Provable Lipschitz Certification for Generative Models

Matt Jordan [1] Alexandros G. Dimakis [1]

Abstract

We present a scalable technique for upper bounding the Lipschitz constant of generative models. We relate this quantity to the maximal norm over the set of attainable vector-Jacobian products of a given generative model. We approximate this set by layerwise convex approximations using zonotopes. Our approach generalizes and improves upon prior work using zonotope transformers and we extend to Lipschitz estimation of neural networks with large output dimension. This provides efficient and tight bounds on small networks and can scale to generative models on VAE and DCGAN architectures.

我们通过逐层凸近似来过度近似这个集合，然后解决优化问题的放松版本

1. Introduction

核心技术难题是这些是具有高维输出的向量值函数，因此标量值神经网络的 Lipschitz 估计技术不能直接转换。

We study the problem of bounding the Lipschitz constant of generative models. The central technical difficulty is that these are vector-valued functions with high-dimensional outputs, so the techniques for Lipschitz estimation of scalar-valued neural networks do not directly translate. Our approach is to frame the Lipschitz constant estimation problem as an optimization over the range of attainable vector-Jacobian products. We overapproximate this set via layerwise convex approximations, and then solve a relaxed version of the optimization problem.

The primary challenge is computing the feasible set of vector-Jacobian products over a range of inputs and vectors. We generate an overapproximation of this set by representing it as a zonotope. While prior work has performed reachability analysis of neural networks using zonotopes, these approaches only consider the forward pass. Our technique generalizes this prior work and is able to yield tighter bounds. Additionally, we are able to apply our approach to backpropagation, where we pass zonotopes backwards in place of vectors.

We present a general algorithm for mapping zonotopes through a variety of nonlinear operators and relate this to a 2-dimensional geometric problem that we solve optimally. The final norm-maximization that arises turns out to be equivalent to a mathematical question called the Grothendieck problem. We demonstrate that the linear-programming relaxation of this problem can be solved in linear time with our machinery. To fairly compare our work against existing Lipschitz estimation techniques, we first compare the estimated Lipschitz value and runtime of our algorithm on networks of increasing size trained on a toy dataset. We observe that our approach favorably trades-off accuracy for efficency and yields tighter bounds compared to previous techniques. Further, it can significantly improve the runtime of exact Lipschitz computations. We scale our technique to generative models on MNIST and CIFAR-10, using well-known architectures like DCGAN and VAEs with both fully-connected and convolutional layers (Kingma & Welling, 2013; Radford et al., 2015). Our approach yields much tighter bounds compared to any other technique that can handle vector-valued networks and can scale to such architecture sizes.

2. Related Work

Robustness Certification: Lipschitz estimation is closely related to robustness certification. Here, the goal is to provide a certificate of robustness against adversarial attacks for a specified network and input region. The techniques of interest in this domain are Lipschitz approximation and reachability analysis. It has been noted multiple times that an upper bound to the Lipschitz constant can provide a guarantee of robustness against adversarial examples (Hein & Andriushchenko, 2017; Weng et al., 2018b;a). Reachability analysis is frequently couched in the language of abstract interpretations where the goal is to develop sound transformations to map sets through the forward pass of a given classifier. The classes of sets considered include hyperboxes and zonotopes, as well as polytopes, imageStars, and linear bounds (Singh et al., 2019a;b; Mirman et al., 2018; Zico Kolter & Wong, 2017; Weng et al., 2018a; Xu et al., 2020; Zhang et al., 2018; Singh et al., 2018; Tran et al., 2020).

Lipschitz Approximation: A number of recent works provide either heuristic estimates or provable upper bounds

[1] University of Texas at Austin. Correspondence to: Matt Jordan <mjordan@cs.utexas.edu>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

可达性分析经常用抽象解释的语言表达，其目标是通过给定分类器的前向传递开发合理的转换以映射集合。

of the Lipschitz constant of neural networks. For ReLU networks, this problem is known to be NP-hard and has strong inapproximability guarantees (Virmaux & Scaman, 2018; Jordan & Dimakis, 2020). However for small networks it has been shown that this quantity may be estimated to a reasonable degree of accuracy. The majority of these works are either unable to handle vector-valued neural networks or are unable to scale to larger networks. Provable upper bounds of the Lipschitz constant may be attained using interval analysis (Weng et al., 2018a), semidefinite programming (Fazlyab et al., 2019), linear or mixed-integer programming (Jordan et al., 2019), or polynomial optimization (Latorre et al., 2019; Chen et al., 2020). Heuristic estimates of the Lipschitz constant may be attained by either randomness and extreme-value theory (Weng et al., 2018b), or by a greedy algorithm over the activation patterns (Virmaux & Scaman, 2018). Our work can be viewed as a spiritual successor to the interval analysis approach of Lipschitz estimation and the zonotope abstract interpretation approaches for robustness certification.

我们的工作可以看作是Lipschitz 估计的区间分析方法和稳健性证明的zonotope 抽象解释方法的精神继承者

3. Problem Statement and Relaxation Strategy

We formally define the problem of estimating the Lipschitz constants of vector-valued functions and provide a broad overview of the strategy for our convex relaxation. We start with the notation we will employ.

Notation: We will denote vectors using lowercase letters, matrices using capital letters and sets using calligraphic letters. We denote the unit-norm ball with respect to the α-norm as B_α. We refer to the dual norm of the α-norm as the norm $\|\cdot\|_{\alpha^*}$, defined as $\|v\|_{\alpha^*} := \sup_{u\in B_\alpha} |u^T v|$. For a function $f(\cdot) : \mathbb{R}^k \to \mathbb{R}^n$, we denote the Jacobian with respect to its argument as $\nabla_x f(x)$, which is a matrix in $\mathbb{R}^{n\times k}$. We will frequently abuse notation and for sets $\mathcal{X}$ and functions f, we write $f(\mathcal{X})$ to denote the set $\{f(x) \mid x \in \mathcal{X}\}$. We use $\odot$ to denote the Hadamard product of two vectors, and $\oplus$ to denote the Minkowksi sum of two sets.

