Our contribution in this paper is summarized as follows.

[*] Corresponding author.

[1] https://github.com/ColinQiyangLi/LConvNet

[2] https://github.com/fra31/auto-attack

- We theoretically characterize the ℓ_1 norm and ℓ_∞ norm of 2D multi-channel convolutional layers. To our knowledge, our approach is the fastest among the existing methods for computing norms of convolutional layers.
- We present a novel regularization method termed norm decay, which can improve generalization of CNNs.
- We *prove* that robust classifiers can be realized with neural networks. Further, our theoretical results and extensive experiments suggest that large norms (compared to norm-regularized networks) of CNN layers do not hurt adversarial robustness.

Related Work

Researches related to the norms of convolutional layers are mostly concerned with the ℓ_2 norm. For example, Miyato et al. (2018) reshape the 4D convolutional kernel into a 2D matrix and use power iterations to compute the ℓ_2 norm of the matrix. Although this method can improve the image quality produced by WGAN (Arjovsky, Chintala, and Bottou 2017), the norm of the reshaped convolutional kernel does not reflect the true norm of the kernel. Based on the observation that the result of power iterations can be computed through gradient back-propagation, Virmaux and Scaman (2018) proposed AutoGrad to compute the ℓ_2 norm. Sedghi, Gupta, and Long (2019) theoretically analyzed the circulant patterns in the unrolled convolutional kernel, based on which they discovered a new approach to compute the singular values of the kernels. Using the computed spectrum of convolution, they proposed singular value clipping, a regularization method which projects a convolution onto the set of convolutions with bounded ℓ_2 norms. It is worth noting that, because of the equivalence of the matrix norms, i.e., $1/\sqrt{m}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$ and $1/\sqrt{n}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty$ for all matrices $A \in \mathbb{R}^{m\times n}$, our approaches to compute the ℓ_1 and ℓ_∞ norm have the same functionalities as those to compute ℓ_2 norm, while our approaches are much more efficient. Gouk et al. (2018) give an analysis on the ℓ_1 and ℓ_∞ norm of convolutional layers but they neglect the padding and strides of convolution, which may lead to incorrect computation results.

All these works have not yet given a clear analysis of how the norms of neural net layers are related to adversarial robustness. To bridge this gap, we first characterize the norms of CNN layers and then analyze theoretically and test empirically if large norms are bad for adversarial robustness.

The ℓ_1 and ℓ_∞ Norm of Convolutional Layers

To understand how norms of CNN layers influence adversarial robustness, we first need to characterize the norms. Sedghi, Gupta, and Long (2019) proposed a method for computing the singular values of convolutional layers, where the largest one is the ℓ_2 norm. However, their method applies to only the case when the stride of convolution is 1, and computing singular values with their algorithm is still computationally expensive and prohibit its usage in large scale deep learning. To alleviate these problems, we theoretically analyze the ℓ_1 norm and ℓ_∞ norm of convolutional layer, and we find that our method of computing norms is much more efficient than that of (Sedghi, Gupta, and Long 2019).

Since 2D multi-channel convolutional layers (Conv2d) (Goodfellow, Bengio, and Courville 2016) are arguably the most widely used convolutional layers in practice, we analyze Conv2d in this paper while the analysis for other types of convolutional layer should be similar.

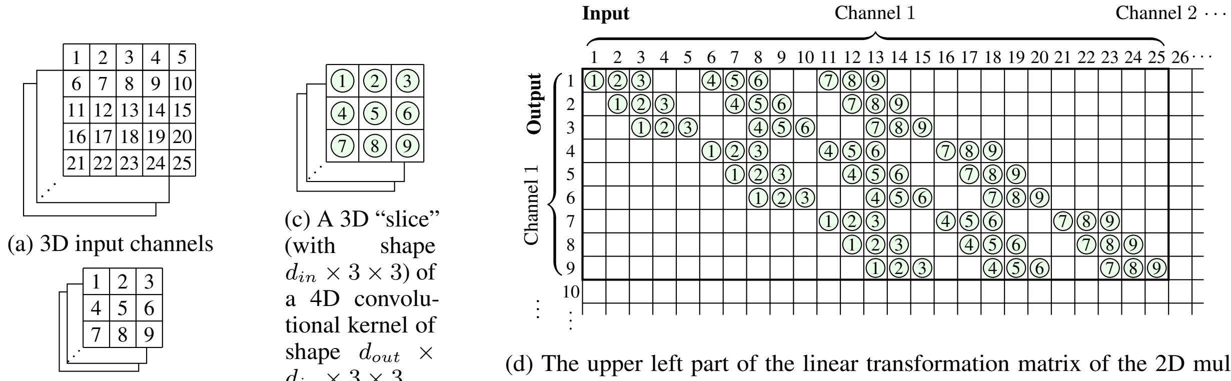
Setting. Let $\mathrm{conv}\colon \mathbb{R}^{d_{in}\times h_{in}\times w_{in}} \to \mathbb{R}^{d_{out}\times h_{out}\times w_{out}}$ be a 2D multi-channel convolutional layer with a 4D kernel $K \in \mathbb{R}^{d_{out}\times d_{in}\times k_1\times k_2}$, where d is the channel dimension, h and w are the spatial dimensions of images, and k_1 and k_2 are the kernel size. Suppose the vertical stride of conv is s_1 and horizontal stride is s_2, and padding size is p_1 and p_2.

