Iteratively Enhanced Semidefinite Relaxations for Efficient Neural Network Verification

Jianglin Lan[1], **Yang Zheng**[2], **Alessio Lomuscio**[3]
[1] James Watt School of Engineering, University of Glasgow, UK
[2] Department of Electrical and Computer Engineering, University of California San Diego, USA
[3] Department of Computing, Imperial College London, UK
Jianglin.Lan@glasgow.ac.uk, zhengy@eng.ucsd.edu, a.lomuscio@imperial.ac.uk

Abstract

We propose an enhanced semidefinite program (SDP) relaxation to enable the tight and efficient verification of neural networks (NNs). The tightness improvement is achieved by introducing a nonlinear constraint to existing SDP relaxations previously proposed for NN verification. The efficiency of the proposal stems from the iterative nature of the proposed algorithm in that it solves the resulting non-convex SDP by recursively solving auxiliary convex layer-based SDP problems. We show formally that the solution generated by our algorithm is tighter than state-of-the-art SDP-based solutions for the problem. We also show that the solution sequence converges to the optimal solution of the non-convex enhanced SDP relaxation. The experimental results on standard benchmarks in the area show that our algorithm achieves the state-of-the-art performance whilst maintaining an acceptable computational cost.

Introduction

The area of verification of neural networks (NNs) is concerned with the development of methods and tools to establish whether a given NN satisfies a given specification on a given input (Li et al. 2020; Liu et al. 2020). If a NN model is verified to be robust, no adversarial attack exists for the model, input and perturbation under analysis (Goodfellow, Shlens, and Szegedy 2014). NN verification has been used in many areas including safety-critical systems (Tran et al. 2020; Julian and Kochenderfer 2021; Kouvaros et al. 2021; Manzanas Lopez et al. 2021).

One type of NN verification methods is the complete approach which guarantees no false negatives or false positives are generated. Existing complete approaches for NN verification build on mixed-integer linear programming (MILP) (Bastani et al. 2016; Lomuscio and Maganti 2017; Tjeng, Xiao, and Tedrake 2019; Anderson et al. 2020; Botoeva et al. 2020) or satisfiability modulo theories (Ehlers 2017; Katz et al. 2021). These complete verifiers can guarantee to resolve any verification query but often come at a high computational cost, thus not scaling well to large-scale NNs.

Another type of verification methods is the incomplete approach that involves overapproximations of the NN and has a better scalability to larger NNs than the complete approach. The existing incomplete verifiers involve abstraction and approximation via linear approximations (Henriksen and Lomuscio 2020, 2021; Wang et al. 2021; Hashemi, Kouvaros, and Lomuscio 2021) or convex approximation (Xu et al. 2020; Dvijotham et al. 2018; Wong and Kolter 2018; Chen et al. 2021; Raghunathan, Steinhardt, and Liang 2018; Fazlyab, Morari, and Pappas 2022; Batten et al. 2021). Some incomplete verifiers can offer completeness guarantees by combining symbolic interval propagation or convex relaxation with input refinement (*e.g.,* `ReluVal` (Wang et al. 2018c)) or neuron refinement (*e.g.,* `Neurify` (Wang et al. 2018a), `VeriNet` (Henriksen and Lomuscio 2020), `DEEPSPLIT` (Henriksen and Lomuscio 2021), β-`CROWN` (Wang et al. 2021) and `OSIP` (Hashemi, Kouvaros, and Lomuscio 2021)), but requiring a very large numbers of splitting refinements.

