Compositional Curvature Bounds for Deep Neural Networks

Taha Entesari [*1] **Sina Sharifi** [*1] **Mahyar Fazlyab** [1]

Abstract

A key challenge that threatens the widespread use of neural networks in safety-critical applications is their vulnerability to adversarial attacks. In this paper, we study the second-order behavior of continuously differentiable deep neural networks, focusing on robustness against adversarial perturbations. First, we provide a theoretical analysis of robustness and attack certificates for deep classifiers by leveraging local gradients and upper bounds on the second derivative (curvature constant). Next, we introduce a novel algorithm to analytically compute provable upper bounds on the second derivative of neural networks. This algorithm leverages the compositional structure of the model to propagate the curvature bound layer-by-layer, giving rise to a scalable and modular approach. The proposed bound can serve as a differentiable regularizer to control the curvature of neural networks during training, thereby enhancing robustness. Finally, we demonstrate the efficacy of our method on classification tasks using the MNIST and CIFAR-10 datasets.

1. Introduction

Neural networks are infamously prone to adversarially designed perturbations (Szegedy et al., 2013). To address this vulnerability, many methods have been proposed to quantify and improve the robustness of these models against adversarial attacks, such as adversarial training (Zhang et al., 2019; Madry et al., 2018), regularization (Leino et al., 2021; Tsuzuku et al., 2018), randomized smoothing (Cohen et al., 2019; Kumar et al., 2021), and many others. One measure of robustness is the Lipschitz constant defined as the smallest $L_f \geq 0$ such that

$$\|f(x) - f(y)\| \leq L_f \|x - y\| \quad \forall x, y.$$

This constant quantifies the sensitivity of the model f to input perturbations, motivating the need to estimate L_f and control it through architecture or the training process.

For continuously differentiable functions, L_f is a tight upper bound on the *first* derivative ($\|Df(x)\| \leq L_f \ \forall x$). However, one can go one step further and leverage the smoothness of the first derivative, i.e., bounds on the *second* derivative to obtain a more refined measure of the function's sensitivity. Indeed, the merit of second-order information in characterizing and enhancing robustness has been established, e.g., (Singla & Feizi, 2020).

Our Contributions: In this work, we seek to characterize the adversarial robustness of continuously differentiable neural network classifiers through the Lipschitz constant of their first derivative defined as

$$\|Df(x) - Df(y)\| \leq L_{Df} \|x - y\| \quad \forall x, y$$

If f is twice differentiable, this constant is a tight upper bound on the second derivative, $\|D^2 f(x)\| \leq L_{Df} \ \forall x$. With a slight abuse of the formal definition, we denote L_{Df} as the "curvature" constant. Our contributions are as follows.

- We provide a theoretical analysis of the interplay between adversarial robustness and smoothness. Specifically, for classification tasks, we derive lower bounds on the margin of correctly classified data points using the first derivative (the Jacobian) and its Lipschitz constant (the curvature constant). We then show that these curvature-based certificates provably improve upon Lipschitz-based certificates, provided the curvature is sufficiently small.
- We propose a novel algorithm to derive analytical upper bounds on the curvature constant of neural networks. This algorithm leverages the compositional structure of the model to compute the bound in a scalable and modular fashion, improving upon previous works that only consider scalar-valued networks with affine-then-activation architectures. The derived bound is differentiable and can be used as a regularizer for training low-curvature neural networks.

[*]Equal contribution [1]Department of Electrical and Computer Engineering, Johns Hopkins University, Baltimore, United States of America. Correspondence to: Mahyar Fazlyab <mahyarfazlyab@jhu.edu>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

- We introduce a relaxed notion of smoothness, the *Anchored Lipschitz* constant, which significantly reduces conservatism in terms of robustness certification. Succinctly, this definition fixes one of the two points involved in the definition of Lipschitz continuity to the point of interest.
- We also present empirical results demonstrating the performance of our method compared to previous works on calculating curvature bounds, and we examine the impact of low curvatures on the robustness of deep classifiers.

To the best of our knowledge, this paper is the first to develop a method for obtaining provable bounds on the second derivative of *general* sequential neural networks. While we consider adversarial robustness as an application domain, the proposed method is also of independent interest for other applications requiring differentiable bounds on the second derivative of neural networks, such as learning-based control for safety-critical applications (Robey et al., 2020).

1.1. Related Work

With respect to the large body of work in this field, here, we focus on the works that are more relevant to our setup.

Adversarial Robustness: The robustness of deep models against adversarial perturbations has been a topic of interest in recent years (Singla & Feizi, 2021; 2022; Xu et al., 2022; Zou et al., 2023). (Huang et al., 2021; Fazlyab et al., 2023) use the Lipschitz constant of the network during the training procedure to induce robustness by bounding the worst-case logits. To achieve robustness, instead of penalizing or constraining the Lipschitz constant during training, some methods directly construct 1-Lipschitz networks. The use of Lipschitz bounded networks has been encouraged by many recent works (Béthune et al., 2022) as they provide desirable properties such as robustness and improved generalization. AOL (Prach & Lampert, 2022) provides a rescaling of the layer weights that makes each linear layer 1-Lipschitz. To obtain Lipschitz bounded networks, many works have utilized LipSDP (Fazlyab et al., 2019) to parameterize 1-Lipschitz layers. SLL (Araujo et al., 2022) proposes 1-Lipschitz residual layers by satisfying LipSDP, and (Fazlyab et al., 2023) generalizes SLL by proposing a $\sqrt{\rho}$-Lipschitz layer. Most recently, (Wang & Manchester, 2023) satisfies the LipSDP condition using Caley Transforms and proposes a non-residual 1-Lipschitz layer.

Other works look beyond the network's first-order properties and control the network's curvature (Moosavi-Dezfooli et al., 2019; Singla et al., 2021). (Srinivas et al., 2022) proposes using centered-soft plus activations and Lipschitz-bounded batch normalizations to cap the curvature and empirically improve robustness.

Lipschitz Constant Calculation: In recent years, there has been a focus on finding accurate bounds on the Lipschitz constant of neural networks. Here we only discuss the ones that can handle continuously-differentiable networks. One of the early works, (Szegedy et al., 2013), provided a bound on the Lipschitz constant using the norm of each layer, which is known to be a loose bound. (Fazlyab et al., 2019) formulated the problem of finding the Lipschitz constant as a semidefinite program (SDP), providing accurate bounds but at the expense of limited scalability. Later, (Hashemi et al., 2021) introduced a local version of LipSDP. Most recently, (Fazlyab et al., 2023) proposed LipLT, an analytic method for bounding the Lipschitz constant through loop transformation, a control-theoretic concept. In this work, we also leverage LipLT to derive upper bounds on the curvature constant.

Most relevant to our setup, (Singla & Feizi, 2020) develops a method to bound the curvature constant of scalar-valued neural networks in the ℓ_2 norm and introduces a numerical optimization scheme to provide curvature-based certificates. In contrast, our method bounds the curvature of arbitrary function compositions, in particular vector-valued feedforward neural networks, in any ℓ_p norm, and provides analytical curvature-based certificates.

1.2. Preliminaries and Notation

We denote the n-dimensional real numbers as $\mathbb{R}^n$. For a vector $x \in \mathbb{R}^n$, x_i is its i-th element. For a matrix $W \in \mathbb{R}^{n\times m}$, $W_{i,:} \in \mathbb{R}^{1\times m}, W_{:,j} \in \mathbb{R}^n, W_{i,j} \in \mathbb{R}$ are the i-th row, j-th column, and the j-th element of $W_{i,:}$, respectively. For a vector x, $\mathrm{diag}(x)$ is the diagonal matrix with $\mathrm{diag}(x)_{ii} = x_i$ and zero otherwise. For an integer n let $[n] = \{1, \cdots, n\}$. Moreover, the operator norm of a matrix A is denoted as $\|A\|_{p\to q} = \sup_{\|x\|_p\leq 1} \|Ax\|_q$. For a real-valued $p \geq 1$, we denote its Hölder conjugate with p^*, i.e., $\frac{1}{p} + \frac{1}{p^*} = 1$. For any vector $x \in \mathbb{R}^n$ and norm $\|\cdot\|_p$, we have $\|x\|_{p^*} = \sup_{\|y\|_p\leq 1} x^\top y$.

A function $f : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz continuous on $\mathcal{C} \subseteq \mathbb{R}^n$ if there exists a non-negative constant $L_f^{p,q}$ such that $\|f(x) - f(y)\|_q \leq L_f^{p,q}\|x - y\|_p \; \forall x, y \in \mathcal{C}$. The smallest such $L_f^{p,q}$ is *the* Lipschitz constant, in the corresponding norms, which is given by

$$L_f^{p,q} = \sup_{x,y\in\mathcal{C}, x\neq y} \frac{\|f(x) - f(y)\|_q}{\|x - y\|_p}.$$

For brevity, we denote $L_f^{p,p}$ as L_f^p. In this work, we define a new notion of Lipschitz continuity at a neighborhood of a point.

Definition 1.1 (Anchored Lipschitz constant)**.** For a function f, the *anchored* Lipschitz constant at a point $x \in \mathcal{C}$ is

defined as

$$L_f^{p,q}(x) = \sup_{y \in \mathcal{C}, x \neq y} \frac{\|f(x) - f(y)\|_q}{\|x - y\|_p}.$$

At any point x, this constant is a lower bound on the Lipschitz constant as one can confirm $L_f^{p,q} = \sup_{x \in \mathcal{C}} L_f^{p,q}(x)$. Figure 1 demonstrates this concept further for the specific case of the $\tanh$ function.

In the following lemma, we establish the relation between the anchored Lipschitz constant and the norm of the derivative.

Lemma 1.2. Consider a differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ and let $L_f^p(x)$ be a corresponding anchored Lipschitz constant. We have

$$\|Df(x)\|_p \leq L_f^p(x).$$

See Appendix A for the proof.

Given bounded numbers $\alpha \leq \beta$, a function $\phi : \mathbb{R} \to \mathbb{R}$ is slope restricted in $[\alpha, \beta]$ if

$$\alpha \leq \frac{\phi(x) - \phi(y)}{x - y} \leq \beta, \quad \forall x, y.$$

The Lipschitz constant of ϕ is then $L_\phi = \max(|\alpha|, |\beta|)$. For simplicity, and based on commonly-used differentiable activation functions such as sigmoid and tanh, we assume that ϕ is monotone, i.e., $\alpha \geq 0$, implying that $L_\phi = \beta$.