We first note that Conv2d without bias is a linear transformation, which can be verified by checking $\mathrm{conv}(\alpha x) = \alpha\,\mathrm{conv}(x)$ and $\mathrm{conv}(x+y) = \mathrm{conv}(x) + \mathrm{conv}(y)$ for any $\alpha \in \mathbb{R}$ and any tensors x and y with appropriate shape. Normally, the input and output of Conv2d are 3D tensors (e.g., images) while the associated linear transformation takes 1D vectors as input. So we reshape the input into a vector (only reshaping the input channel *excluding padding* since padding elements are not variables) and then Conv2d can be represented by $\mathrm{conv}(x) = Mx + b$, where M is the linear transformation matrix and b is the bias vector. Then the norm of Conv2d is just the norm of M. We first state the following well known facts about the norms of a matrix $A \in \mathbb{R}^{m\times n}$: $\|A\|_1 = \max_{1\leq j\leq n}\sum_{i=1}^{m}|A_{ij}|$, $\|A\|_\infty = \max_{1\leq i\leq m}\sum_{j=1}^{n}|A_{ij}|$, and $\|A\|_2 = \sigma_{\max}(A)$, where $\sigma_{\max}(A)$ is the largest singular value of A. While the exact computation of M is complicated, we can analyze how the norm $\|M\|_p$ is related to the convolutional kernel K, which is a 4D tensor in the case of Conv2d.

By carefully inspecting how the output elements of Conv2d are related to the input elements, we find M is basically like the matrix in Figure 1d. The rows of M can be formed by convolving a 3D "slice" (see Figure 1c) of the 4D kernel with the 3D input channels and inspecting which elements on the input channels are being convolved with the 3D kernel slice. If the stride of convolution is 1, M is indeed a doubly circulant matrix like the one in Figure 1d (Goodfellow, Bengio, and Courville 2016; Sedghi, Gupta, and Long 2019). However, when the stride is not 1 or there is padding in the input channel, the patterns in M could be much more complicated, which is not addressed in existing *analytical* formulas (Gouk et al. 2018; Sedghi, Gupta, and Long 2019). We take stride and padding into account and properly address these issues. To obtain a theoretical result of the Lipschitz properties of Conv2d, we present the following assumption, which basically means that the convolutional kernel can be completely covered by the input channel (excluding padding) during convolution. We emphasize that the assumption holds for most convolutional layers used in practice.

Assumption 1. *Let c_1 and c_2 be the smallest positive integers such that $c_1 s_1 \geq p_1$ and $c_2 s_2 \geq p_2$. Assume $k_1 + c_1 s_1 - p_1 \leq h_{in}$ and $k_2 + c_2 s_2 - p_2 \leq w_{in}$, and the padding (if any) for the input of* conv *is zero padding.*

We need the following lemma to present our formula to compute the ℓ_1 norm of Conv2d. The overall idea of the



(a) 3D input channels

(b) 3D output channels

(c) A 3D "slice" (with shape $d_{in} \times 3 \times 3$) of a 4D convolutional kernel of shape $d_{out} \times d_{in} \times 3 \times 3$

(d) The upper left part of the linear transformation matrix of the 2D multi-channel convolutional layer for (a) (b) (c) with stride 1 and no padding. With padding and various strides, the pattern of the linear transformation matrix is more complicated, but these have been properly addressed in our theorem.

Figure 1: An illustration of the linear transformation matrix of a convolutional layer.

lemma is that it links the nonzero elements of every column of M to the elements in the convolutional kernel, which is very useful because the ℓ_1 norm of M is exactly the maximum of the absolute column sum of M.

Lemma 1. *Suppose Assumption 1 holds. The indices set for the last two dimensions of K is $\mathcal{N} := \{(k,t) \colon 1 \le k \le k_1, 1 \le t \le k_2\}$. Let $\sim$ be a binary relation on $\mathcal{N}$ such that, if indices (a,b) and (c,d) satisfy $(a-c) \equiv 0 \pmod{s_1}$ and $(b-d) \equiv 0 \pmod{s_2}$, then $(a,b) \sim (c,d)$. Let $\mathcal{A}_{(a,b)} \subseteq \mathcal{N}$ denote the largest set[3] of indices such that $(a,b) \in \mathcal{A}_{(a,b)}$ and for all $(c,d) \in \mathcal{A}_{(a,b)}$, $(c,d) \sim (a,b)$ and $0 \le c-a \le h_{in} + 2p_1 - k_1$ and $0 \le d-b \le w_{in} + 2p_2 - k_2$. Let $\mathcal{S}$ be a set of indices sets defined as $\mathcal{S} := \{\mathcal{A}_{(a,b)} \colon (a,b) \in \mathcal{N}\}$. Let $M_{:,n}$ be the n-th column of the linear transformation matrix M of* conv*, and let* $\operatorname{nz}(M_{:,n})$ *be the set of nonzero elements of $M_{:,n}$. Then for $n = 1, 2, \dots, d_{in}h_{in}w_{in}$, there exists an indices set $\mathcal{A} \in \mathcal{S}$ such that* $\operatorname{nz}(M_{:,n}) \subseteq \{K_{i,j,k,t} \colon 1 \le i \le d_{out}, (k,t) \in \mathcal{A}\}$*, where $j = \lceil n/(h_{in}w_{in}) \rceil$. Furthermore, for $j = 1, 2, \dots, d_{in}$, for all $\mathcal{A} \in \mathcal{S}$, there exists a column $M_{:,n}$ of M, where $(j-1)h_{in}w_{in} < n \le jh_{in}w_{in}$, such that* $\operatorname{nz}(M_{:,n}) \supseteq \{K_{i,j,k,t} \colon 1 \le i \le d_{out}, (k,t) \in \mathcal{A}\}$.

