Reachability Analysis of Deep Neural Networks with Provable Guarantees

Wenjie Ruan[1], **Xiaowei Huang**[2], **Marta Kwiatkowska**[1]
[1] Department of Computer Science, University of Oxford, UK
[2] Department of Computer Science, University of Liverpool, UK
{wenjie.ruan; marta.kwiatkowska}@cs.ox.ac.uk; xiaowei.huang@liverpool.ac.uk

Abstract

Verifying correctness for deep neural networks (DNNs) is challenging. We study a generic reachability problem for feed-forward DNNs which, for a given set of inputs to the network and a Lipschitz-continuous function over its outputs computes the lower and upper bound on the function values. Because the network and the function are Lipschitz continuous, all values in the interval between the lower and upper bound are reachable. We show how to obtain the safety verification problem, the output range analysis problem and a robustness measure by instantiating the reachability problem. We present a novel algorithm based on adaptive nested optimisation to solve the reachability problem. The technique has been implemented and evaluated on a range of DNNs, demonstrating its efficiency, scalability and ability to handle a broader class of networks than state-of-the-art verification approaches.

1 Introduction

Concerns have been raised about the suitability of deep neural networks (DNNs), or systems with DNN components, for deployment in safety-critical applications, see e.g., [Amodei *et al.*, 2016; Sun *et al.*, 2018]. To ease this concern and gain users' trust, DNNs need to be certified similarly to systems such as airplanes and automobiles. In this paper, we propose to study a generic reachability problem which, for a given DNN, an input subspace and a function over the outputs of the network, computes the upper and lower bounds over the values of the function. The function is generic, with the only requirement that it is Lipschitz continuous. We argue that this problem is fundamental for certification of DNNs, as it can be instantiated into several key correctness problems, including adversarial example generation [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2014], safety verification [Huang *et al.*, 2017; Katz *et al.*, 2017; Ruan *et al.*, 2018], output range analysis [Lomuscio and Maganti, 2017; Dutta *et al.*, 2017a], and robustness comparison.

To certify a system, a certification approach needs to provide not only a result but also a guarantee over the result, such as the error bounds. Existing approaches for analysing DNNs with a guarantee work by either reducing the problem to constraint satisfaction problems that can be solved by MILP [Lomuscio and Maganti, 2017; Cheng *et al.*, 2017; Bunel *et al.*, 2017; Xiang *et al.*, 2017], SAT [Narodytska *et al.*, 2017] or SMT [Katz *et al.*, 2017; Bunel *et al.*, 2017] techniques, or applying search algorithms over discretised vector spaces [Huang *et al.*, 2017; Wicker *et al.*, 2017]. Even though they are able to achieve guarantees, they suffer from two major weaknesses. Firstly, their subjects of study are restricted. More specifically, they can only work with layers conducting linear transformations (such as convolutional and fully-connected layers) and simple non-linear transformations (such as ReLU), and cannot work with other important layers, such as the sigmoid, max pooling and softmax layers that are widely used in state-of-the-art networks. Secondly, for the constraint-based approaches, their scalability is significantly restricted by both the capability of the solvers and the size of the network, and they can only work with networks with a few hundreds of hidden neurons. However, state-of-the-art networks usually have millions, or even billions, of hidden neurons.

This paper proposes a novel approach to tackle the generic reachability problem without suffering from the above weaknesses, and providing provable guarantees in terms of the upper and lower bounds over the errors. The approach is inspired by recent advances made in the area of global optimisation [Gergel *et al.*, 2016; Grishagin *et al.*, 2018]. For the input subspace defined over a set of input dimensions, an adaptive nested optimisation algorithm is developed. The performance of our algorithm is not dependent on the size of the network and therefore it can scale to work with large networks.

Our algorithm assumes certain knowledge about the DNN. However, instead of directly translating the activation functions and their parameters (i.e., weights and bias) into linear constraints, it needs a Lipschitz constant of the network. For this, we show that several layers that cannot be directly translated into linear constraints are actually Lipschitz continuous, and we are able to compute a tight Lipschitz constant by analysing the activation functions and their parameters.

We develop a software tool DeepGO[1] and evaluate its performance by comparing with existing constraint-based approaches, namely, SHERLOCK [Dutta *et al.*, 2017b] and Re-

[1] Available on `https://github.com/trustAI/DeepGO`.

luplex [Katz *et al.*, 2017]. We also demonstrate our tool on DNNs that are beyond the capability of existing tools.

2 Related Works

We discuss several threads of work concerning problems that can be obtained by instantiating our generic reachability problem. Their instantiations are explained in the paper. Due to space limitations, this review is by no means complete.

Safety Verification There are two ways of achieving safety verification for DNNs. The first is to reduce the problem into a constraint solving problem. Notable works include, e.g., [Pulina and Tacchella, 2010; Katz *et al.*, 2017]. However, they can only work with small networks with hundreds of hidden neurons. The second is to discretise the vector spaces of the input or hidden layers and then apply exhaustive search algorithms or Monte-Carlo tree search algorithm on the discretised spaces. The guarantees are achieved by establishing local assumptions such as minimality of manipulations in [Huang *et al.*, 2017] and minimum confidence gap for Lipschitz networks in [Wicker *et al.*, 2017].

Adversarial Example Generation Most existing works, e.g., [Szegedy *et al.*, 2014; Goodfellow *et al.*, 2014; Nguyen *et al.*, 2014; Moosavi-Dezfooli *et al.*, 2016; Carlini and Wagner, 2016], apply various heuristic algorithms, generally using search algorithms based on gradient descent or evolutionary techniques. [Papernot *et al.*, 2015] construct a saliency map of the importance of the pixels based on gradient descent and then modify the pixels. In contrast with our approach based on global optimisation and works on safety verification, these approaches may be able to find adversarial examples efficiently, but are *not able to conclude the nonexistence* of adversarial examples when the algorithm fails to find one.

Output Range Analysis The safety verification approach can be adapted to work on this problem. Moreover, [Lomuscio and Maganti, 2017] consider determining whether an output value of a DNN is reachable from a given input subspace, and reduce the problem to a MILP problem. [Dutta *et al.*, 2017b] considers the range of output values from a given input subspace. Their approach interleaves local search (based on gradient descent) with global search (based on reduction to MILP). Both approaches can only work with small networks.

3 Lipschitz Continuity of DNNs

This section shows that feed-forward DNNs are Lipschitz continuous. Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a N-layer network such that, for a given input $x \in \mathbb{R}^n$, $f(x) = \{c_1, c_2, ..., c_m\} \in \mathbb{R}^m$ represents the confidence values for m classification labels. Specifically, we have $f(x) = f_N(f_{N-1}(...f_1(x; W_1, b_1); W_2, b_2); ...); W_N, b_N)$ where W_i and b_i for $i = 1, 2, ..., N$ are learnable parameters and $f_i(z_{i-1}; W_{i-1}, b_{i-1})$ is the function mapping from the output of layer $i-1$ to the output of layer i such that z_{i-1} is the output of layer $i-1$. Without loss of generality, we normalise the input to lie $x \in [0, 1]^n$. The output $f(x)$ is usually normalised to be in $[0, 1]^m$ with a softmax layer.