2. Curvature-based Robustness Analysis

Consider a continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}^m$ parameterized by a neural network. In this work, our goal is to derive provable upper bounds on the Lipschitz constant of the Jacobian $Df : \mathbb{R}^n \to \mathbb{R}^{m \times n}$, defined as the smallest constant $L_{Df}^{p,q}$ such that

$$\|Df(x_1) - Df(x_2)\|_q \leq L_{Df}^{p,q} \|x_1 - x_2\|_p, \quad \forall x_1, x_2.$$

Furthermore, we can extend this to the anchored Lipschitz constant of the Jacobian, $L_{Df}^{p,q}(x)$, at a given point x, as

$$\|Df(x + \delta) - Df(x)\|_q \leq L_{Df}^{p,q}(x) \|\delta\|_p, \quad \forall \delta.$$

While providing provable upper bounds on these constants can be instrumental in various applications, in this work, we primarily focus on the adversarial robustness of deep classifiers. We develop our methods and certificates based on the Lipschitz continuity of the classifier and its Jacobian. We elaborate more on this in the following subsections.

2.1. Robustness Certificates for Deep Classifiers

Consider a classifier $C(x) := \arg\max_{1 \leq i \leq n_K} f_i(x)$ with n_K classes, where $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_K}$ is a neural network that parameterizes the vector of logits. For a given input x with correct label $y \in [n_K]$, the condition for correct classification is

$$f_{iy}(x) := f_i(x) - f_y(x) < 0, \quad \forall i \neq y.$$

Assuming that x is correctly classified as y, the distance of x to the closest decision boundary measures the classifier's local robustness against additive perturbations. We can compute this distance by solving the following optimization problem,

$$\begin{aligned} \varepsilon^*(x) = \max \quad & \varepsilon \\ \text{s.t.} \quad & \sup_{\|\delta\|_p \leq \varepsilon} f_{iy}(x + \delta) \leq 0, \quad \forall i \neq y. \end{aligned} \tag{1}$$

For ReLU networks and $p \in \{1, \infty\}$, this optimization problem can be encoded as a Mixed-Integer Linear program (MILP) by exploiting the piece-wise nature of the activation functions (Dutta et al., 2018; Fischetti & Jo, 2018; Tjeng et al., 2018). While these MILPs can be solved globally, they suffer from poor scalability. For neural networks with differentiable activation functions, even this mixed-integer structure is absent, making the exact computation of distances effectively intractable. Therefore, we must resort to finding lower bounds on the certified radius to gain tractability.

2.1.1. Lipschitz-based certificates

Suppose the f_{iy}'s are Lipschitz continuous. We can then write

$$f_{iy}(x + \delta) \leq f_{iy}(x) + L_{f_{iy}}^p(x) \|\delta\|_p. \tag{2}$$

where $L_{f_{iy}}^p(x) > 0$ is the *anchored* Lipschitz constant of f_{iy}. By substituting (2) in the constraints of (1), we obtain the following optimization problem to compute a zeroth-order (gradient-free) lower bound,

$$\begin{aligned} \underline{\varepsilon}_0^*(x) = \max \ & \varepsilon \\ \text{s.t.} \ & \sup_{\|\delta\|_p \leq \varepsilon} f_{iy}(x) + L_{f_{iy}}^p(x) \|\delta\|_p \leq 0, \forall i \neq y. \end{aligned}$$

We note that due to constraint tightening, we have $\underline{\varepsilon}_0^*(x) < \varepsilon^*(x)$. Using similar arguments as in (Fazlyab et al., 2023), $\underline{\varepsilon}_0^*(x)$ has the closed-form expression

$$\underline{\varepsilon}_0^*(x) = \min_{i \neq y} \frac{-f_{iy}(x)}{L_{f_{iy}}^p(x)}. \tag{3}$$

2.1.2. Curvature-based certificates

When the model is continuously differentiable, we can exploit its curvature to improve the certificate in (3). Specif-

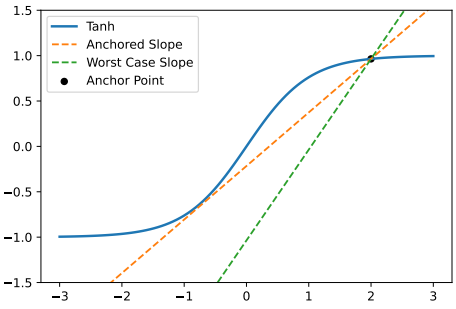

Figure 1: Depiction of anchored Lipschitz constants for $f(x) = \tanh(x)$. The anchored Lipschitz constant at $x = 2$ is less than 0.582, whereas the global Lipschitz constant is 1.

ically, suppose the logit difference f_{iy} is continuously differentiable with Lipschitz gradients and let $L^{p,p^*}_{\nabla f_{iy}}(x)$ be an anchored Lipschitz constant of ∇f_{iy} at x. Then, we can compute an upper bound on $f_{iy}(x+\delta)$ as follows,

$$f_{iy}(x+\delta) \leq \underbrace{f_{iy}(x) + \nabla f_{iy}(x)^\top \delta + \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\|\delta\|_p^2}_{\overline{f_{iy}}(x,\delta;L^{p,p^*}_{\nabla f_{iy}}(x))}. \quad (4)$$

See Appendix A for a derivation of this inequality. In contrast to the zeroth-order bound, this upper bound uses the local first derivative, $\nabla f_{iy}(x)$, as well as bounds on its (anchored) Lipschitz constant, $L^{p,p^*}_{\nabla f_{iy}}(x)$, to obtain a locally more accurate approximation of $f_{iy}(x+\delta)$. By substituting the upper bound (4) in (1), we obtain a first-order (gradient-informed) lower bound on $\varepsilon^*(x)$,

$$\begin{aligned} \underline{\varepsilon}^*_1(x) = \max\ & \varepsilon \\ \text{s.t.}\ & \sup_{\|\delta\|_p \leq \varepsilon} \overline{f_{iy}}(x,\delta;L^{p,p^*}_{\nabla f_{iy}}(x)) \leq 0, \quad \forall i \neq y \end{aligned} \quad (5)$$

As we summarize below, we can compute this lower bound in closed form, provided that we can compute $L^{p,p^*}_{\nabla f_{iy}}(x)$.

Proposition 2.1 (Curvature-based certified radius)**.** Suppose x is classified correctly, i.e., $f_{iy}(x) < 0\ \forall i \neq y$. The optimization problem (5) has the closed-form solution $\underline{\varepsilon}^*_1(x)$ given by

$$\min_{i \neq y} \frac{-\|\nabla f_{iy}(x)\|_{p^*} + (\|\nabla f_{iy}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{iy}}(x) f_{iy}(x))^{\frac{1}{2}}}{L^{p,p^*}_{\nabla f_{iy}}(x)}. \quad (6)$$

See Appendix A for the proof of this proposition.

In the following proposition, we show that if the curvature of the model is sufficiently small, we can certify a larger radius than Lipschitz-based certificates.

Proposition 2.2. Suppose x is classified correctly, i.e., $f_{iy}(x) < 0\ \forall i \neq y$. Fix a $p \geq 1$, and define the zeroth-order $\underline{\varepsilon}^*_0(x)$ and first-order $\underline{\varepsilon}^*_1(x)$ certified radii as in (3) and (6). If the following condition holds,

$$L^{p,p^*}_{\nabla f_{iy}}(x) \leq \frac{-2(\|\nabla f_{iy}(x)\|_{p^*}\underline{\varepsilon}^*_0(x) + f_{iy}(x))}{\underline{\varepsilon}^*_0(x)^2}, \quad i \neq y.$$

Then $\underline{\varepsilon}^*_1(x) \geq \underline{\varepsilon}^*_0(x)$.

See Appendix A for the proof.

2.2. Attack Certificates for Deep Classifiers

Considering the same setup as before, we now aim to obtain the smallest perturbation by which a correctly classified data point can provably be misclassified. This computation can be formulated as the following optimization problem,

$$\begin{aligned} \varepsilon'^*(x) = \min\ & \varepsilon \\ \text{s.t.}\ & \min_{i \neq y} \inf_{\|\delta\|_p \leq \varepsilon} f_{yi}(x+\delta) < 0. \end{aligned} \quad (7)$$

First, we note that problems (1) and (7) are equivalent.

Proposition 2.3. Suppose f correctly classifies the data point x as y, i.e., $f_{iy}(x) < 0$ for $i \neq y$. Then the optimal value of problems (1) and (7) are equal, i.e., $\varepsilon^*(x) = \varepsilon'^*(x)$.

Using the curvature-based upper bound, one can tighten the constraints of the problem and achieve a first-order (gradient-informed) attack certificate as follows,

$$\begin{aligned} \overline{\varepsilon}^*_1(x) = \min\ & \varepsilon \\ \text{s.t.}\ & \min_{i \neq y} \inf_{\|\delta\|_p \leq \varepsilon} \overline{f_{yi}}(x,\delta;L^{p,p^*}_{\nabla f_{yi}}(x)) < 0, \end{aligned} \quad (8)$$

We analytically acquire the optimal value of this problem in the following proposition.

Proposition 2.4 (Curvature-based attack certificate)**.** Suppose x is classified correctly, i.e., $f_{iy}(x) < 0\ \forall i \neq y$. Let $\mathcal{I} = \{i | i \neq y, 2L^{p,p^*}_{\nabla f_{yi}}(x) f_{yi}(x) \leq \|\nabla f_{yi}(x)\|^2_{p^*}\}$. Assuming that $\mathcal{I}$ is non-empty, the optimization problem (8) has the closed-form solution $\overline{\varepsilon}^*_1(x)$ given by

$$\min_{i \in \mathcal{I}} \frac{\|\nabla f_{yi}(x)\|_{p^*} - (\|\nabla f_{yi}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{yi}}(x) f_{yi}(x))^{\frac{1}{2}}}{L^{p,p^*}_{\nabla f_{yi}}(x)}. \quad (9)$$

Given $i^* \in \mathcal{I}$ minimizing (9), the perturbation realizing the attack certificate is obtained through solving $\sup_{\|\delta\|_p \leq \varepsilon} \nabla f_{i^* y}(x)^\top \delta$.

We note that while problem (7) is always feasible (for a non-trivial classifier), problem (8) can be infeasible due to

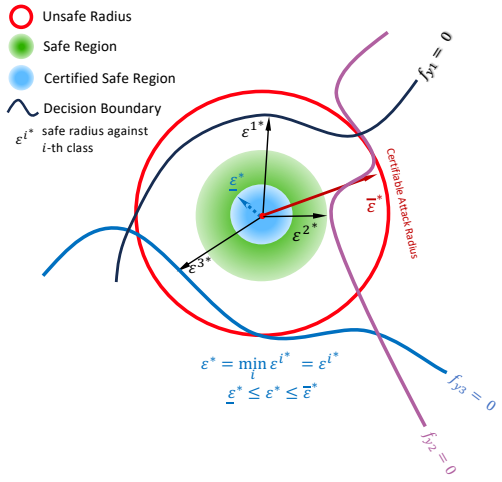

Figure 2: Certified ($\underline{\varepsilon}^*$) and attack ($\bar{\varepsilon}^*$) radii estimates. The green tangent circle denotes the certified radius ε^*.

the tightening of the constraints, which is equivalent to the set $\mathcal{I}$ being empty. Figure 2 illustrates an example scenario for $\underline{\varepsilon}^*$ and $\bar{\varepsilon}^*$.