Now we are ready to show how to calculate the norms of Conv2d.

Theorem 1. *Suppose Assumption 1 holds. Then the ℓ_1 norm and ℓ_∞ norm and an upper bound of the ℓ_2 norm of* conv *are given by*

$$\|\operatorname{conv}\|_1 = \max_{1 \le j \le d_{in}} \max_{\mathcal{A} \in \mathcal{S}} \sum_{(k,t) \in \mathcal{A}} \sum_{i=1}^{d_{out}} |K_{i,j,k,t}|, \tag{1}$$

$$\|\operatorname{conv}\|_\infty = \max_{1 \le i \le d_{out}} \sum_{j=1}^{d_{in}} \sum_{k=1}^{k_1} \sum_{t=1}^{k_2} |K_{i,j,k,t}|, \tag{2}$$

$$\|\operatorname{conv}\|_2 \le \left(h_{out}w_{out} \sum_{i=1}^{d_{out}} \sum_{j=1}^{d_{in}} \sum_{k=1}^{k_1} \sum_{t=1}^{k_2} |K_{i,j,k,t}|^2 \right)^{\frac{1}{2}} \tag{3}$$

where $\mathcal{S}$ is a set of indices sets defined in Lemma 1.

The proofs of Lemma 1 and Theorem 1 are lengthy and deferred to the Appendix.

Do Large Norms Hurt Adversarial Robustness?

Many works mentioned in the Introduction regularize the norms of layers to improve robustness, while some authors (Sokolić et al. 2017; Weng et al. 2018; Yang et al. 2020) pointed out that local Lipschitzness is what really matters to adversarial robustness. In the setting of neural networks, the relations and distinctions between global Lipschitzness, local Lipschitzness, and the norms of layers are unclear. We devote this section to investigate their connections. For completeness, we provide the definition of Lipschitz constant.

Definition 1 (Global and local Lipschitz constant)**.** Given a function $f \colon \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are two finite-dimensional normed spaces equipped with norm $\|\cdot\|_p$, the global Lipschitz constant of f is defined as

$$\|f\|_p := \sup_{x_1, x_2 \in \mathcal{X}} \frac{\|f(x_1) - f(x_2)\|_p}{\|x_1 - x_2\|_p}. \tag{4}$$

We call $\|f\|_p$ a local Lipschitz constant on a compact space $\mathcal{V} \subset \mathcal{X}$ if x_1 and x_2 are confined to $\mathcal{V}$. In the context of neural nets, the norm is usually the ℓ_1, ℓ_2, or ℓ_∞ norm.

To deduce the prevailing claim that large norms hurt adversarial robustness, one must go through the following reasoning: large norms of layers → large global Lipschitz constant of the network → large local Lipschitz constant in the neighborhood of samples → the output of the network changes so sharply around samples that the prediction is changed → reducing adversarial robustness. However, there are at least two serious issues at the first and second arrow in the above reasoning. The first issue is that large norms of individual layers do not necessarily cause the global Lipschitz constant of the network to be large, as demonstrated in the following proposition.

[3] By largest set we mean adding any other indices to $\mathcal{A}_{(a,b)}$ would violate the conditions that follow.

Proposition 1. *There exists a feedforward network with ReLU activation where the norms of all layers can be arbitrarily large while the Lipschitz constant of the network is 0.*

The proof is deferred to the Appendix. Although the network illustrated in the proof of Proposition 1 is a very simple one, it does show that the coupling between layers could make the actual Lipschitz constant of a neural net much smaller than we can expect from the norms of layers. A related discussion of coupling between layers is presented in (Virmaux and Scaman 2018). This proposition breaks the logical chain at the first arrow in the above reasoning of large norms hurting adversarial robustness. The second issue in the reasoning is that, even if the Lipschitz constant of a neural network is very large, it can still be adversarially robust. This is because, *local* Lipschitzness, which means the output of a network does not change sharply in the neighborhood of samples, is *already sufficient* for adversarial robustness, and it has no requirement on the *global* Lipschitz constant (Sokolić et al. 2017; Weng et al. 2018; Yang et al. 2020). In the next paragraph, we will first prove that under a mild assumption, robust classifiers can be achieved with neural networks, and then we will prove that the Lipschitz constant of a *robust* classifier can be arbitrarily large.

Since we are primarily interested in classification tasks, our discussion will be confined to these tasks. We first need some notations. Let $\mathcal{X} \subset \mathbb{R}^n$ be the instance space (data domain) and $\mathcal{Y} = \{1, \ldots, C\}$ be the (finite) label set where C is the number of classes. Let $\mathcal{D}$ be the probability measure of $\mathcal{X}$, i.e., for a subset $A \subset \mathcal{X}$, $\mathcal{D}(A)$ gives the probability of observing a data point $x \in A$. Let $\mathcal{X}$ be endowed with a metric d that will be used in adversarial attack, and let $\mathrm{B}(x, \epsilon) := \{\tilde{x} : d(x, \tilde{x}) \leq \epsilon\}$ be the ϵ-neighborhood of x. Let $f : \mathcal{X} \to \mathcal{Y}$ denote the underlying labeling function (which we do not know), and let $\mathcal{X}^{(c)} \subset \mathcal{X}$ be the set of class c. The robust accuracy is defined as follows, similar to the "astuteness" in (Wang, Jha, and Chaudhuri 2018; Yang et al. 2020).