Lipschitz constants of vector-valued functions: Given a neural network $f : \mathbb{R}^k \to \mathbb{R}^n$, an input domain $\mathcal{X} \subseteq \mathbb{R}^k$ and norms $\|\cdot\|_\alpha$, $\|\cdot\|_\beta$ over $\mathbb{R}^k, \mathbb{R}^n$ respectively, the local Lipschitz constant is formally defined as:

$$L^{(\alpha,\beta)}(f,\mathcal{X}) := \sup_{x,y\in\mathcal{X}} \frac{\|f(x)-f(y)\|_\beta}{\|x-y\|_\alpha}. \tag{1}$$

When f is continuously differentiable and $\mathcal{X}$ is an open set, this quantity may be written as an optimization over matrix norms of the Jacobian. Letting $\nabla_x f(x)$ be the Jacobian of f evaluated at x, we have that

$$L^{(\alpha,\beta)}(f,\mathcal{X}) = \sup_{x\in\mathcal{X}} \|\nabla_x f(x)\|_{\alpha\to\beta}\,. \tag{2}$$

The matrix norm $\|M\|_{\alpha\to\beta}$ is defined as

$$\|M\|_{\alpha\to\beta} := \sup_{\|v\|_\alpha\le 1} \|Mv\|_\beta = \sup_{\|u\|_{\beta^*}\le 1} \left\|M^T u\right\|_{\alpha^*}. \tag{3}$$

where the second equality follows from the definition of the dual norm. Combining Equations 2 and 3, we can formulate the problem of computing the Lipschitz constant as an optimization over vector-Jacobian products:

$$L^{(\alpha,\beta)}(f,\mathcal{X}) = \sup_{x\in\mathcal{X}} \sup_{u\in B_{\beta^*}} \left\|\nabla_x f(x)^T u\right\|_{\alpha*}. \tag{4}$$

When f is nonsmooth, as in the case of neural networks with ReLU nonlinearities, the optimization over the Jacobian in Equation 4 is replaced with an optimization over Clarke generalized subgradients (Jordan & Dimakis, 2020). For our purposes, as we seek only to upper bound this quantity, this distinction is of minimal importance.

The strategy we employ to upper bound Equation 4 will rely on two clear steps. The first step is to generate a sound approximation of the set of vector-Jacobian products of f, and the second step is to bound the maximal $\|\cdot\|_{\alpha^*}$ norm of this set. Concisely, we first develop a set $\mathcal{Y}$ satisfying the containment

$$\{\nabla_x f(x)^T u \mid x \in \mathcal{X}, \quad u \in B_{\beta^*}\} \subseteq \mathcal{Y}, \tag{5}$$

and then we upper bound the maximal α^* norm of $\mathcal{Y}$.

We will focus in particular on the $L^{(\infty,1)}(f,\mathcal{X})$ Lipschitz constant. The choice of $\|\cdot\|_\alpha$ to be the ℓ_∞ norm is standard in the robustness certification literature. The choice of $\|\cdot\|_\beta$ to be the ℓ_1 norm will yield an upper bound for $L^{(\alpha,p)}(f,\mathcal{X})$ for $p \ge 1$ as the dual ball, B_{β^*} is the ℓ_∞ ball and contains all other ℓ_p balls. This approach relies on mapping ℓ_∞ balls through both the forward and backward pass of a network, which zonotopes are well-suited for.

Vector-Jacobian Products of Neural Networks: To handle the step of overapproximating the set of vector-Jacobian products, we turn our attention to the structure of the functions we consider. We consider feedforward neural networks with either fully-connected or convolutional layers and elementwise nonlinearities, σ, such as the ReLU, tanh, or sigmoid operators. A neural network f with L hidden layers may be evaluated according to the following recursion,

$$f(x) := W_L Z_L(x) + b_L \tag{6}$$

$$Z_i(x) = \sigma\Big(\hat{Z}_i(x)\Big) \tag{7}$$

$$\hat{Z}_i(x) = W_i Z_{i-1}(x) + b_i \tag{8}$$

$$Z_0(x) = x, \tag{9}$$

for i in $\{1,\ldots L\}$. When f has convolutional layers, the affine operator in the definition of $\hat{Z}_i(x)$ may be instantiated as a convolution operator.

当f有卷积层时，Z^i(x)定义中的仿射算子可以实例化为卷积算子。

虽然计算完整的雅可比可能很昂贵，但向量雅可比乘积是通过反向传播算法在神经网络上执行的标准操作。

While the full Jacobian may be expensive to compute, vector-Jacobian products are a standard operation performed on neural nets via the backpropagation algorithm. This may be evaluated according to the following recursion,

$$\nabla_x f(x)^T u := W_1^T J_1(x)^T Y_1(x,u) \tag{10}$$

$$Y_i(x,u) = J_{i+1}^T(x)\hat{Y}_i(x,u) \tag{11}$$

$$\hat{Y}_i(x,u) = W_{i+1}^T Y_{i+1}(x,u) \tag{12}$$

$$Y_L(x,u) = u, \tag{13}$$

where $J_i(x)$ is the Jacobian of the i^{th} nonlinearity with respect to its input, $\nabla_{\hat{Z}_i} Z_i(x)$, and i ranges from 1 to $L-1$. For the standard elementwise nonlinearities, $J_i(x)$ is diagonal and may be written as the Hadamard product with a vector. For example, when σ is the ReLU operator, this is a Hadamard product with a vector taking entries in $\{0,1\}$, depending on the sign of the input to each neuron. When convolutional layers are used in place of fully-connected layers, the transpose convolution operator with no bias terms may be used in place of W_{i+1}^T in the definition of $\hat{Y}_i(x,u)$. For generative models that yield images, the outputs are constrained to the hyperbox $[0,1]^n$, usually by applying a sigmoid or tanh layer to the output of f. In this case, $\hat{Y}_{L-1}(x,u)$ is $W_L^T J_L^T(x)u$, for $J_L(x)$ denoting the Jacobian with respect to this final nonlinear layer.

