Towards Fast Computation of Certified Robustness for ReLU Networks

Tsui-Wei Weng [*1] **Huan Zhang** [*2] **Hongge Chen** [1] **Zhao Song** [3 4] **Cho-Jui Hsieh** [2] **Duane Boning** [1]
Inderjit S. Dhillon [4] **Luca Daniel** [1]

Abstract

Verifying the robustness property of a general Rectified Linear Unit (ReLU) network is an NP-complete problem. Although finding the *exact* minimum adversarial distortion is hard, giving a *certified lower bound* of the minimum distortion is possible. Current available methods of computing such a bound are either time-consuming or deliver low quality bounds that are too loose to be useful. In this paper, we exploit the special structure of ReLU networks and provide two computationally efficient algorithms (**Fast-Lin**,**Fast-Lip**) that are able to certify non-trivial lower bounds of minimum adversarial distortions. Experiments show that (1) our methods deliver bounds close to (the gap is 2-3X) exact minimum distortions found by Reluplex in small networks while our algorithms are more than 10,000 times faster; (2) our methods deliver similar quality of bounds (the gap is within 35% and usually around 10%; sometimes our bounds are even better) for larger networks compared to the methods based on solving linear programming problems but our algorithms are 33-14,000 times faster; (3) our method is capable of solving large MNIST and CIFAR networks up to 7 layers with more than 10,000 neurons within tens of seconds on a single CPU core. In addition, we show that there is no polynomial time algorithm that can approximately find the minimum ℓ_1 adversarial distortion of a ReLU network with a $0.99 \ln n$ approximation ratio unless NP=P, where n is the number of neurons in the network.

[*]Equal contribution [1]Massachusetts Institute of Technology, Cambridge, MA [2]UC Davis, Davis, CA [3]Harvard University, Cambridge, MA [4]UT Austin, Austin, TX. Full version is available at https://arxiv.org/pdf/1804.09699. Correspondence to: Tsui-Wei Weng <twweng@mit.edu>, Huan Zhang <ecezhang@ucdavis.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

1. Introduction

Since the discovery of adversarial examples in deep neural network (DNN) image classifiers (Szegedy et al., 2013), researchers have successfully found adversarial examples in many machine learning tasks applied to different areas, including object detection (Xie et al., 2017), image captioning (Chen et al., 2018a), speech recognition (Cisse et al., 2017), malware detection (Wang et al., 2017) and reading comprehension (Jia & Liang, 2017). Moreover, black-box attacks have also been shown to be possible, where an attacker can find adversarial examples without knowing the architecture and parameters of the DNN (Chen et al., 2017; Papernot et al., 2017; Liu et al., 2017b).

The existence of adversarial examples poses a huge threat to the application of DNNs in mission-critical tasks including security cameras, self-driving cars and aircraft control systems. Many researchers have thus proposed defensive or detection methods in order to increase the robustness of DNNs. Notable examples are defensive distillation (Papernot et al., 2016), adversarial retraining/training (Kurakin et al., 2017; Madry et al., 2018) and model ensembles (Tramèr et al., 2018; Liu et al., 2017a). Despite many published contributions that aim at increasing the robustness of DNNs, theoretical results are rarely given and there is no guarantee that the proposed defensive methods can reliably improve the robustness. Indeed, many of these defensive mechanism have been shown to be ineffective when more advanced attacks are used (Carlini & Wagner, 2017c;a;b; He et al., 2017).

The robustness of a DNN can be verified by examining a neighborhood (e.g. ℓ_2 or ℓ_∞ ball) near a data point $\boldsymbol{x_0}$. The idea is to find the largest ball with radius r_0 that guarantees no points inside the neighborhood can ever change classifier decision. Typically, r_0 can be found as follows: given R, a global optimization algorithm can be used to find an adversarial example within this ball, and thus bisection on R can produce r_0. Reluplex (Katz et al., 2017) is one example using such a technique but it is computationally infeasible even on a small MNIST classifier. In general, verifying the robustness property of a ReLU network is NP-complete (Katz et al., 2017; Sinha et al., 2018).

On the other hand, a lower bound β_L of radius r_0 can be given, which guarantees that no examples within a ball of ra-

dius β_L can ever change the network classification outcome. (Hein & Andriushchenko, 2017) is a pioneering work on giving such a lower bound for neural networks that are continuously differentiable, although only a 2-layer MLP network with differentiable activations is investigated. (Weng et al., 2018) has extended theoretical result to ReLU activation functions and proposed a robustness score, CLEVER, based on extreme value theory. Their approach is feasible for large state-of-the-art DNNs but CLEVER is an estimate of β_L without certificates. Ideally, we would like to obtain a *certified* (which guarantees that $\beta_L \leq r_0$) and *non-trivial* (a trivial β_L is 0) lower bound β_L that is reasonably close to r_0 within *reasonable amount of computational time*.

In this paper, we develop two fast algorithms for obtaining a *tight* and *certified* lower bound β_L on ReLU networks. In addition, we also provide a complementary theoretical result to (Katz et al., 2017; Sinha et al., 2018) by further showing there does not even exist a polynomial time algorithm that can approximately find the minimum adversarial distortion with a $0.99 \ln n$ approximation ratio. Our contributions are:

- We fully exploit the ReLU networks to give two computationally efficient methods of computing *tighter* and *guaranteed* robustness lower bounds via (1) **lin**ear approximation on the ReLU units (see Sec 3.3, Algorithm 1, **Fast-Lin**) and (2) bounding network local **Lip**schitz constant (see Sec 3.4, Algorithm 2, **Fast-Lip**). Unlike the per-layer operator-norm-based lower bounds which are often very loose (close to 0, as verified in our experiments) for deep networks, our bounds are much closer to the upper bound given by the best adversarial examples, and thus can be used to evaluate the robustness of DNNs with theoretical guarantee.
- We show that our proposed method is at least *four orders of magnitude faster* than finding the exact minimum distortion (with Reluplex), and also around *two orders of magnitude (or more) faster* than linear programming (LP) based methods. We can compute a reasonable robustness lower bound within a minute for a ReLU network with up to 7 layers or over ten thousands neurons, which is so far the best available result in the literature to our best knowledge.
- We show that there is no polynomial time algorithm that can find a lower bound of minimum ℓ_1 adversarial distortion with a $(1 - o(1)) \ln n$ approximation ratio (where n is the total number of neurons) unless NP=P (see Theorem 3.1).

2. Background and related work

2.1. Solving the minimum adversarial distortion

For ReLU networks, the verification problem can be transformed into a Mixed Integer Linear Programming (MILP) problem (Lomuscio & Maganti, 2017; Cheng et al., 2017; Fischetti & Jo, 2017) by using binary variables to encode the states of ReLU activation in each neuron. (Katz et al., 2017) proposed Reluplex based on satisfiable modulo theory, which encodes the network into a set of linear constraints with special rules to handle ReLU activations and splits the problem into two LP problems based on a ReLU's activation status on demand. Similarly, (Ehlers, 2017) proposed Planet, another splitting-based approach using satisfiability (SAT) solvers. These approaches guarantee to find the exact minimum distortion of an adversarial example, and can be used for formal verification. However, due to NP-hard nature of the underlying problem, these approaches only work on very small networks. For example, in (Katz et al., 2017), verifying a feed-forward network with 5 inputs, 5 outputs and 300 total hidden neurons on a single data point can take a few hours. Additionally, Reluplex can find the minimum distortion only in terms of ℓ_∞ norm (ℓ_1 is possible via an extension) and cannot easily generalize to ℓ_p norm.

2.2. Computing lower bounds of minimum distortion

(Szegedy et al., 2013) gives a lower bound on the minimum distortion in ReLU networks by computing the product of weight matrices operator norms, but this bound is usually too loose to be useful in practice, as pointed out in (Hein & Andriushchenko, 2017) and verified in our experiments (see Table F.1). A tighter bound was given by (Hein & Andriushchenko, 2017) using local Lipschitz constant on a network with one hidden layer, but their approach requires the network to be continuously-differentiable, and thus cannot be directly applied to ReLU networks. (Weng et al., 2018) further provide the lower bound guarantee to non-differentiable functions by Lipschitz continuity assumption and propose the first robustness score, CLEVER, that can evaluate the robustness of DNNs and scale to large ImageNet networks. As also shown in our experiments in Section 4, the CLEVER score is indeed a good robustness estimate close to the true minimum distortion given by Reluplex, albeit without providing certificates. Recently, (Wong & Kolter, 2018) propose a convex relaxation on the MILP verification problem discussed in Sec 2.1, which reduces MILP to LP when the adversarial distortion is in ℓ_∞ norm. They focus on adversarial training, and compute layer-wise bounds by looking into the dual LP problem.