This paper focuses on the incomplete approach. The success of incomplete methods hinges on the tightness of the approximations generated by the technique, because a looser approximation leads to more unsolvable verification queries and a lower scalability capability. The triangle relaxation (Ehlers 2017) offers the tightest possible convex approximation for a single Rectified Linear Unit (ReLU) neuron and has been used for faster bound propagations (Weng et al. 2018; Singh et al. 2019a; Tjandraatmadja et al. 2020; Müller et al. 2021). These incomplete methods are based on polynomial-time solvable linear program (LP) problems and can achieve state-of-the-art (SoA) performance. However, their efficacy is fundamentally limited by the tightness of convex relaxations. This is known as the "convex relaxation barrier" indicating that even optimal convex relaxations on a single neuron fail to obtain tight approximation of the overall model (Salman et al. 2019). This problem can be approached in two ways. The first is applying convex relaxations to multiple neurons. The representative methods include `DeepPoly` (Singh et al. 2019a), `kPoly` (Singh et al. 2019b), `OptC2V` (Tjandraatmadja et al. 2020), and `PRIMA` (Müller et al. 2021). The second is using stronger relaxations, among which the most promising ones achieving tightness and efficiency are based on semidefinite program (SDP) (Raghunathan, Steinhardt, and Liang 2018; Fazlyab, Morari, and Pappas 2022; Batten et al. 2021).

Empirically, the standard SDP relaxation (Raghunathan,

Steinhardt, and Liang 2018) is considerably tighter than the standard LP relaxations. However, as illustrated in (Zhang 2020), SDP relaxations become loose for multiple hidden layers. Linear cuts (Batten et al. 2021) or non-convex cuts (Ma and Sojoudi 2020) were introduced to further tighten SDP relaxations. The standard SDP approaches (Raghunathan, Steinhardt, and Liang 2018; Zhang 2020; Ma and Sojoudi 2020) achieve tighter relaxations than LPs, but incur a considerable extra computational effort, making them not scalable to larger models. To alleviate the computational cost, a memory-efficient first-order algorithm was introduced in (Dathathri et al. 2020). With the same aim, layerwise SDP relaxations were used in (Batten et al. 2021; Newton and Papachristodoulou 2021) by exploiting the cascaded NN structures based on chordal graph decomposition (Zheng, Fantuzzi, and Papachristodoulou 2021). To the best of our knowledge, the `LayerSDP` method (Batten et al. 2021) achieves the tightest relaxations by combining SDP relaxation with triangle relaxation (Ehlers 2017). Even so, it is observed in (Batten et al. 2021) that the relaxation gap of `LayerSDP` is still considerable in large networks. This leads to an increased false negative rate as the model size grows and thus limits applicability of the approach.

In this paper, we advance the SDP approach for NN verification with a new relaxation and an efficient algorithm to solve it. The relaxation we propose combines the `LayerSDP` method (Batten et al. 2021) with a set of valid nonlinear constraints, which gives a provably tighter relaxation than the SoA SDP methods. The inclusion of nonlinear constraints results in a non-convex SDP problem that is generally harder to solve than `LayerSDP`. We further develop an iterative algorithm to obtain the optimal solution of the non-convex SDP problem. The algorithm initialises from the `LayerSDP` solution and solves iteratively an auxiliary convex SDP problem. The auxiliary SDP problem is derived from the original non-convex SDP problem and its size is around the same as `LayerSDP`. Our theoretical analysis shows that the solution sequence produced by the iterative algorithm is tighter than the SoA SDP methods, and can converge to the optimal solution of the non-convex SDP problem. The experiments on various standard benchmarks confirm that the proposed SDP method is considerably tighter than the present SoA, whilst having competitive efficiency. Our previous work (Lan, Zheng, and Lomuscio 2022) constructed effective linear cuts using the linear reformulation technique (RLT) and added them to `LayerSDP` (Batten et al. 2021) to produce a tighter solution. These RLT-based linear cuts can be directly added to the SDP problems solved in this paper, meaning that the method proposed in this paper always gives a provably tighter solution than the method in (Lan, Zheng, and Lomuscio 2022).