As our goal is to generate a set $\mathcal{Y}$ satisfying the containment in equation 5, we can unroll the recursions and iteratively construct sets which serve as sound approximations of each element in the recursion. For an input range of $\mathcal{X}$, our algorithm will yield a collection of sets $\mathcal{Z}_i, \hat{\mathcal{Z}}_i, \mathcal{J}_i, \mathcal{Y}_i, \hat{\mathcal{Y}}_i$ satisfying the containments

$$\mathcal{X} \subseteq \mathcal{Z}_0 \qquad B_{\beta^*} \subseteq \hat{\mathcal{Y}}_L \tag{14}$$

$$W_i\mathcal{Z}_{i-1} + b_i \subseteq \hat{\mathcal{Z}}_i \qquad W_i^T\hat{\mathcal{Y}}_i \subseteq \mathcal{Y}_{i-1} \tag{15}$$

$$\sigma(\hat{\mathcal{Z}}_i) \subseteq \mathcal{Z}_i \qquad \mathcal{J}_i \odot \mathcal{Y}_{i+1} \subseteq \hat{\mathcal{Y}}_i \tag{16}$$

$$\nabla_z\sigma(\mathcal{Z}_i) \subseteq \mathcal{J}_i \tag{17}$$

That is, we first overapproximate the range of attainable values of each layer of the neural net, $Z_i(\mathcal{X})$ and $\hat{Z}_i(\mathcal{X})$. This allows us to create sets that contain the true range of gradients for each nonlinearity, $\nabla_{Z_i}\hat{Z}_i(\mathcal{X})$ as per the left column. Then a similar procedure is used to obtain sets which contain the true range of partial vector-Jacobian products as the backpropagation algorithm is performed, i.e. $Y_i(\mathcal{X}, B_{\beta^*})$. Ultimately this will yield a a set that contains the set of attainable vector-Jacobian products $\nabla_x f(\mathcal{X})^T B_{\beta^*}$. Soundness is encapsulated in the following theorem.

Theorem 1. *For feedforward neural networks f, an input set $\mathcal{X}$ and sets $\mathcal{Z}_i, \hat{\mathcal{Z}}_i, \mathcal{J}_i, \mathcal{Y}_i, \hat{\mathcal{Y}}_\rangle$ satisfying the containments in Equations 14-17, the set of vector-Jacobian products satisfies*

$$\{\nabla_x f(x)^T u \mid x \in \mathcal{X} \quad u \in B_{\beta^*}\} \subseteq \mathcal{Y}_0. \tag{18}$$

For such a $\mathcal{Y}_0$, the Lipschitz constant of f may be upper-bounded by maximizing the $\|\cdot\|_{\alpha^*}$ norm over the set $\mathcal{Y}_0$.

Abstracting this slightly, we notice that each of the recursive containments follow one of four forms. The first is the mapping of sets through affine operators as in equations 15. Next we require the mapping of sets through the nonlinearity σ or an elementwise multiplication as in equation 16. Third we have the Jacobian operator of σ as in equation 17. In the sequel we will describe a family of sets that trades off expressiveness with efficiency of representation, and then we develop a technique to perform each of these four operations in a way that satisfies the required containments.

4. Hyperboxes and Zonotopes

The key idea to handle the sound approximations required by Theorem 1 is to introduce a family of sets and develop transformations that are closed under this family of sets and preserve the necessary containment for the four classes of operators. The families of sets we will consider here are hyperboxes and zonotopes. Throughout our Lipschitz estimation procedure, we will frequently make use of the fact that linear programs and Minkowski sums of these sets are efficiently computable.

Hyperboxes: An axis-aligned hyperbox in $\mathbb{R}^d$ may be defined by a center point $c \in \mathbb{R}^d$ and a radius vector $r \in R^d$ with $r \geq 0$, such that the set $H(c,r)$ may be defined as

$$H(c,r) := \{c + r \odot y \mid \|y\|_\infty \leq 1\}.$$

Hyperboxes have very efficient representations, and enjoy many nice computational properties. Namely, linear programs over $H(c,r)$ admit a closed-form solution computable in time $O(d)$ as do Minkowski sums:

$$\max_{x \in H(c,r)} a^T x = a^T c + \|a \odot r\|_1$$

$$H(c_1, r_1) \oplus H(c_2, r_2) = H(c_1 + c_2, r_1 + r_2).$$

Zonotopes: Zonotopes are a family of sets that can be much more expressive than hyperboxes but also enjoy many of the same efficient subroutines. A zonotope may be defined as the image of an affine operator applied to a hyperbox, or equivalently, a Minkowski sum of line segments. Typically a zonotope in $\mathbb{R}^d$ is represented in the G-representation, where a center $c \in \mathbb{R}^d$ and a generator matrix $G \in \mathbb{R}^{d\times m}$ are supplied. The number of columns, m of E is referred to as the degrees of freedom of a zonotope. Formally, these sets are described as

$$Z(c,E) := \{c + Ey \mid \|y\|_\infty \leq 1\}.$$

Linear programs over zonotopes also admit a closed form solution as

$$\max_{x \in Z(c,E)} a^T x = a^T c + \left\|E^T a\right\|_1$$

which follows from the definitions of dual norms. An important and frequently used application of this fact is that coordinate-wise lower and upper bounds may be efficiently computed. Indeed, the smallest hyperbox containing a zonotope may be written as

$$Z(c, E) \subseteq H(c, |E|\vec{1}).$$

where $|E|$ denotes the absolute value operator applied elementwise to E. Similarly, the Minkowski sum of two zonotopes is efficiently computable as

$$Z(c_1, E_1) \oplus Z(c_2, E_2) = Z(c_1 + c_2, E_1 || E_2)$$

where $E_1 || E_2$ is the concatenation of the columns of E_1 with those of E_2.