The derivation of $\underline{\varepsilon}^*(x)$ and $\bar{\varepsilon}^*(x)$ is significant in two ways. First, we established an analytical solution to the curvature-based certified radius in Proposition 2.1. Although curvature-based certificates have been studied in (Singla & Feizi, 2020), their method involves an iterative algorithm to solve an optimization problem numerically, whereas our method yields closed-form solutions. Second, we introduced curvature-based attack certificates, a novel method for narrowing the *certification gap* of classifiers. The *certification gap* is the (empirical) probability quantifying correctly classified points that lack the desired level of certified defense radii and lack attack certificates at a given perturbation budget ϵ, i.e., $\mathbb{P}_{(x,y)\sim D}\{\underline{\varepsilon}(x) \leq \epsilon \leq \bar{\varepsilon}(x)\}$. Refer to Figure 6 for an illustration.

3. Efficient Estimation of Curvature Bounds

Having established the importance of curvature in providing robustness and attack certificates, in this section, we propose our method to derive upper bounds on the Lipschitz constant of the Jacobian (the curvature constant) of general sequential models. We then curate our algorithm for residual neural networks and explore various Lipschitz estimation techniques to calculate the curvature constant.

3.1. Curvature Bounds for Composed Functions

Let $h = f \circ g$ be the composition of two continuously differentiable functions $g : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$ and $f : \mathbb{R}^{n_2} \to \mathbb{R}^{n_3}$ with Lipschitz Jacobians. The Lipschitz constant of the Jacobian $Dh \in \mathbb{R}^{n_3 \times n_1}$ of h is an upper bound on the *second derivative* of h, assuming it exists, which is a third-order tensor (Srinivas et al., 2022) and difficult to characterize. Our goal is to compute the Lipschitz constant of Dh directly without resorting to any tensor calculus.

Using the chain rule, the Jacobian of h can be written as

$$Dh(x) = Df(g(x))Dg(x).$$

The following theorem establishes a relation between the Lipschitz constant of Dh and the Lipschitz constants of f, g, Df, and Dg.

Theorem 3.1 (Compositional curvature estimation)**.** Given functions f, g, and h as described above, the following inequality holds,

$$L_{Dh}^{p,p^*} \leq L_{Dg}^{p,p^*} L_f^{p^*} + L_{Df}^{p,p^*} L_g^p L_g^{p^*}, \tag{10}$$

where $L_s^{p,q}$ denotes the Lipschitz constant of the function $s(\cdot)$.

Theorem 3.1 provides a basis to recursively calculate a Lipschitz constant for the Jacobian of the composition of multiple functions. In the following, we adapt this result to anchored Lipschitz constants.

Theorem 3.2 (Anchored compositional curvature estimation)**.** Consider functions f, g, and h as in Theorem 3.1. The following inequality holds for the anchored Lipschitz constant of the Jacobian of h at x

$$L_{Dh}^{p,p^*}(x) \leq L_f^{p^*} L_{Dg}^{p,p^*}(x) + \|Dg(x)\|_{p^*} L_{Df}^{p,p^*}(g(x)) L_g^p(x),$$

where $L_s^{p,q}(x)$ denotes the anchored Lipschitz constant of the function $s(\cdot)$ at x.

The structure of Theorem 3.1 (and similarly Theorem 3.2) is of particular interest for sequential neural networks that are the composition of individual layers. In the following section, we will instantiate our framework for such models.

Remark 3.3. We note that in Theorems 3.1 and 3.2, the dual norm p^* is chosen to tailor the bounds specifically for Proposition 2.1 and Proposition 2.4. In general, the same statements hold if we replace p^* with a general $q \geq 1$. See Appendix A for more details.

3.2. Curvature Bounds for Sequential Neural Networks

Consider a sequential residual neural network

$$x^{k+1} = h^k(x^k) = H^k x^k + G^k \Phi(W^k x^k), \tag{11}$$

where $k = 0, 1, \cdots K-1$ and $W^k \in \mathbb{R}^{n_k' \times n_k}, G^k \in \mathbb{R}^{n_{k+1} \times n_k'}$, and $H^k \in \mathbb{R}^{n_{k+1} \times n_k}$ are general matrices. For $x \in \mathbb{R}^n$, $\Phi(x) = [\phi(x_1), \cdots, \phi(x_n)]^\top$, where ϕ is a differentiable monotone activation function slope-restricted in

$[\alpha, \beta]$ $(0 \leq \alpha \leq \beta < \infty)$ with its derivative slope-restricted in $[\alpha', \beta']$ $(-\infty < \alpha' \leq \beta' < \infty)$. By setting $H^k = 0$ and $G^k = I$, we obtain the standard feedforward architecture.

Leveraging Theorem 3.1, we propose a recursive algorithm to compute an upper bound on the Jacobian of the end-to-end map $x^0 \mapsto x^K$. We establish this algorithm in Corollary 3.4.

Corollary 3.4. Let $\overline{L}_{D_k}^{p,p^*}$, $k = 0, \cdots, K-1$ be defined recursively as

$$\overline{L}_{D_{k+1}}^{p,p^*} = \overline{L}_{Dh^k}^{p,p^*} \overline{L}_k^p \overline{L}_k^{p^*} + \overline{L}_{h^k}^{p^*} \overline{L}_{D_k}^{p,p^*}, \tag{12}$$

with $\overline{L}_{D_0}^{p,p^*} = 0$, $\overline{L}_0^p = \overline{L}_0^{p^*} = 1$, where $\overline{L}_k^p, \overline{L}_k^{p^*}$ are Lipschitz constants for the map $x^0 \mapsto x^k$, and $\overline{L}_{Dh^k}^{p,p^*}$ is a Lipschitz constant for the Jacobian of h^k. Then $\overline{L}_{D_k}^{p,p^*}$ is a Lipschitz constant for the Jacobian of the map $x^0 \mapsto x^k$.

Given upper bounds on the Lipschitz constants as $\overline{L}_k^p$, $\overline{L}_k^{p^*}$, $\overline{L}_{h^k}^{p^*}$, and $\overline{L}_{Dh^k}^{p,p^*}$, Corollary 3.4 presents an algorithm to calculate an upper bound on the curvature constant of residual neural networks in a layer-by-layer fashion.

Next, we will compute the individual constants appearing in (12).

3.2.1. Computation of $\overline{L}_{h^k}^p$

$L_{h^k}^p$ is the Lipschitz constant of the k-th layer h^k. Starting from (11), an analytical upper bound on this constant is

$$\overline{L}_{h^k}^{p,\text{naive}} = \|H^k\|_p + \beta \|G^k\|_p \|W^k\|_p. \tag{13}$$

This bound is relatively crude as it does not exploit the monotonicity of the activations, i.e., (13) is agnostic to the value of α. As proposed in (Fazlyab et al., 2023), this bound can be improved by applying a loop transformation on the activation layer Φ. Specifically, we can rewrite h^k as

$$h^k(x^k) = \hat{H}^k x^k + G^k \Psi(W^k x^k), \tag{14}$$

where $\hat{H}^k = H^k + \frac{\alpha+\beta}{2} G^k W^k$ and $\psi(z) = \phi(z) - (\alpha + \beta)z/2$ is the loop transformed activation layer. As a result of this transformation, Ψ is now slope-restricted in $\frac{\beta-\alpha}{2}[-1, 1]$, implying that ψ is $(\frac{\beta-\alpha}{2})$-Lipschitz. An upper bound on the Lipschitz constant of h^k, reformulated as in (14), is then

$$\overline{L}_{h^k}^{p,\text{LT}} = \|H^k + \frac{\alpha+\beta}{2} G^k W^k\|_p + \frac{\beta-\alpha}{2} \|G^k\|_p \|W^k\|_p. \tag{15}$$

As shown in (Fazlyab et al., 2023), this bound, now informed by the monotonicity constant α, is provably better than (13). This can be proved by applying the triangle inequality on the first term.

3.2.2. Computation of $\overline{L}_k^p$.

L_k^p is the Lipschitz constant of the map $x^0 \mapsto x^k$ defined by the composed function $(h^{k-1} \circ \cdots \circ h^0)(x^0)$. A naive bound on L_k^p is the product of the Lipschitz constant of individual layers, i.e., $\overline{L}_k^{p,\text{naive}} = \prod_{i=0}^{k-1} L_{h^i}^p$, where we can upper bound each $L_{h^i}^p$ from (15). However, this bound can grow quickly as the depth increases. To mitigate the adverse effect of depth, we exploit the idea of LipLT. Specifically, we can *unroll* (11) after applying loop transformation to all activation layers, resulting in

$$x^{k+1} = \hat{H}^k \cdots \hat{H}^0 x^0 + \sum_{j=0}^{k} \hat{H}^k \cdots \hat{H}^{j+1} G^j \Psi(W^j x^j).$$

This representation enables us to obtain all the constants $\overline{L}_1^{p,\text{LT}}, \cdots, \overline{L}_K^{p,\text{LT}}$ recursively as follows,

$$\begin{aligned} \overline{L}_{k+1}^{p,\text{LT}} &= \|\hat{H}^k \cdots \hat{H}^0\|_p \\ &\quad + \tfrac{\beta-\alpha}{2} \sum_{j=0}^{k} \|\hat{H}^k \cdots \hat{H}^{j+1} G^j\|_p \|W^j\|_p \overline{L}_j^{p,\text{LT}}. \end{aligned} \tag{16}$$

As shown in (Fazlyab et al., 2023), this bound provably improves the naive bound obtained by the product of Lipschitz constants of individual layers, i.e., $\overline{L}_k^{p,\text{LT}} \leq \prod_{i=0}^{k-1} \overline{L}_{h^i}^{p,\text{naive}}$.

3.2.3. Computation of $\overline{L}_{Dh^k}^{p,p^*}$.

Consider the k-th residual block h^k in (11). The Jacobian of this block is given as

$$Dh^k(x^k) = H^k + G^k \operatorname{diag}(\Phi'(W^k x^k)) W^k. \tag{17}$$

The following proposition provides an upper bound on the Lipschitz constant of this differential operator.

Proposition 3.5. The Jacobian Dh^k defined in (17) is Lipschitz continuous with $\overline{L}_{Dh^k}^{p,p^*}$ being an upper bound on the Lipschitz constant, where

$$\overline{L}_{Dh^k}^{p,p^*} = L_{\phi'} \|G^k\|_{p^*} \|W^k\|_{p^*} \|W^k\|_{p \to \infty},$$

where $L_{\phi'} = \max\{|\alpha'|, |\beta'|\}$.

It is worth mentioning that the upper bound in Proposition 3.5 is tractable and can be calculated efficiently. In particular, for $p = 2$, the matrix norms $\|G^k\|_{p^*}$ and $\|W^k\|_{p^*}$ can be calculated via the power iteration for fully connected and convolutional layers. Furthermore, $\|W^k\|_{p \to \infty}$ is simply the maximum row ℓ_p norm, which is straightforward for fully connected layers and also convolutional layers with respect to the repetitive structure of their Toeplitz matrices. See the proof in Appendix A for more details.

