Safety verification and robustness analysis of neural networks

Brendon Anderson, Jingqi Li, Ziye Ma

12 December 2019

Abstract

In this report, we apply techniques from convex optimization to the problem of verifying a neural network's robustness to adversarial inputs. Specifically, we consider pre-trained deep classification networks with ReLU activation functions. By modeling adversarial attacks as norm-bounded perturbations to a nominal input-output pair, we seek to verify that the resulting classification remains unchanged. Since the verification procedure amounts to a nonconvex optimization problem, we describe various methods for relaxing the problem to a convex program. We then explore various extensions of these relaxation techniques, the first being a tractable model for sparse attacks, and the remainder being new methods for tightening the relaxations. Finally, we describe the basis for a sum-of-squares approach to the problem and provide concluding remarks.

1 Introduction

In this paper, we focus on studying the robustness of multi-layer classification neural networks with ReLU activation functions, one of the most ubiquitous and widely used form of feed-forward neural networks [1]. The notion of robustness stems from the stability of classification outputs given a range a different inputs, all closely defined with respect to a specific metric. To express it in mathematical terms, we first formally define the classification neural network that we want to study:

- A network f with L hidden layers is defined as follows: let $x^0 \in \mathbb{R}^{n_0}$ denote the input and $x^i \in \mathbb{R}^{n_i}$ denote the activation vector at the intermediate layer $i \in \{1, 2, \ldots, L\}$. That is, the network has n_i units in layer i. Then x^i is related to x^{i-1} as $x^i = \phi(W^{i-1}x^{i-1})$, where ϕ is the nonlinear activation function and $W^{i-1} \in \mathbb{R}^{n_i \times n_{i-1}}$ are the weights of the neural network. For the sake of simplicity, we omit the constant bias, and assume that they are included in the variables x^i.

- In our scope, we focus on the nonlinear activation function $\phi(\cdot) = \text{ReLU}(\cdot) = \max\{\cdot, 0\}$, where the maximum is taken element-wise. This activation function is widely used and studied in neural networks.

- The output of the network is $f(x^0) \in \mathbb{R}^k$ such that $f(x^0)_j = c_j^T x^L$ represents the score of class j. There are k classes in total. The class label $y \in \{1, 2, \ldots, k\}$ assigned to the input x^0 is the class with the highest score. A max or softmax function suffices in obtaining y from the output $f(x^0)$.

In this setting, we concern ourselves with the robustness of a pre-trained network, in which the weights are fixed. Given an input x^0, the network deterministically maps it to a specific class y.

Therefore, we ask the following question: If x^0 is reasonably perturbed, does the classification y change?

A key term here is *reasonably perturbed.* In the deep learning literature, these perturbations are also called *adversarial attacks* [2, 3]. There are numerous ways of modeling such attacks, so we have to be specific of what perturbation we are talking about. In this report we study norm-bounded perturbations. As the name suggests, we study perturbations of the input within a bound that is characterized by a norm, i.e. $\|x^0 - \bar{x}\|_p \leq \epsilon$, where x^0 is the perturbed input, $\bar{x}$ is the original input, and $\|\cdot\|_p$ is a vector p-norm.

1.1 Notations

The nonnegative orthant of $\mathbb{R}^n$ is denoted by $\mathbb{R}^n_+$. We use the symbol $\mathbb{S}^n$ to denote the set of all symmetric $n \times n$ matrices with real-valued elements. If the matrix $X \in \mathbb{S}^n$ is positive semidefinite, we write $X \succeq 0$. We denote the vector of diagonal elements of $X \in \mathbb{R}^{n\times n}$ by $\operatorname{diag}(X) \in \mathbb{R}^n$, and we write the rank of the matrix $Y \in \mathbb{R}^{m\times n}$ as $\operatorname{rank}(Y)$. We write the cardinality of a vector $x \in \mathbb{R}^n$ as $\|x\|_0$ (number of nonzero elements), and the cardinality of a set $\mathcal{X}$ as $|\mathcal{X}|$ (number of elements). For two vectors $x, y \in \mathbb{R}^n$, we denote ordering with respect to the nonnegative orthant by the symbol $\leq$. That is, $x \leq y$ if and only if $x_i \leq y_i$ for all $i \in \{1, 2, \ldots, n\}$. Furthermore, we write the Hadamard product (element-wise product) of x and y as $x \odot y$. When adding a scalar $\epsilon \in \mathbb{R}$ to every element in a vector $x \in \mathbb{R}^n$, we use the notation $x + \epsilon$, which is another n-vector. If $\phi\colon \mathbb{R} \to \mathbb{R}$ is a scalar function and $x \in \mathbb{R}^n$ is a vector, we write $y = \phi(x)$ to mean the element-wise operation of ϕ on x, i.e. $y_i = \phi(x_i)$ for all $i \in \{1, 2, \ldots, n\}$.

2 Problem Statement

The network model considered in this report is the L-layer feed-forward neural network $f\colon \mathcal{X} \subseteq \mathbb{R}^{n_0} \to \mathcal{Y} \subseteq \mathbb{R}^{n_L}$ defined by

$$\begin{aligned} x^i &= \phi\left(W^{i-1}x^{i-1}\right), \quad i \in \{1, 2, \ldots, L\}, \\ f(x^0) &= W^L x^L, \end{aligned} \tag{1}$$

where $x^0 \in \mathcal{X}$ is the given input to the network and $W^i \in \mathbb{R}^{n_i \times n_{i-1}}$ is the (trained) weight matrix of the ith layer of the network (where we have again assumed the bias terms are implicitly included). The function ϕ is the nonlinear activation function which is taken to operate element-wise on vector inputs. Popular choices for ϕ include the logistic sigmoid $\phi(x) = \frac{1}{1+e^{-x}}$, the hyperbolic tangent $\phi(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, and rectified linear unit (ReLU) $\phi(x) = \operatorname{ReLU}(x) = \max\{x, 0\}$. The function f of a trained neural network maps the input set $\mathcal{X}$ to the output set $\mathcal{Y} = f(\mathcal{X}) \coloneqq \{y : y = f(x),\ x \in \mathcal{X}\}$.