5. Zonotopes and Sound Pushforward Operators

Now we describe how to construct sound transformations as required by the operations outlined in Equations 14-17.

Affine operators: Both hyperboxes and zonotopes have efficient sound transformations when mapping through affine operators. Consider an affine operator $x \rightarrow Ax + b$. In general, hyperboxes are not closed under affine operation, but the tightest sound operator maps a hyperbox $H(c, r)$ to the hyperbox $H(Ac + b, |A|r)$. Zonotopes, on the other hand, are closed under affine operation and $Z(c, r)$ maps to $Z(Ac + b, AE)$.

Elementwise nonlinearities: In general, zonotopes may not be closed under elementwise nonlinearities. Here we will demonstrate a strategy for these mappings that are optimal in a sense and improve upon the mappings of zonotopes through elementwise nonlinearities from prior works.

In general, the problem we consider is to map a zonotope $Z \subseteq \mathbb{R}^d$ through an operator $\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ where $\Phi(x) = (\phi(x_1), \ldots, \phi(x_d))$. That is, we wish to develop a zonotope Z' satisfying $\Phi(Z) \subseteq Z'$.

The strategy we employ to construct Z' is to retain the structure of Z and incorporate new degrees of freedom. We will scale Z along each coordinate and cover the errors by taking the Minkowski sum with a new zonotope:

$$Z' = (\Lambda \odot Z) \oplus \hat{Z}$$

where Λ is a vector and $\hat{Z}$ represents the a zonotope containing the new degrees of freedom. The following sufficient condition states that this transformation satisfies the desired containment property:

Lemma 1. *For any zonotope $Z \subset \mathbb{R}^d$ and any operator Φ operating over $\mathbb{R}^d$, if $\hat{Z}$ is a zonotope satisfying the containment*

$$\{\Phi(z) - \Lambda \odot z \mid z \in Z\} \subseteq \hat{Z} \tag{19}$$

then

$$\Phi(Z) \subseteq (\Lambda \odot Z) \oplus \hat{Z}.$$

We refer to the set $\{\Phi(z) - \Lambda \odot z \mid z \in Z\}$ as the *residuals*, and reduce the problem to finding a vector Λ and set $\hat{Z}$ containing these residuals. Our strategy is to consider sets $\hat{Z}$ that are axis-aligned hyperboxes, i.e. $\hat{Z} = H(b^*, \mu^*)$. While there are many such residual hyperboxes, a reasonable heuristic would be to choose Λ to yield the smallest hyperbox satisfying Lemma 1.

By considering only transformations that scale each coordinate of Z independently and accounting for the residuals with a hyperbox, it suffices to consider each coordinate individually. In this case, the soundness criterion of Equation 19 reduces to the condition:

$$\{\phi(z_i) - \Lambda_i z_i \mid z \in Z\} \subseteq [b_i - \mu, b_i + \mu].$$

By our heuristic, we would like to minimize the size of the residual interval, 2μ in the above equation. This may be written as a min-max optimization:

$$\min_{\Lambda_i, b_i} \max_{z \in Z} |\phi(z_i) - \Lambda_i z_i - b_i|. \tag{20}$$

Indeed, this may be equivalently be viewed as fitting an affine function $L(z_i) := \Lambda_i z_i + b_i$ to the function $\phi(z_i)$ such that the maximum absolute value deviation between $L(z_i)$ and $\phi(z_i)$ is minimized across all Z. Now assume that the optimal objective value of the above min-max is $2\mu^*$, and the argmin and argmax are (Λ_i^*, b_i^*) and (z_i^*) respectively. Then by definition, we have that

$$\{\phi(z_i) - \Lambda_i^* z_i \mid z \in Z\} \subseteq [b_i^* - \mu^*, b_i^* + \mu^*].$$

By computing the optimal solution to Equation 20 for each coordinate i, we can compute Λ^* and a residual hyperbox $H(b^*, \mu^*)$ satisfying the sufficient condition required by Lemma 1.

It remains to be seen how to efficiently solve the min-max of Equation 20. We consider this problem graphically and notice that any 2-dimensional set of points described as the points with vertical deviation of no more than μ from an affine function $L(z_i)$ is a parallelogram with vertical sides. Indeed, we refer to sets of the form

$$\{(z_i, y_i) \mid z_i \in [l_i, u_i] \quad |y_i - L(z_i)| \leq \mu\}$$

as *vertical parallelograms*, parameterized by the line $L(\cdot)$ and vertical range μ and denoted as $P(L, \mu)$. As we have shown, the min-max can be reduced to an instance of the vertical-parallelogram fitting problem.

Definition 1. *The* ***vertical parallelogram fitting problem*** *asks the following question. Given a 2-dimensional set S, we seek to find the vertical parallelogram with minimal area that contains the set S.*

Since the horizontal range of the provided set S is fixed, the area of a vertical parallelogram hinges only upon the length of its vertical side. We discuss how to solve this problem in the next section.

Applying this problem to the particular form of the min-max in Equation 20, we arrive at the following theorem:

Theorem 2. *When S is the set $\{(z_i, \phi(z_i)) \mid z_i \in [l_i, u_i]\}$, the solution to the vertical parallelogram fitting problem yields the optimal solution to Equation 20. Repeated calls to this subroutine yields the tightest hyperbox fitting the residuals as in equation 19.*

Jacobians of Elementwise Operators: Now we consider sound operators for the Jacobians of these elementwise nonlinearities, which will ultimately yield the sets $\mathcal{J}_i$ as in equation 17. Specifically the goal is to develop a set containing

$$\{\nabla_z \Phi(z) \mid z \in Z\}$$

for any zonotope Z. While the strategy presented above certainly applies to this case, we choose to develop a hyperbox approximation for this containment. As hyperboxes allow for independence of coordinates and the Φ operator is an elementwise operator, this reduces to computing, for each coordinate i, the values:

$$j_i^{(l)} := \min_{z \in Z} \phi'(z_i) \qquad j_i^{(u)} := \max_{z \in Z} \phi'(z_i).$$

When the i^{th} coordinate of Z is bounded in $[l_i, u_i]$, the above minimum and maximum may be solved efficiently for common nonlinearities. Indeed for ReLU, we have that $j_i^{(l)} = \texttt{sign}(l_i)$ and $j_i^{(u)} = \texttt{sign}(u_i)$. For the sigmoid and tanh operators, this is $j_i^{(l)} = \phi'(\max(\{|l_i|, |u_i|\}))$ and $j_i^{(u)} = \phi'(\min(\{|l_i|, |u_i|\})$. This may be computed for every coordinate i and yields the hyperbox with center $\frac{j^{(u)}+j^{(l)}}{2}$ and radius $\frac{j^{(u)}-j^{(l)}}{2}$).

