Proceedings of Machine Learning Research vol 144:1–9, 2021

Certifying robustness of neural networks via optimization

Yuexin Bian YUBIAN@UCSD.EDU
University of California, San Diego

Abstract

Though neural network has strong represent ability of non-linear mapping and many work has seen its impressive performance on diverse tasks, it fails catastrophically in the presence of adversarial inputs. As a result, certifying the robustness of neural networks against input uncertainties has attracted surging attention in recent years. To ensure the certificate of robustness, researchers have been dedicated to estimating the bound of the worst-case margin of neural networks via optimization. They utilize linear programming, semidefinite programming and some other techniques to make the original tight bound estimate problem (non-convex) more tractable. In this paper, we provide the description of methods in previous work and show how the work connect each other. We also reproduce the method in previous work and test on a number of tasks.

Keywords: certificate robustness for neural network, semidefinite programming, optimization.

1. Introduction

Goal: This review is mainly based on material in Wong and Kolter (2018); Raghunathan et al. (2018b); Fazlyab et al. (2020); Batten et al. (2021). We try to illustrate the methodology, provide further exposition and connections between those works, and reconstruct the methods and show them in examples.

2. Related Work

Certification methods which evaluate the robustness of a given network can be separated into two categories. The first category utilizes formal verification techniques, aims to provide tight certificates for any network using discrete optimization. These works typically employ either Satisfiability Modulo Theories (SMT) solvers Ehlers (2017); Huang et al. (2017) or integer programming approaches Lomuscio and Maganti (2017); Tjeng et al. (2017); Cheng et al. (2017). However, the main drawback is that, while these techniques provide *exact tight* certificates on neural networks, they are often very slow and the complexity scales exponentially with the size of the network in the worst-case.

The review will mainly focus on the material in second category, which leverages convex optimization, aims at computing *tractable bounds* on the possible perturbation regions of neural networks. A notable work is Wong and Kolter (2018), in which the authors propose a linear-programming (LP) relaxation of piece-wise linear networks and provide upper bounds on the worst-case loss. Raghunathan et al. (2018a) propose an semidefinite programming (SDP) relaxation of one-layer sigmoid-based neural networks based on bounding the worst-case loss with a first-order Taylor expansion. Raghunathan et al. (2018b) proposes a semidefinite relaxation for certifying robustness of piece-wise linear multi-layer neural networks. Fazlyab et al. (2020) accordingly proposes a SDP framework via Quadratic Constraints, which achieves tighter bound in deep networks. Batten et al. (2021) introduced layer-based semidefinite relaxation for verifying robustness of neural networks, and showed the new SDP relaxation is provably tighter than previous work.

3. Problem Formulation

Neural network classifier model. Consider a function $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_f}$ parameterized by a feed-forward neural network of the form

$$x_0 = x \quad x_{k+1} = \phi_k(W_k x_k + b_k) \quad k = 0, ..., L-1 \quad f(x_0) = x_L \tag{1}$$

where $W_k \in \mathbb{R}^{n_{k+1} \times n_k}, b_k \in \mathbb{R}^{n_{k+1}}$ are the weight and bias of the k-th layer, $n_0 = n_x, n_l = n_f$, and $\phi_k : \mathbb{R}^{n_{k+1}} \to \mathbb{R}^{n_{k+1}}$ is the layer of activation functions. When f functions as a n_f-class classifier, then the class label $i(x) = \arg\max_i f(x)_i$, where $f(x)_i = e_i^\top f(x)$, e_i denotes the vector with a 1 in i-th coordinate and 0's elsewhere.

Certificate of robustness for neural network classifier. For a clean input label pair $(\overline{x}, \overline{y})$, if the neural network classifies correctly, then $f(x)_{\overline{y}} > f(x)_y, \forall y \neq \overline{y}$. When the input is adversarially perturbed $x \in N_\epsilon(\overline{x})$, where $N_\epsilon(\overline{x}) = \{x | \|x - \overline{x}\|_\infty \leq \epsilon\}$, the output of the neural network may change. The neural network maintains robust when all samples in the neighborhood of a given input $\overline{x}$ are classified with the same label $\overline{y}$, $f(x)_{\overline{y}} > f(x)_y, \forall x \in N_\epsilon(\overline{x}), y \neq \overline{y}$.