2.3. Hardness and approximation algorithms

NP $\neq$ P is the most important and popular assumption in computational complexity in the last several decades. It can be used to show that the decision of the exact case of a problem is hard. However, in several cases, solving one problem approximately is much easier than solving it exactly. For example, there is no polynomial time algorithm to solve the MAX-CUT problem, but there is a simple 0.5-approximation polynomial time algorithm. Previous works (Katz et al., 2017; Sinha et al., 2018) show that there is no polynomial time algorithm to find the minimum adversarial

distortion r_0 exactly. A natural question to ask is: does there exist a polynomial time algorithm to solve the robustness problem approximately? In other words, can we give a lower bound of r_0 with a guaranteed approximation ratio?

From another perspective, NP $\neq$ P only rules out the polynomial running time. Some problems might not even have a sub-exponential time algorithm. To rule out that, the most well-known assumption used is the "Exponential Time Hypothesis" (Impagliazzo et al., 1998). The hypothesis states that 3SAT cannot be solved in sub-exponential time in the worst case. Another example is that while tensor rank calculation is NP-hard (Håstad, 1990), a recent work (Song et al., 2017b) proved that there is no $2^{o(n^{1-o(1)})}$ time algorithm to give a constant approximation of the rank of the tensor. There are also some stronger versions of the hypothesis than ETH, e.g., Strong ETH (Impagliazzo & Paturi, 2001), Gap ETH (Dinur, 2016; Manurangsi & Raghavendra, 2017), and average case ETH (Feige, 2002; Razenshteyn et al., 2016).

3. Robustness guarantees for ReLU networks

Overview of our results. We begin with a motivating theorem in Sec 3.1 showing that there does NOT exist a polynomial time algorithm able to find the minimum adversarial distortion with a $(1 - o(1)) \ln n$ approximation ratio. We then introduce notations in Sec 3.2 and state our main results in Sec 3.3 and 3.4, where we develop two approaches that guarantee to obtain a lower bound of minimum adversarial distortion. In Sec 3.3, we first demonstrate a general approach to *directly* derive the output bounds of a ReLU network with linear approximations when inputs are perturbed by a general ℓ_p norm noise. The analytic output bounds allow us to develop a fast algorithm **Fast-Lin** to compute certified lower bound. In Sec 3.4, we present **Fast-Lip** to obtain a certified lower bound of minimum distortion by deriving upper bounds for the local Lipschitz constant. Both methods are highly efficient and allow fast computation of certified lower bounds on large ReLU networks.

3.1. Finding the minimum distortion with a $0.99 \ln n$ approximation ratio is hard

(Katz et al., 2017) shows that verifying robustness for ReLU networks is NP-complete; in other words, there is no efficient (polynomial time) algorithm to find the exact minimum adversarial distortion. Here, we further show that even *approximately* finding the minimum adversarial distortion with a guaranteed approximation ratio can be hard. Suppose the ℓ_p norm of the true minimum adversarial distortion is r_0, and a robustness verification program A gives a guarantee that no adversarial examples exist within an ℓ_p ball of radius r (r is a lower bound of r_0). The approximation ratio $\alpha := \frac{r_0}{r} > 1$. We hope that α is close to 1 with a guarantee; for example, if α is a constant regardless of the scale of the network, we can always be sure that r_0 is at most α times as large as the lower bound r found by A. Here we relax this requirement and allow the approximation ratio to increase with the number of neurons n. In other words, when n is larger, the approximation becomes more inaccurate, but this "inaccuracy" can be bounded. However, the following theorem shows that no efficient algorithms exist to give a $0.99 \ln n$ approximation in the special case of ℓ_1 robustness:

Theorem 3.1. *Unless* P = NP*, there is no polynomial time algorithm that gives* $(1 - o(1)) \ln n$*-approximation to the* ℓ_1 ReLU *robustness verification problem with* n *neurons.*

Our proof is based on a well-known in-approximability result of SET-COVER problem (Raz & Safra, 1997; Alon et al., 2006; Dinur & Steurer, 2014) and a novel reduction from SET-COVER to our problem. We defer the proof into Appendix A. The formal definition of the ℓ_1 ReLU robustness verification problem can be found in Definition A.7. Theorem 3.1 implies that any efficient (polynomial time) algorithm cannot give better than $(1 - o(1)) \ln n$-approximation guarantee. Moreover, by making a stronger assumption of Exponential Time Hypothesis (ETH), we can state an explicit result about running time using existing results from SET-COVER (Moshkovitz, 2012a;b),

Corollary 3.2. *Under* ETH*, there is no* $2^{o(n^c)}$ *time algorithm that gives* $(1 - o(1)) \ln n$*-approximation to the* ℓ_1 ReLU *robustness verification problem with* n *neurons, where* $c \in (0, 1)$ *is some fixed constant.*

3.2. ReLU Networks and Their Activation Patterns

Let $\boldsymbol{x} \in \mathbb{R}^{n_0}$ be the input vector for an m-layer neural network with $m - 1$ hidden layers and let the number of neurons in each layer be $n_k, \forall k \in [m]$. We use $[n]$ to denote set $\{1, 2, \cdots, n\}$. The weight matrix $\mathbf{W}^{(k)}$ and bias vector $\boldsymbol{b}^{(k)}$ for the k-th layer have dimension $n_k \times n_{k-1}$ and n_k, respectively. Let $\phi_k : \mathbb{R}^{n_0} \to \mathbb{R}^{n_k}$ be the operator mapping from input layer to layer k and $\sigma(\boldsymbol{y})$ be the coordinate-wise activation function; for each $k \in [m-1]$, the relation between layer $k-1$ and layer k can be written as $\phi_k(\boldsymbol{x}) = \sigma(\mathbf{W}^{(k)}\phi_{k-1}(\boldsymbol{x}) + \boldsymbol{b}^{(k)})$, where $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}, \boldsymbol{b}^{(k)} \in \mathbb{R}^{n_k}$. For the input layer and the output layer, we have $\phi_0(\boldsymbol{x}) = \boldsymbol{x}$ and $\phi_m(\boldsymbol{x}) = \mathbf{W}^{(m)}\phi_{m-1}(\boldsymbol{x}) + \boldsymbol{b}^{(m)}$. The output of the neural network is $f(\boldsymbol{x}) = \phi_m(\boldsymbol{x})$, which is a vector of length n_m, and the j-th output is its j-th coordinate, denoted as $f_j(\boldsymbol{x}) = [\phi_m(\boldsymbol{x})]_j$. For ReLU activation, the activation function $\sigma(\boldsymbol{y}) = \max(\boldsymbol{y}, \mathbf{0})$ is an element-wise operation on the input vector $\boldsymbol{y}$.

Given an input data point $\boldsymbol{x_0} \in \mathbb{R}^{n_0}$ and a bounded ℓ_p-norm perturbation $\epsilon \in \mathbb{R}_+$, the input $\boldsymbol{x}$ is constrained in an ℓ_p ball $B_p(\boldsymbol{x_0}, \epsilon) := \{\boldsymbol{x} \mid \|\boldsymbol{x} - \boldsymbol{x_0}\|_p \leq \epsilon\}$. With all possible perturbations in $B_p(\boldsymbol{x_0}, \epsilon)$, the pre-ReLU activation of each neuron has a lower and upper bound $l \in \mathbb{R}$ and $u \in \mathbb{R}$, where $l \leq u$. Let us use $\boldsymbol{l}_r^{(k)}$ and $\boldsymbol{u}_r^{(k)}$ to de-

note the lower and upper bound for the r-th neuron in the k-th layer, and let $\boldsymbol{z}_r^{(k)}$ be its pre-ReLU activation, where $\boldsymbol{z}_r^{(k)} = \mathbf{W}_{r,:}^{(k)}\phi_{k-1}(\boldsymbol{x}) + \boldsymbol{b}_r^{(k)}$, $\boldsymbol{l}_r^{(k)} \le \boldsymbol{z}_r^{(k)} \le \boldsymbol{u}_r^{(k)}$, and $\mathbf{W}_{r,:}^{(k)}$ is the r-th row of $\mathbf{W}^{(k)}$. There are three categories of possible activation patterns – (i) the neuron is always activated: $\mathcal{I}_k^+ := \{r \in [n_k] \mid \boldsymbol{u}_r^{(k)} \ge \boldsymbol{l}_r^{(k)} \ge 0\}$, (ii) the neuron is always inactivated: $\mathcal{I}_k^- := \{r \in [n_k] \mid \boldsymbol{l}_r^{(k)} \le \boldsymbol{u}_r^{(k)} \le 0\}$, and (iii) the neuron could be either activated or inactivated: $\mathcal{I}_k := \{r \in [n_k] \mid \boldsymbol{l}_r^{(k)} < 0 < \boldsymbol{u}_r^{(k)}\}$. Obviously, $\{\mathcal{I}_k^+, \mathcal{I}_k^-, \mathcal{I}_k\}$ is a partition of set $[n_k]$.