2.1 Robustness Certificates

When discussing robustness, we seek to determine whether a norm-bounded perturbation causes the network's classification to change. However, there exists various perspectives to this problem, much like those encountered in robust and stochastic optimization. In particular, the perturbation may be viewed as advantageous, a probabilistic event, or adversarial. Since many classification networks are employed in safety-critical applications, we take the viewpoint that the perturbation is worst-case, hence the terminology of *adversarial attacks.* In this setting, we'd like to ensure that all perturbations within the bound do not change the classification output. If all perturbations are safe, then surely the worst-case or maximum perturbation is safe. This line of reasoning allows us to naturally formulate the safety verification problem of interest as the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & (c_y - c_{\bar{y}})^T x^L \\ \text{subject to} \quad & x^i = \text{ReLU}(W^{i-1}x^{i-1}), \quad i \in \{1, 2, \dots, L\}, \quad \text{(Network constraints)} \\ & \|x^0 - \bar{x}\|_p \leq \epsilon, \quad \text{(Perturbation bound)} \end{aligned} \tag{2}$$

where the optimization variables are $x^0, x^1, \dots, x^L$. Here, $(\bar{x}, \bar{y})$ is a clean, unperturbed input-output pair, x^0 is the perturbed input, and y is a fixed class such that $y \neq \bar{y}$. We denote the optimal value of this problem (associated with class y) by $\ell_y^*(\bar{x}, \bar{y})$.

Definition 1 (Neural network robustness)**.** A neural network is said to be *certifiably robust* on $(\bar{x}, \bar{y})$ if $\ell_y^*(\bar{x}, \bar{y}) \leq 0$ for all $y \neq \bar{y}$.

We note that throughout the remainder of this report we use $\|\cdot\|_p = \|\cdot\|_\infty$ since this is the most natural norm to work with in a variety of real-life applications such as pixel value perturbations in image classification. The optimization problem (2) is nonconvex due to the ReLU equality constraints. Therefore, one cannot find a global maximum in general. In this work we study convex relaxations to efficiently compute an upper bound $\hat{\ell}_y^*(\bar{x}, \bar{y}) \geq \ell_y^*(\bar{x}, \bar{y})$. When $\hat{\ell}_y^*(\bar{x}, \bar{y}) \leq 0$, the robustness of the network on input $(\bar{x}, \bar{y})$ is certified. In this report, we explore several ways to relax problem (2), each of which being discussed in detail in the following sections. In the sequel, we will drop the $\hat{\ell}$ notation and simply write the optimal value to each certification optimization (whether relaxed or not) as $\ell_y^*(\bar{x}, \bar{y})$.

2.2 Linear Program Relaxation

Since the nonconvexity in problem (2) comes from the nonlinear ReLU activation, we convexify (2) by finding equivalent or approximate representations of the ReLU constraint that are convex. In [4], the authors proposed replacing the ReLU constraint with its upper convex envelope, given that we know the boundaries u and l of the input to the ReLU function.

Proposition 1 (Convex envelope of ReLU)**.** *The ReLU constraint $z = \text{ReLU}(Wx)$ on the box $l \leq x \leq u$ has the convex envelope defined by*

$$\begin{aligned} z &\geq 0, \\ z &\geq Wx, \\ -u \odot (Wx) + (u - l) \odot z &\leq -u \odot l. \end{aligned}$$

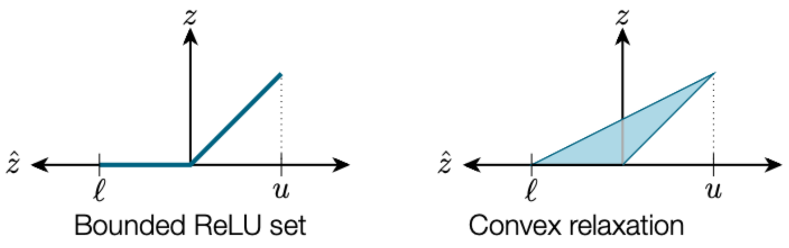


Figure 1: Illustration of the convex envelope of the ReLU constraint. [4]

By replacing the ReLU constraints in (2) with the relaxation introduced in Proposition 1, we obtain the following convex linear program:

$$\begin{array}{rl} \text{maximize} & (c_y - c_{\bar{y}})^T x^L \\ \text{subject to} & \bar{x} - \epsilon \le x^0 \le \bar{x} + \epsilon, \\ & x^i \ge 0, \\ & x^i \ge W^{i-1} x^{i-1}, \\ & -u^i \odot (W^{i-1} x^{i-1}) + (u^i - l^i) \odot x^i \le -u^i \odot l^i, \\ \text{for all} & i \in \{1, 2, \dots, L\}, \end{array} \tag{3}$$

where the optimization variables are $x^0, x^1, \dots, x^L$. Note in (3), the vectors u^i and l^i correspond to the upper and lower bounds on x^i as dictated by the input bounds $u^0 = \bar{x} + \epsilon$ and $l^0 = \bar{x} - \epsilon$. There are various methods in the literature for computing or approximating these bounds. Depending on the technique, the overall relaxation may become tighter. We present one of the methods in (7) and (8).

2.3 Semidefinite Program Relaxation

In the current section, we still focus on convexifying the ReLU constraint, albeit with a different approach. In [5], the authors proposed the following equivalent representation of the ReLU constraint:

Proposition 2 (Equivalent form of ReLU)**.** *The constraint* $z = \text{ReLU}(Wx)$ *is equivalent to the following system of constraints:*

$$\begin{aligned} z &\ge 0, \\ z &\ge Wx, \\ z \odot (z - Wx) &= 0. \end{aligned}$$

Using Proposition 2 to rewrite the ReLU constraints in (2), we obtain the following (for $L = 1$):

$$\begin{array}{rl} \text{maximize} & c^T z \\ \text{subject to} & z \ge 0, \\ & z \ge Wx, \\ & z \odot z = z \odot (Wx), \\ & x \odot x \le (l + u) \odot x - l \odot u, \end{array} \tag{4}$$

where the optimization variables are $x \in \mathbb{R}^{n_0}$ and $z \in \mathbb{R}^{n_1}$. Again, the vectors $l, u \in \mathbb{R}^{n_0}$ are the lower and upper bounds on the input such that $l \le x \le u$. Problem (4) is a nonconvex QCQP and the source of nonconvexity is quadratic equality constraint $z \odot z = z \odot (Wx)$. To make this problem more tractable, we convert it to a semidefinite program. To do this, we define the new matrix variable $P \in \mathbb{S}^{n_0 + n_1 + 1}$ as follows:

$$\begin{aligned} v &:= (1, x, z), \\ P &:= vv^T = \begin{bmatrix} 1 & x^T & z^T \\ x & xx^T & xz^T \\ z & zx^T & zz^T \end{bmatrix}. \end{aligned}$$

We use the block-indexing for the variable P given by

$$P = \begin{bmatrix} P[1] & P[x^T] & P[z^T] \\ P[x] & P[xx^T] & P[xz^T] \\ P[z] & P[zx^T] & P[zz^T] \end{bmatrix}.$$