Elementwise multiplication: Using the above machinery, we can handle the elementwise multiplication operator in a sound fashion. Given a zonotope Z and a hyperbox H, we wish to develop a zonotope which contains the set

$$\{x \odot z \mid x \in H, \quad z \in Z\}$$

Parallel to Lemma 1, we employ a strategy where we seek to find the hyperbox that most tightly fits the residual set. This soundness criterion is proved in the following lemma:

Lemma 2. *For any zonotope $Z \subseteq \mathbb{R}^d$ and hyperbox $H \subseteq \mathbb{R}^d$, if $\hat{Z}$ is a zonotope satisfying the containment*

$$\{x \odot z - \Lambda \odot z \mid x \in H \quad z \in Z\} \subseteq \hat{Z} \tag{21}$$

then

$$\{x \odot z \mid x \in H \quad z \in Z\} \subseteq (\Lambda \odot Z) \oplus \hat{Z}.$$

lemma

Since $\odot$ acts elementwise, H is a hyperbox, and we only seek to fit a hyperbox residual, we may again consider each coordinate independently. This reduces to another min-max problem where the maximum is now taken over both Z and H.

$$\min_{\Lambda_i, b_i} \max_{z \in Z, x \in H} |x_i \cdot z_i - \Lambda_i z_i - b_i|. \tag{22}$$

We may again solve this via a reduction vertical-parallelogram fitting problem. We can suppose that z_i is contained in the interval $[l_i^{(z)}, u_i^{(z)}]$ and x_i is contained in the interval $[l_i^{(x)}, u_i^{(x)}]$. Then the following theorem relates the vertical parallelogram fitting problem to the elementwise multiplication operator.

Theorem 3. *When S is the set $\{(z, x \cdot z) \mid l^{(z)} \leq z \leq u^{(z)} \quad l^{(x)} \leq x \leq u^{(x)}\}$, the solution to the vertical-parallelogram fitting problem yields the optimal solution to Equation 22. Repeated calls to this subroutine yields the tightest hyperbox fitting the residuals as in Equation 21.*

theorem In this sense, we may once again compute the vertical parallelogram fit for each coordinate i to generate the scaling factor Λ and residual hyperbox Z'.

6. Vertical-Parallelogram fitting problem

In the cases of elementwise operators or elementwise multiplication by a hyperbox, we have reduced the problem of tightly fitting the residuals to the vertical parallelogram fitting problem. Here we describe our algorithm to solve this problem and illustrate its use on two examples. Prior work has optimally solved this problem for the ReLU nonlinearity, but is unnecessarily loose for differentiable nonlinearities. We provide full derivations for the ReLU, sigmoid, tanh, and absolute value operators in the appendix.

Algorithm Consider some general 2-dimensional set S. We will walk through the major components of our algorithm and describe the necessary subroutines to fit a vertical parallelogram to S. Recall that vertical-parallelograms are parameterized by a scalar vertical side-length, 2μ, and an affine function $L(\cdot)$ which is parameterized by a slope λ and intercept b.

First we compute the vertical-side length μ. Since vertical parallelograms are convex, any vertical parallelogram containing the target set S must contain its convex hull. Assuming that the set S is bounded in the horizontal dimension by $[l_x, u_x]$, the convex hull of S may be decomposed into a convex upper and lower hull, $h^+(x), h^-(x)$ such that

$$\texttt{conv}(S) = \{(x, y) \mid x \in [l_x, u_x], \quad h^-(x) \leq y \leq h^+(x)\},$$

where the upper hull is concave and the lower hull is convex. The vertical deviation between these hulls, which we refer

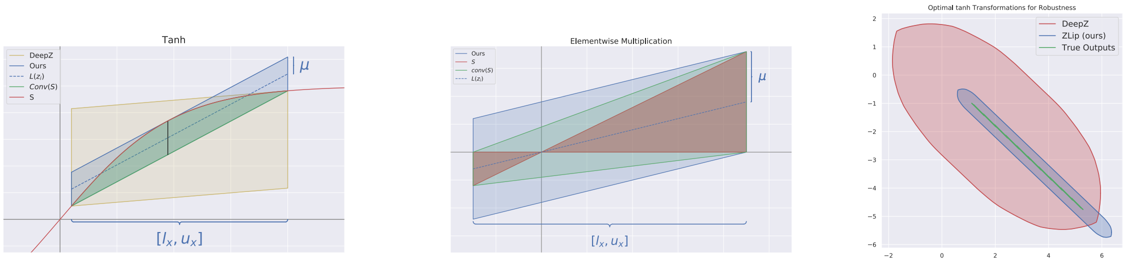

Figure 1. Examples of the vertical parallelogram fitting algorithm for the tanh (left) and elementwise multiplication (middle) operators. The sets S are drawn in red, with their convex hulls in green. The optimal vertical parallelogram for each is drawn in blue. The bound yielded by DeepZ is in yellow for the tanh operator. For classifiers trained on the toy dataset, our improved tanh bounds yield tighter reachability analysis compared to prior work (right). The set of attainable outputs of a network for a specified region is plotted in green, DeepZ yields the set shaded in red, whereas our reachable set is shaded in blue.

to as the altitude at x, is denoted by their difference $h^+(x) - h^-(x)$. The vertical side-length 2μ is lower bounded by the altitude at x for every $x \in [l_x, u_x]$. If the maximum of the altitude across $[l_x, u_x]$ is $2\mu^*$, attained at x^*, the vertical side-length of P must satisfy $2\mu^* \leq 2\mu$. It turns out this is always attainable with equality, i.e. there always exists an affine function L such that $P(L, \mu^*)$ contains S.