For the choice $p = p^* = 2$, we propose an alternative approach to acquire better Lipschitz estimates for Dh^k. To

this end, we propose to rewrite Dh^k as a standard network block.

Lemma 3.6 (Vectorized Jacobian)**.** The Jacobian matrix in (17) can be rewritten as a standard neural network layer

$$dh^k(x^k) := \text{vec}(Dh^k(x^k)) = b^k + A^k\Phi'(W^k x^k),$$

where $dh^k(x^k) \in \mathbb{R}^{\hat{n}_k}$ with $\hat{n}_k = n_{k+1} \times n_k$. For all $i \in [n_{k+1}], j \in [n_k]$, and $l \in [n'_k]$ let $m = (j-1)\times n_{k+1}+i$. Then $A \in \mathbb{R}^{\hat{n}_k \times n'_k}$ and $b^k \in \mathbb{R}^{\hat{n}_k}$ are given by $b^k_m = H^k_{ij}$, $A^k_{ml} = G^k_{il}W^k_{lj}$.

The following lemma establishes the relation between the Lipschitz constant of the Jacobian matrix ($L^p_{Dh^k}$) and its vectorized representation ($L^p_{dh^k}$) when $p = 2$.

Lemma 3.7. Let $p = 2$, and suppose $L^p_{dh^k}$ is the Lipschitz constant of the vectorized Jacobian function $x^k \mapsto dh^k(x^k)$ defined in Lemma 3.6. Then $L^p_{dh^k}$ is a valid Lipschitz constant for the Jacobian function $x^k \mapsto Dh^k(x^k)$.

We can improve the Lipschitz bound provided in Proposition 3.5 using the previous lemmas.

Theorem 3.8. For $p = 2$, $\overline{L}^p_{dh^k} = L_{\phi'}\|A^k\|_2\|W^k\|_2$ is a Lipcshitz constant for the Jaocbian matrix Dh^k. Furthermore, $\overline{L}^p_{dh^k} \leq \overline{L}^p_{Dh^k}$.

Leveraging the vectorized representation dh^k of the Jacobian Dh^k, we can utilize more advanced techniques for Lipschitz estimation such as LipSDP to further reduce the conservatism. Specifically, for non-residual building blocks, i.e., when $H^k = 0$ and $G^k = I$, we can extract a feasible solution (optimal when $\alpha' = -\beta'$) to the LipSDP formulation for $dh^k(x)$.

Theorem 3.9. Let $h^k(x) = \Phi(W^k x)$. Define $\overline{L}^{2,\text{SDP}}_{dh^k} = L_{\phi'}\|TW^k\|_2$, where T is a diagonal matrix with $T_{ii} = \|W^k_{i,:}\|_2$. Then $\overline{L}^{2,\text{SDP}}_{dh^k}$ is a valid Lipschitz constant for dh^k in ℓ_2 norm.

3.2.4. Summary of Algorithm

It now remains to combine all the components developed thus far to obtain upper bounds on the curvature of the whole network. This is summarized in Algorithm 1. First, we use LipLT to calculate the Lipschitz constants of the individual layers ($\overline{L}^{p^*}_{h^k}$) and the subnetworks ($\overline{L}^p_k$ and $\overline{L}^{p^*}_k$). Then, using Proposition 3.5, we provide an upper bound on $L^{p,p^*}_{Dh^k}$. Finally, we calculate $\overline{L}^{p,p^*}_{D^{k+1}}$ using (12).

In the algorithm, we can easily swap the use of Proposition 3.5 with any Lipschitz constant acquired based on the theoretical ground of Lemma 3.7, such as that of Theorem 3.8 or Theorem 3.9 (if the network is non-residual).

Algorithm 1 Compositional Curvature Estimation of Neural Networks

Input: K-layer neural network in the form of (11).
Initialize $\overline{L}^p_0 = \overline{L}^{p^*}_0 = 1, \overline{L}^{p,p^*}_{D_0} = 0$.
for $k = 0$ **to** $K-1$ **do**
  Calculate $\overline{L}^{p^*}_{h^k}$ using (15).
  Calculate a bound on $\overline{L}^{p,p^*}_{Dh^k}$ using Proposition 3.5.
  Update $\overline{L}^{p,p^*}_{D_{k+1}} = \overline{L}^{p,p^*}_{Dh^k}\overline{L}^p_k\overline{L}^{p^*}_k + \overline{L}^{p^*}_{h^k}\overline{L}^{p,p^*}_{D_k}$.
  Calculate $\overline{L}^p_{k+1}$ and $\overline{L}^{p^*}_{k+1}$ using (16).
end for
Return $\overline{L}^{p,p^*}_{D_K}$, the Lipschitz constant of the Jacobian of $x^0 \mapsto x^K$.

3.2.5. Comparison with existing approaches

Unlike the previous work by Singla et al. (Singla & Feizi, 2020), which is limited to scalar-valued and non-residual architectures, our framework accommodates vector-valued general sequential models. Additionally, although Singla et al. (Singla & Feizi, 2020) could theoretically handle convolutional neural networks by expressing such layers as equivalent fully connected layers using their Toeplitz matrices (Chen et al., 2020), this approach would be computationally prohibitive. In contrast, our method readily applies to convolutional layers.

Moreover, our method does *not* require twice differentiability. This is particularly relevant for functions with Lipschitz continuous first derivatives but undefined second derivatives. For instance, consider the well-known Exponential Linear Unit (ELU):

$$f(z) = \begin{cases} z & z \geq 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases}$$

The ELU has a Lipschitz continuous first derivative, but its second derivative is not defined at $z = 0$. Therefore, Hessian-based analysis would fail for this function, whereas our Jacobian Lipschitz analysis is applicable to networks using this activation function.

In the following section, we will utilize Algorithm 1 to bound the Lipschitz constant of the Jacobian and exploit it during the training phase to control the curvature of the neural network.

3.3. Curvature-Controlled Networks

As established in Section 2, models with low curvature constants can elicit more robust behavior against norm-bounded perturbations. Driven by this observation, we can design a curvature-based regularizer that would promote robustness during training. One approach is to reward large certified radii in the objective similar to (Fazlyab et al., 2023; Xu

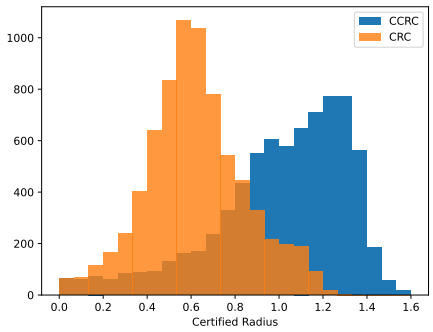

Figure 3: Certified radius comparison on a 6-layer neural network.

et al., 2022), giving rise to the training loss function

$$\mathcal{L}(x, y; f) = \mathcal{L}_{\text{CE}}(f(x), u_y) + \lambda 1_{\{C(x)=y\}} g(\underline{\varepsilon}_1^*(x)),$$

where $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss, u_y is the one-hot vector corresponding to y, $\lambda > 0$ is the regularization constant, and $g\colon \mathbb{R}_+ \to \mathbb{R}$ is a convex decreasing function (e.g., $g(z) = \exp(-z)$). The role of the indicator function $1_{\{C(x)=y\}}$ is to restrict the regularizer to correctly classified points only. This regularizer is differentiable due to the closed-form expression for $\varepsilon_1^*(x)$ given in (6). Nonetheless, computing it for each data point in the training dataset can be computationally costly for large-scale instances. A more efficient approach is to regularize the bound on the global curvature of the network during training,

$$\mathcal{L}(x, y; f) = \mathcal{L}_{\text{CE}}(f(x), u_y) + \lambda \overline{L}_{Df}^{p,p^*},$$

To further improve the efficiency, we propose to use 1-Lipschitz layers to build the architecture. Specifically, when $p = 2$, if h^k in (11) is modified to be a 1-Lipschitz function ($\overline{L}_{h^k}^p = 1$), the concatenation of all layers will be 1-Lipschitz ($\overline{L}_k^p = 1$), and thus (12) yields the curvature bound $\overline{L}_{Df}^p = \sum_{k=0}^{L-1} \overline{L}_{Dh^k}^p$, which can be readily computed.

4. Experiments

In this section, we evaluate the performance of our proposed methods via a series of experiments. We contrast our methods against the state-of-the-art Lipschitz constant estimation, curvature estimation, and neural network robustness certification algorithms. In our experiments we set $p = 2$ for all norms and Lipschitz calculations. We defer the discussion of hyperparameters to Appendix D.1. Our code is available at https://github.com/o4lc/Compositional-Curvature-Bounds-for-DNNs.

We first showcase the application and superiority of our Jacobian Lipschitz constant estimation on MNIST (LeCun et al., 1998). Next, to further motivate the use of anchored Lipschitz constant estimation, we train several networks with different depths to portray its effectiveness on both Lipschitz constant estimation and Jacobian Lipschitz constant estimation. Finally, we compare our robustness certification method with the state-of-the-art classification on the CIFAR (Krizhevsky et al., 2009) dataset and provide attack certificates on the same networks.

Table 1: Comparison of certified accuracies obtained from state-of-the-art methods SLL (Araujo et al., 2022) and CRM (Fazlyab et al., 2023) on CIFAR-10.

Model	Methods	Accuracy	Certified Accuracy (ε)			Parameters
			$\frac{36}{255}$	$\frac{72}{255}$	$\frac{108}{255}$	
6C2F	Standard	**79.95**	0	0	0	0.7M
	CRM	58.57	36.25	18.36	7.37	0.7M
	CCRC (Ours)	61.15	**49.53**	**33.36**	**16.95**	0.7M
Lip-3C1F	SLL	**57.2**	45.0	35.0	26.5	1M
	SLL + CCRC (Ours)	53.2	**46.6**	**39.3**	**31.6**	1M
6F	Standard	61.89	0	0	0	4M
	CRM	60.63	42.73	24.75	12.6	4M
	CCRC (Ours)	**62.1**	**52.09**	**40.8**	**29.17**	4M

Comparison with other Curvature-based Methods In this experiment, we compare our proposed compositional curvature calculation method with the previous works. Consequently, we train a 6-layer fully connected network on MNIST with curvature regularization and compute the certified radii of the test data points for this network using two curvature calculation algorithms. We denote (Singla & Feizi, 2020) as *Curvature-based Robustness Certificate (CRC)*, and our method as *Compositional Curvature-based Robustness Certificate (CCRC)*. Next, to focus the experiment on comparing the curvature bounds, we use the method of (Singla & Feizi, 2020) to obtain the certified radius for each point.

Figure 3 compares the certified radii of these methods and confirms the superior performance of the compositional curvature calculation algorithm.