Let $l_y^*(\overline{x}, \overline{y})$ denote the worst-case margin of an incorrect class y that can be achieved by adversarial perturbation:

$$l_y^*(\overline{x}, \overline{y}) = \max_{x \in N_\epsilon(\overline{x})} f(x)_y - f(x)_{\overline{y}} = \max_{x \in N_\epsilon(\overline{x})} (e_y - e_{\overline{y}}) f(x). \tag{2}$$

A network is *certifiably robust* (Raghunathan et al., 2018b) on $(\overline{x}, \overline{y})$ if $l_y^*(\overline{x}, \overline{y}) < 0, \forall y \neq \overline{y}$. Since computing $l_y^*(\overline{x}, \overline{y})$ for a neural network involves solving a non-convex optimization problem, which is intractable in general. The robustness of f can be quantified by estimating an upper bound of $l_y^*(\overline{x}, \overline{y})$. If the upper bound $L_y(\overline{x}, \overline{y})$ have $l_y^*(\overline{x}, \overline{y}) \leq L_y(\overline{x}, \overline{y}) \leq 0$, then we have a certificate of robustness of the network on input $(\overline{x}, \overline{y})$.

4. Approaches

The perturbed input $x \in N_\epsilon(\overline{x})$ is a convex set, to estimate the upper bound of $l_y^*(\overline{x}, \overline{y})$, researchers resort to relax the non-convex equalities resulted from activation functions ($x_{k+1} = \phi_k(W_k x_k + b_k)$). In this section, we mainly focus on the ReLU activation functions. We will also have a discussion on other activation functions at the end of this section.

4.1. LP relaxation

Fig 1 illustrates the relaxation of ReLU function. For ReLU activation, we have $\phi(x) = \max(x, 0)$. Obviously, it's a non-convex function. The authors Wong and Kolter relax the output into the triangular convex sets. Consider the input x is bounded by l and u, then the output could represent as

$$\phi(x) \geq x, \phi(x) \geq 0, \phi(x) \leq \frac{\phi(u) - \phi(l)}{u - l}(x - l) + \phi(l). \tag{3}$$

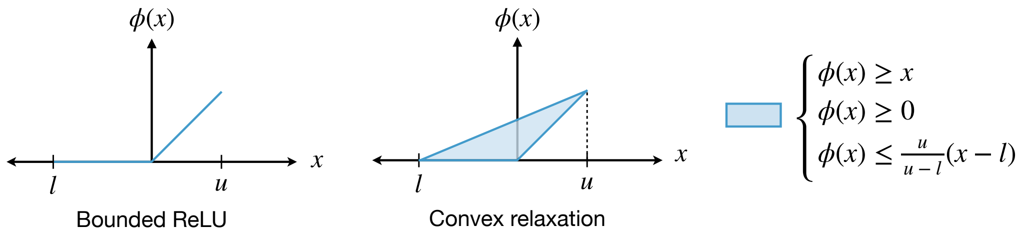

Figure 1: Illustration of the convex triangular ReLU relaxation over the bounded set $[l, u]$, where $l < 0, u > 0$.

Using the relaxation (3), we could reformulate the optimization problem for computing the lower bound of $l_y^*(\overline{x}, \overline{y})$ into a linear program (LP),

$$\begin{aligned} l_{\text{LP}} = \max \quad & c^\top x_L && \text{(4a)} \\ \text{s.t.} \quad & \|x_0 - \overline{x}\|_\infty \leq \epsilon, \text{(Input constraint)} && \text{(4b)} \\ & x_{k+1} \geq 0, x_{k+1} \geq W_k x_k + b_k, k \in [L], \text{(ReLU relaxation)} && \text{(4c)} \\ & x_{k+1} \leq \frac{\phi(u_k) - \phi(l_k)}{u_k - l_k}(W_k x_k + b_k - l_k) + \phi(l_k), k \in [L]. \text{(ReLU relaxation)} && \text{(4d)} \end{aligned}$$