Definition 1 (Lipschitz Continuity) *Given two metric spaces (X, d_X) and (Y, d_Y), where d_X and d_Y are the metrics on the sets X and Y respectively, a function $f : X \rightarrow Y$ is called* Lipschitz continuous *if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in X$:*

$$d_Y(f(x_1), f(x_2)) \leq K d_X(x_1, x_2) \tag{1}$$

K is called the Lipschitz constant *for the function f. The smallest K is called* the Best Lipschitz constant*, denoted as K_{best}.*

[Szegedy *et al.*, 2014] show that deep neural networks with half-rectified layers (*i.e.,* convolutional or fully connected layers with ReLU activation functions), max pooling and contrast-normalization layers are Lipschitz continuous. They prove that the upper bound of the Lipschitz constant can be estimated via the operator norm of learned parameters W.

Next, we show that the softmax layer, sigmoid and Hyperbolic tangent activation functions also satisfy Lipschitz continuity. First we need the following lemma [Sohrab, 2003].

Lemma 1 *Let $f : \mathbb{R}^n \rightarrow \mathbb{R}^m$, if $||\partial f(x)/\partial x|| \leq K$ for all $x \in [a, b]^n$, then f is Lipschitz continuous on $[a, b]^n$ and K is its Lipschitz constant, where $||*||$ represents a norm operator.*

Based on this lemma, we have the following theorem.

Theorem 1 *Convolutional or fully connected layers with the sigmoid activation function $s(Wx + b)$, Hyperbolic tangent activation function $t(Wx + b)$, and softmax function $p(x)_j$ are Lipschitz continuous and their Lipschitz constants are $\frac{1}{2}\|W\|, \|W\|$, and $\sup_{i,j}(\|x_i\| + \|x_i x_j\|)$, respectively.*

Proof 1 *First of all, we show that the norm operators of their Jacobian matrices are bounded.*

(1) Layer with sigmoid activation $s(q) = 1/(1+e^{-q})$ with $q = Wx + b$:

$$\begin{aligned} \left\| \frac{\partial s(x)}{\partial x} \right\| &= \left\| \frac{\partial s(q)}{\partial q} \frac{\partial q}{\partial x} \right\| \leq \left\| \frac{\partial s(q)}{\partial q} \right\| \left\| \frac{\partial q}{\partial x} \right\| \\ &\leq \left\| s(q) \circ (\mathbf{1} - s(q)) \right\| \|W\| \leq \frac{1}{2}\|W\| \end{aligned} \tag{2}$$

(2) Layer with Hyperbolic tangent activation function $t(q) = 2/(1 + e^{-2q}) - 1$ with $q = Wx + b$:

$$\begin{aligned} \left\| \frac{\partial t(x)}{\partial x} \right\| &= \left\| \frac{\partial t(q)}{\partial q} \frac{\partial q}{\partial x} \right\| \leq \left\| \frac{\partial t(q)}{\partial q} \right\| \left\| \frac{\partial q}{\partial x} \right\| \\ &\leq \left\| \mathbf{1} - t(q) \circ t(q)) \right\| \|W\| \leq \|W\| \end{aligned} \tag{3}$$

(3) Layer with softmax function $p(x)_j = e^{x_j}/(\sum_{k=1}^{n} e^{x_k})$ for $j = 1, ..., m$ and $n = m$ (dimensions of input and output of softmax are the same):

$$\left\| \frac{\partial p(x)_j}{\partial x_i} \right\| = \begin{cases} x_i(1 - x_j), & i = j \\ -x_i x_j, & i \neq j \end{cases} \leq \sup_{i,j}(\|x_i\| + \|x_i x_j\|) \tag{4}$$

Since the softmax layer is the last layer of a deep neural network, we can estimate its supremum based on Lipschitz constants of previous layers and box constraints of DNN's input.

The final conclusion follows by Lemma 1 and a fact that all those layer functions are bounded on their Jacobian matrix.

4 Problem Formulation

Let $o : [0,1]^m \to \mathbb{R}$ be a Lipschitz continuous function statistically evaluating the outputs of the network. Our problem is to find its upper and lower bounds given the set X' of inputs to the network. Because both the network f and the function o are Lipschitz continuous, all values between the upper and lower bounds have a corresponding input, i.e., are reachable.

Definition 2 (Reachability of Neural Network) *Let $X' \subseteq [0,1]^n$ be an input subspace and $f : \mathbb{R}^n \to \mathbb{R}^m$ a network. The reachability of f over the function o under an error tolerance $\epsilon \geq 0$ is a set $R(o, X', \epsilon) = [l, u]$ such that*

$$l \geq \inf_{x' \in X'} o(f(x')) - \epsilon \text{ and } u \leq \sup_{x' \in X'} o(f(x')) + \epsilon. \quad (5)$$

We write $u(o, X', \epsilon) = u$ and $l(o, X', \epsilon) = l$ for the upper and lower bound, respectively. Then the reachability diameter is

$$D(o, X', \epsilon) = u(o, X', \epsilon) - l(o, X', \epsilon). \quad (6)$$

Assuming these notations, we may write $D(o, X', \epsilon; f)$ if we need to explicitly refer to the network f.

In the following, we instantiate o with a few concrete functions, and show that several key verification problems for DNNs can be reduced to our reachability problem.

Definition 3 (Output Range Analysis) *Given a class label $j \in [1, .., m]$, we let $o = \Pi_j$ such that $\Pi_j((c_1, ..., c_m)) = c_j$.*

We write $c_j(x) = \Pi_j(f(x))$ for the network's confidence in classifying x as label j. Intuitively, output range [Dutta *et al.*, 2017a] quantifies how a certain output of a deep neural network (*i.e.,* classification probability of a certain label j) varies in response to a set of DNN inputs with an error tolerance ϵ. Output range analysis can be easily generalised to logit [2] range analysis.

We show that the safety verification problem [Huang *et al.*, 2017] can be reduced to solving the reachability problem.

Definition 4 (Safety) *A network f is safe with respect to an input x and an input subspace $X' \subseteq [0,1]^n$ with $x \in X'$, written as $\mathtt{S}(f, x, X')$, if*

$$\forall x' \in X' : \arg\max_j c_j(x') = \arg\max_j c_j(x) \quad (7)$$

We have the following reduction theorem.