Anchored Lipschitz/Curvature Estimation Next, we study the impact of localizing the computations via the concept of anchored Lipschitz constant introduced in this paper. To achieve this, we train fully connected neural networks of varying depths and calculate upper bounds on the Lipschitz and curvature constants of the network, both globally and in an anchored manner. Figure 4 illustrates the results on the MNIST dataset. For the anchored bounds, we average the values over the test dataset. The results demonstrate that using the anchored counterparts significantly improves the bounds.

Attack Certification on CIFAR-10 In this experiment, we provide radii for provable attacks on a 6F model. Fig-

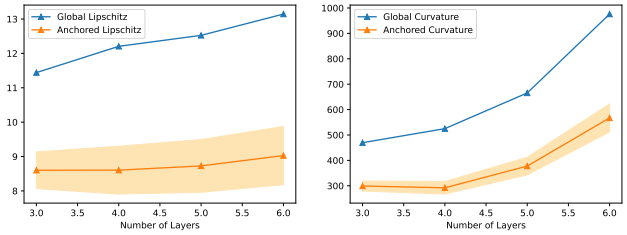

Figure 4: Comparison of global Lipschitz and Curvature estimation against their anchored counterparts. The shaded areas denote the standard deviation over the whole dataset.

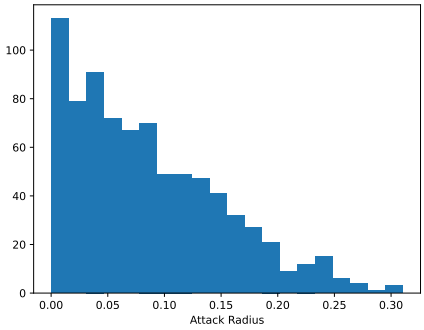

Figure 5: Histogram of the certified attack radii for a 6-layer neural network trained via curvature regularization on CIFAR-10.

ure 5 shows the results. This model has an accuracy of 56.19%. Furthermore, with the perturbation budget $\frac{36}{255}$ the model has certified and PGD accuracy of 47.16% and 48.46%, respectively. By analyzing the attack certificates we find that our method is able to provide an attack certificate for a total of 808 samples, of which 645 require a perturbation budget of at most $\frac{36}{255}$. Using this information, the robust accuracy of the model with this perturbation budget is at most $56.19 - \frac{645}{10000} \times 100 = 49.74\%$. This is illustrated in Figure 6, where A_c is the clean accuracy, A_v^* is the verified accuracy, and $\underline{A}_v$ and $\overline{A}_v$, are lower and upper bounds on the verified accuracy, respectively.

This has two main implications. First, having attack certificates for any data eliminates the need to perform an attack on that data as the existence of an attack was verified by our proposition. Second, the attack certificates further narrow down the uncertainty of the model accuracy. As the certified accuracy is a lower bound on the actual certified robustness of the model, we conclude that the actual certified accuracy of this model is in the range $[47.16, 49.74]$, *regardless* of the certification method.

Robustness Certification on CIFAR-10 The final experiment aims to compare the certified accuracy of models

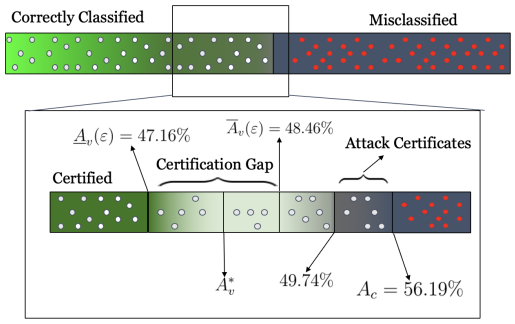

Figure 6: Illustration of the certificates provided by our method. A_c and A_v are the clean and verified accuracies, respectively. The certification radius is $\varepsilon = \frac{36}{255}$.

with different architectures on CIFAR-10 with the state-of-the-art. We train two 6C2F and 6F non-residual and a 1-Lipschitz neural network with *Lip-3C1F* architecture on the CIFAR-10 dataset with the following loss function

$$\mathcal{L}(x, y; f) = \mathcal{L}_{\text{CE}}^{\tau,\nu}(f(x), y) + \lambda \overline{L}_{Df}^{p}, \tag{18}$$

where $\mathcal{L}_{\text{CE}}^{\tau,\nu}$ is a modified variant of the cross entropy loss function (Prach & Lampert, 2022) and $\overline{L}_{Df}^{p}$ is the curvature bound acquired through Algorithm 1. Refer to Appendix D.1 for more on the loss function details. Table 1 shows the comparison between these models. We find that incorporating the additional regularization term leads to higher certified accuracies, smaller certification gaps, and often, higher clean accuracies.

5. Conclusion

In this work, we proposed a novel method to calculate provable upper bounds on the curvature constant of smooth deep neural networks, i.e., the Lipschitz constant of their first derivative. Our method leverages the compositional structure of the model to compute the curvature bound in a scalable and modular fashion. The generality of our curvature estimation algorithm can enable its use for compositional functions beyond neural networks. Furthermore, we provided analytical robustness certificates for deep classifiers based on the curvature of the model. In the future, we aim to further tighten the estimated gap of the curvature bound, enabling the algorithm to produce tighter bounds on the curvature of even deeper neural networks.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. We do not foresee any societal implications arising solely from our work.

References

Araujo, A., Havens, A. J., Delattre, B., Allauzen, A., and Hu, B. A unified algebraic perspective on lipschitz neural networks. In *The Eleventh International Conference on Learning Representations*, 2022.

Béthune, L., Boissin, T., Serrurier, M., Mamalet, F., Friedrich, C., and Gonzalez Sanz, A. Pay attention to your loss: understanding misconceptions about lipschitz neural networks. *Advances in Neural Information Processing Systems*, 35:20077–20091, 2022.

Chen, Y., Xie, Y., Song, L., Chen, F., and Tang, T. A survey of accelerator architectures for deep neural networks. *Engineering*, 6(3):264–274, 2020. ISSN 2095-8099. doi: https://doi.org/10.1016/j.eng.2020.01.007.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pp. 1310–1320. PMLR, 2019.

Dutta, S., Jha, S., Sankaranarayanan, S., and Tiwari, A. Output range analysis for deep feedforward neural networks. In *NASA Formal Methods Symposium*, pp. 121–138. Springer, 2018.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. Efficient and accurate estimation of lipschitz constants for deep neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Fazlyab, M., Entesari, T., Roy, A., and Chellappa, R. Certified robustness via dynamic margin maximization and improved lipschitz regularization, 2023.

Fischetti, M. and Jo, J. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018.

Hashemi, N., Ruths, J., and Fazlyab, M. Certifying incremental quadratic constraints for neural networks via convex optimization. In *Learning for Dynamics and Control*, pp. 842–853. PMLR, 2021.

Huang, Y., Zhang, H., Shi, Y., Kolter, J. Z., and Anandkumar, A. Training certifiably robust neural networks with efficient local lipschitz bounds. *Advances in Neural Information Processing Systems*, 34:22745–22757, 2021.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kumar, A., Levine, A., and Feizi, S. Policy smoothing for provably robust reinforcement learning. In *International Conference on Learning Representations*, 2021.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Leino, K., Wang, Z., and Fredrikson, M. Globally-robust neural networks. In *International Conference on Machine Learning*, pp. 6212–6222. PMLR, 2021.

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

Moosavi-Dezfooli, S.-M., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9078–9086, 2019.

Prach, B. and Lampert, C. H. Almost-orthogonal layers for efficient general-purpose lipschitz networks. In *European Conference on Computer Vision*, pp. 350–365. Springer, 2022.

Robey, A., Hu, H., Lindemann, L., Zhang, H., Dimarogonas, D. V., Tu, S., and Matni, N. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pp. 3717–3724. IEEE, 2020.

Singla, S. and Feizi, S. Second-order provable defenses against adversarial attacks. In *International conference on machine learning*, pp. 8981–8991. PMLR, 2020.

Singla, S. and Feizi, S. Skew orthogonal convolutions. In *International Conference on Machine Learning*, pp. 9756–9766. PMLR, 2021.

Singla, S. and Feizi, S. Improved techniques for deterministic l2 robustness. *Advances in Neural Information Processing Systems*, 35:16110–16124, 2022.

Singla, V., Singla, S., Feizi, S., and Jacobs, D. Low curvature activations reduce overfitting in adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16423–16433, 2021.

Srinivas, S., Matoba, K., Lakkaraju, H., and Fleuret, F. Efficient training of low-curvature neural networks. *Advances in Neural Information Processing Systems*, 35: 25951–25964, 2022.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tjeng, V., Xiao, K. Y., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming.

In *International Conference on Learning Representations*, 2018.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. *Advances in neural information processing systems*, 31, 2018.

Wang, R. and Manchester, I. Direct parameterization of lipschitz-bounded deep networks. In *International Conference on Machine Learning*, pp. 36093–36110. PMLR, 2023.

Xu, Y., Sun, Y., Goldblum, M., Goldstein, T., and Huang, F. Exploring and exploiting decision boundary dynamics for adversarial robustness. In *The Eleventh International Conference on Learning Representations*, 2022.

Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., and Jordan, M. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pp. 7472–7482. PMLR, 2019.

Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

A. Theorems and Proofs

Proof of Lemma 1.2

Proof. For any direction $d \in \mathbb{R}^n$, let $g_d(t) = f(x + td)$. Evidently, we have $g'_d(t) = \frac{d}{dt} g_d(t) = Df(x + td)d$, where $Df(x) \in \mathbb{R}^{m \times n}$. Moreover, we have

$$\|Df(x)d\|_p = \|g'_d(0)\|_p = \| \lim_{t \to 0} \frac{g_d(t) - g_d(0)}{t - 0} \|_p = \lim_{t \to 0} \frac{\|f(x + td) - f(x)\|_p}{t} \leq L_f^p(x)\|d\|_p,$$

where the third equality follows from continuity of $\| \cdot \|_p$. Consequently,

$$\|Df(x)\|_p = \sup_{\|d\|_p \leq 1} \|Df(x)d\|_p = \sup_{\|d\|_p \leq 1} \|g'_d(0)\|_p \leq L_f^p(x).$$

□

Corollary A.1. Consider a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ and let $L_f^p(x)$ be a corresponding anchored Lipschitz constant in some p-norm. We have

$$\|\nabla f(x)\|_{p^*} \leq L_f^p(x).$$

Proof. The proof follows from the fact that $Df(x) = \nabla f(x)^\top$ and that for matrix A and norm $\| \cdot \|_p$, we have $\|A^\top\|_p = \|A\|_{p^*}$. □

Derivation of equation (4)

Proof. To prove the quadratic upper bound on the logit gap with the anchored Lipschitz constant we utilize the mean value theorem. For any x and δ, we can write

$$f_{iy}(x + \delta) = f_{iy}(x) + \nabla f_{iy}(x)^\top \delta + \int_0^1 (\nabla f_{iy}(x + t\delta) - \nabla f_{iy}(x))^\top \delta \, dt.$$

We can then write

$$\begin{aligned}
|f_{iy}(x + \delta) - f_{iy}(x) - \nabla f_{iy}(x)^\top \delta| &\leq \int_0^1 |(\nabla f_{iy}(x + t\delta) - \nabla f_{iy}(x))^\top \delta| dt \\
&\xrightarrow[\frac{1}{p^*} + \frac{1}{p} = 1]{\text{Hölder's Inequality}} \leq \int_0^1 \|(\nabla f_{iy}(x + t\delta) - \nabla f_{iy}(x)\|_{p^*} \|\delta\|_p dt \\
&\leq \int_0^1 L^{p,p^*}_{\nabla f_{iy}}(x)\|\delta\|_p^2 \, t \, dt \\
&= \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\|\delta\|_p^2.
\end{aligned}$$

Thus, we have

$$f_{iy}(x + \delta) \leq f_{iy}(x) + \nabla f_{iy}(x)^\top \delta + \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\|\delta\|_p^2.$$

□

Proof of Proposition 2.1

Proof. Define $\Delta = \{\delta \mid f_{iy}(x + \delta) \leq 0\}$ and $\overline{\Delta} = \{\delta \mid \overline{f_{iy}}(x, \delta; L^{p,p^*}_{\nabla f_{iy}}(x)) \leq 0\}$. It is clear that $\overline{\Delta} \subseteq \Delta$. Consequently, $\{\varepsilon \mid \sup_{\|\delta\|_p \leq \varepsilon} \overline{f_{iy}}(x, \delta; L^{p,p^*}_{\nabla f_{iy}}(x)) \leq 0\} \subseteq \{\varepsilon \mid \sup_{\|\delta\|_p \leq \varepsilon} f_{iy}(x + \delta) \leq 0\}$, and thus, (5) yields a valid lower bound on the original radius (1).