where $c^\top = e_y - e_{\overline{y}}$, $[L] = \{0, \ldots, L-1\}$. l_k, u_k are the lower bound and upper bound of the input of k-th activation layer, respectively. $l_k = \min(x_k), u_k = \max(x_k)$ can be computed directly:

$$\begin{aligned} l_0 &= \overline{x} - \epsilon \mathbf{1}, & u_0 &= \overline{x} - \epsilon \mathbf{1}, && \text{(5a)} \\ l_{k+1} &= [W_k]_+ l_k + [W_k]_- u_k + b_k, & u_{k+1} &= [W_k]_+ u_k + [W_k]_- l_k + b_k, k \in [L]. && \text{(5b)} \end{aligned}$$

where $([W]_+)_{ij} = \max(W_{ij}, 0)$ and $([W]_-)_{ij} = \min(W_{ij}, 0)$.

4.2. SDP relaxation

The ReLU equality can be expressed equivalently as the three linear and quadratic constraints between x and $\phi(x)$:

$$\phi(x) \geq x, \phi(x) \geq 0, \phi(x)(\phi(x) - x) = 0. \tag{6}$$

The first two linear constraints ensure that the output of activation layer is at least as large as both 0 and x, the third quadratic constraint ensures that $\phi(x)$ is equal to x or 0. This key observation leads researchers Raghunathan et al. to reformulate the non-trivial optimization problem (2) into a quadratically constrained quadratic program (QCQP), and later the QCQP can be relaxed into a semidefinite program (SDP).

First, we can transform the linear input constraint to the quadratic constraint,

$$\|x_0 - \overline{x}\|_\infty \leq \epsilon \Leftrightarrow l_0 \leq x_0 \leq u_0 \Leftrightarrow (x_0 - l_0) \cdot (x_0 - u_0) \leq 0 \Leftrightarrow x_0^2 \leq (l_0 + u_0) \cdot x_0 - l_0 \cdot u_0 \tag{7}$$

l_0, u_0 can be computed using equation (5a). Similarly, for $k \in [L]$, we have $x_k^2 \leq (\bar{l}_k + \bar{u}_k) \cdot x_k - \bar{l}_k \cdot \bar{u}_k$, where $\bar{l}_k, \bar{u}_k$ are the lower bound and upper bound of the output of k-th activation layer, respectively. We have $\bar{l}_k = \phi_k(l_k), \bar{u}_k = \phi_k(u_k)$. The non-convex QCQP can be formulated using (7) together with (6), noted that f_{QCQP} is equivalent to $l_y^*(\overline{x}, \overline{y})$:

$$\begin{aligned} f_{\text{QCQP}} = \max \quad & c^\mathsf{T} x_L && \text{(8a)} \\ \text{s.t.} \quad & \text{for} \quad k \in [L], && \\ & x_k^2 \leq (\bar{l}_k + \bar{u}_k) \cdot x_k - \bar{l}_k \cdot \bar{u}_k, \text{(Input constraint)} && \text{(8b)} \\ & x_{k+1} \geq 0, x_{k+1} \geq W_k x_k + b_k, \text{(ReLU constraint, linear)} && \text{(8c)} \\ & x_{k+1}^2 = x_{k+1} \cdot (W_k x_k + b_k). \text{(ReLU constraint, quadratic)} && \text{(8d)} \end{aligned}$$

where $\cdot$ represents the elementwise product operation.

The key idea of relaxing the non-convex QCQP (8) to a convex SDP is to introduce a new *convex set* of variables representing all linear and quadratic monomials in $x_0, \dots, x_L$, and then constraints can be represented as linear functions of the new variables. From this idea, matrix P is formulated as follows:

$$\begin{aligned} P &= vv^\mathsf{T} \\ &= \begin{bmatrix} P[1] & P[x_0^\mathsf{T}] & P[x_1^\mathsf{T}] & \dots & P[x_L^\mathsf{T}] \\ P[x_0] & P[x_0 x_0^\mathsf{T}] & P[x_0 x_1^\mathsf{T}] & \dots & P[x_0 x_L^\mathsf{T}] \\ P[x_1] & P[x_1 x_0^\mathsf{T}] & P[x_1 x_1^\mathsf{T}] & \dots & P[x_1 x_L^\mathsf{T}] \\ \dots & \dots & \dots & \dots & \dots \\ P[x_L] & P[x_L x_0^\mathsf{T}] & P[x_L x_1^\mathsf{T}] & \dots & P[x_L x_L^\mathsf{T}] \end{bmatrix} \end{aligned} \tag{9}$$

where $v = [1, x_0, x_1, \dots, x_L]$. We can substitute the variable w.r.t. $x_0, x_1, \dots x_L$ in (8) by the new variable which is the entries in the matrix P. For instance, x_0^2 is equivalent to diag$(P[x_0 x_0^\mathsf{T}])$, $x_0 = P[x_0]$.

The source of non-convexity comes from the rank-one constraint on matrix P. It can be relaxed such that $P = VV^\mathsf{T} \succeq 0$, where V can be full rank. Since the new set $P = VV^\mathsf{T} \succeq 0$ is convex and is a superset of the original non-convex set ($P \supseteq P' = vv^\mathsf{T}$), the following formulated relaxation would result in $f_{\text{SDP}} \geq f_{QCQP} = l_y^*(\overline{x}, \overline{y})$. This formulation provides an upper bound for which $l_y^*(\overline{x}, \overline{y})$ can serve as a certificate of robustness.

$$\begin{aligned} f_{\text{SDP}} = \max \quad & c^\mathsf{T} P[x_L] && \text{(10a)} \\ \text{s.t.} \quad & \text{for} \quad k \in [L], && \\ & \text{diag}(P[x_k x_k^\mathsf{T}]) \leq (\bar{l}_k + \bar{u}_k) \cdot P[x_k] - \bar{l}_k \cdot \bar{u}_k, \text{(Input constraint)} && \text{(10b)} \\ & P[x_{k+1}] \geq 0, P[x_{k+1}] \geq W_k P[x_k] + b_k, \text{(ReLU constraint)} && \text{(10c)} \\ & \text{diag}(P[x_{k+1} x_{k+1}^\mathsf{T}]) = \text{diag}(W_k P[x_k x_{k+1}^\mathsf{T}]) + b_k \cdot P[x_{k+1}]. \text{(ReLU constraint)} && \text{(10d)} \\ & P[1] = 1, P \succeq 0. \text{(convex relaxation)} && \text{(10e)} \end{aligned}$$

4.3. SDP relaxation with modifications

In section 4.2, matrix P is formed and rank-1 constraint is relaxed to ensure the formulated SDP convex. Recall that P has the same size as the original non-convex $P' = vv^\mathsf{T} \in \mathbb{R}^{(1+\sum_{k=0}^{L} n_k) \times (1+\sum_{k=0}^{L} n_k)}$, the resulted SDP may not be tractable (10) for large networks. In the meantime, in some cases (e.g.

one layer NN), SDP relaxation is looser than LP relaxation, $f_{\text{SDP}} \geq f_{\text{LP}} \geq l_y^*(\overline{x}, \overline{y})$, that is, LP relaxation provides better upper bound in such cases. Consider the mentioned issues, Batten et al. introduce several modifications that provide the tighter upper bound and enable faster computation.

4.3.1. LINEAR CUTS

In both Raghunathan et al. (2018b); Batten et al. (2021), some tested instances illustrate that SDP relaxation can be looser than LP relaxation. Motivated by this, upper bound of triangular constraint is added to the previous formulated SDP.

$$f_{\text{SDP2}} = \max \quad c^{\mathsf{T}} P[x_L] \tag{11a}$$

$$\text{s.t.} \quad \text{for} \quad k \in [L],\ 10b, 10c, 10d, 10e$$

$$P[x_{k+1}] \leq \frac{\phi(u_k) - \phi(l_k)}{u_k - l_k}(W_k P[x_k] + b_k - l_k) + \phi(l_k), k \in [L] \text{(Linear cuts)} \tag{11b}$$

It is obviously $f_{\text{SDP}} \geq f_{\text{SDP2}} \geq l_y^*(\overline{x}, \overline{y})$, Batten et al. proved that given a non-convex NN verification instance, we have that $\min(f_{\text{SDP}, f_{\text{LP}}}) \geq f_{\text{SDP2}} \geq l_y^*(\overline{x}, \overline{y})$.