Revisiting (4), we can rewrite the problem in terms of P as

$$\begin{array}{rl} \text{maximize} & c^T P[z] \\ \text{subject to} & P[z] \geq 0, \quad P[z] \geq WP[x], \\ & \operatorname{diag}(P[zz^T]) = \operatorname{diag}(WP[xz^T]), \\ & \operatorname{diag}(P[xx^T]) \leq (l+u) \odot P[x] - l \odot u, \\ & P[1] = 1, \\ & P \succeq 0, \\ & \operatorname{rank}(P) = 1, \end{array} \tag{5}$$

where now the optimization variable is $P \in \mathbb{S}^{n_0+n_1+1}$. Note this is an equivalent problem to that of (4). The source of nonconvexity here is the rank constraint. By dropping the rank constraint, we obtain a convex semidefinite program relaxation of problem. Extending this idea to the general L-layer setting, the SDP relaxation of problem (2) is found to be

$$\begin{array}{rl} \text{maximize} & (c_y - c_{\bar{y}})^T P[x^L] \\ \text{subject to} & P \succeq 0, \\ & P[1] = 1, \\ & P[x^i] \geq 0, \\ & P[x^i] \geq W^{i-1} P[x^{i-1}], \\ & \operatorname{diag}(P[x^i(x^i)^T]) = \operatorname{diag}(WP[x^{i-1}(x^i)^T]), \\ & \operatorname{diag}(P[x^{i-1}(x^{i-1})^T]) \leq (l^{i-1} + u^{i-1}) \odot P[x^{i-1}] - l^{i-1} \odot u^{i-1}, \\ \text{for all} & i \in \{1, 2, \ldots, L\}, \end{array} \tag{6}$$

where the optimization variable is $P \in \mathbb{S}^{1+\sum_{i=1}^{L} n_i}$. This equation is adapted from [5]. Another important question arising from problem (6) is how to compute u^i and l^i for each layer. It is obvious to first define

$$\begin{aligned} l^0 &= \bar{x} - \epsilon, \\ u^0 &= \bar{x} + \epsilon. \end{aligned} \tag{7}$$

Then, according to the authors of [5], a sufficient method for computing the bounds at subsequent layers is as follows:

$$\begin{aligned} l^i &= [W^{i-1}]_+ l^{i-1} + [W^{i-1}]_- u^{i-1}, \\ u^i &= [W^{i-1}]_+ u^{i-1} + [W^{i-1}]_- l^{i-1}, \end{aligned} \tag{8}$$

for $i \in \{1, 2, \ldots, L\}$. Here, for a matrix M we use the element-wise notation $[M]_+ = \max\{M, 0\}$, and $[M]_- = \min\{M, 0\}$. One can think of this as the propagation of l^0 and u^0 through the network.

2.4 Semidefinite Relaxation via Quadratic Constraints

2.4.1 Big Picture

Relaxing the network robustness verification problem to an SDP via quadratic constraints requires three steps:

1. Formulate a mathematical description of elements of the neural network, including the input set $\mathcal{X}$, the safety specification set $\mathcal{S}$ that restricts the outputs from the perturbed inputs, and the activation function $\phi(x)$.

2. Formulate quadratic constraints. We will show that for the three elements of the network defined above, we can form quadratic constraints in the form
$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0,$$
which is an outer approximation of the original set or output space.

3. Formulate a linear matrix inequality feasibility problem based on the quadratic constraints in the form
$$\begin{array}{ll} \text{minimize} & 0 \\ \text{subject to} & M \preceq 0, \end{array}$$
where the optimization variable is the matrix M. This is an SDP and thus can be solved efficiently.

In the sequel we discuss the second and third steps in detail.

2.4.2 Abstraction of Neural Networks via Quadratic Constraints

Recall that our ultimate goal is to formulate a tractable convex optimization problem which can be used to certify the robustness of a neural network. Here, we aim to formulate the problem as an SDP. It turns out that quadratic constraints are useful for deriving the SDP, leading to the following definition.

Definition 2 (Quadratic constraint)**.** Let $\mathcal{X} \subset \mathbb{R}^n$ be a nonempty set. Suppose $\mathcal{P}$ is the set of all symmetric matrices P such that
$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } x \in \mathcal{X}.$$
Then we say that $\mathcal{X}$ satisfies the *quadratic constraint (QC)* defined by $\mathcal{P}$.

2.4.3 Input and Safety Set Quadratic Constraints

Instead of using a fixed input set as mentioned in the running example, we consider an over-approximation of $\mathcal{X}$ by the finitely many intersections of sets defined by quadratic inequalities:
$$\mathcal{X} \subseteq \bigcap_{P \in \mathcal{P}} \left\{ x : \begin{bmatrix} x \\ 1 \end{bmatrix}^T P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \right\}.$$

For our specific parametrization of the input set (ℓ_∞-norm ball) the form of the class of matrices P is given in (7) in [6]. Here, we mainly discuss the pros and cons when using this strategy.

- *Advantages*: Here the matrix $P \in \mathcal{P}$ is treated as a decision variable in the SDP. The optimal outer approximation can be found by solving the optimization problem, thus reducing the conservativeness.
- *Disadvantages*: Making P a decision variable increases the complexity of the optimization problem, leading to a longer solving time.

Due to space limitations, we omit the derivation of QC for the safety specification set, which is very similar to the case of input set despite that the quadratic inequalities do not contain any decision variables.

2.4.4 Activation Quadratic Constraints

To derive the QC for the activation functions that we consider (ReLU), a convex relaxation procedure must be performed. First, one can show that a ReLU function can be represented by

$$z = \phi(x) = \max\{x, 0\} \iff \begin{cases} z \geq 0, \\ z \geq x, \\ z \odot z = z \odot x. \end{cases} \tag{9}$$

Next, a set of multipliers $(\lambda, \nu, \eta) \in \mathbb{R} \times \mathbb{R}_+ \times \mathbb{R}_+$ is associated with (9) such that

$$\lambda \left(z \odot z - z \odot x\right) + \nu(z - x) + \eta z \geq 0. \tag{10}$$

Notice that for different values of (λ, ν, η), (10) describes a different outer-approximation of the output space of the original ReLU function ϕ. We adopt the same idea as we did for the input set that the multipliers (λ, ν, η) will be left as decision variables which leads to the optimal (tightest) approximation of the activation functions. The key feature of (10) is that the LHS function is linear in (λ, ν, η), which enables us to derive the QC for the scalar-valued activation function $z_i = \phi(x_i)$:

$$\begin{bmatrix} x_i \\ z_i \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & \lambda_i & -\nu_i \\ \lambda_i & -2\lambda_i & \nu_i + \eta_i \\ -\nu_i & \nu_i + \eta_i & 0 \end{bmatrix} \begin{bmatrix} x_i \\ z_i \\ 1 \end{bmatrix} \geq 0, \tag{11}$$

in which the middle matrix of the quadratic constraint is linear in $(\lambda_i, \nu_i, \eta_i)$. As we will see in the sequel, this directly leads to the LMI constraint that appears in the SDP formulation. By simple linear transformations, (11) can be generalized for vector-valued activation functions, leading to Lemma 3 in the paper, which is indeed the QC defined for the ReLU activation functions.