3.3. Approach 1 (Fast-Lin): Certified lower bounds via linear approximations

3.3.1. Derivation of the output bounds via linear upper and lower bounds for ReLU

In this section, we propose a methodology to *directly* derive upper bounds and lower bounds of the output of an m-layer feed-forward ReLU network. The central idea is to derive an *explicit* upper/lower bound based on the linear approximations for the neurons in category (iii) and the signs of the weights associated with the activations.

We start with a 2-layers network and then extend it to m layers. The j-th output of a 2-layer network is:

$$f_j(\boldsymbol{x}) = \sum_{r \in \mathcal{I}_1^+, \mathcal{I}_1^-, \mathcal{I}_1} \mathbf{W}_{j,r}^{(2)} \sigma(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) + \boldsymbol{b}_j^{(2)}.$$

For neurons $r \in \mathcal{I}_1^+$, we have $\sigma(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) = \mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}$; for neurons $r \in \mathcal{I}_1^-$, we have $\sigma(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) = 0$. For the neurons in category (iii), we propose to use the following linear upper bound and a linear lower bound to replace the ReLU activation $\sigma(y)$:

$$\frac{u}{u-l}y \le \sigma(y) \le \frac{u}{u-l}(y-l). \tag{1}$$

Let $\boldsymbol{d}_r^{(1)} := \frac{\boldsymbol{u}_r^{(1)}}{\boldsymbol{u}_r^{(1)} - \boldsymbol{l}_r^{(1)}}$, we have

$$\begin{aligned} \boldsymbol{d}_r^{(1)}(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) \le\ & \sigma(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) \\ & \le \boldsymbol{d}_r^{(1)}(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)} - \boldsymbol{l}_r^{(1)}). \end{aligned} \tag{2}$$

To obtain an upper bound and lower bound of $f_j(\boldsymbol{x})$ with (1), set $\boldsymbol{d}_r^{(1)} = 1$ for $r \in \mathcal{I}_1^+$, and we have

$$\begin{aligned} f_j^U(\boldsymbol{x}) = & \sum_{r \in \mathcal{I}_1^+, \mathcal{I}_1} \mathbf{W}_{j,r}^{(2)} \boldsymbol{d}_r^{(1)}(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) \\ & - \sum_{r \in \mathcal{I}_1, \mathbf{W}_{j,r}^{(2)} > 0} \mathbf{W}_{j,r}^{(2)} \boldsymbol{d}_r^{(1)} \boldsymbol{l}_r^{(1)} + \boldsymbol{b}_j^{(2)}, \end{aligned} \tag{3}$$

$$\begin{aligned} f_j^L(\boldsymbol{x}) = & \sum_{r \in \mathcal{I}_1^+, \mathcal{I}_1} \mathbf{W}_{j,r}^{(2)} \boldsymbol{d}_r^{(1)}(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)}) \\ & - \sum_{r \in \mathcal{I}_1, \mathbf{W}_{j,r}^{(2)} < 0} \mathbf{W}_{j,r}^{(2)} \boldsymbol{d}_r^{(1)} \boldsymbol{l}_r^{(1)} + \boldsymbol{b}_j^{(2)}, \end{aligned} \tag{4}$$

where $f_j^L(\boldsymbol{x}) \le f_j(\boldsymbol{x}) \le f_j^U(\boldsymbol{x})$. To obtain $f_j^U(\boldsymbol{x})$, we take the upper bound of $\sigma(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)})$ for $r \in \mathcal{I}_1, \mathbf{W}_{j,r}^{(2)} > 0$ and its lower bound for $r \in \mathcal{I}_1, \mathbf{W}_{j,r}^{(2)} \le 0$. Both cases share a common term of $\boldsymbol{d}_r^{(1)}(\mathbf{W}_{r,:}^{(1)}\boldsymbol{x} + \boldsymbol{b}_r^{(1)})$, which is combined into the first summation term in (3) with $r \in \mathcal{I}_1$. Similarly we get the bound for $f_j^L(\boldsymbol{x})$.

For a general m-layer ReLU network with the linear approximation (1), we will show in Theorem 3.5 that the network output can be bounded by two explicit functions when the input $\boldsymbol{x}$ is perturbed with a ϵ-bounded ℓ_p noise. We start by defining the activation matrix $\mathbf{D}^{(k)}$ and the additional equivalent bias terms $\mathbf{T}^{(k)}$ and $\mathbf{H}^{(k)}$ for the k-th layer in Definition 3.3 and the two explicit functions in 3.4.

Definition 3.3 ($\mathbf{A}^{(k)}, \mathbf{T}^{(k)}, \mathbf{H}^{(k)}$)**.** *Given matrices $\mathbf{W}^{(k)} \in \mathbb{R}^{n_k \times n_{k-1}}$ and vectors $\boldsymbol{b}^{(k)} \in \mathbb{R}^{n_k}, \forall k \in [m]$. We define $\mathbf{D}^{(0)} \in \mathbb{R}^{n_0 \times n_0}$ as an identity matrix. For each $k \in [m-1]$, we define matrix $\mathbf{D}^{(k)} \in \mathbb{R}^{n_k \times n_k}$ as follows*

$$\mathbf{D}_{r,r}^{(k)} = \begin{cases} \frac{\boldsymbol{u}_r^{(k)}}{\boldsymbol{u}_r^{(k)} - \boldsymbol{l}_r^{(k)}} & \text{if } r \in \mathcal{I}_k; \\ 1 & \text{if } r \in \mathcal{I}_k^+; \\ 0 & \text{if } r \in \mathcal{I}_k^-. \end{cases} \tag{5}$$

We define matrix $\mathbf{A}^{(m-1)} \in \mathbb{R}^{n_m \times n_{m-1}}$ to be $\mathbf{W}^{(m)}\mathbf{D}^{(m-1)}$, and for each $k \in \{m-1, m-2, \cdots, 1\}$, matrix $\mathbf{A}^{(k-1)} \in \mathbb{R}^{n_m \times n_{k-1}}$ is defined recursively as $\mathbf{A}^{(k-1)} = \mathbf{A}^{(k)}\mathbf{W}^{(k)}\mathbf{D}^{(k-1)}$. For each $k \in [m-1]$, we define matrices $\mathbf{T}^{(k)}, \mathbf{H}^{(k)} \in \mathbb{R}^{n_k \times n_m}$, where

$$\mathbf{T}_{r,j}^{(k)} = \begin{cases} \boldsymbol{l}_r^{(k)} & \text{if } r \in \mathcal{I}_k,\ \mathbf{A}_{j,r}^{(k)} > 0; \\ 0 & \text{otherwise .} \end{cases}$$

$$\mathbf{H}_{r,j}^{(k)} = \begin{cases} \boldsymbol{l}_r^{(k)} & \text{if } r \in \mathcal{I}_k,\ \mathbf{A}_{j,r}^{(k)} < 0; \\ 0 & \text{otherwise .} \end{cases}$$