Theorem 2 *A network f is safe with respect to x and X' s.t. $x \in X'$ if and only if $u(\oplus, X', \epsilon) \leq 0$, where $\oplus(c_1, ..., c_m) = \max_{i \in \{1..m\}}(\Pi_i(c_1, ..., c_m) - \Pi_j(c_1, ..., c_m))$ and $j = \arg\max_j c_j(x)$. The error bound of the safety decision problem by this reduction is 2ϵ.*

It is not hard to see that the adversarial example generation [Szegedy *et al.*, 2014], which is to find an input $x' \in X'$ such that $\arg\max_j c_j(x') \neq \arg\max_j c_j(x)$, is the dual problem of the safety problem.

The following two problems define the robustness comparisons between the networks and/or the inputs.

[2] Logit output is the output of the layer before the softmax layer. The study of Logit outputs is conducted in, e.g., [Papernot *et al.*, 2015; Dutta *et al.*, 2017b].

Definition 5 (Robustness) *Given two homogeneous[3] networks f and g, we say that f is strictly more robust than g with respect to a function o, an input subspace X' and an error bound ϵ, written as $\mathtt{R}_{o,X',\epsilon}(f, g)$, if $D(o, X', \epsilon; f) < D(o, X', \epsilon; g)$.*

Definition 6 *Given two input subspaces X' and X'' and a network f, we say that f is more robust on X' than on X'' with respect to a statistical function o and an error bound ϵ, written as $\mathtt{R}_{f,o,\epsilon}(X', X'')$, if $D(o, X', \epsilon) < D(o, X'', \epsilon)$.*

Thus, by instantiating the function o, we can quantify the output/logit range of a network, evaluate whether a network is safe, and compare the robustness of two homogeneous networks or two input subspaces for a given network.

5 Confidence Reachability with Guarantees

Section 3 shows that a trained deep neural network is Lipschitz continuous regardless of its layer depth, activation functions and number of neurons. Now, to solve the reachability problem we need to find the *global* minimum and maximum values given an input subspace, assuming that we have a Lipschitz constant K for the function $o \cdot f$. In the following, we let $w = o \cdot f$ be the concatenated function. Without loss of generality, we assume the input space X' is a box-constraint, which is clearly feasible since images are usually normalized into $[0,1]^n$ before being fed into a neural network.

The computation of the minimum value is reduced to solving the following optimization problem with guaranteed convergence to the global minimum (the maximization problem can be transferred into a minimization problem).

$$\min_x \ w(x), \quad s.t. \ x \in [a,b]^n \quad (8)$$

However, the above problem is very difficult since $w(x)$ is a highly non-convex function which cannot be guaranteed to reach the global minimum by regular optimization schemes based on gradient descent. Inspired by an idea from optimisation, see e.g., [Piyavskii, 1972; Torn and Zilinskas, 1989], we design another continuous function $h(x, y)$, which serves as a lower bound of the original function $w(x)$. Specifically, we need

$$h(x, y) \leq w(x), \ \forall x, y \in [a,b]^n, \ h(x, x) = w(x) \quad (9)$$

Furthermore, for $i \geq 0$, we let $\mathcal{Y}_i = \{y_0, y_1, ..., y_i\}$ be a finite set containing $i+1$ points from the input space $[a,b]^n$, and let $\mathcal{Y}_i \subseteq \mathcal{Y}_k$ when $k > i$, then we can define a function $H(x; \mathcal{Y}_i) = \max_{y \in \mathcal{Y}_i} h(x, y)$ which satisfies the following relation:

$$H(x; \mathcal{Y}_i) < H(x; \mathcal{Y}_k) \leq w(x), \forall i < k \quad (10)$$

We use $l_i = \inf_{x \in [a,b]^n} H(x; \mathcal{Y}_i)$ to denote the minimum value of $H(x; \mathcal{Y}_i)$ for $x \in [a,b]^n$. Then we have

$$l_0 < l_1 < ... < l_{i-1} < l_i \leq \inf_{x \in [a,b]^n} w(x) \quad (11)$$

[3] Here, two networks are homogeneous if they are applied on the same classification task but may have different network architectures (layer numbers, layer types, etc) and/or parameters.

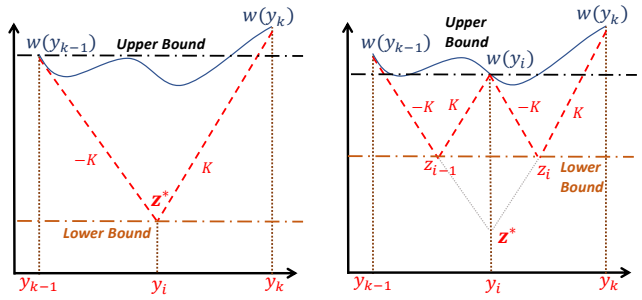


Figure 1: A lower-bound function designed via Lipschitz constant

Similarly, we need a sequence of upper bounds u_i to have

$$l_0 < ... < l_i \leq \inf_{x\in[a,b]^n} w(x) \leq u_i < ... < u_0 \tag{12}$$

By Expression (12), we can have the following:

$$\lim_{i\to\infty} l_i = \min_{x\in[a,b]^n} w(x) \text{ and } \lim_{i\to\infty}(u_i - l_i) = 0 \tag{13}$$

Therefore, we can asymptotically approach the global minimum. Practically, we execute a finite number of iterations by using an error tolerance ϵ to control the termination. In next sections, we present our approach, which constructs a sequence of lower and upper bounds, and show that it can converge with an error bound. To handle the high-dimensionality of DNNs, our approach is inspired by the idea of adaptive nested optimisation in [Gergel *et al.*, 2016], with significant differences in the detailed algorithm and convergence proof.

5.1 One-dimensional Case

We first introduce an algorithm which works over one dimension of the input, and therefore is able to handle the case of $x \in [a, b]$ in Eqn. (8). The multi-dimensional optimisation algorithm will be discussed in Section 5.2 by utilising the one-dimensional algorithm.

We define the following lower-bound function.

$$\begin{aligned} h(x, y) &= w(y) - K|x - y| \\ H(x; \mathcal{Y}_i) &= \max_{y\in\mathcal{Y}_i} \; w(y) - K|x - y| \end{aligned} \tag{14}$$

where $K > K_{best}$ is a Lipschitze constant of w and $H(x; \mathcal{Y}_i)$ intuitively represents the lower-bound sawtooth function shown as Figure 1. The set of points $\mathcal{Y}_i$ is constructed recursively. Assuming that after $(i-1)$-th iteration, we have $\mathcal{Y}_{i-1} = \{y_0, y_1, .., y_{i-1}\}$, whose elements are ascendingly ordered, and sets

$$w(\mathcal{Y}_{i-1}) = \{w(y_0), w(y_1), .., w(y_{i-1})\}$$

$$\mathcal{L}_{i-1} = \{l_0, l_1, ..., l_{i-1}\}$$

$$\mathcal{U}_{i-1} = \{u_0, u_1, ..., u_{i-1}\}$$

$$\mathcal{Z}_{i-1} = \{z_1, ..., z_{i-1}\}$$

The elements in sets $w(\mathcal{Y}_{i-1})$, $\mathcal{L}_{i-1}$ and $\mathcal{U}_{i-1}$ have been defined earlier. The set $\mathcal{Z}_{i-1}$ is to record the smallest values z_k computed in an interval $[y_{k-1}, y_k]$.