Definition 2 (Robust accuracy)**.** We say a classifier $g : \mathbb{R}^n \to \mathbb{R}$ have robust accuracy γ under adversarial attack of magnitude $\epsilon \geq 0$ if $\gamma = \mathcal{D}\big(\{x \in \mathcal{X} : |g(\tilde{x}) - f(x)| < 0.5 \text{ for all } \tilde{x} \in \mathrm{B}(x, \epsilon)\}\big)$.

Here, for convenience of proof, we use a classifier that outputs a real number, and its prediction is determined by choosing the nearest label to its output. Thus, if the output of g is at most 0.5 apart from the true label, then g gives the correct label. This definition and the following theorem and proposition can be easily generalized to the widely used classifiers with vectors as outputs. Intuitively, robust accuracy is the probability measure of the set of "robust points", which are the points whose ϵ-neighbors can be correctly classified by g. Our next theorem shows that, under a mild assumption similar to that in (Yang et al. 2020), there exits a neural network that can achieve robust accuracy 1 (i.e., the highest accuracy).

Assumption 2 (2-epsilon separable)**.** *The data points of any two different classes are 2-epsilon separable:* $\inf\{d(x^{(i)}, x^{(j)}) : x^{(i)} \in \mathcal{X}^{(i)}, x^{(j)} \in \mathcal{X}^{(j)}, i \neq j\} > 2\epsilon$.

Intuitively, Assumption 2 states any two epsilon-balls centered at data points from two different classes do not have overlap. We would like to provide an explanation for why the assumption holds for a reasonable attack size ϵ in computer vision tasks. We say the attack size ϵ is reasonable, if for all $x \in \mathcal{X}$ and for all $s \in \mathrm{B}(x, \epsilon)$, the label of s given by humans is the same as that of x. Thus, if ϵ is reasonable (as in our definition), the two balls $\mathrm{B}(x_1, \epsilon)$ and $\mathrm{B}(x_2, \epsilon)$ for x_1 and x_2 coming from two different classes would not have overlap, which means the 2-epsilon separable assumption should hold for a *reasonable* ϵ. In our analysis, we do not rely on the number of classes, so the assumption should hold for any number of classes. But we do think in reality, the training of adversarially robust classifiers may be more difficult for larger number of classes because intuitively, the neighborhood $\mathrm{B}(x, \epsilon)$ of x from different classes are more likely to be close to each other if the number of classes are larger.

Theorem 2 (Realizability of robust classifiers)**.** *Let* $\rho : \mathbb{R} \to \mathbb{R}$ *be any non-affine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. If Assumption 2 holds, then there exists a feedforward neural network with* ρ *being the activation function that has robust accuracy* 1.

The proof is deferred to the Appendix. We notice that Yang et al. (2020) showed a related result that there exists a function that has small local Lipschitz constants and achieves robust accuracy 1. Our result (Theorem 2) is different from theirs in that we prove that a neural network that can be realized in a digital computer can obtain robust accuracy 1 while they proved an *abstract* function f can obtain robust accuracy 1, where the definition of f relies on knowing the data distribution $\mathcal{D}$ and f may not be realized in a digital computer. Yang et al. (2020) also empirically showed that real-world image datasets are typically 2ϵ-separable and thus there should exist neural networks that achieve high robust accuracy. Using Theorem 2, we are ready to show that a neural network having robust accuracy 1 can have arbitrarily large Lipschitz constant, as in the following proposition.

Proposition 2. *Let* $\rho : \mathbb{R} \to \mathbb{R}$ *be any non-affine continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. If Assumption 2 holds, then for all* $\xi > 0$*, there exists a feedforward neural network with* ρ *being the activation function that achieves robust accuracy* 1 *and its Lipschitz constant is at least* ξ.

The proof is deferred to the Appendix. Proposition 2 shows that neural networks that have large Lipschitz constant can be adversarially robust because they can have small local Lipschitz constants *in the instance domain*. This proposition implies that what really matters is the local Lipschitz property of the network instead of the global one. Yang et al. (2020) also stressed the importance of controlling local Lipschitzness of neural nets, by showing a function that has small local Lipschitz constant can achieve robust accuracy 1.

On the other hand, although enforcing a small global Lipschitz constant can ensure local Lipschitzness, it may reduce the expressive power of the network and hurt standard accuracy. Let us consider fitting the function $f(x) = 1/x$ in the interval $(0.5, 1)$; then no 1-Lipschitz function could fit it well since the slope of the function in that interval is as large as 4. Thus, enforcing global Lipschitzness may result in hurting standard accuracy a lot while obtaining only a slight improvement in robustness (e.g., as in (Li et al. 2019)). In order to further investigate how norms influence the adversarial robustness in practice, we further propose a novel norm-regularization method in the next section.