Definition 3.4 (Two explicit functions : $f^U(\cdot)$ and $f^L(\cdot)$)**.** *Let matrices $\mathbf{A}^{(k)}$, $\mathbf{T}^{(k)}$ and $\mathbf{H}^{(k)}$ be defined as in Definition 3.3. We define two functions $f^U, f^L : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$ as follows. For each input vector $\boldsymbol{x} \in \mathbb{R}^{n_0}$,*

$$f_j^U(\boldsymbol{x}) = \mathbf{A}_{j,:}^{(0)}\boldsymbol{x} + \boldsymbol{b}_j^{(m)} + \sum_{k=1}^{m-1} \mathbf{A}_{j,:}^{(k)}(\boldsymbol{b}^{(k)} - \mathbf{T}_{:,j}^{(k)}),$$

$$f_j^L(\boldsymbol{x}) = \mathbf{A}_{j,:}^{(0)}\boldsymbol{x} + \boldsymbol{b}_j^{(m)} + \sum_{k=1}^{m-1} \mathbf{A}_{j,:}^{(k)}(\boldsymbol{b}^{(k)} - \mathbf{H}_{:,j}^{(k)}).$$

Now, we are ready to state our main theorem,

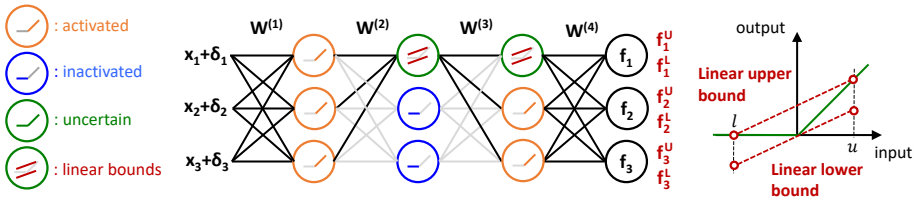

Figure 1. Illustration of deriving output bounds for ReLU networks in Section 3.3. The final output upper bounds (f_j^U) and lower bounds (f_j^L) can be derived by considering the activation status of the neurons with input perturbation $\|\delta\|_p \leq \epsilon$. For neurons in $\mathcal{I}_k^+$, their outputs are identical to their inputs; for neurons in $\mathcal{I}_k^-$, they can be removed during computation as their outputs are always zero; for neurons in $\mathcal{I}_k$, their outputs can be bounded by corresponding linear upper bounds and lower bounds considering the signs of associated weights.

Theorem 3.5 (Explicit upper and lower bounds)**.** *Given an m-layer* ReLU *neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, there exists two explicit functions $f^L : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$ and $f^U : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$ (see Definition 3.4) such that $\forall j \in [n_m]$, $f_j^L(\boldsymbol{x}) \leq f_j(\boldsymbol{x}) \leq f_j^U(\boldsymbol{x})$, $\forall \boldsymbol{x} \in B_p(\boldsymbol{x_0}, \epsilon)$.*

The proof of Theorem 3.5 is in Appendix B. Since the input $\boldsymbol{x} \in B_p(\boldsymbol{x_0}, \epsilon)$, we can maximize (3) and minimize (4) within this set to obtain a global upper and lower bound of $f_j(\boldsymbol{x})$, which has analytical solutions for any $1 \leq p \leq \infty$ and the result is formally shown in Corollary 3.7 (proof in Appendix C). In other words, we have *analytic* bounds that can be computed efficiently without resorting to any optimization solvers for general ℓ_p distortion, and this is the key to enable fast computation for layer-wise output bounds.

We first formally define the global upper bound γ_j^U and lower bound γ_j^L of $f_j(\boldsymbol{x})$, and then obtain Corollary 3.7.

Definition 3.6 (γ_j^L, γ_j^U)**.** *Given a point $\boldsymbol{x_0} \in \mathbb{R}^{n_0}$, a neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, parameters p, ϵ. Let matrices $\mathbf{A}^{(k)}$, $\mathbf{T}^{(k)}$ and $\mathbf{H}^{(k)}$, $\forall k \in [m-1]$ be defined as in Definition 3.3. We define γ_j^L, γ_j^U, $\forall j \in [n_m]$ as*

$$\gamma_j^L = \mu_j^- + \nu_j - \epsilon \|\mathbf{A}_{j,:}^{(0)}\|_q \text{ and } \gamma_j^U = \mu_j^+ + \nu_j + \epsilon \|\mathbf{A}_{j,:}^{(0)}\|_q,$$

where $1/p + 1/q = 1$ and ν_j, μ_j^+, μ_j^- are defined as

$$\mu_j^+ = -\sum_{k=1}^{m-1} \mathbf{A}_{j,:}^{(k)} \mathbf{T}_{:,j}^{(k)}, \quad \mu_j^- = -\sum_{k=1}^{m-1} \mathbf{A}_{j,:}^{(k)} \mathbf{H}_{:,j}^{(k)} \tag{6}$$

$$\nu_j = \mathbf{A}_{j,:}^{(0)} \boldsymbol{x_0} + \boldsymbol{b}_j^{(m)} + \sum_{k=1}^{m-1} \mathbf{A}_{j,:}^{(k)} \boldsymbol{b}^{(k)} \tag{7}$$

Corollary 3.7 (Two side bounds in closed-form)**.** *Given a point $\boldsymbol{x_0} \in \mathbb{R}^{n_0}$, an m-layer neural network function $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_m}$, parameters p and ϵ. For each $j \in [n_m]$, there exist two fixed values γ_j^L and γ_j^U (see Definition 3.6) such that $\gamma_j^L \leq f_j(\boldsymbol{x}) \leq \gamma_j^U$, $\forall \boldsymbol{x} \in B_p(\boldsymbol{x}_0, \epsilon)$.*

3.3.2. Computing pre-ReLU activation bounds

Theorem 3.5 and Corollary 3.7 give us a global lower bound γ_j^L and upper bound γ_j^U of the j-th neuron at the m-th layer if we know all the pre-ReLU activation bounds $\boldsymbol{l}^{(k)}$ and $\boldsymbol{u}^{(k)}$, from layer 1 to $m-1$, as the construction of $\mathbf{D}^{(k)}$, $\mathbf{H}^{(k)}$ and $\mathbf{T}^{(k)}$ requires $\boldsymbol{l}^{(k)}$ and $\boldsymbol{u}^{(k)}$ (see Definition 3.3). Here, we show how this can be done easily and layer-by-layer. We start from $m = 1$ where $\mathbf{A}^{(0)} = \mathbf{W}^{(1)}$, $f^U(\boldsymbol{x}) = f^L(\boldsymbol{x}) = \mathbf{A}^{(0)}\boldsymbol{x} + \boldsymbol{b}^{(1)}$. Then, we can apply Corollary 3.7 to get the output bounds of each neuron and set them as $\boldsymbol{l}^{(1)}$ and $\boldsymbol{u}^{(1)}$. Then, we can proceed to $m = 2$ with $\boldsymbol{l}^{(1)}$ and $\boldsymbol{u}^{(1)}$ and compute the output bounds of second layer by Corollary 3.7 and set them as $\boldsymbol{l}^{(2)}$ and $\boldsymbol{u}^{(2)}$. Repeating this procedure for all $m-1$ layers, we will get all the $\boldsymbol{l}^{(k)}$ and $\boldsymbol{u}^{(k)}$ needed to compute the output range of the m-th layer.

Note that when computing $\boldsymbol{l}^{(k)}$ and $\boldsymbol{u}^{(k)}$, the constructed $\mathbf{W}^{(k)}\mathbf{D}^{(k-1)}$ can be saved and reused for bounding the next layer, which facilitates efficient implementations. Moreover, the time complexity of computing the output bounds of an m-layer ReLU network with Theorem 3.5 and Corollary 3.7 is *polynomial* time in contrast to the approaches in (Katz et al., 2017) and (Lomuscio & Maganti, 2017) where SMT solvers and MIO solvers have *exponential* time complexity. The major computation cost is to form $\mathbf{A}^{(0)}$ for the m-th layer, which involves multiplications of layer weights in a *similar cost of forward propagation*. See the "ComputeTwoSideBounds" procedure in Algorithm 1 in Appendix D.