In i-th iteration, we do the following sequentially:

- Compute $y_i = \arg\inf_{x\in[a,b]} H(x; \mathcal{Y}_{i-1})$ as follows. Let $z^* = \min \mathcal{Z}_{i-1}$ and k be the index of the interval $[y_{k-1}, y_k]$ where z^* is computed. Then we let

$$y_i = \frac{y_{k-1} + y_k}{2} - \frac{w(y_k) - w(y_{k-1})}{2K} \tag{15}$$

  and have that $y_i \in (y_{k-1}, y_k)$.
- Let $\mathcal{Y}_i = \mathcal{Y}_{i-1} \cup \{y_i\}$, and reorder $\mathcal{Y}_i$ in an ascending order, and update $w(\mathcal{Y}_i) = w(\mathcal{Y}_{i-1}) \cup \{w(y_i)\}$.
- Calculate

$$z_{i-1} = \frac{w(y_i) + w(y_{k-1})}{2} - \frac{K(y_i - y_{k-1})}{2} \tag{16}$$

$$z_i = \frac{w(y_k) + w(y_i)}{2} - \frac{K(y_k - y_i)}{2} \tag{17}$$

  and update $\mathcal{Z}_i = (\mathcal{Z}_{i-1} \setminus \{z^*\}) \cup \{z_{i-1}, z_i\}$.
- Calculate the new lower bound $l_i = \inf_{x\in[a,b]} H(x; \mathcal{Y}_i)$ by letting $l_i = \min \mathcal{Z}_i$, and updating $\mathcal{L}_i = \mathcal{L}_{i-1} \cup \{l_i\}$.
- Calculate the new upper bound $u_i = \min_{y\in\mathcal{Y}_i} w(y)$ by letting $u_i = \min\{u_{i-1}, w(y_i)\}$.

We terminate the iteration whenever $|u_i - l_i| \leq \epsilon$, and let the global minimum value be $y^* = \min_{x\in[a,b]} H(x; \mathcal{Y}_i)$ and the minimum objective function be $w^* = w(y^*)$.

Intuitively, as shown in Fig. 1, we iteratively generate lower bounds (which is the lowest point in the saw-tooth function in the figure) by continuously refining a piecewise-linear lower bound function which is guaranteed to below the original function with Lipschitz continuity. And the upper bound is the lowest evaluation value of the original function so far.

Convergence Analysis

In the following, we show the convergence of this algorithm to the global minimum by proving the following conditions:

- Convergence Condition 1: $\lim_{i\to\infty} l_i = \min_{x\in[a,b]} w(x)$
- Convergence Condition 2: $\lim_{i\to\infty}(u_i - l_i) = 0$

Proof 2 (Monotonicity of Lower/Upper Bound Sequences) *First, we prove that the lower bound sequence $\mathcal{L}_i$ is strictly monotonic. Because*

$$l_i = \min \mathcal{Z}_i = \min\{(\mathcal{Z}_{i-1} \setminus \{z^*\}) \cup \{z_{i-1}, z_i\}\} \tag{18}$$

and $l_{i-1} = \min \mathcal{Z}_i$. To show that $l_i > l_{i-1}$, we need to prove $z_{i-1} > z^*$ and $z_i > z^*$. By the algorithm, z^* is computed from interval $[y_{k-1}, y_k]$, so we have

$$z^* = \frac{w(y_k) + w(y_{k-1})}{2} - \frac{K(y_k - y_{k-1})}{2} \tag{19}$$

We then have

$$z_{i-1} - z^* = \frac{w(y_i) - w(y_k) - K(y_i - y_k)}{2} \tag{20}$$

Since $y_i < y_k$ and $K > K_{best}$, based on Lipschitz Continuity, we have $z_{i-1} > z^*$. Similarly, we can prove $z_i > z^*$. Thus $l_i > l_{i-1}$ is guaranteed.

Second, the monotonicity of upper bounds u_i can be seen from the algorithm where u_i is updated into $\min\{u_i, w(y_i)\}$ in every iteration.

Proof 3 (Convergence Condition 1)
Since $\mathcal{Y}_{i-1} \subseteq \mathcal{Y}_i$, we have $H(x;\mathcal{Y}_{i-1}) \leq H(x;\mathcal{Y}_i)$. Based on Proof 2, we also have $l_{i-1} < l_i$. And because

$$l_i = \inf_{x\in[a,b]} H(x;\mathcal{Y}_i) \leq \min_{x\in[a,b]} w(x) \tag{21}$$

the lower bound sequence $\{l_0, l_1, ..., l_i\}$ is strictly monotonically increasing and are bounded from above by $\min_{x\in[a,b]} w(x)$. Thus $\lim_{i\to\infty} l_i = \min_{x\in[a,b]} w(x)$ holds.

Proof 4 (Convergence Condition 2)
Since $\lim_{i\to\infty} l_i = \min_{x\in[a,b]} w(x)$, we show $\lim_{i\to\infty}(u_i - l_i) = 0$ by showing that $\lim_{i\to\infty} u_i = \min_{x\in[a,b]} w(x)$. By $\mathcal{Y}_i = \mathcal{Y}_{i-1} \cup \{y_i\}$ and $y_i \in X = [a,b]$, we have $\lim_{i\to\infty} \mathcal{Y}_i = X$. Then we have $\lim_{i\to\infty} u_i = \lim_{i\to\infty} \inf_{y\in\mathcal{Y}_i} w(y) = \inf w(X)$. Since $X = [a,b]$ is a closed interval, then we can prove $\lim_{i\to\infty} u_i = \inf w(X) = \min_{x\in[a,b]} w(x)$.

Dynamically Improve Lipschitz Constant
A Lipschitz constant closer to K_{best} can greatly improve the speed of convergence of algorithm. We design a practical approach to dynamically update the current Lipschitz constant according to the information obtained from the previous iteration:

$$K = \eta \max_{j=1,...,i-1} \left| \frac{w(y_j) - w(y_{j-1})}{y_j - y_{j-1}} \right| \tag{22}$$

where $\eta > 1$. Please be noted that, because

$$\lim_{i\to\infty} \max_{j=1,...,i-1} \eta \left| \frac{w(y_j) - w(y_{j-1})}{y_j - y_{j-1}} \right| = \eta \sup_{y\in[a,b]} \frac{dw}{dy} > K_{best}$$

this dynamic update does not compromise the convergence.