A Regularization Method: Norm Decay

Equipped with Eq. (1) and Eq. (2), we present an algorithm termed norm decay to control (or regularize) the norm of fully-connected layers and convolutional layers. Then we investigate how norm decay influences generalization and adversarial robustness in experiments.

The norm decay approach is to add a regularization term to the original loss function $\mathcal{L}(\theta)$, where θ is the parameter, to form an augmented loss function:

$$\min_{\theta} \quad \mathcal{L}(\theta) + \frac{\beta}{N} \sum_{i=1}^{N} \|\theta^{(i)}\|_p \tag{5}$$

where $\theta^{(i)}$ denotes the linear transformation matrix of the i-th layer and β is a hyperparameter, and the summation is over all fully-connected layers and convolutional layers.

Form Eq. (1) and Eq. (2), we can see that the ℓ_1 and ℓ_∞ norm depends on only some elements in the kernel, which means the gradient of norm w.r.t. kernel elements ($\nabla_\theta \|\theta^{(i)}\|_p$) are typically sparse. Besides, since the norm is the sum of the absolute values of these elements, the gradient w.r.t. a single kernel element is either 1 or -1 or 0, which makes the computation of gradient very efficient. After updating the kernel parameters using an optimizer such as stochastic gradient descent (SGD), the elements that contribute to the norm may become completely different from those before the update (due to the $\max$ operation in Eq. (1) and Eq. (2)), which could cause non-smoothness (i.e., rapid change) of the gradient $\nabla_\theta \|\theta^{(i)}\|_p$. To smooth the gradient change and stabilize training, we introduce a momentum γ to keep a moving average of the gradient of the norms. The details are shown in Algorithm 1.

Algorithm 1 Norm Decay

```
Input: loss function $\mathcal{L}$ (assuming it is to be minimized),
    parameters $\theta$, momentum $\gamma$, regularization parameter $\beta$
Output: parameters $\theta$
h ← 0 (initialize the gradient of norms of layers)
repeat
    $g \leftarrow \nabla_\theta \mathcal{L}$
    Compute $p$, the gradient of $\ell_1$ or $\ell_\infty$ norm of each
    fully-connected and convolutional layer
    $h \leftarrow \gamma \cdot h + (1 - \gamma) \cdot p$
    $g \leftarrow g + \beta / N \cdot h$
    $\theta \leftarrow \mathrm{SGD}(\theta, g)$
until convergence
```

kernel size	ℓ_2(VS)	ℓ_2(SGL)	ℓ_1(ours)	ℓ_∞(ours)
3, 3, 32, 32	26.5	5.75	0.00605	0.00576
3, 3, 32, 128	27.4	6.92	0.00682	0.00575
3, 3, 128, 256	29.0	98.0	0.00576	0.00560
3, 3, 256, 512	59.4	490	0.0117	0.00898
5, 5, 256, 128	59.7	91.5	0.0103	0.00729
5, 5, 512, 256	255	523	0.0239	0.0180

Table 1: Computation time (seconds) of 100 runs of computing different norms for various kernels. The experimental setup is shown in the next subsection and the computation is run on GPU. The input image has the same shape as a CIFAR-10 image. The kernel size is represented by (kernel height, kernel width, # input channels, # output channels). VS denotes the method of Virmaux and Scaman (2018) and SGL denotes the method of Sedghi, Gupta, and Long (2019).

Experiments

Firstly, we show our approaches for computing norms of Conv2d are very efficient. In the second part, we conduct extensive experiments to investigate if regularizing the norms of CNN layers is effective in improving adversarial robustness. In the third part, we compare the norms of the layers of adversarially robust CNNs against their non-adversarially robust counterparts.

Algorithmic Efficiency Comparison

We compare the efficiency of three methods that can compute the exact norms of convolutional layers, including computing the ℓ_2 norm with power iteration (Virmaux and Scaman 2018) and circulant matrix (Sedghi, Gupta, and Long 2019) and computing the ℓ_1 norm and ℓ_∞ norm with Eq. (1) and Eq. (2). The result is shown in Table 1, which shows that our approaches are much faster (up to 14,000 times faster) than the others, while our approaches are theoretically and empirically equivalent to the others in computing norms.

Regularizing Norms Improves Generalization but Can Hurt Adversarial Robustness

To better understand the effect of regularizing the norm of CNN layers, we conduct experiments with various models on CIFAR-10 (Krizhevsky and Hinton 2009). Specially, we use three approaches, including weight decay (WD), singular value clipping (SVC) (Sedghi, Gupta, and Long 2019), and norm decay (ND), to regularize the norms. Here, we only use the norm-regularization methods that do not change the architecture of the network, and thus exclude the GroupSort (Anil, Lucas, and Grosse 2019) and BCOP (Li et al. 2019). We also exclude the methods that may not regularize the true norms (e.g., reshaping the convolutional kernel into a matrix) such as Parseval Regularization (Cisse et al. 2017) and (Gouk et al. 2018).