3.3.3. Deriving maximum certified lower bounds of minimum adversarial distortion

Suppose c is the predicted class of the input data point $\boldsymbol{x_0}$ and the class is j. With Theorem 3.5, the maximum possible lower bound for the targeted attacks $\widetilde{\epsilon}_j$ and un-targeted attacks $\widetilde{\epsilon}$ are

$$\widetilde{\epsilon}_j = \max_{\epsilon} \epsilon \text{ s.t. } \gamma_c^L(\epsilon) - \gamma_j^U(\epsilon) > 0 \text{ and } \widetilde{\epsilon} = \min_{j \neq c} \widetilde{\epsilon}_j.$$

Though it is hard to get analytic forms of $\gamma_c^L(\epsilon)$ and $\gamma_j^U(\epsilon)$ in terms of ϵ, fortunately, we can still obtain $\widetilde{\epsilon}_j$ via a binary search. This is because Corollary 3.7 allows us to efficiently compute the numerical values of $\gamma_c^L(\epsilon)$ and $\gamma_j^U(\epsilon)$ given ϵ. It is worth noting that we can further improve the bound by considering $g(\boldsymbol{x}) := f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$ at the last layer and apply the same procedure to compute the lower bound of

$g(\boldsymbol{x})$ (denoted as $\widetilde{\gamma}^L$); this can be done easily by redefining the last layer's weights to be a row vector $\bar{\boldsymbol{w}} := \mathbf{W}^{(m)}_{c,:} - \mathbf{W}^{(m)}_{j,:}$. The corresponding maximum possible lower bound for the targeted attacks is $\widetilde{\epsilon}_j = \max \epsilon$ s.t. $\widetilde{\gamma}^L(\epsilon) > 0$. We list our complete algorithm, **Fast-Lin**, in Appendix D.

3.3.4. Discussions

We have shown how to derive explicit output bounds of ReLU network (Theorem 3.5) with the proposed linear approximations and obtain analytical certified lower bounds (Corollary 3.7), which is the key of our proposed algorithm **Fast-Lin**. (Wong & Kolter, 2018) presents a similar algorithmic result on computing certified bounds, but our framework and theirs are entirely different – we use direct computation of layer-wise linear upper/lower bounds in Sec 3.3 with binary search on ϵ, while their results is achieved via the lens of dual LP formulation with Newton's method. Interestingly, when we choose a special set of lower and upper bounds as in (2) and they choose a special dual LP variable in their equation (8), the two different frameworks coincidentally produce the same procedure for computing layer-wise bounds (the "ComputeTwoSideBounds" procedure in **Fast-Lin** and Algorithm 1 in (Wong & Kolter, 2018)). However, our choice of bounds (2) is due to computation efficiency, while (Wong & Kolter, 2018) gives a quite different justification. We encourage the readers to read Appendix A.3 in their paper on the justifications for this specific selection of dual variables and understand this robustness verification problem from different perspectives.

3.4. Approach 2 (Fast-Lip): Certified lower bounds via bounding the local Lipschitz constant

(Weng et al., 2018) shows a non-trivial lower bound of minimum adversarial distortion for an input example $\boldsymbol{x_0}$ in targeted attacks is $\min\left(g(\boldsymbol{x_0})/L^j_{q,x_0}, \epsilon\right)$, where $g(\boldsymbol{x}) = f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$, L^j_{q,x_0} is the local Lipschitz constant of $g(\boldsymbol{x})$ in $B_p(\boldsymbol{x_0}, \epsilon)$, j is the target class, c is the original class, and $1/p + 1/q = 1$. For un-targeted attacks, the lower bound can be presented in a similar form. (Weng et al., 2018) uses sampling techniques to estimate the local Lipschitz constant and compute an estimated lower bound without certificates.

Here, we propose a new algorithm to compute a *certified* lower bound of the minimum adversarial distortion by upper bounding the local Lipschitz constant. To start with, let us rewrite the relations of subsequent layers in the following form: $\phi_k(\boldsymbol{x}) = \boldsymbol{\Lambda}^{(k)}(\mathbf{W}^{(k)}\phi_{k-1}(\boldsymbol{x}) + \boldsymbol{b}^{(k)})$, where $\sigma(\cdot)$ is replaced by the diagonal activation pattern matrix $\boldsymbol{\Lambda}^{(k)}$ that encodes the status of neurons r in k-th layer:

$$\boldsymbol{\Lambda}^{(k)}_{r,r} = \begin{cases} 1 \text{ or } 0 & \text{if } r \in \mathcal{I}_k \\ 1 & \text{if } r \in \mathcal{I}^+_k \\ 0 & \text{if } r \in \mathcal{I}^-_k \end{cases} \tag{8}$$

and $\boldsymbol{\Lambda}^{(m)} = \boldsymbol{I}_{n_m}$. With a slight abuse of notation, let us define $\boldsymbol{\Lambda}^{(k)}_a$ as a diagonal activation matrix for neurons in the k-th layer who are always **a**ctivated, i.e. the r-th diagonal is 1 if $r \in \mathcal{I}^+_k$ and 0 otherwise, and $\boldsymbol{\Lambda}^{(k)}_u$ as the diagonal activation matrix for k-th layer neurons whose status are **u**ncertain, i.e. the r-th diagonal is 1 or 0 (to be determined) if $r \in \mathcal{I}_k$, and 0 otherwise. Therefore, we have $\boldsymbol{\Lambda}^{(k)} = \boldsymbol{\Lambda}^{(k)}_a + \boldsymbol{\Lambda}^{(k)}_u$. We can obtain $\boldsymbol{\Lambda}^{(k)}$ for $\boldsymbol{x} \in B_p(\boldsymbol{x_0}, \epsilon)$ by applying Algorithm 1 and check the lower and upper bounds for each neuron r in layer k.

3.4.1. A general upper bound of Lipschitz constant in ℓ_q norm

The central idea is to compute upper bounds of L^j_{q,x_0} by exploiting the three categories of activation patterns in ReLU networks when the allowable inputs are in $B_p(\boldsymbol{x_0}, \epsilon)$. $L^j_{q,\boldsymbol{x}_0}$ can be defined as the maximum norm of directional derivative as shown in (Weng et al., 2018). For the ReLU network, the maximum directional derivative norm can be found by examining all the possible activation patterns and take the one (the worst-case) that results in the largest gradient norm. However, as all possible activation patterns grow exponentially with the number of the neurons, it is impossible to examine all of them in brute-force. Fortunately, computing the worst-case pattern on each element of $\nabla g(\boldsymbol{x})$ (i.e. $[\nabla g(x)]_k$, $k \in [n_0]$) is much easier and more efficient. In addition, we apply a simple fact that the maximum norm of a vector (which is $\nabla g(\boldsymbol{x}), \boldsymbol{x} \in B_p(\boldsymbol{x_0}, \epsilon)$ in our case) is upper bounded by the norm of the maximum value for each components. By computing the worst-case pattern on $[\nabla g(\boldsymbol{x})]_k$ and its norm, we can obtain an upper bound of the local Lipschitz constant, which results in a certified lower bound of minimum distortion.

Below, we first show how to derive an upper bound of the Lipschitz constant by computing the worst-case activation pattern on $[\nabla g(\boldsymbol{x})]_k$ for 2 layers. Next, we will show how to apply it repeatedly for a general m-layer network, and the algorithm is named **Fast-Lip**. Note that for simplicity, we will use $[\nabla f_j(\boldsymbol{x})]_k$ to illustrate our derivation; however, it is easy to extend to $[\nabla g(\boldsymbol{x})]_k$ as $g(\boldsymbol{x}) = f_c(\boldsymbol{x}) - f_j(\boldsymbol{x})$ by simply replacing last layer weight vector by $\mathbf{W}^{(m)}_{c,:} - \mathbf{W}^{(m)}_{j,:}$.