Preliminaries

We use the symbol $\mathbb{R}^n$ to denote the n-dimensional Euclidean space, $\|\cdot\|_\infty$ to denote the standard ℓ_∞ norm, and $\mathbb{I}_b$ to denote a sequence of non-zero integers from 0 to b. We use $\text{diag}(X)$ to stack the main diagonals of matrix X as a column, and $\odot$ to refer to the element-wise product. We use $\mathbf{0}_{n\times m}$ to denote a $n \times m$ zero matrix and $\mathbf{1}_n$ to denote a $n \times 1$ vector of ones. We use $P_i[z]$ to represent the elements of matrix P_i corresponding to the vector or matrix z.

We focus on feed-forward ReLU NNs. A network $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$ with L hidden layers, n_0 inputs and n_{L+1} outputs is defined as follows: $f(x_0) = W_L x_L + b_L$ with $x_{i+1} = \text{ReLU}(\hat{x}_{i+1})$ and $\hat{x}_{i+1} = W_i x_i + b_i$, $i \in \mathbb{I}_{L-1}$, where x_0 is the input, $f(x_0)$ is the output, $\hat{x}_{i+1} \in \mathbb{R}^{n_{i+1}}$ is the pre-activation vector, $x_{i+1} \in \mathbb{R}^{n_{i+1}}$ is the post-activation vector, $W_i \in \mathbb{R}^{n_{i+1}\times n_i}$ is the weight and $b_i \in \mathbb{R}^{n_{i+1}}$ is the bias. We focus on classification NNs whereby an input x_0 is assigned to the class $j^\star$ whose corresponding output has the highest value: $j^\star = \arg\max_{j=1,2,\ldots,n_{L+1}} f(x_0)_j$.

This paper contributes to solving the local robustness verification problem. Given a NN f and a clean input $\bar{x}$ under the ℓ_∞-norm perturbation ϵ, the local robustness problem concerns determining whether f is robust on $\bar{x}$, *i.e.*, whether $f(x_0)_{j^\star} - f(x_0)_j > 0$ holds for all $j = 1, 2, \ldots, n_{L+1}$ with $j \neq j^\star$, and for all x_0 satisfying $\|x_0 - \bar{x}\|_\infty \leq \epsilon$. This verification problem can be formulated and solved as an optimisation problem (Raghunathan, Steinhardt, and Liang 2018; Batten et al. 2021):

$$
\begin{aligned}
\gamma^* := \quad & \min_{\{x_i\}_{i=0}^{L}} \; c^\mathsf{T} x_L + c_0 \\
\text{subject to} \quad & x_{i+1} = \text{ReLU}(W_i x_i + b_i),\ i \in \mathbb{I}_{L-1}, && \text{(1a)} \\
& \|x_0 - \bar{x}\|_\infty \leq \epsilon, && \text{(1b)} \\
& l_{i+1} \leq x_{i+1} \leq u_{i+1},\ i \in \mathbb{I}_{L-1}, && \text{(1c)}
\end{aligned}
$$

where $c^\mathsf{T} = W_L(j^\star, :) - W_L(j, :)$ and $c_0 = b_L(j^\star) - b_L(j)$, and l_{i+1} and u_{i+1} are the lower and upper activation vector bounds that can be computed by using bound propagation methods (Henriksen and Lomuscio 2020; Wang et al. 2018b). Without loss of generality, we assume that the considered verification problem is feasible. We solve the problem (1) for every potential adversarial target $j = 1, 2, \ldots, n_{L+1}$, where $j \neq j^\star$, to obtain the optimal objective value γ^*. If $\gamma^* > 0$ in all the cases, the NN is verified to be robust on the input $\bar{x}$ under the adversarial input x_0.