Now, to acquire the analytical result we have

$$\sup_{\|\delta\|_p \leq \varepsilon} f_{iy}(x) + \nabla f_{iy}(x)^\top \delta + \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\|\delta\|_p^2 = f_{iy}(x) + \varepsilon\|\nabla f_{iy}(x)\|_{p^*} + \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\varepsilon^2.$$

This equality holds due to the fact that for the δ^* achieving $\sup_{\|\delta\|_p \leq \varepsilon} \nabla f_{iy}(x)^\top \delta$ we have $\|\delta^*\|_p = \varepsilon$, implying that δ^* is also a maximizer of the other term $\frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\|\delta\|_p^2$.

As a result, we obtain the following optimization problem that is equivalent to (5)

$$\begin{aligned} \max \quad & \varepsilon \\ \text{s.t.} \quad & \frac{L^{p,p^*}_{\nabla f_{iy}}(x)}{2}\varepsilon^2 + \|\nabla f_{iy}(x)\|_{p^*}\varepsilon + f_{iy}(x) \leq 0, \forall i \neq y. \end{aligned}$$

Using elementary calculations, we obtain the optimal solution

$$\min_{i \neq y} \frac{-\|\nabla f_{iy}(x)\|_{p^*} + (\|\nabla f_{iy}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{iy}}(x) f_{iy}(x))^{\frac{1}{2}}}{L^{p,p^*}_{\nabla f_{iy}}(x)}.$$

□

Proof of Proposition 2.2

Proof. Suppose x is correctly classified, i.e., $f_{iy}(x) < 0$ for $i \neq y$. To enforce the condition $\underline{\varepsilon}^*_0(x) \leq \underline{\varepsilon}^*_1(x)$ we must have

$$\underline{\varepsilon}^*_0(x) \leq \frac{-\|\nabla f_{iy}(x)\|_{p^*} + (\|\nabla f_{iy}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{iy}}(x) f_{iy}(x))^{\frac{1}{2}}}{L^{p,p^*}_{\nabla f_{iy}}(x)}, \qquad \forall i \neq y.$$

Or equivalently,

$$L^{p,p^*}_{\nabla f_{iy}}(x)\underline{\varepsilon}^*_0(x)^2 + 2\|\nabla f_{iy}(x)\|_{p^*}\underline{\varepsilon}^*_0(x) + 2f_{iy}(x) \leq 0.$$

The above inequality holds if and only if

$$L^{p,p^*}_{\nabla f_{iy}}(x) \leq \frac{-2(\|\nabla f_{iy}(x)\|_{p^*}\underline{\varepsilon}^*_0(x) + f_{iy}(x))}{\underline{\varepsilon}^*_0(x)^2}. \tag{19}$$

Utilizing Corollary A.1, we note that $\underline{\varepsilon}^*_0(x) = \min_{i \neq y} \frac{-f_{iy}(x)}{L^p_{f_{iy}}(x)} \leq \frac{-f_{iy}(x)}{L^p_{f_{iy}}(x)} \leq \frac{-f_{iy}(x)}{\|\nabla f_{iy}(x)\|_{p^*}}$. This ensures that the R.H.S. of (19) is positive.

□

Proof of Proposition 2.3

Proof. We prove this in two steps:

1. $\varepsilon^*(x) \leq \varepsilon'^*(x)$: Suppose this does not hold. Then $\varepsilon^*(x) > \varepsilon'^*(x)$. However, having a solution for (7) implies that there exists a pair δ and j with $\|\delta\|_p \leq \varepsilon'^*(x) < \varepsilon^*(x)$ such that $f_{yj}(x+\delta) < 0$. This perturbation-index pair is then a violation for the constraint of problem (1). Thus, we must have $\varepsilon^*(x) \leq \varepsilon'^*(x)$.
2. $\varepsilon^*(x) \geq \varepsilon'^*(x)$: Similarly, if this does not hold, then $\forall i \neq y, \delta, \|\delta\|_p \leq \varepsilon^*(x) < \varepsilon'^*(x)$ we have $\sup_{\|\delta\|_p \leq \varepsilon^*(x)} f_{iy}(x+\delta) \leq 0$. But having a solution for (7) asserts that there exists one such index that $f_{yi}(x+\delta) < 0$. Thus, $\varepsilon^*(x) \geq \varepsilon'^*(x)$ must hold.

As a result, we must have $\varepsilon^*(x) = \varepsilon'^*(x)$. □

Proof of Proposition 2.4

Proof. To prove this proposition, we first find the analytical solution to the inner optimization problems, namely

$$\inf_{\|\delta\|_p \leq \varepsilon} \overline{f_{yi}}(x, \delta; L^{p,p^*}_{\nabla f_{iy}}(x)) = \inf_{0 \leq \lambda \leq 1} \inf_{\|\delta\|_p = \lambda\varepsilon} f_{yi}(x) + \nabla f_{yi}(x)^\top \delta + \frac{L^{p,p^*}_{\nabla f_{yi}}(x)}{2} \|\delta\|_p^2. \tag{20}$$

Let $\delta^* = \arg\min_{\|\delta\|_p \leq 1} \nabla f_{yi}(x)^\top \delta$, which yields $\nabla f_{yi}(x)^\top \delta^* = -\|\nabla f_{yi}(x)\|_{p^*}$. The inner optimization problem is minimized at $\delta = \lambda\varepsilon\delta^*$. Consequently, we can frame problem (20) equivalently as

$$g_i^*(\varepsilon) = g_i(\varepsilon, \lambda^*) = \min_{0 \leq \lambda \leq 1} \underbrace{f_{yi}(x) - \varepsilon\|\nabla f_{yi}(x)\|_{p^*}\lambda + \frac{L^{p,p^*}_{\nabla f_{yi}}(x)}{2}\varepsilon^2\lambda^2}_{g_i(\varepsilon,\lambda)}. \tag{21}$$

There are three possible solutions

1. $\hat{\lambda} = 0$. In this scenario
$$g_i(\varepsilon, \hat{\lambda}) = f_{yi}(x).$$

2. $\hat{\lambda} = \dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)\varepsilon}$. In this scenario
$$g_i(\varepsilon, \hat{\lambda}) = f_{yi}(x) - \frac{\|\nabla f_{yi}(x)\|^2_{p^*}}{2L^{p,p^*}_{\nabla f_{yi}}(x)}.$$
For this solution we must have $\hat{\lambda} \leq 1$. This imposes the condition $\dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)} \leq \varepsilon$.

3. $\hat{\lambda} = 1$. In this scenario we have
$$g_i(\varepsilon, \hat{\lambda}) = f_{yi}(x) - \varepsilon\|\nabla f_{yi}(x)\|_{p^*} + \frac{L^{p,p^*}_{\nabla f_{yi}}(x)}{2}\varepsilon^2.$$

Putting together the different conditions, we find that

$$g_i^*(\varepsilon) = \begin{cases} f_{yi}(x) - \dfrac{\|\nabla f_{yi}(x)\|^2_{p^*}}{2L^{p,p^*}_{\nabla f_{yi}}(x)} & , \dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)} \leq \varepsilon \\ f_{yi}(x) - \varepsilon\|\nabla f_{yi}(x)\|_{p^*} + \dfrac{L^{p,p^*}_{\nabla f_{yi}}(x)}{2}\varepsilon^2 & , \dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)} > \varepsilon \end{cases}$$

The next step is solving

$$\begin{aligned} \min \quad & \varepsilon \\ \text{s.t.} \quad & \min_{i \neq y} g_i^*(\varepsilon) < 0. \end{aligned}$$

For each $i \neq y$, if $\dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)} \leq \varepsilon$ then $g_i^*(\varepsilon) \leq 0$ requires $2L^{p,p^*}_{\nabla f_{yi}}(x) f_{yi}(x) \leq \|\nabla f_{yi}(x)\|^2_{p^*}$, and if $\dfrac{\|\nabla f_{yi}(x)\|_{p^*}}{L^{p,p^*}_{\nabla f_{yi}}(x)} > \varepsilon$ the smallest value of ε yielding $g_i^*(\varepsilon) \leq 0$ is

$$\frac{\|\nabla f_{yi}(x)\|_{p^*} - \sqrt{\|\nabla f_{yi}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{yi}}(x) f_{yi}(x)}}{L^{p,p^*}_{\nabla f_{yi}}(x)}, \tag{22}$$

which similarly requires the condition $2L^{p,p^*}_{\nabla f_{yi}}(x)f_{yi}(x) \leq \|\nabla f_{yi}(x)\|^2_{p^*}$ for realizability. Consequently, if $2L^{p,p^*}_{\nabla f_{yi}}(x)f_{yi}(x) \leq \|\nabla f_{yi}(x)\|^2_{p^*}$ holds for some $i \neq y$, the smallest valid ε is given as in (22). Thus, we conclude that

$$\bar{\varepsilon}^*_1(x) = \min_{i\in\mathcal{I}} \frac{\|\nabla f_{yi}(x)\|_{p^*} - \sqrt{\|\nabla f_{yi}(x)\|^2_{p^*} - 2L^{p,p^*}_{\nabla f_{yi}}(x)f_{yi}(x)}}{L^{p,p^*}_{\nabla f_{yi}}(x)}, \tag{23}$$

where $\mathcal{I} = \{i | i \neq y, 2L^{p,p^*}_{\nabla f_{yi}}(x)f_{yi}(x) \leq \|\nabla f_{yi}(x)\|^2_{p^*}\}$. If $\mathcal{I} = \varnothing$, the problem is infeasible.