Bounds for a 2-layer ReLU Network. The gradient is:

$$[\nabla f_j(\boldsymbol{x})]_k = \mathbf{W}^{(2)}_{j,:}\boldsymbol{\Lambda}^{(1)}_a\mathbf{W}^{(1)}_{:,k} + \mathbf{W}^{(2)}_{j,:}\boldsymbol{\Lambda}^{(1)}_u\mathbf{W}^{(1)}_{:,k}.$$

The first term $\mathbf{W}^{(2)}_{j,:}\boldsymbol{\Lambda}^{(1)}_a\mathbf{W}^{(1)}_{:,k}$ is a constant and all we need to bound is the second term $\mathbf{W}^{(2)}_{j,:}\boldsymbol{\Lambda}^{(1)}_u\mathbf{W}^{(1)}_{:,k}$. Let $\mathbf{C}^{(1)}_{j,k} = \mathbf{W}^{(2)}_{j,:}\boldsymbol{\Lambda}^{(1)}_a\mathbf{W}^{(1)}_{:,k}$, $\mathbf{L}^{(1)}_{j,k}$ and $\mathbf{U}^{(1)}_{j,k}$ be the lower and upper bounds of the second term, we have

$$\mathbf{L}^{(1)}_{j,k} = \sum_{i\in\mathcal{I}_1, \mathbf{W}^{(2)}_{j,i}\mathbf{W}^{(2)}_{i,k}<0} \mathbf{W}^{(2)}_{j,i}\mathbf{W}^{(2)}_{i,k},\ \mathbf{U}^{(1)}_{j,k} = \sum_{i\in\mathcal{I}_1, \mathbf{W}^{(2)}_{j,i}\mathbf{W}^{(2)}_{i,k}>0} \mathbf{W}^{(2)}_{j,i}\mathbf{W}^{(2)}_{i,k}$$

$$\max_{\boldsymbol{x}\in B_p(\boldsymbol{x_0},\epsilon)} |[\nabla f_j(\boldsymbol{x})]_k| \le \max(|\mathbf{C}^{(1)}_{j,k}+\mathbf{L}^{(1)}_{j,k}|, |\mathbf{C}^{(1)}_{j,k}+\mathbf{U}^{(1)}_{j,k}|).$$

Bounds for 3 layers or more. For 3 or more layers, we can apply the above 2-layer results recursively, layer-by-layer. For example, for a 3-layer ReLU network,

$$[\nabla f_j(\boldsymbol{x})]_k = \mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}\mathbf{W}^{(2)}\mathbf{\Lambda}^{(1)}\mathbf{W}^{(1)}_{:,k},$$

if we let $\mathbf{Y}^{(1)}_{:,k} = \mathbf{W}^{(2)}\mathbf{\Lambda}^{(1)}\mathbf{W}^{(1)}_{:,k}$, then $[\nabla f_j(\boldsymbol{x})]_k$ is reduced to the following form that is similar to 2 layers:

$$[\nabla f_j(\boldsymbol{x})]_k = \mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}\mathbf{Y}^{(1)}_{:,k} \tag{9}$$

$$= \mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}_a\mathbf{Y}^{(1)}_{:,k} + \mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}_u\mathbf{Y}^{(1)}_{:,k} \tag{10}$$

To obtain the bound in (9), we first need to obtain a lower bound and upper bound of $\mathbf{Y}^{(1)}_{:,k}$, where we can directly apply the 2-layer results to get an upper and an lower bound for each component i as $\mathbf{C}^{(1)}_{i,k}+\mathbf{L}^{(1)}_{i,k} \le \mathbf{Y}^{(1)}_{i,k} \le \mathbf{C}^{(1)}_{i,k}+\mathbf{U}^{(1)}_{i,k}$. Next, the first term $\mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}_a\mathbf{Y}^{(1)}_{:,k}$ in (10) can be lower bounded and upper bounded respectively by

$$\sum_{i\in\mathcal{I}_2^+} \mathbf{W}^{(3)}_{j,i}\mathbf{C}^{(1)}_{i,k} + \sum_{i\in\mathcal{I}_2^+,\mathbf{W}^{(3)}_{j,i}>0} \mathbf{W}^{(3)}_{j,i}\mathbf{L}^{(1)}_{i,k} + \sum_{i\in\mathcal{I}_2^+,\mathbf{W}^{(3)}_{j,i}<0} \mathbf{W}^{(3)}_{j,i}\mathbf{U}^{(1)}_{i,k} \tag{11}$$

$$\sum_{i\in\mathcal{I}_2^+} \mathbf{W}^{(3)}_{j,i}\mathbf{C}^{(1)}_{i,k} + \sum_{i\in\mathcal{I}_2^+,\mathbf{W}^{(3)}_{j,i}>0} \mathbf{W}^{(3)}_{j,i}\mathbf{U}^{(1)}_{i,k} + \sum_{i\in\mathcal{I}_2^+,\mathbf{W}^{(3)}_{j,i}<0} \mathbf{W}^{(3)}_{j,i}\mathbf{L}^{(1)}_{i,k} \tag{12}$$

whereas the second term $\mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}_u\mathbf{Y}^{(1)}_{:,k}$ in (10) is bounded by $\sum_{i\in\mathcal{P}} \mathbf{W}^{(3)}_{j,i}(\mathbf{C}^{(1)}_{i,k}+\mathbf{L}^{(1)}_{i,k}) + \sum_{i\in\mathcal{Q}} \mathbf{W}^{(3)}_{j,i}(\mathbf{C}^{(1)}_{i,k}+\mathbf{U}^{(1)}_{i,k})$ with lower/upper bound index sets $\mathcal{P}_L$, $\mathcal{Q}_L$ and $\mathcal{P}_U$, $\mathcal{Q}_U$:

$$\begin{aligned}
\mathcal{P}_L &= \{i \mid i\in\mathcal{I}_2, \mathbf{W}^{(3)}_{j,i}>0, \mathbf{C}^{(1)}_{i,k}+\mathbf{L}^{(1)}_{i,k}<0\},\\
\mathcal{Q}_L &= \{i \mid i\in\mathcal{I}_2, \mathbf{W}^{(3)}_{j,i}<0, \mathbf{C}^{(1)}_{i,k}+\mathbf{U}^{(1)}_{i,k}>0\}; \qquad (13)\\
\mathcal{P}_U &= \{i \mid i\in\mathcal{I}_2, \mathbf{W}^{(3)}_{j,i}<0, \mathbf{C}^{(1)}_{i,k}+\mathbf{L}^{(1)}_{i,k}<0\},\\
\mathcal{Q}_U &= \{i \mid i\in\mathcal{I}_2, \mathbf{W}^{(3)}_{j,i}>0, \mathbf{C}^{(1)}_{i,k}+\mathbf{U}^{(1)}_{i,k}>0\}. \qquad (14)
\end{aligned}$$

Let $\mathbf{C}^{(2)}_{j,k} = \sum_{i\in\mathcal{I}_2^+} \mathbf{W}^{(3)}_{j,i}\mathbf{C}^{(1)}_{i,k}$, $\mathbf{U}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k}$ and $\mathbf{L}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k}$ be the upper and lower bound of $[\nabla f_j(\boldsymbol{x})]_k$, we have

$$\mathbf{U}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k} = (12)+(14) \text{ and } \mathbf{L}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k} = (11)+(13),$$

$$\max_{\boldsymbol{x}\in B_p(\boldsymbol{x_0},\epsilon)} |[\nabla f_j(\boldsymbol{x})]_k| \le \max(|\mathbf{L}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k}|, |\mathbf{U}^{(2)}_{j,k}+\mathbf{C}^{(2)}_{j,k}|).$$

Thus, this technique can be used iteratively to solve $\max_{\boldsymbol{x}\in B_p(\boldsymbol{x_0},\epsilon)} |[\nabla f_j(\boldsymbol{x})]_k|$ for a general m-layer network, and we can easily bound any q norm of $\nabla f_j(\boldsymbol{x})$ by the q norm of the vector of maximum values. For example,

$$\max_{\boldsymbol{x}\in B_p(\boldsymbol{x_0},\epsilon)} \|\nabla f_j(\boldsymbol{x})\|_q \le \left(\sum_k (\max_{\boldsymbol{x}\in B_p(\boldsymbol{x_0},\epsilon)} |[\nabla f_j(\boldsymbol{x})]_k|)^q\right)^{\frac{1}{q}}$$

We list our full procedure, **Fast-Lip**, in Appendix D.