4.3.2. LAYER-BASED SDP RELAXATIONS

As before mentioned, P is a $(1+\sum_{k=0}^{L} n_k)$ square matrix, as neural network gets larger, it is difficult to solve the formulated SDP. Based on it, Batten et al. proposed layer-based SDP relaxations. The key idea is to introduce layer-based P_k for each layer k, where $P_k = v_k v_k^{\mathsf{T}}, v_k = [1, x_k, x_{k+1}]^{\mathsf{T}} \in \mathbb{R}^{1+n_k+n_{k+1}}$. Then instead of solving the big matrix P, the optimization solver here focuses on solving smaller $P_k, k \in [L]$, which is more tractable. Note that P_k, P_{k+1} both have variable 1 and x_{k+1}:

$$P_k = \begin{bmatrix} 1 & x_k^{\mathsf{T}} & x_{k+1}^{\mathsf{T}} \\ x_k & x_k x_k^{\mathsf{T}} & x_k x_{k+1}^{\mathsf{T}} \\ x_{k+1} & x_{k+1} x_k^{\mathsf{T}} & x_{k+1} x_{k+1}^{\mathsf{T}} \end{bmatrix}, P_{k+1} = \begin{bmatrix} 1 & x_{k+1}^{\mathsf{T}} & x_{k+2}^{\mathsf{T}} \\ x_{k+1} & x_{k+1} x_{k+1}^{\mathsf{T}} & x_{k+1} x_{k+2}^{\mathsf{T}} \\ x_{k+2} & x_{k+2} x_{k+1}^{\mathsf{T}} & x_{k+2} x_{k+2}^{\mathsf{T}} \end{bmatrix}. \tag{12}$$

To maintain the consistency in common variables between P_k and P_{k+1}, the following constraint is added:

$$P_k[\hat{x}_{k+1} \hat{x}_{k+1}^{\mathsf{T}}] = P_{k+1}[\hat{x}_{k+1} \hat{x}_{k+1}^{\mathsf{T}}], k \in [L-1] \tag{13}$$

where $\hat{x_{k+1}} = [1, x_{k+1}^{\mathsf{T}}]^{\mathsf{T}}$.

Similarly as section 4.2, the authors relax rank-one P_k to $P_k \succeq 0$. Relying on chordal decomposition result for sparse PSD matric, the authors proved $f_{\text{SDP2}} = f_{\text{SDP3}} \geq l_y^*(\overline{x}, \overline{y})$.

$$
\begin{aligned}
f_{\text{SDP3}} = \max \quad & c^{\mathsf{T}} P_{L-1}[x_L] && \text{(14a)}\\
\text{s.t.} \quad & \text{for} \quad k \in [L], && \\
& \operatorname{diag}(P_k[x_k x_k^{\mathsf{T}}]) \le (\bar{l}_k + \bar{u}_k) \cdot P_k[x_k] - \bar{l}_k \cdot \bar{u}_k, \text{(Input constraint)} && \text{(14b)}\\
& P_k[x_{k+1}] \ge 0, P_k[x_{k+1}] \ge W_k P_k[x_k] + b_k, \text{(ReLU constraint)} && \text{(14c)}\\
& \operatorname{diag}(P_k[x_{k+1} x_{k+1}^{\mathsf{T}}]) = \operatorname{diag}(W_k P_k[x_k x_{k+1}^{\mathsf{T}}]) + b_k \cdot P_k[x_{k+1}], \text{(ReLU constraint)} && \text{(14d)}\\
& P_k[x_{k+1}] \le \frac{\phi(u_k) - \phi(l_k)}{u_k - l_k}(W_k P_k[x_k] + b_k - l_k) + \phi(l_k), k \in [L] \text{(Linear cuts)} && \text{(14e)}\\
& P_k[1] = 1, P_k \succeq 0, \text{(matrix constraint)} && \text{(14f)}\\
& (13). \quad \text{(consistency constraint)} && \text{(14g)}
\end{aligned}
$$