5.2 Multi-dimensional Case

The basic idea is to decompose a multi-dimensional optimization problem into a sequence of nested one-dimensional subproblems. Then the minimuma of those one-dimensional minimization subproblems are back-propagated into the original dimension and the final global minimum is obtained.

$$\min_{x\in[a_i,b_i]^n} w(x) = \min_{x_1\in[a_1,b_1]} ... \min_{x_n\in[a_n,b_n]} w(x_1, ..., x_n) \tag{23}$$

We first introduce the definition of k-th Level Subproblem.

Definition 7 *The k-th level optimization subproblem, written as $\phi_k(x_1, ..., x_k)$, is defined as follows: for $1 \leq k \leq n-1$,*

$$\phi_k(x_1, ..., x_k) = \min_{x_{k+1}\in[a_{k+1},b_{k+1}]} \phi_{k+1}(x_1, ..., x_k, x_{k+1})$$

and for $k = n$,

$$\phi_n(x_1, ..., x_n) = w(x_1, x_2, ..., x_n)$$

Combining Expression (23) and Definition 7, we have that

$$\min_{x\in[a_i,b_i]^n} w(x) = \min_{x_1\in[a_1,b_1]} \phi_1(x_1)$$

which is actually an one-dimensional optimization problem and therefore can be solved by the method in Section 5.1.

However, when evaluating the objective function $\phi_1(x_1)$ at $x_1 = a_1$, we need to project a_1 into the next one-dimensional subproblem

$$\min_{x_2\in[a_2,b_2]} \phi_2(a_1, x_2)$$

We recursively do the projection until reaching the n-th level one-dimensional subproblem,

$$\min_{x_n\in[a_n,b_n]} \phi_n(a_1, a_2, ..., a_{n-1}, x_n)$$

Once solved, we back-propagate objective function values to the first-level $\phi_1(a_1)$ and continue searching from this level until the error bound is reached.

Convergence Analysis
We use mathematical induction to prove the convergence for multi-dimension case.

- Base case: for all $x \in \mathbb{R}$, $\lim_{i\to\infty} l_i = \inf_{x\in[a,b]} w(x)$ and $\lim_{i\to\infty}(u_i - l_i) = 0$ hold.
- Inductive step: if for all $x \in \mathbb{R}^k$, $\lim_{i\to\infty} l_i = \inf_{x\in[a,b]^k} w(x)$ and $\lim_{i\to\infty}(u_i - l_i) = 0$ are satisfied, then for all $x \in \mathbb{R}^{k+1}$, $\lim_{i\to\infty} l_i = \inf_{x\in[a,b]^{k+1}} w(x)$ and $\lim_{i\to\infty}(u_i - l_i) = 0$ hold.

The base case (*i.e.,* one-dimension case) is already proved in Section 5.1. Now we prove the inductive step.

Proof 5 *By the nested optimization scheme, we have*

$$\min_{\mathbf{x}\in[a_i,b_i]^{k+1}} w(\mathbf{x}) = \min_{x\in[a,b]} \Phi(x)$$

$$\Phi(x) = \min_{\mathbf{y}\in[a_i,b_i]^k} w(x, \mathbf{y})$$

Since $\min_{\mathbf{y}\in[a_i,b_i]^k} w(x,\mathbf{y})$ is bounded by an interval error $\epsilon_{\mathbf{y}}$, assuming $\Phi^*(x)$ is the accurate global minimum, then we have

$$\Phi^*(x) - \epsilon_{\mathbf{y}} \leq \Phi(x) \leq \Phi^*(x) + \epsilon_{\mathbf{y}}$$

So $k+1$-dimensional problem is decomposed as problem $\min_{x\in[a,b]} \Phi(x)$. The difference to real one-dimenisonal case is that evaluation of $\Phi(x)$ is not accurate but bounded by $|\Phi(x) - \Phi^*(x)| \leq \epsilon_{\mathbf{y}}, \forall x \in [a,b]$, where $\Phi^*(x)$ is accurate function evaluation.

Assuming that the minimal value obatined from our method is $\Phi^*_{min} = \min_{x\in[a,b]} \Phi^*(x)$ under an accurate function evaluation case, and the corresponding lower bound sequence, upper bound sequence are$\{l^*_0, ..., l^*_i\}$ and $\{u^*_0, ..., u^*_i\}$.

For the inaccurate evaluation case, we assume $\Phi_{min} = \min_{x\in[a,b]} \Phi(x)$, and its lower bound sequence and upper bound sequence are$\{l_0, ..., l_i\}$ and $\{u_0, ..., u_i\}$. The termination criteria for both cases are $|u^*_i - l^*_i| \leq \epsilon_x$ and $|u_i - l_i| \leq \epsilon_x$, and ϕ^* represent the ideal global minimum. Then we have $\phi^* - \epsilon_x \leq l_i$. Assuming that $l^*_i \in [x_k, x_{k+1}]$ and x_k, x_{k+1} are adjacent evaluation points, due to that $l^*_i = \inf_{x\in[a,b]} H(x;\mathcal{Y}_i)$, we have

$$\phi^* - \epsilon_x \leq l^*_i = \frac{\Phi^*(x_k) + \Phi^*(x_{k+1})}{2} - \frac{L(x_{k+1} - x_k)}{2}$$

Since $|\Phi(x_i) - \Phi^*(x_i)| \leq \epsilon_{\mathbf{y}}, \forall i = k, k+1$, thus we have

$$\phi^* - \epsilon_x \leq \frac{\Phi(x_k) + \Phi(x_{k+1})}{2} + \epsilon_{\mathbf{y}} - \frac{L(x_{k+1} - x_k)}{2}$$

Based on the search scheme, we know that

$$l_i = \frac{\Phi(x_k) + \Phi(x_{k+1})}{2} - \frac{L(x_{k+1} - x_k)}{2} \tag{24}$$

thus we have $\phi^* - l_i \leq \epsilon_{\mathbf{y}} + \epsilon_x$.