Since the ReLU constraints in (1a) are nonlinear, the optimisation problem (1) is non-convex and in general hard to solve. A promising direction is converting problem (1) into a convex relaxation problem that can be efficiently solved. Solving the convex relaxation gives an optimal value γ^*_{cvx} that is a valid lower bound to γ^*, *i.e.*, $\gamma^* \geq \gamma^*_{\text{cvx}}$. When $\gamma^*_{\text{cvx}} > 0$, then $\gamma^* > 0$ always holds and the input is robust. Clearly, a tighter convex relaxation (*i.e.*, with a smaller relaxation gap $\gamma^* - \gamma^*_{\text{cvx}}$) increases the number of verification queries that can be solved. Tightness is thus the key design objective of a convex relaxation method given that its computational cost is acceptable.

The LP relaxation (Ehlers 2017) is a widely used convex relaxation method in the literature. Its key idea is using a triangle relaxation to approximate each ReLU constraint in (1a) with a convex hull:

$$
\begin{aligned}
& x_{i+1} \geq 0,\ x_{i+1} \geq W_i x_i + b_i, \\
& x_{i+1} \leq k_i \odot (W_i x_i + b_i - \hat{l}_{i+1}) + \text{ReLU}(\hat{l}_{i+1}),
\end{aligned} \tag{2}
$$

where $k_i = (\text{ReLU}(\hat{u}_{i+1}) - \text{ReLU}(\hat{l}_{i+1}))/(\hat{u}_{i+1} - \hat{l}_{i+1})$.

The vectors $\hat{u}_{i+1}$ and $\hat{l}_{i+1}$ are the upper and lower bounds of the pre-activation vector $\hat{x}_{i+1}$, respectively.

Replacing the nonlinear constraint (1a) with the linear constraints in (2) leads to an LP relaxation to the non-convex optimisation problem (1). The LP is relatively easy to solve, but there is usually a large relaxation gap for many NNs, known as the *convex relaxation barrier* in (Salman et al. 2019). This paper aims to develop a tighter convex relaxation method based on semidefinite program (SDP) (Parrilo 2000; Lasserre 2009).

Convex SDP Relaxations and a New Non-convex Enhancement

In this section we describe a few SDP relaxations to the non-convex optimisation problem (1), including the existing SDP methods and our new non-convex SDP relaxation.

Convex SDP Relaxations. The starting point is to observe that the nonlinear ReLU constraints in (1a) can equivalently be replaced by the set of linear and quadratic constraints:

$$\begin{aligned} &x_{i+1} \geq 0,\ x_{i+1} \geq W_i x_i + b_i, \\ &x_{i+1} \odot (x_{i+1} - W_i x_i - b_i) = 0,\ i \in \mathbb{I}_{L-1}. \end{aligned} \tag{3}$$

Given a verification instance, we can know the *strictly active neurons* at each layer. If the j-th neuron, $1 \leq j \leq n_{i+1}$, at layer $i+1$ is strictly active, then for this neuron the set of constraints in (3) can equivalently be replaced by a single linear equality constraint $x_{i+1}(j) = W_i(j,:)x_i + b_i(j)$.

The input constraints in (1b) and (1c) can also be reformulated as the quadratic constraints:

$$x_i \odot x_i - (l_i + u_i) \odot x_i + l_i \odot u_i \leq 0,\ i \in \mathbb{I}_L, \tag{4}$$

where $l_0 = \bar{x} - \epsilon \mathbf{1}_{n_0}$ and $u_0 = \bar{x} + \epsilon \mathbf{1}_{n_0}$.

The techniques of polynomial lifting (Parrilo 2000; Lasserre 2009) can be used to reformulate the quadratic constraints in (3) and (4) as linear constraints in terms of new variables. Specifically, to couple all the ReLU constraints of the network together, we define a single positive semidefinite (PSD) matrix P as:

$$P = \mathbf{x}\mathbf{x}^\mathsf{T}, \tag{5}$$

with $\mathbf{x} = [1,\ x_0^\mathsf{T},\ x_1^\mathsf{T},\ \cdots,\ x_L^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{1+\sum_{i=0}^{L} n_i}$. By using (5), the constraints (3) and (4) can be reformulated as linear constraints of P. This idea was first utilised for NN verification in (Raghunathan, Steinhardt, and Liang 2018). Dropping the rank-one constraint on P results in a global SDP relaxation to the non-convex optimisation problem (1), with a potentially smaller relaxation gap than the LP relaxation, as empirically observed in (Raghunathan, Steinhardt, and Liang 2018). However, the global SDP relaxation is very computationally expensive for large NNs due to high dimensionality of P, thus limiting applicability of the approach.