For maximum altitude $2\mu^*$ attained at x^*, we compute an affine function L such that the shifted affine function $L(x) + \mu^*$ is greater than $h^+(x)$ for all $x \in [l_x, u_x]$ and vice versa for the lower convex hull, $L(x) - \mu^* \leq h^-(x)$. We first solve for the slope of L. The subgradient of a convex function f, denoted $\delta f(x)$, is the set of slopes of affine functions passing through the point $(x, f(x))$ that lower bound f. Thus, the slope λ must lie in both the subgradient of $h^-(x)$ and $-h^+(x)$. The intersection of these subgradients is guaranteed to be nonempty as x^* minimizes the convex function $h^-(x) - h^+(x)$, implying $0 \in \delta(h^-(x^*) - h^+(x^*))$. Any element in the intersection of these subgradients suffices as a choice for the slope of the fitting parallelogram. We arbitrarily choose such an element and denote it as λ^*.

Finally we compute the intercept of the line L. For the bounds to be tight, the constraint that $L(x) + \mu^* \geq h^+(x)$ must be tight at x^* and likewise for h^-. Hence the line L must pass through the point $(x^*, \frac{h^+(x^*)+h^-(x^*)}{2})$. Provided with a slope λ^*, it is trivial to compute the intercept of a line passing through that point. This concludes the algorithm and yields the desired parallelogram attaining the tightest possible vertical deviation.

The running time of this algorithm depends crucially on the structure of the set S. Computing the upper and lower convex hulls of S may be challenging and algorithms to compute these functions need to be designed by hand. However, the sets we consider admit efficient algorithms to compute these functions. Multiple prior works have correctly computed the convex hull for the set defined by the ReLU operator, but have provided suboptimal bounds for differentiable nonlinearities. To illustrate its use, we will apply our algorithm to the tanh operator and the elementwise multiplication operator.

Tanh example: Consider the $\tanh$ function and the horizontal coordinate bounds $[l_x, u_x]$ where $l_x \geq 0$. In this case, we desire to fit a parallelogram to the set $\{(x, \tanh(x)) \mid x \in [l_x, u_x]\}$. The $\tanh$ function is concave for $x \geq 0$, so the convex hull of S has an upper hull of $h^+(x) = \tanh(x)$. The lower hull, $h^-(x)$, is the secant line passing through the points $(l_x, \tanh(l_x))$ and $(u_x, \tanh(u_x))$, which has slope $\lambda = \frac{\tanh(u_x) - \tanh(l_x)}{u_x - l_x}$. The altitude is maximized when the slope of $\tanh(x)$ is equal to λ, which is attained at $x^* = \tanh^{-1}(\sqrt{1-\lambda})$. This yields the maximal vertical deviation $2\mu^* = \tanh(x^*) - h^-(x^*)$. Since the lower hull is a secant line, the subgradients everywhere are λ, so the slope of $L(\cdot)$ must be λ. The intercept of the parallelogram's line $L(\cdot)$ must pass through the point $(x^*, \tanh(x^*) - \mu^*)$. This is illustrated graphically in figure 1 (left), where the set S is drawn in red and its convex hull is green. The parallelogram yielded by our algorithm is in blue, whereas the parallelogram yielded by prior work is yellow (Singh et al., 2018).

Elementwise multiplication example: Now consider the case of elementwise multiplication. Consider the x-interval with $l_x < 0 < u_x$ multiplied by a value y in the range $[\alpha, \beta]$ for $\alpha \geq 0$. The set $S = \{(x, x \cdot y) \mid l_x \leq x \leq u_x \quad \alpha \leq y \leq \beta\}$ is the union of the two triangles shown in red in Figure 1 (middle). The convex hull of S is the trapezoid, in green in Figure 1, and the upper hull is the line connecting the points $(l_x, \alpha l_x)$ and $(u_x, \alpha u_x)$. The lower hull is the line connecting the points $(l_x, \beta l_x)$ and $(u_x, \alpha u_x)$. The maximum altitude is attained at one of the endpoints: in the example in Figure 1 this is u_x because $u_x > |l_x|$. The altitude is then $(\alpha - \beta) \cdot u_x$. The slope λ must lie between the slopes of the directional derivatives of the upper and lower hulls, yielding an admissible range of $[\frac{\alpha u_x - \beta l_x}{u_x - l_x}, \frac{\beta u_x - \alpha l_x}{u_x - l_x}]$. We choose λ to be the midpoint of this interval. The intercept of the center line may then be chosen such that it passes through the point $(u_x, \frac{\alpha+\beta}{2} u_x)$. We plot the resulting parallelogram in blue.

7. Maximizing Norms over Zonotopes

The final step to upper bounding the Lipschitz constant of a network is to compute a maximization of the $\|\cdot\|_{\alpha^*}$ norm over a zonotope Z, which contains the set of all attainable vector-Jacobian products. While maximizing a convex norm over a convex set may be hard in general, it suffices to upper bound this value. We may always upper bound this norm efficiently by transforming Z into the tightest containing hyperbox and computing the norm over this hyperbox. Any maximal-ℓ_p norm of a hyperbox is efficiently computable, so this technique is quite efficient. However as we typically consider the α to be the ℓ_∞ norm, we focus on techniques to maximize the dual ℓ_1 norm specifically. First we show that this problem is equivalent to computing the Grothendieck problem, i.e. to compute the matrix norm $\|\cdot\|_{\infty \to 1}$.