Similarly, we can get

$$\phi^* + \epsilon_x \geq u_i^* = \inf_{y \in \mathcal{Y}_i} \Phi^*(y) \geq u_i - \epsilon_{\mathbf{y}} \tag{25}$$

so $u_i - \phi^* \leq \epsilon_x + \epsilon_{\mathbf{y}}$. *By* $\phi^* - l_i \leq \epsilon_{\mathbf{y}} + \epsilon_x$ *and the termination criteria* $u_i - l_i \leq \epsilon_x$, *we have* $l_i - \epsilon_{\mathbf{y}} \leq \phi^* \leq u_i + \epsilon_{\mathbf{y}}$, i.e., *the accurate global minimum is also bounded.*

The proof indicates that the overall error bound of the nested scheme only increases linearly w.r.t. the bounds in the one-dimensional case. Moreover, an adaptive approach can be applied to optimise its performance without compromising convergence. The key observation is to relax the strict subordination inherent in the nested scheme and simultaneously consider all the univariate subproblems arising in the course of multidimensional optimization. For all the generated subproblems that are active, a numerical measure is applied. Then an iteration of the multidimensional optimization consists in choosing the subproblem with maximal measurement and carrying out a new trial within this subproblem. The measure is defined to be the maximal interval characteristics generated by the one-dimensional optimisation algorithm.

5.3 Proof of the NP-compleness

Here we describe the proof idea. For the upper bound, we first show that finding the optimal value for the one-dimensional case can be done in polynomial time with respect to the error bound ϵ. Then for the multiple-dimensional case, we have a non-deterministic algorithm to first guess a subset of dimensions and then conduct the one-dimensional optimisation one by one. The entire procedure can be done in polynomial time with a nondeterministic automaton, i.e., in NP.

For the lower bound, we show a reduction from the 3-SAT problem. For any instance φ of 3-SAT, we can construct a network f and an evaluation function o, such that the satisfiability of φ is equivalent to un-reachability of value 0 for the function $w = o \cdot f$.

6 Experiments

6.1 Comparison with State-of-the-art Methods

Two methods are choose as baseline methods in this paper:

- Reluplex [Katz *et al.*, 2017]: an SMT-based method for solving queries on DNNs with ReLU activations; we apply a bisection scheme to compute an interval until an error is reached
- SHERLOCK [Dutta *et al.*, 2017b]: a MILP-based method dedicated to output range analysis on DNNs with ReLU activations

Our software is implemented in Matlab 2018a, running on a notebook computer with i7-7700HQ CPU and 16GB RAM. Since Reluplex and SHERLOCK (not open-sourced) are designed on different software platforms, we take their experimental results from [Dutta *et al.*, 2017b], whose experimental environment is a Linux workstation with 63GB RAM and 23-Cores CPU (more powerful than ours) and $\epsilon = 0.01$. Following the experimental setup in [Dutta *et al.*, 2017b], we use their data (2-input and 1-output functions) to train six neural networks with various numbers and types of layers and neurons. The input subspace is $X' = [0, 10]^2$.

NN ID	Layer No.	Neutron No.	Time by SHERLOCK	Time by Reluplex	Our method
N-0	1	100	1.9s	1m 55s	0.4s
N-1	1	200	2.4s	13m 58s	1.0s
N-2	1	500	17.8s	Timeout	6.8s
N-3	1	500	7.6s	Timeout	5.3s
N-4	1	1000	7m 57.8s	Timeout	1.8s
N-5	6	250	9m 48.4s	Timeout	15.1s

Figure 2: Comparison with SHERLOCK and Reluplex

The comparison results are given in Fig. 2. It shows that, while the performance of both Reluplex and SHERLOCK are considerably affected by the increase of neuron numbers and layers, our method is not. For the benchmarked six neural networks, our averaging computation time is around $5s$, 36 fold improvement over SHERLOCK and nearly 100 fold improvement over Reluplex (excluding timeouts). Please note that, the result of our method is running on a notebook PC, which is significant less powerful than the 23-core CPU stations used for SHELOCK and Reluplex.

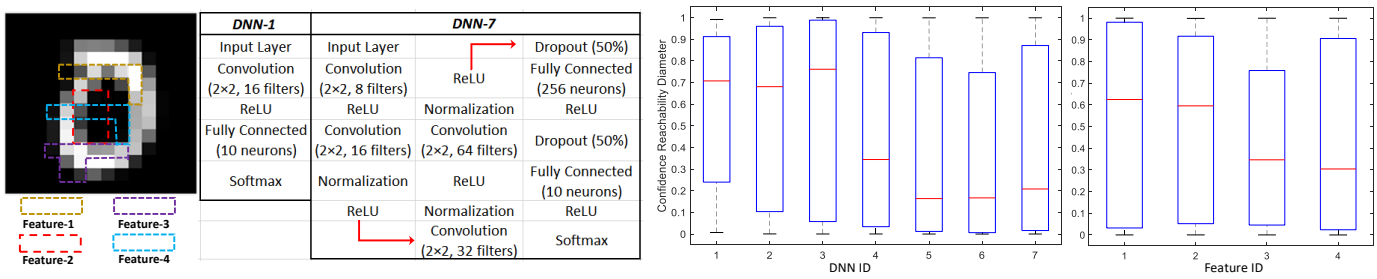
6.2 Safety and Robustness Verification by Reachability Analysis

We use our tool to conduct logit and output range analysis. Seven convolutional neural networks, represented as DNN-1,...,DNN-7, were trained on MNIST dataset. Images are resized into 14×14 so that a DNN with deeper layers tends to over-fit. The networks have different layer types including ReLu, dropout and normalization layers, and the layer numbers range from 5 to 19. Testing accuracies range from 95% to 99%, and $\epsilon = 0.05$ is used in our experiments.

We randomly choose 20 images (2 images per label) and manually choose 4 features such that each feature contains 8 pixels, i.e., $X' = [0, 1]^8$. Fig. 3 (a) illustrates the four features and the structures of two DNNs with the shallowest and deepest layers, *i.e.,* DNN-1 and DNN-7.

Safety Verification Fig. 4 (a) shows an example: for DNN-1, Feature-4 is *guaranteed to be safe* with respect to the image x and the input subspace X'. Specifically, the reachability interval is $R(\Pi_0, X', \epsilon) = [74.36\%, 99.98\%]$, which means that $l(\Pi_0, X', \epsilon) = 74.36\%$. By this, we have $u(\oplus_{-0}, X', \epsilon) \leq (1 - 0.7436) < 0.7436 = l(\Pi_0, X', \epsilon)$. Then, by Theorem 2, we have $\mathtt{S}(\text{DNN-1}, x, X')$. Intuitively, no matter how we manipulate this feature, the worst case is to reduce the confidence of being '0' from 99.95% (it original confidence probability) to 74.36%.

Statistical Comparison of Safety Fig. 4 (b) compares the ratios of safe images for different DNNs and features. It shows that: *i)* no DNN is 100% safe on those features; DNN-6 is the safest one and DNN-1, DNN-2 and DNN-3 are less safe, which means a DNN with well chosen layers are safer than those DNNs with very shallow or deeper layers; *ii)* the safety



(a) (b)

Figure 3: (a)The four features and the structures of DNN-1 and DNN-7. (b) Left: boxplots of confidence reachability diameters for 7 DNNs, each DNNs has 4×20 analysis; Right: boxplot of confidence reachability diameters for 4 features, each feature has 7×20 analysis. Red line represents the median value: a lower value indicates a more robust model or feature

performance of different DNNs is consistent for the same feature, which suggests that the feature matters - some features are easily perturbed to generate adversarial examples, e.g., Feature-1 and Feature-2.