2.4.5 Safety Verification of Neural Networks via SDP

Before we (re)state the main theorem proposed in the paper, recall the abstraction of the original network (1) via QCs that we developed in Section 2.4.2. For simplicity, consider the one-layer network $z = W^1\phi\left(W^0 x\right)$ whose input set $\mathcal{X}$ satisfies the QC defined by $\mathcal{P}$, i.e.

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T P \begin{bmatrix} x \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } x \in \mathcal{X},\ P \in \mathcal{P}, \tag{12}$$

and the activation function ϕ satisfies the QC defined by $\mathcal{Q}$, i.e.

$$\begin{bmatrix} y \\ \phi(y) \\ 1 \end{bmatrix}^T Q \begin{bmatrix} y \\ \phi(y) \\ 1 \end{bmatrix} \geq 0 \quad \text{for all } y \in \mathbb{R}^n,\ Q \in \mathcal{Q}. \tag{13}$$

Finally, satisfaction of the quadratic inequality of the safety set implies the inclusion $\mathcal{Y} \subseteq \mathcal{S}$, i.e.

$$\begin{bmatrix} z \\ 1 \end{bmatrix}^T S \begin{bmatrix} z \\ 1 \end{bmatrix} \leq 0 \quad \text{for all } x \in \mathcal{X} \implies \mathcal{Y} \subseteq \mathcal{S}. \tag{14}$$

Now we are ready to state the main result of the paper:

Proposition 3 (SDP feasibility [6])**.** *If the SDP problem*

$$\begin{aligned} &\text{minimize} && 0 \\ &\text{subject to} && M_{in}(P) + M_{mid}(Q) + M_{out}(S) \preceq 0 \end{aligned} \tag{15}$$

over the optimization variables P and Q is feasible, then the inclusion $\mathcal{Y} \subseteq \mathcal{S}$ holds, where in (15) *the matrices $M_{in}(P)$ and $M_{mid}(Q)$ are linear in the multipliers (λ, ν, η) in P and Q, as we introduced in the previous sections, and $M_{out}(S)$ is a constant matrix.*

This proposition justifies that (15) is an SDP problem that can be solved efficiently by, e.g. interior point methods. The derivation of (15) is sketched as follows. First, it can be verified easily that (12), (13), and (14) all can be rewritten in the following form:

$$\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^T \widetilde{M} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix} \geq 0, \quad \widetilde{M} \in \{M_{in}(P), M_{mid}(Q), M_{out}(S)\}. \tag{16}$$

Summing (16) over all $\widetilde{M} \in \{M_{in}(P), M_{mid}(Q), M_{out}(S)\}$ gives

$$\underbrace{\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^T M_{in} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}}_{\geq 0} + \underbrace{\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^T M_{mid} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}}_{\geq 0} + \underbrace{\begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}^T M_{out} \begin{bmatrix} x^0 \\ x^1 \\ 1 \end{bmatrix}}_{\leq 0} \leq 0, \tag{17}$$

in which the first and second terms in $M_{in}(P)$ and $M_{mid}(Q)$ being nonnegative is due to the satisfaction of QCs defined for the input set and activation functions, respectively. Moreover, the entire LHS function being nonpositive implies that the last term in $M_{out}(S)$ must be non-positive as well, which is exactly equivalent to the safety specification

$$\begin{bmatrix} z \\ 1 \end{bmatrix}^T S \begin{bmatrix} z \\ 1 \end{bmatrix} \leq 0 \quad \text{for all } x \in \mathcal{X} \tag{18}$$

that we want to verify. Finally, by the property of negative semidefinite matrices, the LMI constraint in (15) is equivalent to (17).

3 Results

3.1 Sparse Attacks

In this section, we consider the setting in which an adversary's attacks are not only limited in terms of their magnitude (via a norm-bound), but also in terms of their number. In other words, we are interested in the case where an attacker has access to only $k \leq n_0$ of the inputs to the network.

To the best of our knowledge, this model has not been well formulated or studied in the literature. Mathematically, the most natural model for sparse attacks is given by the additional constraint

$$\|x^0 - \bar{x}\|_0 \leq k, \tag{19}$$

where $\|\cdot\|_0$ denotes the cardinality, or number of nonzero entries, of a vector.

The solution to the cardinality-constrained verification problem not only certifies the robustness of a network to sparse attacks, but it naturally shows which of the k inputs of x^0 are the most sensitive to perturbations. This additional information may very well allow a network designer to robustify the most sensitive inputs in an *a posteriori* fashion without having to retrain the entire network. However, as written, the constraint (19) is nonconvex due to the nonconvexity of the cardinality function. To formulate the sparse attack model in a tractable manner, we outline two approaches: brute force enumeration and ℓ_1-relaxation.

3.1.1 Brute Force Enumeration

In this approach, we consider iterating through all possible choices of k attacks, solving the problem for each possibility separately, then finding the worst-case attack among them. Formally, we define the set of all subsets of $\{1, 2, \ldots, n_0\}$ with k elements as

$$\mathcal{I} = \{I \subseteq \{1, 2, \ldots, n_0\} : |I| = k\}.$$

We remark that there are n_0-choose-k such choices, i.e. $|\mathcal{I}| = \binom{n_0}{k}$. We then solve the sequence of problems:

```
for all $I \in \mathcal{I}$ :
  for all $y \neq \bar{y}$ :
    solve certification program (3)
      subject to $x_i^0 = \bar{x}_i$ for all $i \notin I$.
    store optimal objective $\ell_y^*(\bar{x}, \bar{y})$.
  store $\ell_I^*(\bar{x}, \bar{y}) = \max_{y \neq \bar{y}} \ell_y^*(\bar{x}, \bar{y})$
set $\ell^*(\bar{x}, \bar{y}) = \max_{I \in \mathcal{I}} \ell_I^*(\bar{x}, \bar{y})$.                (20)
```

Note that in the above formulation, we solve the LP problem (3), but given enough computational resources, one may alternatively use the SDP relaxation (6). Furthermore, we note that the additional constraints added to each subproblem (i.e. $x_i^0 = \bar{x}_i$ for all $i \notin I$) are linear in the optimization variables. Therefore, each subproblem is convex, successfully bypassing the nonconvexity of the constraint (19). Additionally, no relaxation has been made the the constraint (19). As a result, the cardinality constraint remains exact in the sense that the only relaxation error associated with the approach in (20) is that from the relaxation of the ReLU constraints in (3) (or in (6)). The primary drawback to this approach is the combinatorial number of subproblems that we must solve, in particular $\binom{n_0}{k}$ of them. This imposes a dramatic computational hurdle. To bypass this, we propose using an ℓ_1-relaxation.