Theorem 4. *The problem of computing the maximal ℓ_1 norm of a zonotope is equivalent to the $\|\cdot\|_{\infty \to 1}$ matrix norm: both problems are NP-hard in general. Additionally, any approximation algorithm with approximation ratio α for the Grothendieck problem will yield an approximation algorithm with ratio α for the zonotope ℓ_1 maximization problem and vice versa.*

theorem The Grothendieck problem is well-studied and it has been shown that the semidefinite relaxation yields an approximation ratio of < 1.783 (Braverman et al., 2013). However, this relaxation may be quite slow. We present a novel result that states that the linear-programming relaxation for the ℓ_1 zonotope norm maximization problem, and equivalently the Grothendieck problem, may be computed in linear time.

Theorem 5. *For a zonotope, $Z(c, E)$, the linear programming relaxation of $\max_{z \in Z(c,E)} \|z\|_1$ is computable in time $O(|E|)$ where $|E|$ denotes the number of elements in E.*

theorem The proof follows from applying the vertical-parallelogram fitting algorithm to the absolute value operator and then solving a linear program over the resulting zonotope.

8. Experiments

We highlight that our algorithm, which we refer to as ZLip, is specifically designed to provide Lipschitz estimates of networks with large output dimension. However, the approaches outlined above are applicable to scalar functions as well. As much of the literature focuses on classifiers, we first compare our approach on a binary classification task against other Lipschitz estimation techniques. Then we apply ZLip to generative models for MNIST and CIFAR-10. A full description of the experimental details and additional experiments on MNIST and CIFAR-10 classifiers are present in the supplementary.

Toy Network Benchmarks: To fairly compare against existing Lipschitz estimation techniques, we present results on the 2-dimensional Circle dataset from (Aziznejad et al., 2020), where the binary classification is resolved by taking the sign of the output. We consider the $L^{(\infty,|\cdot|)}(f, \mathcal{X})$ Lipschitz constant for fully-connected networks f with input dimension 2 and a varying amount of layers of width 100 and the ReLU nonlinearity. We report the average Lipschitz estimate and compute time for the following techniques: Fast-Lip, LipSDP, SeqLip, CLEVER, and LipMIP. Fast-Lip and LipSDP provide provable upper bounds (Weng et al., 2018a; Fazlyab et al., 2019). SeqLip and CLEVER provide heuristic estimates and LipMIP computes this quantity exactly (Virmaux & Scaman, 2018; Weng et al., 2018b; Jordan & Dimakis, 2020). LipMIP leverages interval analysis as a first step, so we also consider a modified version that instead uses the layerwise approximations yielded by ZLip.

In Figure 2, we plot the reported Lipschitz estimate and runtime of these other techniques applied on input regions that are random hyperboxes of ℓ_∞ radius 0.1 centered at elements in the test set. These plots can demonstrate where each technique lies with respect to the efficiency-accuracy tradeoff. In varying the architecture size, we observe that ZLip yields the tightest provable upper bounds for small networks, and only begins to provide looser bounds than LipSDP at 9 hidden layers, at which point LipSDP is three orders of magnitude slower than ZLip. Additionally, using ZLip in place of interval analysis in LipMIP can provide speedups of up to 100x while preserving the exactness of Lipschitz computation.

Generative Models: We now scale our approach to generative models for the MNIST and CIFAR-10 datasets. We train multiple VAEs and GANs using fully-connected and convolutional layers and the ReLU and tanh nonlinearities (Kingma & Welling, 2013; Radford et al., 2015). To evaluate over VAEs, we consider input sets $\mathcal{X}$ that are ℓ_∞ balls surrounding the encodings of images taken from the test set. For GAN evaluation, we consider ℓ_∞ balls surrounding random inputs from the training distribution and evaluate $L^{(\infty,1)}(f, \mathcal{X})$ of the generator. The only other nontrivial Lipschitz estimation approach that tolerates vector-valued networks and is able to scale to networks of this size is Fast-Lip. Full experimental details are presented in the Appendix, as well as experiments with input sets of different radii.

Table 1 displays results for random inputs with ℓ_∞ radius of

Table 1. Evaluation of ZLip and Fast-Lip on generative models trained on the MNIST and CIFAR-10 datasets, evaluated over inputs of radius 0.05. Times are reported in seconds The column F/Z denotes the ratio of the estimate returned by Fast-Lip to our estimate. For larger networks, ZLip can yield upper bounds that are several orders of magnitude tighter than Fast-Lip.

	MNIST					CIFAR-10				
Network	ZLip		Fast-Lip			ZLip		Fast-Lip		
	Value	Time	Value	Time	F/Z	Value	Time	Value	Time	F/Z
VAESmall	6.81×10^2	0.831	6.30×10^3	0.0017	**10.29**	4.07×10^3	4.97	1.13×10^4	0.0029	**2.79**
VAEMed	2.39×10^3	1.22	1.06×10^6	0.0029	**1024**	8.28×10^3	5.99	1.06×10^6	0.0042	**133.6**
VAEBig	6.56×10^3	1.38	5.71×10^7	0.0042	**46746**	8.39×10^e	5.41	1.74×10^7	0.0054	**2227.8**
VAECNN	8.75×10^2	1.37	3.58×10^4	0.0028	**47.75**	5.97×10^3	10.3	6.16×10^4	0.0031	**10.38**
FFGAN	1.55×10^7	0.455	2.95×10^8	0.0044	**52.99**	1.24×10^7	1.14	1.21×10^8	0.081	**10.01**
DCGAN	4.34×10^5	3.46	2.24×10^7	0.0056	**55.33**	2.11×10^6	8.15	4.63×10^7	0.0127	**31.19**
VAETanh	3.55×10^3	2.17	7.96×10^4	0.0032	**24.19**	2.26×10^3	3.90	2.97×10^5	0.0044	**132.8**
VC-Tanh	2.40×10^3	2.95	7.71×10^4	0.0031	**37.25**	5.76×10^3	1.97	1.86×10^5	0.0033	**32.47**