Consistency constraint can also be relaxed. The authors also consider a linear number of consistency constraints as

$$P_k[x_{k+1}] = P_{k+1}[x_{k+1}], k \in [L-1] \tag{15}$$

Then new layer-based SDP relaxation can be formulated as

$$
\begin{aligned}
f_{\text{SDP4}} = \max \quad & c^{\mathsf{T}} P_{L-1}[x_L] && \text{(16a)}\\
\text{s.t.} \quad & \text{for} \quad k \in [L], 14b, 14c, 14d, 14e, 14f, && \\
& (15). \quad \text{(consistency constraint)} && \text{(16b)}
\end{aligned}
$$

Here SDP4 is looser then SDP3 but is faster to solve and it is still provably better than the LP relaxation.

4.4. Discussion on approaches

The discussed relaxation is summarized in Table 1.

	$P \succeq 0$	linear cut	layer-based $P_k \succeq 0$	consistency constraint
LP		✓		
SDP	✓			
SDP2	✓	✓		
SDP3		✓	✓	quadratic number
SDP4		✓	✓	linear number

Table 1: Discussed relaxation about finding an upper bound of $l_y^*(\overline{x}, \overline{y})$, which provide certificate for NN robustness.

4.5. Discussion on different activation functions

Previous subsections mainly focus on the ReLU activation function. Here we introduce the idea when the activation function lies in other types. Using linear functions to bound a general activation function is proposed in Zhang et al. (2018). Here we describe the non-conservative QCs for general activation functions.

Fazlyab et al. introduced tighter quadratic constraints for ReLU activation function. The key idea is that the ReLU function precisely lies on the boundary of the sector $[0, 1]$, and repeated nonlinearities for sector bound and slope restricted inequalities is introduced. Recall that in section 4.2, the ReLU function can be represented as two linear constraint and one quadratic constraint, equivalently (6). More specifically, let consider $\phi(x) = \max(\alpha x, \beta x), x \in \mathbb{R}^{n_k}$ be the concatenation of n_k ReLU activation functions (For ReLU, we have $\alpha = 0, \beta = 1$). We have

$$\phi(x^i) = \max(\alpha x^i, \beta x^i) \Leftrightarrow \begin{cases} \beta x^i \leq \phi(x^i), \alpha x^i \leq \phi(x^i) & \text{(linear constraint)} \\ (\phi(x^i) - \alpha x^i)(\phi(x^i) - \beta x^i) = 0 & \text{(quadratic constraint)} \end{cases} \tag{17}$$

where x^i is i-th element of x, $i \in [n_k]$. Furthermore, since ReLU is slope-restricted in $[\alpha, \beta](\alpha \leq \beta)$, for any two distinct indices $i \neq j$, we have

$$(\phi(x^j) - \phi(x^i) - \alpha(x^j - x^i))(\phi(x^j) - \phi(x^i) - \beta(x^j - x^i)) \leq 0. \tag{18}$$

We can see the formulated constraints (17), (18) can considerably reduce conservatism, especially for networks, as (18) considers the *the coupling between the neurons throughout the entire network*.

Although deriving non-conservative QCs for general activation functions (other than ReLU) is more complicated as they are not on the boundary of any sector. Fazlyab et al. iilustrated that by bounding these functions at multiple points by sector bounds, we can obtain a substantially better over-approximation.