Further speed-up. Note that in the 3-layer example, we compute the bounds from right to left, i.e. we first get the bound of $\mathbf{W}^{(2)}\mathbf{\Lambda}^{(1)}\mathbf{W}^{(1)}_{:,k}$, and then bound $\mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}\mathbf{Y}^{(1)}_{:,k}$. Similarly, we can compute the bounds from left to right – get the bound of $\mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}\mathbf{W}^{(2)}$ first, and then bound $\mathbf{Y}^{(2)}_{j,:}\mathbf{\Lambda}^{(1)}\mathbf{W}^{(1)}_{:,k}$, where $\mathbf{Y}^{(2)}_{j,:} = \mathbf{W}^{(3)}_{j,:}\mathbf{\Lambda}^{(2)}\mathbf{W}^{(2)}$. Since the dimension of the output layer (n_m) is typically far less than the dimension of the input vector (n_0), computing the bounds from left to right is more efficient as the matrix $\mathbf{Y}$ has a smaller dimension of $n_m \times n_k$ rather than $n_k \times n_0$.

4. Experiments

In this section, we perform extensive experiments to evaluate the performance of our proposed two lower-bound based robustness certificates on networks with different sizes and with different defending techniques during training process. Specifically, we compare our proposed bounds[1] (**Fast-Lin**, **Fast-Lip**) with Linear Programming (LP) based methods (**LP**, **LP-Full**), formal verification methods (**Reluplex**), lower bound by global Lipschitz constant (**Op-norm**), estimated lower bounds (**CLEVER**) and attack algorithms (**Attacks**) for toy networks (2-3 layers with 20 neurons in each layer) and large networks (2-7 layers with 1024 or 2048 neurons in each layer) in Table 1. The evaluation on the effects of defending techniques is presented in Table 2. All bound numbers are the average of 100 random test images with random attack targets, and running time (per image) for all methods is measured on a single CPU core. We include detailed setup of experiments, descriptions of each method, additional experiments and discussions in Appendix F (See Tables F.1 and F.2). The results suggest that our proposed robustness certificates are of high qualities and are computationally efficient even in large networks up to 7 layers or more than 10,000 neurons. In particular, we show that:

- Our certified lower bounds (**Fast-Lin**, **Fast-Lip**) are close to (gap is only 2-3X) the exact minimum distortion computed by **Reluplex** for small networks (**Reluplex** is only feasible for networks with less 100 neurons for MNIST), but our algorithm is more than 10,000 times faster than **Reluplex**. See Table 1a and Table F.1.
- Our certified lower bounds (**Fast-Lin**, **Fast-Lip**) give similar quality (the gap is within 35%, and usually around 10%; sometimes our bounds are even better) compared with the LP-based methods (**LP**, **LP-Full**); however, our algorithm is 33 - 14,000 times faster. The LP-based methods are infeasible for networks with more than 4,000 neurons. See Table 1b and Table F.2.
- When the network goes larger and deeper, our proposed methods can still give non-trivial lower bounds comparing to the upper bounds founded by attack algorithms on large

[1] https://github.com/huanzhang12/CertifiedReLURobustness

Table 1. Comparison of methods of computing certified lower bounds (**Fast-Lin**, **Fast-Lip**, **LP**, **LP-Full**,**Op-norm**), estimated lower bound (**CLEVER**), exact minimum distortion (**Reluplex**) and upper bounds (**Attack**: CW for $p = 2, \infty$, EAD for $p = 1$) on (a) 2, 3 layers *toy* MNIST networks with 20 neurons per layer and (b) *large* networks with 2-7 layers, 1024 or 2048 nodes per layer. Differences of lower bounds and speedup are measured on the best bound from our proposed algorithms and **LP**-based approaches (the **bold** numbers in each row). In (a), we show how close our fast bounds are to exact minimum distortions (**Reluplex**) and the bounds that are slightly tighter but very expensive (**LP-Full**). In (b), **LP-Full** and **Reluplex** are *computationally infeasible* for all the networks reported here.

Toy Networks			Average Magnitude of Distortions on 100 Images							
Network	p	Target	Certified Lower Bounds				difference	Exact	Uncertified	
			Our bounds		Our Baselines		ours vs.	Reluplex	CLEVER	Attacks
			Fast-Lin	Fast-Lip	LP	LP-Full	LP(-Full)	(Katz et al., 2017)	(Weng et al., 2018)	CW/EAD
MNIST $2 \times [20]$	∞	rand	**0.0309**	0.0270	**0.0319**	0.0319	-3.2%	0.07765	0.0428	0.08060
	2	rand	**0.6278**	0.6057	0.7560	**0.9182**	-31.6%	-	0.8426	1.19630
	1	rand	3.9297	**4.8561**	4.2681	**4.6822**	+3.7%	-	5.858	11.4760
MNIST $3 \times [20]$	∞	rand	**0.0229**	0.0142	0.0241	**0.0246**	-6.9%	0.06824	0.0385	0.08114
	2	rand	**0.4652**	0.3273	0.5345	**0.7096**	-34.4%	-	0.7331	1.22570
	1	rand	**2.8550**	2.8144	3.1000	**3.5740**	-20.1%	-	4.990	10.7220

(a) Toy networks. **Reluplex** is designed to verify ℓ_∞ robustness so we omit its numbers for $p = 2, 1$.

Large Networks		Average Magnitude of Distortion on 100 Images							Average Running Time per Image			
Network	p	Certified Bounds				diff	Uncertified		Certified Bounds			Speedup
		Our bounds		LP	Op-norm	ours	CLEVER	Attacks	Our bounds		LP	ours
		Fast-Lin	Fast-Lip	(Baseline)	(Szegedy et al., 2013)	vs. LP	(Weng et al., 2018)	CW/EAD	Fast-Lin	Fast-Lip	(Baseline)	vs. LP
MNIST $2 \times [1024]$	∞	**0.03083**	0.02512	**0.03386**	0.00263	-8.9%	0.0708	0.1291	**156 ms**	219 ms	**20.8 s**	133X
	2	**0.63299**	0.59033	**0.75164**	0.40201	-15.8%	1.2841	1.8779	**128 ms**	234 ms	**195 s**	1523X
	1	3.88241	**5.10000**	**4.47158**	0.35957	+14.1%	7.4186	17.259	139 ms	**1.40 s**	**48.1 s**	34X
MNIST $3 \times [1024]$	∞	**0.02216**	0.01236	**0.02428**	0.00007	-8.7%	0.0717	0.1484	**1.12 s**	1.11 s	**52.7 s**	47X
	2	**0.43892**	0.26980	**0.49715**	0.10233	-11.7%	1.2441	2.0387	**906 ms**	914 ms	**714 s**	788X
	1	**2.59898**	2.25950	**2.91766**	0.01133	-10.9%	7.2177	17.796	**863 ms**	3.84 s	**109 s**	126X
MNIST $4 \times [1024]$	∞	**0.00823**	0.00264	-	0.00001	-	0.0793	0.1303	**2.25 s**	3.08 s	-	-
	2	**0.18891**	0.06487	-	0.17734	-	1.4231	1.8921	**2.37 s**	2.72 s	-	-
	1	**1.57649**	0.72800	-	0.00183	-	8.9764	17.200	**2.42 s**	2.91 s	-	-
CIFAR $5 \times [2048]$	∞	**0.00170**	0.00030	-	0.00000	-	0.0147	0.02351	**26.2 s**	78.1 s	-	-
	2	**0.07654**	0.01417	-	0.00333	-	0.6399	0.9497	**36.8 s**	49.4 s	-	-
	1	**1.18928**	0.31984	-	0.00000	-	9.7145	21.643	**37.5 s**	53.6 s	-	-
CIFAR $6 \times [2048]$	∞	**0.00090**	0.00007	-	0.00000	-	0.0131	0.01866	**37.0 s**	119 s	-	-
	2	**0.04129**	0.00331	-	0.01079	-	0.5860	0.7635	**60.2 s**	95.6 s	-	-
	1	**0.72178**	0.08212	-	0.00000	-	8.2507	17.160	**61.4 s**	88.2 s	-	-
CIFAR $7 \times [1024]$	∞	**0.00134**	0.00008	-	0.00000	-	0.0112	0.0218	**10.6 s**	29.2 s	-	-
	2	**0.05938**	0.00407	-	0.00029	-	0.5145	0.9730	**16.9 s**	27.3 s	-	-
	1	**0.86467**	0.09239	-	0.00000	-	8.630	22.180	**17.6 s**	26.7 s	-	-

(b) Larger networks. "**-**" indicates the corresponding method is computationally infeasible for that network.