3.1.2 Direct Relaxation

Instead of enumerating the cardinality constraint (19), we appeal to the well-known theoretical and empirical results suggesting the ℓ_1-norm acts as a good sparsity-promoting surrogate for the

ℓ_0-norm, $\|\cdot\|_0$ (which we remind the reader is not actually a norm). In particular, we propose replacing (19) by the constraint

$$\|x^0 - \bar{x}\|_1 \leq k\epsilon, \tag{21}$$

where ϵ is the same scalar used in the ℓ_∞-norm bound. We include this factor of ϵ to take care of scaling issues. In particular, consider an input x^0 such that $\|x^0 - \bar{x}\|_0 \leq k$ and $\|x^0 - \bar{x}\|_\infty \leq \epsilon$. Without loss of generality we assume the first k elements of $x^0 - \bar{x}$ are the nonzero ones. Then

$$\|x^0 - \bar{x}\|_1 = \sum_{i=1}^{n_0} |x_i^0 - \bar{x}_i| = \sum_{i=1}^{k} |x_i^0 - \bar{x}_i| \leq k\epsilon, \tag{22}$$

where we applied the ℓ_∞-bound k times to obtain the inequality. Therefore, we see that $\{x^0 : \|x^0 - \bar{x}\|_0 \leq k, \|x^0 - \bar{x}\|_\infty \leq \epsilon\} \subseteq \{x^0 : \|x^0 - \bar{x}\|_1 \leq k\epsilon, \|x^0 - \bar{x}\|_\infty \leq \epsilon\}$. Hence, (21) together with the ℓ_∞-bound is indeed a convex relaxation of the cardinality constraint (19) together with the ℓ_∞-bound. Therefore, appending (21) to the LP relaxation (3) gives the following sparse attack verification problem:

$$\begin{aligned} \text{maximize} \quad & (c_y - c_{\bar{y}})^T x^L \\ \text{subject to} \quad & \|x^0 - \bar{x}\|_\infty \leq \epsilon, \\ & \|x^0 - \bar{x}\|_1 \leq k\epsilon, \\ & x^i \geq 0, \\ & x^i \geq W^{i-1}x^{i-1}, \\ & -u^i \odot (W^{i-1}x^{i-1}) + (u^i - l^i) \odot x^i \leq -u^i \odot l^i, \\ \text{for all} \quad & i \in \{1, 2, \ldots, L\}, \end{aligned} \tag{23}$$

where the optimization variables are $x^0, x^1, \ldots, x^L$.

The benefits to this approach, versus that in Section 3.1.1, include the fact that (23) requires the solution of only one optimization problem, whereas (20) requires $\binom{n_0}{k}$. Therefore the ℓ_1-relaxation roughly maintains the scalability and computational complexity of the original verification problem. The drawback of the ℓ_1-relaxation is the additional loosening of the bound on the true value of $\ell_y^*(\bar{x}, \bar{y})$. That is, by using the ℓ_1-norm constraint, we enlarge the feasible set of the problem, possibly making the optimal value of (23) no longer match that of (20). In addition to the value of the solution, the structure of the solution may no longer match the true optimal solution; the optimal input x^0 for (23) may not have cardinality less than or equal to k. It is due to these properties of the problem that we say the relaxation in (23) is generally not exact.

3.1.3 Experimental Results

In this section we compare the two sparse attacks models: the brute force approach in (20) and the ℓ_1-relaxation in (23). We randomly initialized $N = 100$ different $5 \times 5 \times 2$ networks with ReLU activations. On each network, we solved the brute force ℓ_0-enumeration and the ℓ_1-relaxation problems for all $k \in \{1, 2, \ldots, n_0 = 5\}$. The average optimization runtime for each method is shown at various sparsity levels in Table 1. As expected, the ℓ_1 runtime is roughly constant over k, whereas the ℓ_0 runtime is combinatorial and generally higher. For $k = 5$, the ℓ_0 runtime is shorter since this requires only one optimization and has one less constraint than the ℓ_1 problem.

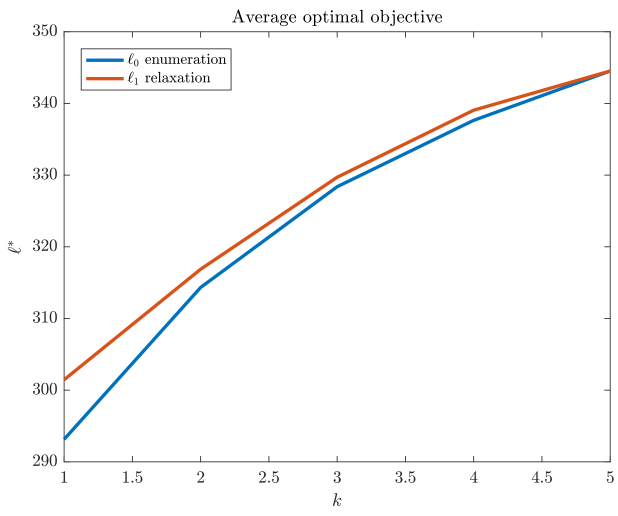
After verifying that indeed the optimal objective value of the ℓ_1-relaxation upper bounds that of the ℓ_0-enumeration in every case, we computed the average objective value for each k value and plotted the results in Figure 2a. As seen, the ℓ_1-relaxation closely matches the exact ℓ_0-enumeration

Table 1: Average runtime of ℓ_0 and ℓ_1 sparse attack models for various sparsity levels k.