Experimental setup. We set the regularization parameter to different values and test generalization and adversarial robustness of the models on test set. In norm decay, we simply set the hyperparameter γ (momentum) to 0.5 and test the other hyperparameter β in $\{10^{-5}, \ldots, 10^{-2}\}$. We

		plain	weight decay				singular value clipping				ℓ_1 norm decay				ℓ_∞ norm decay			
model	ACC	—	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0.5	1.0	1.5	2.0	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
vgg	Clean	90.4	91.6	91.7	90.1	90.2	87.6	89.1	90.0	89.9	88.1	91.1	90.6	90.8	91.8	91.1	90.8	90.6
	Robust	60.2	56.3	60.5	60.6	60.3	48.8	52.2	54.1	56.7	56.5	62.5	61.1	60.1	56.9	60.0	60.8	60.1
resnet	Clean	93.2	94.3	94.1	93.1	92.7	93.6	94.0	94.2	93.8	92.5	93.4	93.5	93.4	93.0	93.8	93.1	93.0
	Robust	37.0	28.2	33.7	33.9	40.9	35.2	41.7	43.2	39.8	24.5	37.7	38.3	37.5	20.0	34.7	38.9	37.6
senet	Clean	93.1	94.2	93.9	93.0	92.4	93.8	94.2	93.8	94.2	92.3	93.8	93.3	93.3	93.0	93.6	92.8	93.2
	Robust	35.7	23.5	32.8	37.0	34.8	30.5	35.6	35.2	37.4	33.6	36.0	38.2	36.7	28.6	31.0	37.6	37.4
regnet	Clean	91.8	93.6	94.4	92.3	91.3	93.9	93.4	93.0	92.4	93.7	92.3	91.6	91.9	93.4	92.0	91.8	91.9
	Robust	34.8	23.7	30.3	30.0	31.0	27.7	28.8	29.0	28.8	29.2	31.1	28.1	34.3	23.2	27.7	27.9	30.6

Table 2: Comparison of clean accuracy (%) and robust accuracy (%) of 4 CNN models trained with different norm-regularization methods on CIFAR-10. The second row corresponds to the values of regularization parameters. Robust accuracy is tested with standard Auto Attack (Croce and Hein 2020) under ℓ_∞ metric with $\epsilon = 1/255$.

also test the regularization parameter of weight decay in $\{10^{-5}, \dots, 10^{-2}\}$ and test SVC by clipping the singular values to $\{2.0, 1.5, 1.0, 0.5\}$, respectively, following the setting in the original paper. We use four CNN architectures in our experiments, including VGG-11 (Simonyan and Zisserman 2015), ResNet-18 (He et al. 2016), SENet-18 (Hu, Shen, and Sun 2018), and RegNetX-200MF (Radosavovic et al. 2020). We use the SGD optimizer with momentum of 0.9 and set the initial learning to 0.01. We train the models for 120 epochs and decay the learning rate by a factor of 0.1 at epoch 75, 90, and 100. After finishing training, we use the state-of-the-art attack "Auto Attack" (Croce and Hein 2020) to attack the trained CNNs. The experiments are conducted on a machine a GTX 1080 Ti GPU and an Intel Core i5-9400F 6-core CPU and 32GB RAM.

The result is shown in Table 2. Since we find that all models trained with WD, SVC, and ND have basically zero robust accuracy under ℓ_∞ attack with $\epsilon = 8/255$ and $\epsilon = 4/255$, we set $\epsilon = 1/255$ to see the actual effect of regularizing norms. Because of that, we first conclude that these regularization methods cannot improve adversarial robustness by reducing norms when facing large attack (in the sense of large ϵ). From Table 2, we can see that the four regularization methods typically improve generalization. However, as the regularization becomes stronger, the norm of all layers becomes smaller (see Appendix for the changes of norms during training) while the robust accuracy could slightly decrease. The reduction in robust accuracy is especially evident when the regularization is the strongest and the norms are the smallest (in the first column of each regularization method in Table 2). This result is very surprising and contradicts the prevailing claim that small norms of CNN layers improve robustness (Szegedy et al. 2014; Cisse et al. 2017; Anil, Lucas, and Grosse 2019; Li et al. 2019). We can see that there seems to be a trade-off between standard (clean) accuracy and robust accuracy. When the clean accuracy gets a higher value, the robust accuracy typically gets a lower value. This trade-off has been pointed out by Tsipras et al. (2019), and they proved that the trade-off is inevitable when the distribution of two different classes is "mixed". However, Yang et al. (2020) have shown that the CIFAR-10 training set and test set are both 2ϵ-separable for ϵ much larger than the typical values used in adversarial attack. Therefore, by Theorem 2, there should exist a neural network that achieves robust accuracy 1 and there should be no *intrinsic* trade-off.

The reason for this phenomenon may be that regularizing the norms in fact suppresses the power of CNNs to become local Lipschitz. From the results in the last section, we know that large norms do not necessarily result in large local Lipschitz constants. Thus, in an unconstrained parameter space (in the case of no regularization) the network may be able to find a minimizer (w.r.t. the loss) that has better local Lipschitzness. When the parameter space is constrained (due to regularization), the network may need to sacrifice local Lipschitzness to retain standard accuracy, which is the training target.

Although the proposed norm decay *may slightly* reduce the adversarial robustness, it still serves as a novel and promising regularizer for CNNs in improving standard generalization.