Table 2. Comparison of the lower bounds for ℓ_∞ distortion found by our algorithms on models with defensive distillation (DD) (Papernot et al., 2016) with temperature = 100 and adversarial training (Madry et al., 2018) with $\epsilon = 0.3$ for three targeted attack classes.

		runner-up target			random target			least-likely target		
Network	Method	Undefended	DD	Adv. Training	Undefended	DD	Adv. Training	Undefended	DD	Adv. Training
MNIST 3*[1024]	Fast-Lin	0.01826	0.02724	**0.14730**	0.02211	0.03827	**0.17275**	0.02427	0.04967	**0.20136**
	Fast-Lip	0.00965	0.01803	0.09687	0.01217	0.02493	0.11618	0.01377	0.03207	0.13858
MNIST 4*[1024]	Fast-Lin	0.00715	0.01561	**0.09579**	0.00822	0.02045	**0.11209**	0.00898	0.02368	**0.12901**
	Fast-Lip	0.00087	0.00585	0.04133	0.00145	0.00777	0.05048	0.00183	0.00903	0.06015

networks. See Table 1b and Table F.2.

- For defended networks, especially for adversarial training (Madry et al., 2018), our methods give significantly larger bounds, validating the effectiveness of this defending method. Our algorithms can thus be used for evaluating defending techniques. See Table 2.

5. Conclusions

In this paper we have considered the problem of verifying the robustness property of ReLU networks. By exploiting the special properties of ReLU networks, we have here presented two computational efficient methods **Fast-Lin** and **Fast-Lip** for this problem. Our algorithms are two orders of magnitude (or more) faster than LP-based methods, while obtaining solutions with similar quality; meanwhile, our bounds qualities are much better than the previously proposed operator-norm based methods. Additionally, our methods are efficient and easy to implement: we compute the bounds layer-by-layer, and the computation cost for each layer is similar to the cost of matrix products in forward propagation; moreover, we do not need to solve any integer programming, linear programming problems or their duals. Future work could extend our algorithm to handle the structure of convolutional layers and apply our algorithm to evaluate the robustness property of large DNNs such as ResNet on the ImageNet dataset.

Acknowledgment

The authors sincerely thank Aviad Rubinstein for the suggestion of using set-cover to prove hardness. The authors sincerely thank Dana Moshkovitz for pointing out some references about the hardness result of set-cover. The authors would also like to thank Mika Göös, Rasmus Kyng, Zico Kolter, Jelani Nelson, Eric Price, Milan Rubinstein, Jacob Steinhardt, Zhengyu Wang, Eric Wong and David P. Woodruff for useful discussions. Luca Daniel and Tsui-Wei Weng acknowledge the partial support of MIT-Skoltech program and MIT-IBM Watson AI Lab. Huan Zhang and Cho-Jui Hsieh acknowledge the support of NSF via IIS-1719097 and the computing resources provided by Google Cloud and NVIDIA.

References

Ailon, N., Bhattacharya, A., Jaiswal, R., and Kumar, A. Approximate clustering with same-cluster queries. In *ITCS*, 2018.

Alon, N., Moshkovitz, D., and Safra, S. Algorithmic construction of sets for k-restrictions. *ACM TALG*, 2(2): 153–177, 2006.

Ambühl, C., Mastrolilli, M., and Svensson, O. Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. *SIAM Journal on Computing*, 40(2):567–596, 2011.

Arora, S. and Safra, S. Probabilistic checking of proofs: A new characterization of np. *JACM*, 45(1):70–122, 1998.

Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and the hardness of approximation problems. *JACM*, 45(3):501–555, 1998.

Carlini, N. and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *AISec CCS*, 2017a.

Carlini, N. and Wagner, D. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*, 2017b.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, 2017c.

Chen, H., Zhang, H., Chen, P.-Y., Yi, J., and Hsieh, C.-J. Show-and-fool: Crafting adversarial examples for neural image captioning. In *ACL*, 2018a.

Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., and Hsieh, C.-J. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *AISec*, 2017.

Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., and Hsieh, C.-J. Ead: Elastic-net attacks to deep neural networks via adversarial examples. In *AAAI*, 2018b.

Cheng, C.-H., Nührenberg, G., and Ruess, H. Maximum resilience of artificial neural networks. *arXiv preprint arXiv:1705.01040*, 2017.

Cisse, M. M., Adi, Y., Neverova, N., and Keshet, J. Houdini: Fooling deep structured visual and speech recognition models with adversarial examples. In *NIPS*, 2017.

Cohen-Addad, V., De Mesmay, A., Rotenberg, E., and Roytman, A. The bane of low-dimensionality clustering. In *SODA*. SIAM, 2018.

Dinur, I. Mildly exponential reduction from gap 3sat to polynomial-gap label-cover. In *ECCC*, 2016.

Dinur, I. and Steurer, D. Analytical approach to parallel repetition. In *STOC*. ACM, 2014.

Ehlers, R. Formal verification of piece-wise linear feed-forward neural networks. In *ATVA*, 2017.

Feige, U. Relations between average case complexity and approximation complexity. In *STOC*. ACM, 2002.

Fischetti, M. and Jo, J. Deep neural networks as 0-1 mixed integer linear programs: A feasibility study. *arXiv preprint arXiv:1712.06174*, 2017.

Håstad, J. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

He, W., Wei, J., Chen, X., Carlini, N., and Song, D. Adversarial example defenses: Ensembles of weak defenses are not strong. In *USENIX WOOT*, 2017.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. *arXiv preprint arXiv:1705.08475*, 2017.

Impagliazzo, R. and Paturi, R. On the complexity of k-sat. *Journal of Computer and System Sciences*, 62(2): 367–375, 2001.

Impagliazzo, R., Paturi, R., and Zane, F. Which problems have strongly exponential complexity? In *FOCS*. IEEE, 1998.

Jia, R. and Liang, P. Adversarial examples for evaluating reading comprehension systems. In *EMNLP*, 2017.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *CAV*, 2017.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *ICLR*, 2017.

首篇 Lipschitz 认证

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. Towards robust neural networks via random self-ensemble. *arXiv preprint arXiv:1712.00673*, 2017a.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017b.

Lokshtanov, D., Marx, D., and Saurabh, S. Lower bounds based on the exponential time hypothesis. *Bulletin of EATCS*, 3(105), 2013.

Lomuscio, A. and Maganti, L. An approach to reachability analysis for feed-forward relu neural networks. *arXiv preprint arXiv:1706.07351*, 2017.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Manurangsi, P. and Raghavendra, P. A birthday repetition theorem and complexity of approximating dense csps. In *ICALP*, 2017.

Moshkovitz, D. The projection games conjecture and the np-hardness of ln n-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 276–287. Springer, 2012a.

Moshkovitz, D. The projection games conjecture and the np-hardness of ln n-approximating set-cover. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 276–287. Springer, 2012b.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. Practical black-box attacks against machine learning. In *AsiaCCS*, 2017.

Raz, R. and Safra, S. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *STOC*. ACM, 1997.

Razenshteyn, I., Song, Z., and Woodruff, D. P. Weighted low rank approximations with provable guarantees. In *STOC*, 2016.

Sinha, A., Namkoong, H., and Duchi, J. Certifiable distributional robustness with principled adversarial training. In *ICLR*, 2018.

Song, Z., Woodruff, D. P., and Zhong, P. Low rank approximation with entrywise ℓ_1-norm error. In *STOC*. ACM, 2017a.

Song, Z., Woodruff, D. P., and Zhong, P. Relative error tensor low rank approximation. *arXiv preprint arXiv:1704.08246*, 2017b.

Song, Z., Woodruff, D. P., and Zhong, P. Towards a zero-one law for entrywise low rank approximation. 2018.

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.

Tramèr, F., Kurakin, A., Papernot, N., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

Wang, Q., Guo, W., Zhang, K., Ororbia II, A. G., Xing, X., Liu, X., and Giles, C. L. Adversary resistant deep neural networks with an application to malware detection. In *SIGKDD*. ACM, 2017.

Weng, T.-W., Zhang, H., Chen, P.-Y., Jinfeng, Y., Su, D., Gao, Y., Hsieh, C.-J., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. In *ICLR*, 2018.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*. https://arxiv.org/pdf/1711.00851v2, 2018.

Xie, C., Wang, J., Zhang, Z., Zhou, Y., Xie, L., and Yuille, A. Adversarial examples for semantic segmentation and object detection. In *ICCV*, 2017.

Clever