□

Proof of Theorem 3.1

Proof. Writing the definition of Lipschitz continuity, we have

$$\begin{aligned}
\|Dh(x) - Dh(y)\|_q &= \|Df(g(x))Dg(x) - Df(g(y))Dg(y)\|_q \\
&= \|Df(g(x))Dg(x) - Df(g(x))Dg(y) + Df(g(x))Dg(y) - Df(g(y))Dg(y)\|_q \\
&\leq \|Df(g(x))Dg(x) - Df(g(x))Dg(y)\|_q + \|Df(g(x))Dg(y) - Df(g(y))Dg(y)\|_q \\
&\leq \|Df(g(x))\|_q\|Dg(x) - Dg(y)\|_q + \|Df(g(x)) - Df(g(y))\|_q\|Dg(y)\|_q \\
&\leq L^{p,q}_{Dg}\|Df(g(x))\|_q\|x - y\|_p + L^{q',q}_{Df}\|g(x) - g(y)\|_{q'}\|Dg(y)\|_q \\
&\leq L^{p,q}_{Dg}\|Df(g(x))\|_q\|x - y\|_p + L^{q',q}_{Df}L^{p,q'}_{g}\|Dg(y)\|_q\|x - y\|_p.
\end{aligned}$$

Based on Lemma 1.2, we note that L^q_f is an upper bound on $\|Df(\cdot)\|_q$ and that L^q_g is an upper bound on $\|Dg(\cdot)\|_q$. Thus, we arrive at

$$\|Dh(x) - Dh(y)\|_q \leq \left(L^{p,q}_{Dg}L^q_f + L^{q',q}_{Df}L^{p,q'}_{g}L^q_g\right)\|x - y\|_p.$$

Finally, by setting $q = p^*$ and $q' = p$, we obtain $L^{p,p^*}_{Dh} \leq L^{p,p^*}_{Dg}L^{p^*}_f + L^{p,p^*}_{Df}L^p_gL^{p^*}_g$. □

Proof of Theorem 3.2

Proof. Taking the same steps as the proof of Theorem 3.1, we have

$$\begin{aligned}
\|Dh(x+\delta) - Dh(x)\|_q &\leq \|Df(g(x+\delta))\|_q\|Dg(x+\delta) - Dg(x)\|_q + \|Df(g(x+\delta)) - Df(g(x))\|_q\|Dg(x)\|_q \\
&\leq L^{p,q}_{Dg}(x)\|Df(g(x+\delta))\|_q\|\delta\|_p + L^{q',q}_{Df}(g(x))L^{p,q'}_{g}(x)\|Dg(x)\|_q\|\delta\|_p \\
&\leq \left(L^q_fL^{p,q}_{Dg}(x) + \|Dg(x)\|_qL^{q',q}_{Df}(g(x))L^{p,q'}_{g}(x)\right)\|\delta\|_p.
\end{aligned}$$

Setting $q = p^*$ and $q' = p$ yields the result. □

Proof of Proposition 3.5

Proof. We drop the superscript k for simplicity here. Writing out the definition of Lipschitz continuity we have

$$\begin{aligned}
\|Dh(x) - Dh(y)\|_q &= \|G\,\mathrm{diag}(\Phi'(Wx))W - G\,\mathrm{diag}(\Phi'(Wy))W\|_q \\
&\leq \|G\|_q\|W\|_q\|\mathrm{diag}(\Phi'(Wx)) - \mathrm{diag}(\Phi'(Wy))\|_q \\
&= \|G\|_q\|W\|_q\max_i|\Phi'(Wx)_i - \Phi'(Wy)_i| \\
&\leq L_{\phi'}\|G\|_q\|W\|_q\max_i|W_{i,:}(x - y)|,
\end{aligned} \tag{24}$$

where $L_{\phi'} = \max\{|\alpha'|, |\beta'|\}$ is the Lipschitz constant of ϕ'. Next, we can write

$$\max_i|W_{i,:}(x - y)| = \|W(x - y)\|_\infty \leq \|W\|_{p\to\infty}\|x - y\|_p,$$

Setting $q = p^*$ yields the desired result.

□

Proof of Lemma 3.6

Proof. To perform the conversion to a standard layer, we consider individual entries of the output:

$$Dh^k(x)_{ij} = H^k_{ij} + \sum_{l=1}^{n'_k} G^k_{il} W^k_{lj} \Phi'(W^k x)_l.$$

Thus, by flattening the matrix $Dh^k(x)$ into a vector $dh^k(x)$, where the ij-th element of $Dh^k(x)$ is mapped to the $\big((j-1)\times n_{k+1}+i\big)$-th element of $dh^k(x)$, $\forall i \in [n_{k+1}], j \in [n_k]$, we obtain the desired result. The vector b^k and matrix A^k defined in the lemma yield the correct map. Importantly, we have the identity $A^k = \hat{G}^k \tilde{W}^k$, where

$$\hat{G}^k = \begin{bmatrix} G^k & 0 & \cdots & 0 \\ 0 & G^k & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & G^k \end{bmatrix}, \quad \tilde{W}^k = \begin{bmatrix} \mathrm{diag}(W^k_{:,1}) \\ \mathrm{diag}(W^k_{:,2}) \\ \vdots \\ \mathrm{diag}(W^k_{:,n_k}) \end{bmatrix}.$$

Evidently, we have $A^k_{ml} = \big(G^k \mathrm{diag}(W^k_{:,j})\big)_{il} = G^k_{il} W^k_{lj}$. □

Proof of Lemma 3.7

Proof. By vectorization, we have $\|dh^k(x) - dh^k(y)\|_2 = \|Dh^k(x) - Dh^k(y)\|_F$, where $\|\cdot\|_F$ is the Frobenius norm. We can write

$$\|Dh^k(x) - Dh^k(y)\|_2 \le \|Dh^k(x) - Dh^k(y)\|_F = \|dh^k(x) - dh^k(y)\|_2 \le L^p_{dh^k} \|x - y\|_2,$$

where we have used the fact that for a given matrix A, $\|A\|_2 = \sigma_{\max}(A) \le \sqrt{\sum_i \sigma_i^2(A)} = \|A\|_F$. □

Proof of Theorem 3.8

Proof. Using identity $A^k = \hat{G}^k \tilde{W}^k$ from the proof of Lemma 3.6, we have

$$\|A^k\|_2 \le \|\hat{G}^k\|_2 \|\tilde{W}^k\|_2 = \|G^k\|_2 \|\tilde{W}^k\|_2.$$

For $\|\tilde{W}^k\|_2$ we have $(\tilde{W}^{k\top}\tilde{W}^k)_{ij} = \sum_l \tilde{W}^k_{li}\tilde{W}^k_{lj}$. With respect to the sparsity pattern of the matrix $\tilde{W}^k$, $\tilde{W}^{k\top}\tilde{W}^k$ is only non-zero on its diagonal with $(\tilde{W}^{k\top}\tilde{W}^k)_{ii} = \|W_{i,:}\|_2^2$. Thus $\|\tilde{W}^k\|_2 = \max_i \|W_{i,:}\|_2 = \|W\|_{2\to\infty}$. This concludes the proof. □

Proof of Theorem 3.9

Proof. Using Lemma 3.6 to vectorize the Jacobian of the layer $h^k(x) = \Phi(W^k x)$, we obtain $A^k = \tilde{W}^k$ (see proof of Lemma 3.6). As stated in the proof of Theorem 3.8, $A^{k\top}A^k$ is diagonal with $(A^{k\top}A^k)_{ii} = \|W_{i,:}\|_2$. Next, we state the semidefinite program of (Fazlyab et al., 2019) for calculating the Lipschitz constant of $dh^k = A^k \phi'(Wx)$. Define

$$M(\rho, T) = \begin{bmatrix} -\alpha'\beta' W^{k\top} T W^k - \rho I & \frac{(\alpha'+\beta')}{2} W^{k\top} T \\ \frac{(\alpha'+\beta')}{2} T W^k & -T + A^{k\top} A^k \end{bmatrix},$$

where T is a diagonal non-negative matrix of appropriate dimensions. We have the following optimization problem.

$$\begin{aligned} \min_{\rho, T} \quad & \rho \\ \text{s.t.} \quad & M(\rho, T) \preceq 0. \end{aligned}$$

As stated in (Fazlyab et al., 2019), for any given feasible pair (ρ, T), $\sqrt{\rho}$ is an upper bound on the Lipschitz constant of the desired map.

We first assume that $\alpha' = -\beta'$, implying that the off-diagonal terms of $M(\rho, T)$ will be zero. As a result, the linear matrix inequality constraint $M(\rho, T) \preceq 0$ simplifies to satisfying two semidefinite conditions as follows,

$$\begin{cases} \beta'^2 W^{k\top} T W^k \preceq \rho I, \\ A^{k\top} A^k \preceq T. \end{cases}$$

We claim that the optimal solution for this system of constraints is given by

$$\begin{cases} T^* = A^{k\top} A^k, \\ \rho^* = \beta'^2 \|W^{k\top} T^* W^k\|_2. \end{cases}$$

The choice of ρ^* is trivial. Next, it is easy to see that if we instead use $T' = T^* + E$, where E is another non-negative diagonal matrix, we will have $W^{k\top} T' W^k = W^{k\top}(T^* + E)W^k = W^{k\top} T^* W^k + W^{k\top} E W^k \succeq W^{k\top} T^* W^k$, where the last inequality follows as $W^{k\top} E W^k$ is a real symmetric positive semidefnite matrix.This concludes the proof of optimality of the proposed solution for the case in which $\alpha' = -\beta'$.

Next, we consider the scenario in which $|\alpha'| < \beta'$, we observe that a function that is slope-restricted in $[\alpha', \beta']$ is also slope-restricted in $[-\beta', \beta']$. Consequently, the proposed ρ^* is a feasible point in this case, although it may not be optimal. The case in which $|\beta'| < |\alpha'|$ follows the same argument, yielding a feasible solution $\rho^* = |\alpha'| \|W^k T^* W^k\|_2$.
Thus, we always have $L^p_{dh^k} \leq L_{\phi'} \|\sqrt{T^*} W^k\|_2 = \overline{L}^{p,\text{SDP}}_{dh^k}$. □

Corollary A.2. Let $p = 2$, and consider the map $h^k(x) = \Phi(W^k x)$ where the derivative of the ith activation function is slope-restricted in $[\alpha'_i, \beta'_i]$. Then $L^p_{dh^k}(x) \leq \|D\sqrt{T^*} W^k\|_2$, where D is a diagonal matrix with $D_{ii} = \max\{|\alpha'_i|, |\beta'_i|\}$.

B. Calculation of Anchored Lipschitz

In this section, we elaborate on some aspects of the anchored Lipschitz calculation that we introduced in the main text.