The Norms of Adversarially Robust Networks

Equipped with our efficient approaches to computing norms of convolutional layers, we further test how the norms of adversarially robust CNNs differ from their non-adversarially robust counterparts. Specifically, we use three adversarial training frameworks, namely, PGD-AT (Madry et al. 2018), ALP (Kannan, Kurakin, and Goodfellow 2018), and TRADES (Zhang et al. 2019) to train the four models, namely, VGG-11, ResNet-18, SENet-18, and RegNetX-200MF. The experimental setting is the same as that in the last subsection except the initial learning rate is set to 0.1 by following the setting of Pang et al. (2020). After finishing training, we compute the ℓ_∞ norms of all layers in the CNNs with/without adversarial training. The result is shown in Figure 2. We can see that the norms of layers of adversarially robust CNNs are comparable or even larger than their non-adversarially robust counterparts (e.g., the adversarially robust ResNet and SENet have especially larger norms while having much higher robust accuracy than the plain models). Due to space limitation, we put the comparison of the norms of individual layers in the supplementary material. These findings consistently show that large norms of CNNs do not hurt adversarial robustness and what really matters is the local Lipschitzness of the networks.

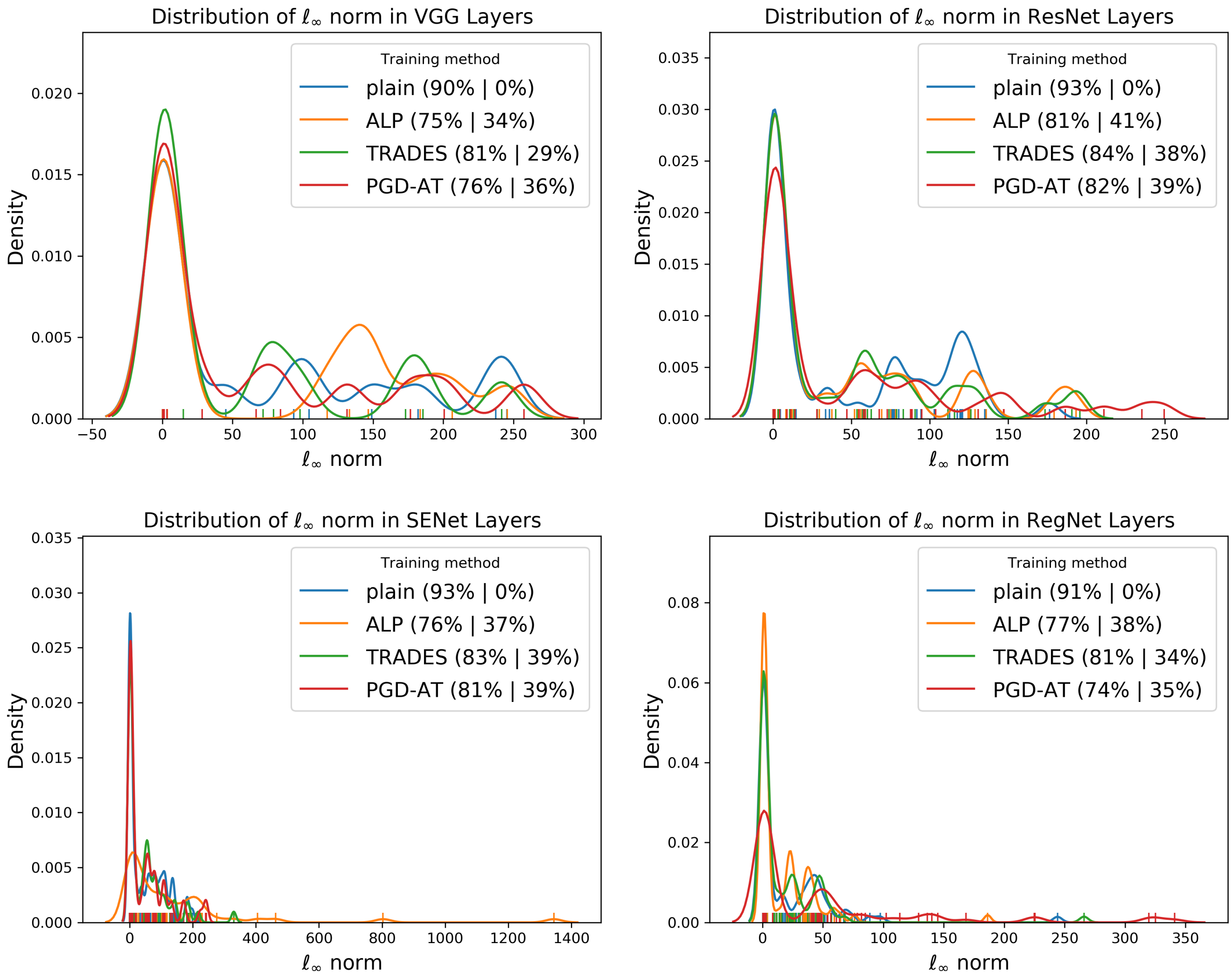


Figure 2: Comparison of the distribution of norms of the layers of four CNN architectures trained with different adversarial training methods on CIFAR-10. The density is fitted using Gaussian kernel density estimation. The small bars on the bottom of the plots indicate the values of the norms. The two numbers beside each training method are the clean accuracy and robust accuracy, respectively. The robust accuracy is evaluated with standard Auto Attack (Croce and Hein 2020) under ℓ_∞ metric with $\epsilon = 8/255$.