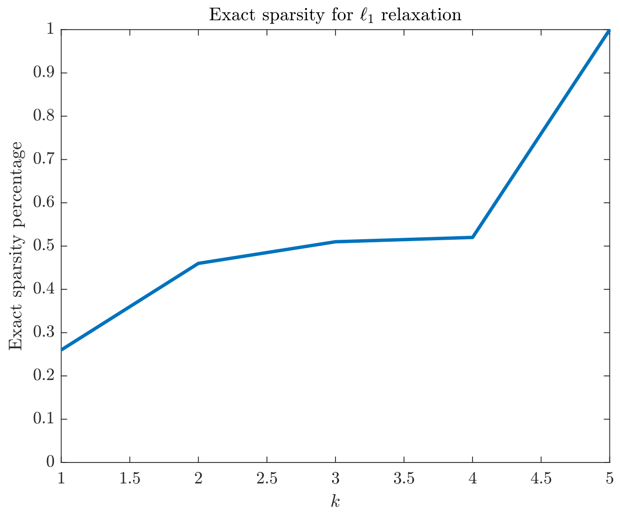
Sparsity Level k	ℓ_0 Runtime (seconds)	ℓ_1 Runtime (seconds)
1	0.9248	0.2883
2	1.9205	0.2634
3	1.9515	0.2636
4	0.9875	0.2678
5	0.1910	0.2610

for all k (in the average relative magnitude sense), and that the relaxation gap decreases as k increases to n_0.

For each sparsity level k, we also computed the percentage of the networks that the ℓ_1-relaxation recovered the sparsity pattern of the true sparse attack solution x^0, as determined from the exact ℓ_0-enumeration. As seen in Figure 2b, the percentage of exact recoveries increases as the sparsity level k increases. It it interesting to note that this percentage appears to increase from about 25% to 50% for $k \in \{1, 2, \dots, n_0 - 1\}$, and that it jumps to 100% at $k = n_0$. We leave the theoretical justification for these empirical findings to future work.



(a) Optimal cost of ℓ_0 and ℓ_1 sparse attack models for various sparsity levels k.



(b) Percentage of networks on which ℓ_1-relaxation is exact.

Figure 2: Sparse attack experiments.

3.2 Tightening Semidefinite Relaxation via Added Constraints

The main source of compromise in the SDP relaxation (6) comes from the lack of tightness in the rank constraint relaxation. For small networks, SDP relaxations might be tight already, but for larger networks, they are generally not. Therefore, we seek techniques to tighten the relaxation. A key observation here is that when we relaxed the problem (2), we only dropped the rank constraint. Therefore, we'd like to know if there's any implicit way to add this constraint back, as we can't do it explicitly due to its nonconvexity. To do so, we consider the existing constraint

$$P[z] - WP[x] \geq 0, \tag{24}$$

in the case of $L = 1$. By multiplying two of the same constraints together, we obtain the redundant constraint

$$\begin{aligned} &(P[z] - WP[x])(P[z] - WP[x])^T \geq 0 \\ \implies &P[z]P[z]^T - WP[x]P[z]^T - P[z]P[x]^TW^T + WP[x]P[x]^TW^T \geq 0. \end{aligned}$$

If P is rank-1, then $P[z]P[z]^T = P[zz^T]$, $P[x]P[z]^T = P[xz^T]$, etc. Therefore, by replacing these terms in the above redundant constraint, we create a new constraint that is affine in P and is found to implicitly push the solution P closer to rank-1. Using this technique to lift all nonconvex terms to their convex counterparts gives

$$P[zz^T] - WP[xz^T] - P[zx^T]W^T + WP[xx^T]W^T \geq 0.$$

In the general L-layer case, we obtain for any two layers i and j

$$P[x^j(x^j)^T] - WP[x^i(x^j)^T] - P[x^j(x^i)^T]W^T + WP[x^i(x^i)^T]W^T \geq 0. \tag{25}$$

Note we are limited to multiplying together linear constraints since the entries of P are at most quadratic in the x^is.

Our empirical findings show that the more complex the network is (the more hidden layers and input dimensions), the more effective this trick is. We measure the effectiveness by comparing the optimal values in (6) after applying these new constraints. Our results also show that cross-layer constraints (such that $i \neq j$ in (25)) consistently outperform same-layer constraints (where $i = j$).

Here we present some of our experimental results using the Iris flower dataset. We trained a $4 \times 10 \times 10 \times 10 \times 10 \times 3$ neural network with RELU activations for classification based on 4 features of a flower. In this experiment, we added redundant constraints for every two adjacent layers. The results are shown in Table 2, where SDP+ is the cost associated with the problem using redundant constraints. We see that by utilizing this technique, we are able to lower the optimal value in (6) from positive to negative in some cases, thus successfully certifying robustness that we couldn't have done without this added constraint.

Table 2: SDP versus SDP plus added constraints for $\epsilon = 0.75$.

Datapoint Number	SDP Cost	SDP+ Cost
1	0.0094	−0.0200
5	0.1100	−0.0900
20	0.0320	0.0012
22	0.2683	0.2376
31	6.5294	6.4972

3.3 Implementation of Semidefinite Relaxation via Quadratic Constraints

In this section, we implement the QC-based SDP formulation of the network verification problem and investigate the impact of different network properties on the quality of the outer approximation. This involves plotting the image of a perturbation set $\mathcal{X}$ (shown in blue throughout) and four supporting hyperplanes that are determined via the convex relaxation described in the paper (shown in red). Figure 3 shows the impact of increasing the hidden dimension of a one-layer multilayer perceptron (MLP) with ReLU activation functions. As can be seen, the approximation becomes less tight as the number of hidden units increases. Similarly, Figure 4 shows the impact of increasing the size of the perturbation set on the resulting outer approximation. This figure shows that as ϵ increases, i.e. as the perturbation set becomes larger, the tightness of the outer approximation decreases. This agrees with our intuition for this problem.

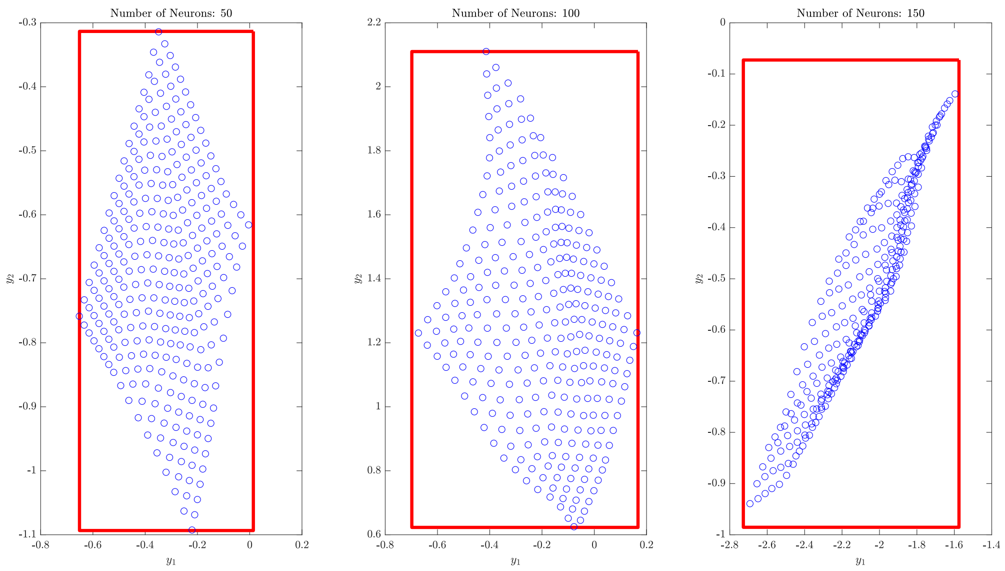


Figure 3: Impact of increasing the hidden neurons in a one-layer MLP with ReLU activation and random weights.