Statistical Comparison of Robustness Fig. 3 (b) compares the robustness of networks and features with two boxplots over the reachability diameters, for the function o being Π_j for suitable j. We can see that DNN-6 and DNN-5 are the two most robust ones while DNN-1, DNN-2 and DNN-3 are less robust. Moreoer, Feature-1 and Feature-2 are less robust than Feature-3 and Feature-4.

Together with the above, we can see that reachability analysis with our tool can be used to quantify the safety and robustness of deep learning models. In the following, we address the comparison of networks over a fixed feature.

Safety Comparison of Networks By Fig. 4 (c), DNN-4 and DNN-6 are guaranteed to be safe w.r.t. the subspace defined by Feature-3. Moreover, the output range of DNN-7 is $[1.8\%, 100.0\%]$, which means that we can generate adversarial images by only perturbing this feature, among which the worst one is as shown in the figure with a confidence 1.8%. Thus, reachability analysis not only enables safety verification in a binary way (*i.e.,* safe or not safe) but also allows benchmarking of safety of different deep learning models in a principled, quantitive manner (*i.e.,* how safe) by quantifying the 'worst' adversarial example. Moreover, compared with retraining the model with 'regular' adversarial images, those 'worst' adversarial images are more effective in improving the robustness of DNNs [Kolter and Wong, 2017].

Robustness Comparison of Networks The bar chart in Fig. 4 (c) shows the reachability diameters of the networks over Feature-3, for the function o being Π_j. DNN-4 is the most robust one, with its output range being $[94.2\%, 100\%]$.

6.3 A Comprehensive Comparison with the State-of-the-arts

This section presents a comprehensive, high-level comparison of our approach with several existing approaches, which have been used in either range analysis or verification of DNNs, including SHERLOCK [Dutta *et al.*, 2017b], Reluplex [Katz *et al.*, 2017], Planet [Ehlers, 2017], MIP [Cheng *et al.*, 2017; Lomuscio and Maganti, 2017], BaB [Bunel *et al.*, 2017], as shown in Fig. 5.

Core Techniques Most existing approaches (SHERLOCK, Reluplex, Planet, MIP) are based on reduction to constraint solving, except for BaB which mixes constraint solving with local search. On the other hand, we are based on global optimization and the Lipschitz continuity of the networks. As indicated in Section 3 of the paper, all known layers used in classification tasks are Lipschitz continuous.

Workable Layer Types While we are able to work with all known layers used in classification tasks because they are Lipschitz continuous (proved in Section 3 of the paper), Planet, MIP and BaB can only work with Relu and Maxpooling and SHERLOCK and Reluplex can only work with Relu.

Running Time on ACAS-Xu Network We collect running time data from [Bunel *et al.*, 2017] on the ACAS-Xu network, and find that our approach has similar performance to BaB, and better than the others. No experiments for SHERLOCK is available. Please note that, comparing to their experimental platform (Desktop PC with i7-5930K CPU, 32GB RAM), ours is less powerful (Laptop PC with i7-7700HQ CPU, 16GB RAM). Please note that, although our approach performs well on this network, the actual strength of our approach is not the running time on small networks such as the ACAS-Xu network, but the capability to work with large-scale networks (such as those shown in Section 6.2).

Computational Complexity While all the approaches are in the same complexity class, NP, our approach is with respect to the number of input dimensions to be changed. This is compared with the other approaches which are with respect to the number of hidden neurons. It is known that the number of hidden neurons is much larger than the number of input dimensions, *e.g.,* there are nearly 6.5×10^6 neurons in AlexNet.

Applicable to state-of-the-art networks We are able to work with state-of-the-art networks with millions of neurons. However, other tools (Reluplex, Planet, MIP, BaB) can only work with hundreds of neurons. SHERLOCK can work with thousands of neurons thanks to its mixing of the MILP reduction with local search.

Maximum number of layers in tested DNNs We have validated our approaches on networks with 19 layers, while other approaches are validated on networks with up to 6 layers.

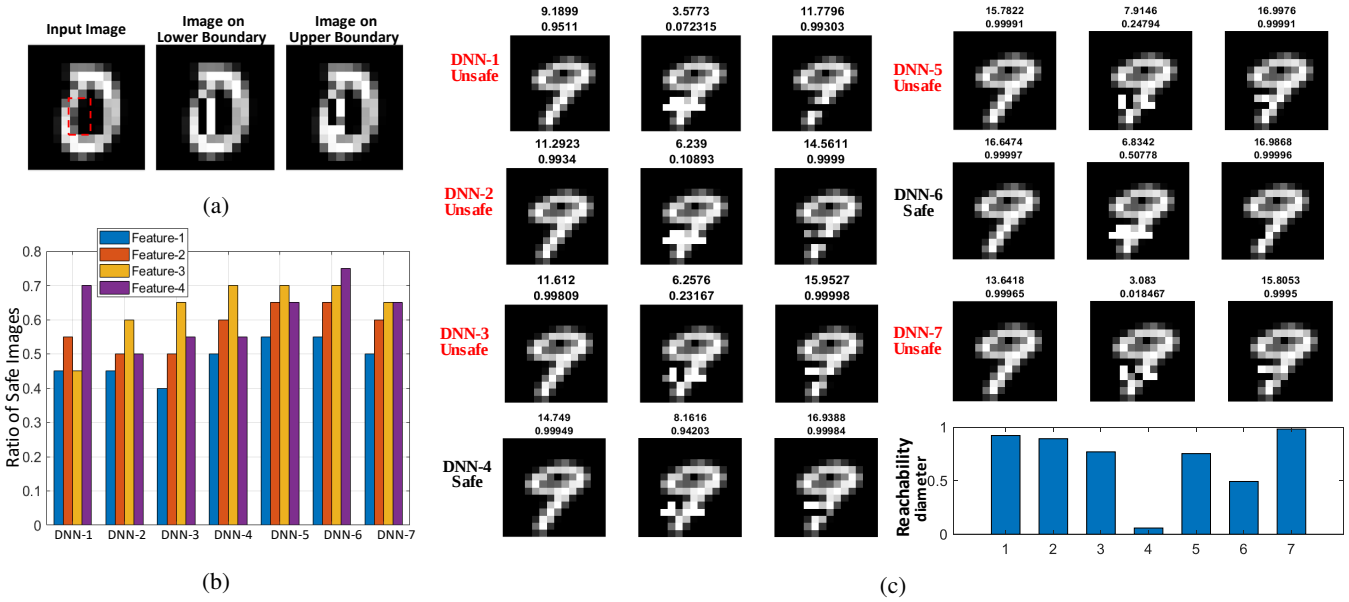


Figure 4: (a) Left: an original image (logit is 11.806, confidence of being '0' is 99.95%), area marked by dashed line is the feature. Middle: an image on the confidence lower bound. Right: an image on the confidence upper bound; for the output of label 0, the feature's output range is $[74.36\%, 99.98\%]$, and logit reachability is $[7.007, 13.403]$. (b) Ratios of safe images for 7 DNNs and 4 features. (c) A detailed example of comparing the safety and robustness of DNNs for Image '9' and feature-3: the first number in the title is logits and the second one is confidence; those unsafe cases are all misclassfied as '8'; the last bar chart shows their confidence reachability diameters.