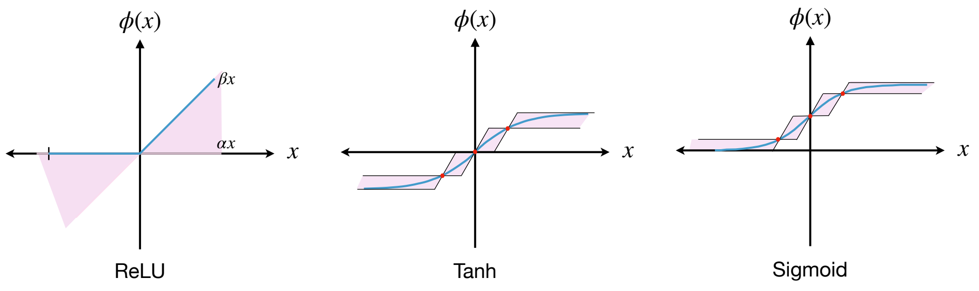

Figure 2: Illustration on bounding the general activation function at multiple points by sector bounds. (Left) ReLU function, bounded by the sector [0,1]; (Middle) The curve of the tanh function overapproximated on $\mathbb{R}$ by the intersection of three sectors. (Right) The curve of the sigmoid function similar as tanh function.

5. Numerical Experiments

In this section, we reproduce the evaluation of the performance of the discussed certificate (Table 1) on neural networks trained using different robust training procedures. The source code and the model is available at `https://github.com/alwaysbyx/NN-verification`.

Networks. We consider feedforward networks that are trained on the MNIST dataset of handwritten digits using three different robust training procedures, the same setting as Raghunathan et al. (2018b). The weights of the models are obtained in Raghunathan et al. (2018b).

1. Grad-NN. It's a two-layer network with 500 hidden nodes from Raghunathan et al. (2018a), obtained by using an SDP-based bound on the gradient of the network (different from the SDP presented here) as a regularizer. The weights of grad-NN is obtained from Raghunathan et al. (2018a).

2. LP-NN. It's a two-layer network with 500 hidden nodes trained via the LP-based robust training procedure of Wong and Kolter (2018).

3. PGD-NN. It's a four-layer network with 200,100 and 50 hidden nodes (i.e., the architecture is 784-200-100-50-10). The network is trained using adversarial training Goodfellow et al. (2014) against the strong PGD attack. The stepsize of the PGD attack was set to 0.1, number of iterations to 40, perturbation size = 0.3 and weight on adversarial loss to $\frac{1}{3}$.

Optimization setup. We use the YALMIP toolbox Lofberg (2004) with MOSEK as a backend to solve the different convex programs that arise in these certification procedures. The experiments are performed on a 6-Core Intel Core i7 2.6 GHz machine with 16 GB of RAM.

	Grad-NN		LP-NN		PGD-NN	
	optimal value	time	optimal value	time	optimal value	time
LP	-0.0647	1.3155	-5.1875	0.7401	not feasible	0.2327
SDP	-0.5873	8239.3	0.7401	9654.5	-8.8549	2097.4
SDP3					-11.4675	3716.3
SDP4		same as SDP3				1218.4

Table 2: Report on optimal value and solving time (seconds) using different certificate on one test sample. All numbers are reported for l_∞ attacks at = 0.1.

	Grad-NN		LP-NN		PGD-NN	
	verified	time	verified	time	verified	time
Attack	94%		80%	-	80%	
LP	35%	-	79%	-	65%	-
SDP	80%	17726	80%	2453	82%	3974
SDP3	91%	3392	80%	12.1	84%	159.8
SDP4	91%	3392	80%	12.1	72%	81.1

Table 3: Report on accuracy and solving time (seconds) using different certificate. All numbers are reported for l_∞ attacks at = 0.1, this report is generated by Batten et al. (2021).

6. Discussion

References

Ben Batten, Panagiotis Kouvaros, Alessio Lomuscio, and Yang Zheng. Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In *International Joint Conference on Artificial Intelligence (IJCAI21)*, pages 2184–2190, 2021.

Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 251–268. Springer, 2017.

Ruediger Ehlers. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis*, pages 269–286. Springer, 2017.

Mahyar Fazlyab, Manfred Morari, and George J Pappas. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatic Control*, 2020.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer, 2017.

Johan Lofberg. Yalmip: A toolbox for modeling and optimization in matlab. In *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, pages 284–289. IEEE, 2004.

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

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *arXiv preprint arXiv:1801.09344*, 2018a.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:1811.01057*, 2018b.

Vincent Tjeng, Kai Xiao, and Russ Tedrake. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. *Advances in neural information processing systems*, 31, 2018.