3.4 Input Gridding

Motivated by the observation from Section 3.3 that a small ϵ-bound on the input uncertainty set often yields tight outer approximations, we propose a trick to improve the accuracy of the outer approximation by gridding the input uncertainty set. Specifically, we partition the input uncertainty set $\mathcal{X} = \{x : \|x - \bar{x}\|_\infty \leq \epsilon\}$ into disjoint sets $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_k$ such that $\mathcal{X} = \cup_{i=1}^k \mathcal{X}_i$. For each $i \in \{1, 2, \ldots, k\}$, let $\mathcal{Y}_i$ be the outer approximation of the input uncertainty set $\mathcal{X}_i$. We then take the union of the outer approximations of each subarea, namely $\cup_{i=1}^k \mathcal{Y}_i$, as the outer approximation of the true output uncertainty set $\mathcal{Y}$.

In what follows, we present an illustrative example of the proposed trick. Suppose we have a 3-layer network with 40 neurons in each layer, associated with input $x \in \mathbb{R}^2$ and output $y \in \mathbb{R}^2$, i.e. a $2 \times 40 \times 40 \times 40 \times 2$ network. Consider $\bar{x} = (0.5, 0.5)$, and perturbation set $\mathcal{X} = \{x \in \mathbb{R}^2 : \|x - \bar{x}\|_\infty \leq 0.4\}$. We plot the real output of the perturbed inputs as yellow circles in Figure 5, and the decision boundary as the black dashed line. In the left plot in Figure 5, the red polytope is the outer approximation of the output corresponding to the input uncertainty set $\mathcal{X}$. We can see that the real output are all safe but the outer approximation of the output predicts an unsafe network due to its intersection with the decision boundary. However, in the right plot in Figure 5, we partition $\mathcal{X}$ into 4 regions. For each region we compute an outer approximation of its output. Then, we take the union of the four outer approximations as the outer approximation of $\mathcal{X}$. We can see that the outer approximation is tighter than the unpartitioned case, and that the resulting outer approximation shows that the network is indeed safe.

We remark that the input gridding can be implemented in a parallel or distributed fashion, and the empirical results support our conjecture that the outer approximation of the output converges to the exact output set as the maximum volume of the partitioned regions goes to zero.

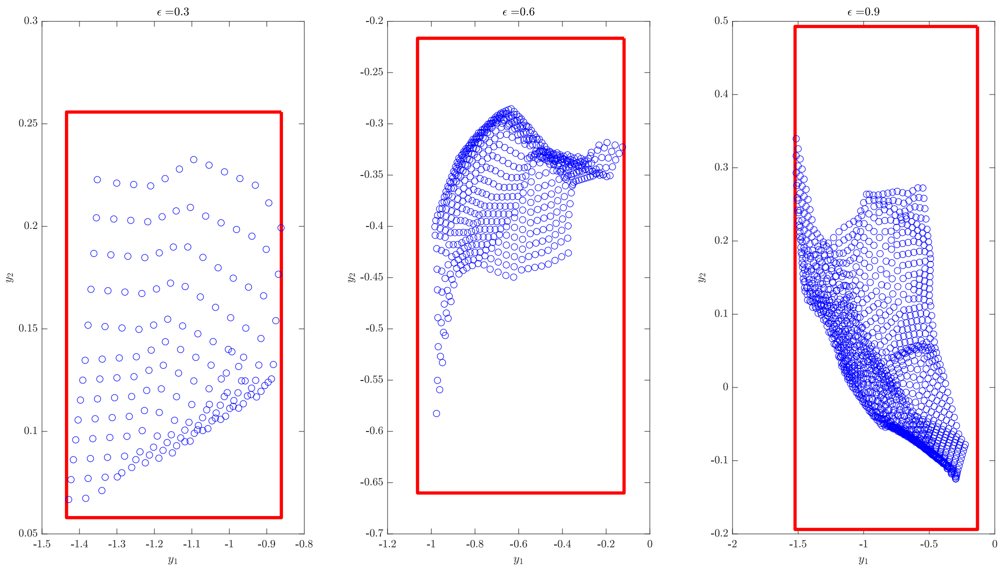


Figure 4: Impact of increasing the radius ϵ of the perturbation set $\mathcal{X} = \{x : \|x - \bar{x}\|_\infty \leq \epsilon\}$ for a one hidden-layer MLP with ReLU activation and random weights. This network has 100 neurons in the hidden layer.

3.5 Sum-of-Squares Optimization

An interesting observation is that in this particular robustness certification problem, we should not focus on optimization, but rather certification. More concretely, we only want to know if $\ell_y^*(\bar{x}, \bar{y}) \leq 0$ for all $y \neq \bar{y}$. That is, we only seek the certificate that $\ell_y^*(\bar{x}, \bar{y}) < 0$ or the certificate that $\ell_y^*(\bar{x}, \bar{y}) \geq 0$, and we don't really need to know the optimal value $\ell_y^*(\bar{x}, \bar{y})$ itself.

In the LP and SDP methods we've outlined so far, we can only obtain the certificate through solving the optimization problem. However, using ideas in sum-of-squares (SOS) optimization, we can utilize Positivstellensatz, a special type of theorem of alternatives, to produce this certificate. We start by presenting the most ubiquitously known Positivstellensatz:

Theorem 1 (Stengle's Positivstellensatz [7])**.** *The closed basic semialgebraic set*

$$\mathcal{S} = \{x \in \mathbb{R}^n : g_1(x) \geq 0, \ldots, g_m(x) \geq 0\}$$

is empty if and only if there exist sum-of-squares polynomials $s_0(x), \ldots, s_m(x)$, $s_{12}(x), s_{13}(x), \ldots, s_{123\ldots m}(x)$ *such that*

$$-1 = s_0(x) + \sum_i s_i(x) g_i(x) + \sum_{i,j} s_{ij}(x) g_i(x) g_j(x) + .. + s_{123..m}(x) g_1(x)..g_m(x).$$

The power of Positivstellensatz is that by considering the basic semialgebraic set $\mathcal{S} = \{x \in \mathbb{R}^n : f(x) \geq 0,\ x \in F\}$, where $f(x)$ is the original objective function in problem (2) and F is the original feasible set in problem (2), we can exactly solve our certification problem. Note that the SOS variables in Positivstellensatz are constrained to be even-degree polynomials. Thus, if they are upper-bounded by degree $2d$, we call this the dth degree SOS hierarchy, which asymptotically approaches the true optimum.