Conclusion and Future Work

In this paper, we theoretically characterize the ℓ_1 norm and ℓ_∞ norm of convolutional layers and present efficient approaches for computing the exact norms. Our methods are extremely efficient among the existing methods for computing norms of convolutional layers. We present norm decay, a novel regularization method, which can improve generalization of CNNs. We *prove* that robust classifiers can be realized with neural networks – a piece of encouraging news to the deep learning community.

We theoretically analyze the relationship between global Lipschitzness, local Lipschitzness, and the norms of layers. In particular, we show that large norms of layers do not necessarily lead to a large global Lipschitz constant and a large global Lipschitz constant does not necessarily incur small robust accuracy. In the experiments, we find that regularizing the norms may not improve adversarial robustness and may even slightly hurt adversarial robustness. Moreover, CNNs trained with adversarial training frameworks actually have comparable and even larger layer norms than their non-adversarially robust counterparts, which shows that large norms of layers do not matter. Our theoretical result (Proposition 2) also suggests that imposing local Lipschitzness on neural nets may be an effective approach in adversarial training, which sheds light on future research.

Acknowledgments

This work was supported by the NSFC under Grant 61976097.

References

Akhtar, N.; and Mian, A. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access* 6: 14410–14430.

Anil, C.; Lucas, J.; and Grosse, R. 2019. Sorting out lipschitz function approximation. In *International Conference on Machine Learning*.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* .

Bietti, A.; Mialon, G.; Chen, D.; and Mairal, J. 2019. A kernel perspective for regularizing deep neural networks. In *International Conference on Machine Learning*, 664–674.

Carmon, Y.; Raghunathan, A.; Schmidt, L.; Duchi, J. C.; and Liang, P. S. 2019. Unlabeled data improves adversarial robustness. In *Advances in Neural Information Processing Systems*, 11192–11203.

Cisse, M.; Bojanowski, P.; Grave, E.; Dauphin, Y.; and Usunier, N. 2017. Parseval networks: Improving robustness to adversarial examples. *International Conference on Machine Learning* .

Croce, F.; and Hein, M. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. *arXiv preprint arXiv:2003.01690* .

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1625–1634.

Fazlyab, M.; Robey, A.; Hassani, H.; Morari, M.; and Pappas, G. 2019. Efficient and accurate estimation of lipschitz constants for deep neural networks. In *Advances in Neural Information Processing Systems*, 11423–11434.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep learning*. MIT press.

Gouk, H.; Frank, E.; Pfahringer, B.; and Cree, M. 2018. Regularisation of neural networks by enforcing lipschitz continuity. *arXiv preprint arXiv:1804.04368* .

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Huang, S.; Papernot, N.; Goodfellow, I.; Duan, Y.; and Abbeel, P. 2017. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284* .

Kannan, H.; Kurakin, A.; and Goodfellow, I. 2018. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373* .

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto* .

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553): 436–444.

Li, Q.; Haque, S.; Anil, C.; Lucas, J.; Grosse, R. B.; and Jacobsen, J.-H. 2019. Preventing gradient attenuation in lipschitz constrained convolutional networks. In *Advances in Neural Information Processing Systems*, 15390–15402.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*.

Miyato, T.; Kataoka, T.; Koyama, M.; and Yoshida, Y. 2018. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1765–1773.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2574–2582.

Pang, T.; Yang, X.; Dong, Y.; Xu, K.; Su, H.; and Zhu, J. 2020. Boosting adversarial training with hypersphere embedding. *arXiv preprint arXiv:2002.08619* .

Radosavovic, I.; Kosaraju, R. P.; Girshick, R.; He, K.; and Dollár, P. 2020. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10428–10436.

Sedghi, H.; Gupta, V.; and Long, P. M. 2019. The Singular Values of Convolutional Layers. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=rJevYoA9Fm.

Simonyan, K.; and Zisserman, A. 2015. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.

Sokolić, J.; Giryes, R.; Sapiro, G.; and Rodrigues, M. R. 2017. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing* 65(16): 4265–4280.

Su, J.; Vargas, D. V.; and Sakurai, K. 2019. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation* 23(5): 828–841.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Tsipras, D.; Santurkar, S.; Engstrom, L.; Turner, A.; and Madry, A. 2019. Robustness may be at odds with accuracy. In *International Conference on Learning Representations*.

Virmaux, A.; and Scaman, K. 2018. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In

Advances in Neural Information Processing Systems, 3835–3844.

Wang, Y.; Jha, S.; and Chaudhuri, K. 2018. Analyzing the robustness of nearest neighbors to adversarial examples. In *International Conference on Machine Learning*, 5133–5142.

Wang, Y.; Zou, D.; Yi, J.; Bailey, J.; Ma, X.; and Gu, Q. 2019. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*.

Weng, T.-W.; Zhang, H.; Chen, P.-Y.; Yi, J.; Su, D.; Gao, Y.; Hsieh, C.-J.; and Daniel, L. 2018. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*.

Yang, Y.-Y.; Rashtchian, C.; Zhang, H.; Salakhutdinov, R. R.; and Chaudhuri, K. 2020. A Closer Look at Accuracy vs. Robustness. In *Advances in Neural Information Processing Systems*, volume 33, 8588–8601. URL https://proceedings.neurips.cc/paper/2020/file/61d77652c97ef636343742fc3dcf3ba9-Paper.pdf.

Zhang, H.; Yu, Y.; Jiao, J.; Xing, E. P.; Ghaoui, L. E.; and Jordan, M. I. 2019. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*.