Consider a continuously differentiable function $\phi : \mathbb{R} \mapsto \mathbb{R}$. The anchored Lipschitz constant of ϕ at a x is given by

$$L_\phi(x) = \max_{y \neq x} \frac{|\phi(y) - \phi(x)|}{|y - x|}. \tag{25}$$

This optimization problem can be solved on a case-by-case basis. For example, for the case of the $\tanh$ function, the maximizer of (25) is the point from which the tangent passes through $(x, \phi(x))$. This is given by solving the nonlinear equation

$$|\phi'(y)| = \lim_{t \to y} \frac{|\phi(t) - \phi(x)|}{|t - x|}. \tag{26}$$

See Figure 1 for reference. A similar idea follows for other bounded activation functions. We note that (26) is in general a nonlinear equation without a closed-form solution. In practice, we use a numerical method (like bisection) to solve this nonlinear equation at initiation for a grid of points of the real line and then query these values whenever they are needed for Lipschitz calculation.

Next, consider a single residual block as in (11)

$$h(x) = Hx + G\Phi(Wx).$$

With the definition of anchored Lipschitz, one can use the local naive bound for the Lipschitz constant to obtain the following bound on the Lipschitz constant of h,

$$\overline{L}^{p,\text{naive}}_h(x) = \|H\|_p + \|G\|_p \|\text{diag}(L_\Phi(x))W\|_p,$$

where $L_\Phi(x) = [L_{\phi_1}(x), \cdots, L_{\phi_{n_1}}(x)]$. Then $\overline{L}^{p,\text{naive}}_h(x)$ would be an upper bound on the anchored Lipschitz of h. This can be adapted to LipSDP (Fazlyab et al., 2019) or LipLT (Fazlyab et al., 2023).

The above analysis can extended to multi-layer residual neural networks. For example, multiplying the layer-wise anchored Lipschitz bounds will yield the anchored naive Lipschitz bound for the whole network.

C. Time Complexity

We analyze the time complexity of Algorithm 1 when the ℓ_2 norm is used ($p = 2$). We utilize the power method to calculate the matrix norms. We assume that we perform only a single loop of power iteration to calculate the matrix norms. This assumption is justified in previous work (Fazlyab et al., 2023; Huang et al., 2021). We provide the complexity in terms of the number of multiplications. For a general neural network of the form of equation (11) we have the following calculations:

- Lipschitz constant of each residual block, i.e. $\overline{L}^p_{h^k} : \mathcal{O}(n_{k+1}n_k + n_{k+1}n'_k + n'_k n_k)$.
- Lipschitz constant of the Jacobian of each residual block, i.e., $\overline{L}^p_{Dh^k} : \mathcal{O}(n_{k+1}n'_k + 2n'_k n_k)$ or $\overline{L}^p_{dh^k} : \mathcal{O}(n_{k+1}n'_k + n_{k+1}n'_k n_k)$.
- Lipschitz constant of the subnetwork from the first layer to the $(k+1)$-th layer, i.e.,

$$\overline{L}^p_{k+1} : \mathcal{O}(\sum_{j=0}^{k} \left[n_{j+1}n_j + n'_j n_j + \sum_{i=j+1}^{k} (n_{i+1}n_i + n_{i+1}n'_i + n'_i n_i)\right] + \sum_{i=0}^{k} (n_{i+1}n_i + n_{i+1}n'_i + n'_i n_i)).$$

However, it is worth mentioning that the computational complexity of a forward pass through the network is also a sum of quadratic terms, i.e., $\mathcal{O}(\sum_{k=0}^{K} n_{k+1}n_k + n_{k+1}n'_k + n'_k n_k)$. Thus, the main bottleneck would be the calculation of $\overline{L}^p_{k+1}$ and $\overline{L}^p_{dh^k}$. We leverage the specialized GPU implementation of LipLT (Fazlyab et al., 2023), which substantially reduces the time complexity of calculating $\overline{L}^p_k$, $k = 0, \cdots K$.

Furthermore, by using 1-Lipschitz networks as done in some of the experiments, the time complexity of $\overline{L}^p_{k+1}$ and $\overline{L}^p_{h^k}$ would be $\mathcal{O}(1)$.

D. Experiments

In this section, we provide the details of our methods and provide further supporting experiments.

D.1. Implementation Details

We used three different architectures in our experiments. We show convolutional layers in the form $C(c, k, s, p)$, where c is the number of filters, k is the size of the square kernel, s is the stride length, and p is the symmetric padding. Fully connected layers are of the form $L(n)$, where n is the number of output neurons of this layer. Furthermore, residual layers of the form (11) is denoted by an extra character 'R', i.e., CR and LR for residual convolutional and fully-connected layers, respectively. The details of our architectures are as follows:

- 6C2F: C(32, 3, 1, 1), C(32, 4, 2, 1), C(64, 3, 1, 1), C(64, 4, 2, 1), C(64, 3, 1, 1) C(64, 4, 2, 1), L(512), L(10).
- 6F: L(1024), L(512), L(256), L(256), L(128), L(10).
- Lip-3C1F: CR(15, 3, 1, 1), CR(15, 3, 1, 1), CR(15, 3, 1, 1), LR(1024).

For the hyperparameter λ used in the loss (18), we employ a primal-dual approach. That is, after each mini-batch we update

$$\lambda^+ = \min\{\lambda + \eta(A_c - \varepsilon), \lambda_{\text{min}}\},$$

where λ is the current value of the regularizer, η is a step size, A_c is a moving average of the training accuracy of the mini-batches, ε is the minimum train accuracy that we expect from the model, and λ_{min} is the smallest value for the regularizer. For training on CIFAR-10, we choose $(\eta, \epsilon, \lambda_{\text{min}}) = (0.05, 0.6, 0.01)$.

The rest of the training details are as follows. We use the modified cross-entropy loss function from (Prach & Lampert, 2022),

$$\mathcal{L}^{\tau,\nu}_{\text{CE}}(f(x), y) = \tau \cdot \text{CE}(\frac{f(x) - \nu\sqrt{2}\,\overline{L}u_y}{\tau}, y), \tag{27}$$

where τ is a temperature constant, u_y is the one-hot encoding of the value of y, and $\overline{L}$ is an ℓ_2 Lipschitz constant of the model. ν is a zero-one variable based on the architecture and mode of training. For the 6C2F and 6F models, as we want

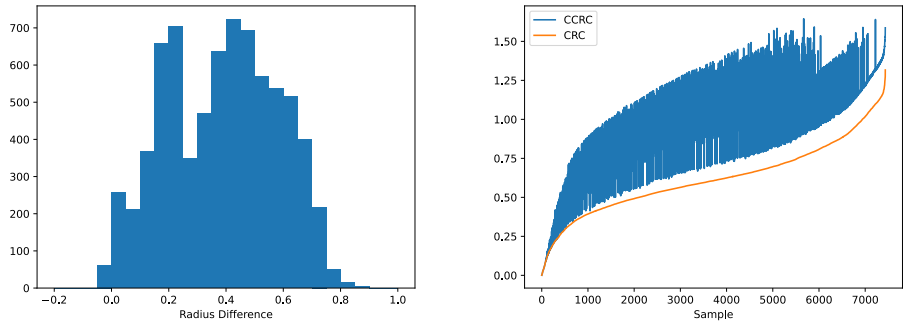

Figure 7: Comparison of certified radii acquired via CRC/CCRC on a 6-layer neural network trained via curvature regularization on MNIST. *(Left)* Histogram of per-sample radius improvement of our method over (Singla & Feizi, 2020). *(Right)* Plot of certified radii for correctly classified data. The data are sorted according to the CRC radii.

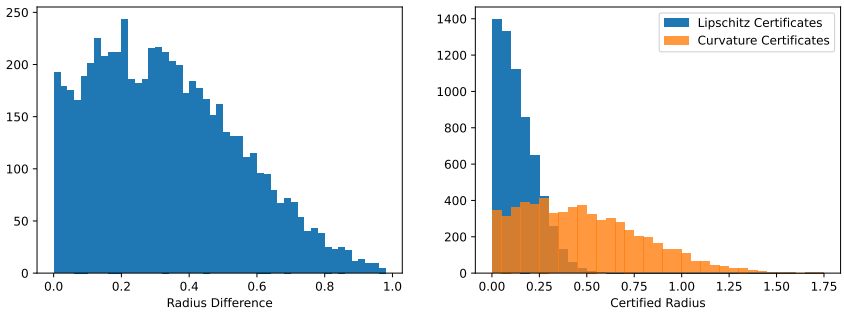

Figure 8: Comparison of certified radii calculated through Curvature Certificates versus Lipschitz Certificates for a 6-layer neural network trained via curvature regularization on CIFAR-10. *(Left)* Histogram of the per-sample radii improvements. *(Right)* Histogram of certified radii.

regularization directly through the curvature constant, we set $\nu = 0$. For the Lip-3C1F model, we set $\nu = 1$ as the original (Araujo et al., 2022) work. We use $\tau = 0.25$. Furthermore, we train our models for 1000 epochs with a batch size of 256 with a cosine annealing strategy with an initial learning rate of 10^{-4} and a final learning rate 10^{-5}, and report the average results on two seed in Table 1.

D.2. Per-sample Improvement

Expanding on the "Comparison with other Curvature-based Methods" experiment in Section 4, we provide the per-sample improvements of the certified radii in Figure 7, corresponding to Figure 3.

D.3. Training with Direct Curvature Regularization

We observed that for the models that are trained with direct regularization of the curvature, first-order certificates are significantly better than zeroth-order certificates, i.e., by regularizing the model's curvature, Proposition 2.2 would hold for all points of the test dataset. This is shown in Figure 8 for the 6F model.

D.4. Attack Certificates on 1-Lipschitz models

In this experiment, we provide radii for provable attacks on a 1-Lipschitz model trained on the CIFAR-10 dataset. The curvature required for this certificate was calculated using Algorithm 1 and utilizing the 1-Lipschitz structure. Figure 9 shows the budget required for a subset of the correctly classified data points that can certifiably be attacked. We consider the test samples of the CIFAR-10 dataset, which includes 10,000 samples. The model's accuracy on the test set is approximately 50%, resulting in about 5,000 correctly labeled samples. Of these 5,000 samples, we can provide an attack certificate for

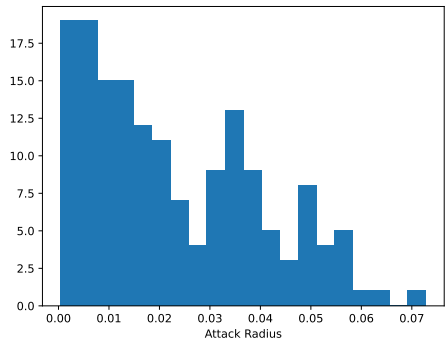

Figure 9: Attack radii certificates for a 1-Lipschitz structure.

approximately 150 of them. This translates to a 3% success rate (150/5000) for the attack certificate among the correctly classified test samples. It is worth noting that these samples can all be provably misclassified with an attack budget of less than 0.07, even on 1-Lipschitz networks.