An import result from [8] and [9] is that verifying Positivstellensatz can be written in a form of an SDP. The complexity of this SDP grows very quickly as the degree d increases, thus becoming

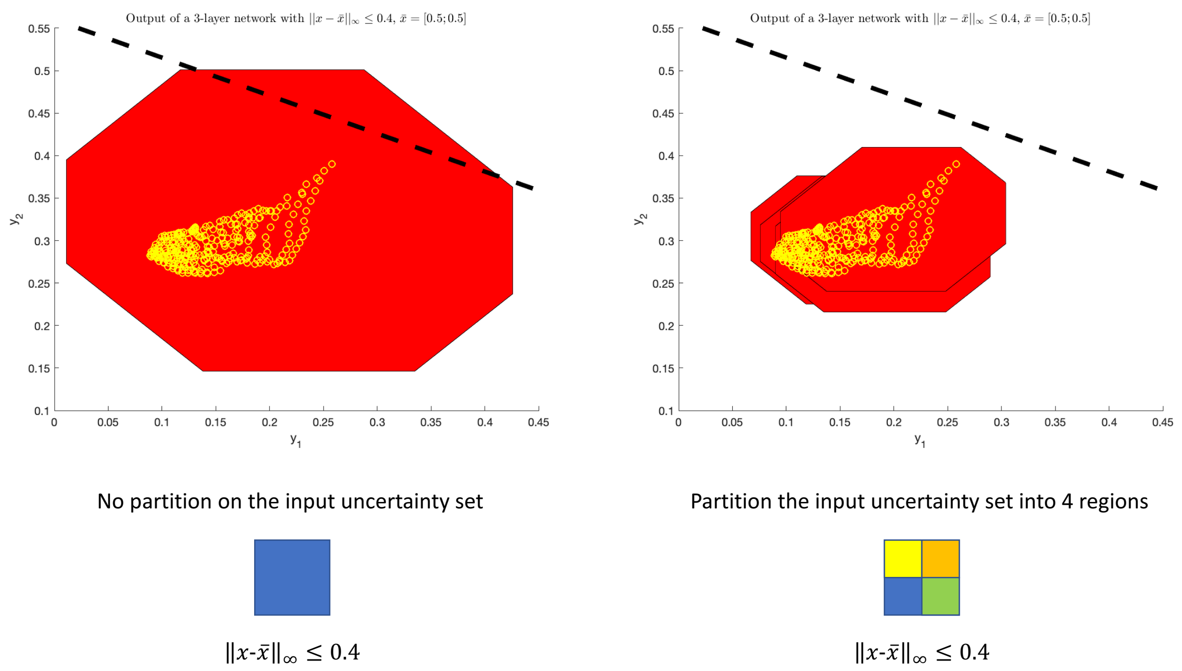


Figure 5: Tightened outer approximation resulting from the input gridding trick.

intractable for real-world problems. In our experiments, we used SOS optimization with SOS-TOOLS in MATLAB on a 2-dimensional input problem with one hidden layer and observed that the optimal value is the same to the SDP relaxation when $d = 1$. When $d = 2$, the optimal value becomes much lower. Beyond this scale, the solver takes too much time, thereby rendering this direct application of SOS optimization useless for realistic problems. However, we can potentially borrow matrix structures from higher order (≥ 2) hierarchies of SOS optimization to guide us on constructing new constraints to make the relaxation even tighter in both the SDP and QC case.

4 Conclusions and Future Work

In this report we've reviewed the motivation and current state-of-the-art methods for certifying the robustness of neural networks against adversarial attacks. We demonstrated three methods to formulate the problem into tractable convex programs: LP relaxation, SDP relaxation, and SDP relaxation via quadratic constraints. Based on these three approaches, we've proposed various extensions to address current gaps in the literature. First, we proposed two models for addressing sparse attacks, and we empirically demonstrated the advantages and disadvantages of both. Next, we showed that by intelligently forming redundant constraints, we can tighten the SDP relaxation even further than the state-of-the-art. Finally, motivated by the SDP via quadratic constraints, we show that by gridding the input uncertainty set and solving multiple smaller relaxations, we can again tighten the overall relaxation further than the current literature suggests.

The findings in this report are largely empirical, and therefore we leave for future work the theoretical justification of the interesting observations we've discussed. Additionally, as introduced in Section 3.5, sum-of-squares optimization provides a new approach to the verification problem with guarantees on the certificate's tightness. It is therefore our intent to explore this line of analysis in future work in an effort to further enhance the techniques available for certifying robustness.

Acknowledgements

A special thanks is in order to our advisor, Professor Somayeh Sojoudi. We would also like to thank the other EE 227BT instructors, Professor Laurent El Ghaoui and Armin Askari.

References

[1] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, “A survey of deep neural network architectures and their applications,” *Neurocomputing*, vol. 234, pp. 11–26, 2017.

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

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

[4] E. Wong and J. Z. Kolter, “Provable defenses against adversarial examples via the convex outer adversarial polytope,” *arXiv preprint arXiv:1711.00851*, 2017.

[5] A. Raghunathan, J. Steinhardt, and P. S. Liang, “Semidefinite relaxations for certifying robustness to adversarial examples,” in *Advances in Neural Information Processing Systems*, pp. 10877–10887, 2018.

[6] M. Fazlyab, M. Morari, and G. J. Pappas, “Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming,” *arXiv preprint arXiv:1903.01287*, 2019.

[7] G. Stengle, “A nullstellensatz and a positivstellensatz in semialgebraic geometry,” *Mathematische Annalen*, vol. 207, no. 2, pp. 87–97, 1974.

[8] P. A. Parrilo, “Semidefinite programming relaxations for semialgebraic problems,” *Mathematical programming*, vol. 96, no. 2, pp. 293–320, 2003.

[9] J. B. Lasserre, “Global optimization with polynomials and the problem of moments,” *SIAM Journal on optimization*, vol. 11, no. 3, pp. 796–817, 2001.