In summary, our approach shows advantages regarding mainly the following aspects: *i)* the capability of working with state-of-the-art networks of large scale; *ii)* lower computational complexity, i.e., NP-complete with respect to the input dimensions to be changed, instead of the number of hidden neurons; and *iii)* the capability of working with more types of layers.

7 Conclusion

We propose, design and implement a reachability analysis tool for deep neural networks, which has provable guarantees and can be applied to neural networks with very deep layers and non-linear activation functions. The experiments demonstrate that our tool can be utilized to verify the safety of deep neural networks and quantitatively compare their robustness. We envision that this work marks an important step towards a practical, guaranteed safety verification on DNNs. Future work includes parallelizing this method in GPUs to test its scalability on large-scale models trained on ImageNet. Generalisation of this method to work with other deep learning models such as RNNs and deep reinforcement learning will also be explored.

References

[Amodei *et al.*, 2016] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[Bunel *et al.*, 2017] Rudy Bunel, Ilker Turkaslan, Philip HS Torr, Pushmeet Kohli, and M Pawan Kumar. Piecewise linear neural network verification: A comparative study. *arXiv preprint arXiv:1711.00455*, 2017.

[Carlini and Wagner, 2016] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[Cheng *et al.*, 2017] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. Maximum resilience of artificial neural networks. In Deepak D'Souza and K. Narayan Kumar, editors, *Automated Technology for Verification and Analysis*, pages 251–268, Cham, 2017. Springer International Publishing.

[Dutta *et al.*, 2017a] Souradeep Dutta, Susmit Jha, Sriram Sanakaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks. *CoRR*, abs/1709.09130, 2017.

[Dutta *et al.*, 2017b] Souradeep Dutta, Susmit Jha, Sriram Sanakaranarayanan, and Ashish Tiwari. Output range analysis for deep neural networks. *arXiv preprint arXiv:1709.09130*, 2017.

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

[Gergel *et al.*, 2016] Victor Gergel, Vladimir Grishagin, and Alexander Gergel. Adaptive nested optimization scheme for multidimensional global search. *Journal of Global Optimization*, 66(1):35–51, 2016.

	Core Techniques	Workable Layer Types	Running time on ASAC Xu	Computational Complexity	applicable to state-of-the-art DNNs?	Maximal No. of Layers in tested DNNs
SHERLOCK	MILP + Local Search	ReLu	No experiment	NP w.r.t. neuron no.	No (~6845 neurons)	6
Reluplex	SMT + LP	ReLu	O(10^4)-O(10^6)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
Planet	SAT + LP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
MIP	MIP	ReLu, maxpooling	O(10^3)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
BaB	MIP + BaB	ReLu, maxpooling	O(10^2)	NP w.r.t. neuron no.	No (~ 300 neurons)	6
DeepGO (this paper)	GO + Lipschitz Continuty	Layer with Lipschitz Continuty (Sigmod, Tanh, max-pooling, ReLu, etc)	O(10^2)	NP w.r.t. changed input dimensions	Yes (~10^5 neurons)	19

Figure 5: A high-level comparison with state-of-the-arts: SHERLOCK [Dutta *et al.*, 2017b], Reluplex [Katz *et al.*, 2017], Planet [Ehlers, 2017], MIP [Cheng *et al.*, 2017; Lomuscio and Maganti, 2017], BaB [Bunel *et al.*, 2017]

[Goodfellow *et al.*, 2014] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and Harnessing Adversarial Examples. *ArXiv e-prints*, December 2014.

[Grishagin *et al.*, 2018] Vladimir Grishagin, Ruslan Israfilov, and Yaroslav Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

[Huang *et al.*, 2017] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*, pages 3–29, 2017.

[Katz *et al.*, 2017] Guy Katz, Clark Barrett, David Dill, Kyle Julian, and Mykel Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. *arXiv preprint arXiv:1702.01135*, 2017.

[Kolter and Wong, 2017] J Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[Lomuscio and Maganti, 2017] Alessio Lomuscio and Lalit Maganti. An approach to reachability analysis for feed-forward relu neural networks. *CoRR*, abs/1706.07351, 2017.

[Moosavi-Dezfooli *et al.*, 2016] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016.

[Narodytska *et al.*, 2017] Nina Narodytska, Shiva Prasad Kasiviswanathan, Leonid Ryzhyk, Mooly Sagiv, and Toby Walsh. Verifying properties of binarized deep neural networks. *CoRR*, abs/1709.06662, 2017.

[Nguyen *et al.*, 2014] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR*, abs/1412.1897, 2014.

[Papernot *et al.*, 2015] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[Piyavskii, 1972] SA Piyavskii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12(4):57–67, 1972.

[Pulina and Tacchella, 2010] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. In Tayssir Touili, Byron Cook, and Paul Jackson, editors, *Computer Aided Verification*, pages 243–257, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Ruan *et al.*, 2018] Wenjie Ruan, Min Wu, Youcheng Sun, Xiaowei Huang, Daniel Kroening, and Marta Kwiatkowska. Global robustness evaluation of deep neural networks with provable guarantees for L0 norm. *arXiv preprint arXiv:1804.05805v1*, 2018.

[Sohrab, 2003] Houshang H Sohrab. *Basic real analysis*, volume 231. Springer, 2003.

[Sun *et al.*, 2018] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. Concolic testing for deep neural networks. *arXiv preprint arXiv:1805.00089v1*, 2018.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *In ICLR*. Citeseer, 2014.

[Torn and Zilinskas, 1989] Aimo Torn and Antanas Zilinskas. *Global Optimization*. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[Wicker *et al.*, 2017] Matthew Wicker, Xiaowei Huang, and Marta Kwiatkowska. Feature-guided black-box safety testing of deep neural networks. *CoRR*, abs/1710.07859, 2017.

[Xiang *et al.*, 2017] Weiming Xiang, Hoang-Dung Tran, and Taylor T Johnson. Output reachable set estimation and verification for multi-layer neural networks. *arXiv preprint arXiv:1708.03322*, 2017.