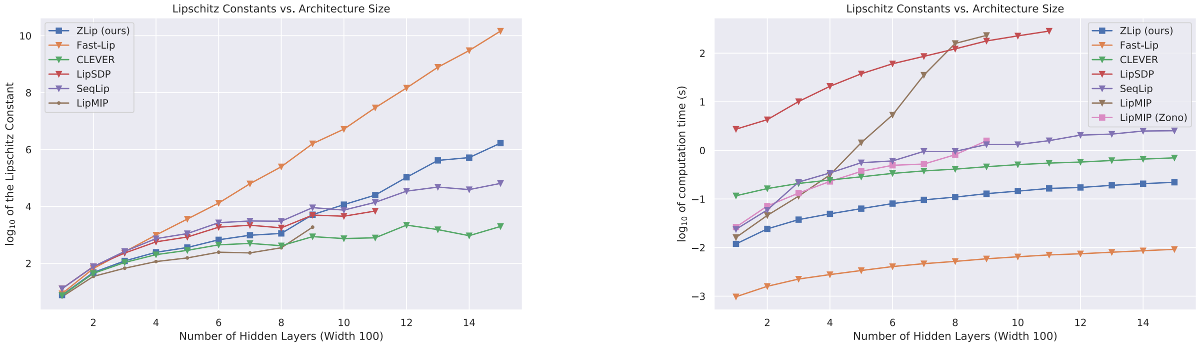

Figure 2. Comparisons of Lipschitz estimate (left) and running time (right) for various Lipschitz estimation techniques on networks trained on the Circle dataset. The horizontal axis is the number of hidden layers of the networks considered and the vertical axis is in log-scale. Our approach is scalable to larger networks while still providing reasonably tight bounds. Also note that using our approach within LipMIP dramatically improves the efficiency.

0.05. We report the average Lipschitz bound and runtime, as well as the ratio of the Fast-Lip value to the ZLip value, denoted F/Z. As the network size scales, we notice that the values reported by both ZLip and Fast-Lip increase. This is likely due to both the true Lipschitz constant of the network increasing as well as the bounds becoming looser. However the bounds provided by ZLip are comparatively much tighter than Fast-Lip for larger networks. We attribute this to the fact that the interval analysis of Fast-Lip introduces error in both the affine and nonlinear layers, whereas our approach only introduces error in the nonlinear layers. While an increase in network size accompanies an increase in runtime for both techniques and ZLip is indeed slower than Fast-Lip, ZLip remains tractable for even the largest networks we consider.

9. Conclusion

We have presented a principled way to efficiently upper bound the Lipschitz constant of neural networks with large output dimension. Our technique improves upon prior works for zonotope approximations of neural networks and is applicable to the operators required by the backpropagation algorithm. This approach yields tighter provable Lipschitz bounds for classifiers and is able to scale effectively to familiar generative models for the MNIST and CIFAR10 datasets, yielding improvements of up to three orders of magnitude for Lipschitz estimation of these networks.

Acknowledgements: This research has been supported by NSF Grants CCF 1763702,1934932, AF 1901292, 2008710, 2019844 research gifts by Western Digital, WNCG IAP, computing resources from TACC and the Archie Straiton Fellowship.

References

Aziznejad, S., Gupta, H., Campos, J., and Unser, M. Deep neural networks with trainable activations and controlled lipschitz constant. January 2020.

Braverman, M., Makarychev, K., Makarychev, Y., and Naor, A. THE GROTHENDIECK CONSTANT IS STRICTLY SMALLER THAN KRIVINE'S BOUND. *Forum of Mathematics, Pi*, 1, 2013.

Chen, T., Lasserre, J.-B., Magron, V., and Pauwels, E. Semialgebraic optimization for lipschitz constants of ReLU networks. February 2020.

Fazlyab, M., Robey, A., Hassani, H., Morari, M., and Pappas, G. Efficient and accurate estimation of lipschitz constants for deep neural networks. In Wallach, H., Larochelle, H., Beygelzimer, A., d Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 11423–11434. Curran Associates, Inc., 2019.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. May 2017.

Jordan, M. and Dimakis, A. G. Exactly computing the local lipschitz constant of ReLU networks. March 2020.

Jordan, M., Lewis, J., and Dimakis, A. G. Provable certificates for adversarial examples: Fitting a ball in the union of polytopes. March 2019.

Kingma, D. P. and Welling, M. Auto-Encoding variational bayes. December 2013.

Latorre, F., Rolland, P., and Cevher, V. Lipschitz constant estimation of neural networks via sparse polynomial optimization. September 2019.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3578–3586, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. November 2015.

Singh, G., Gehr, T., Mirman, M., Püschel, M., and Vechev, M. T. Fast and effective robustness certification. *NeurIPS*, 1(4):6, 2018.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proc. ACM Program. Lang.*, 3(POPL):41:1–41:30, January 2019a.

Singh, G., Gehr, T., Püschel, M., and Vechev, M. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, January 2019b.

Tran, H.-D., Bak, S., Xiang, W., and Johnson, T. T. Verification of deep convolutional neural networks using ImageStars. In *Computer Aided Verification*, pp. 18–42. Springer International Publishing, 2020.

Virmaux, A. and Scaman, K. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 3835–3844. Curran Associates, Inc., 2018.

Weng, T.-W., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Boning, D., Dhillon, I. S., and Daniel, L. Towards fast computation of certified robustness for ReLU networks. April 2018a.

Weng, T.-W., Zhang, H., Chen, P.-Y., Yi, J., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. January 2018b.

Xu, K., Zhang, H., Wang, S., Wang, Y., Jana, S., Lin, X., and Hsieh, C.-J. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. November 2020.

Zhang, H., Weng, T.-W., Chen, P.-Y., Hsieh, C.-J., and Daniel, L. Efficient neural network robustness certification with general activation functions. November 2018.

Zico Kolter, J. and Wong, E. Provable defenses against adversarial examples via the convex outer adversarial polytope. November 2017.