Thanks to the inherent cascading structure of feed-forward NNs, the activation vector of layer $i+1$ depends only on its preceding layer i, for all $i \geq 0$. We can exploit this cascading structure to derive a layer-based SDP relaxation with a set of PSD matrices that are of smaller sizes than the matrix P. The layer-based PSD matrices are defined as:

$$P_i = \mathbf{x}_i \mathbf{x}_i^\mathsf{T},\ i \in \mathbb{I}_{L-1}, \tag{6}$$

with $\mathbf{x}_i = [1,\ x_i^\mathsf{T}, x_{i+1}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{\bar{n}_i}$ and $\bar{n}_i = 1 + n_i + n_{i+1}$. By using (6), we reformulate (3) and (4) as a set of linear constraints on the elements of P_i:

$$P_i[x_{i+1}] \geq 0,\ P_i[x_{i+1}] \geq W_i P_i[x_i] + b_i,\ i \in \mathbb{I}_{L-1}, \tag{7a}$$

$$\mathrm{diag}(P_i[x_{i+1}x_{i+1}^\mathsf{T}] - W_i P_i[x_i x_{i+1}^\mathsf{T}]) - b_i \odot P_i[x_{i+1}] = 0,\ i \in \mathbb{I}_{L-1}, \tag{7b}$$

$$\mathrm{diag}(P_i[x_i x_i^\mathsf{T}]) - (l_i + u_i) \odot P_i[x_i] + l_i \odot u_i \leq 0,\ i \in \mathbb{I}_{L-1}, \tag{7c}$$

$$\mathrm{diag}(P_{L-1}[x_L x_L^\mathsf{T}]) - (l_L + u_L) \odot P_{L-1}[x_L] + l_L \odot u_L \leq 0, \tag{7d}$$

$$P_i[\bar{x}_{i+1}\bar{x}_{i+1}^\mathsf{T}] = P_{i+1}[\bar{x}_{i+1}\bar{x}_{i+1}^\mathsf{T}],\ i \in \mathbb{I}_{L-2}, \tag{7e}$$

$$P_i[1] = 1,\ P_i \succeq 0,\ \mathrm{rank}(P_i) = 1,\ i \in \mathbb{I}_{L-1}, \tag{7f}$$

where $\bar{x}_{i+1} = [1,\ x_{i+1}^\mathsf{T}]^\mathsf{T}$. The constraint (7e) ensures input-output consistency (Batten et al. 2021) and (7f) is equivalent to (6) as shown in (Horn and Johnson 2012).

By replacing (3) and (4) with (7) and dropping the rank-one constraint $\mathrm{rank}(P_i) = 1$, the non-convex problem (1) is relaxed to a convex layer SDP (Batten et al. 2021):

$$\begin{aligned} \gamma^*_{\text{LayerSDP}} := \min_{\{P_i\}_{i=0}^{L-1}}\ & c^\mathsf{T} P_{L-1}[x_L] + c_0 \\ \text{subject to}\quad & \text{(7a), (7b), (7c), (7d), (7e)}, && \text{(8a)} \\ & P_i[1] = 1,\ P_i \succeq 0,\ i \in \mathbb{I}_{L-1}, && \text{(8b)} \\ & P_i[x_{i+1}] \leq A_i P_i[x_i] + B_i,\ i \in \mathbb{I}_{L-1}, && \text{(8c)} \end{aligned}$$

where (8c) is reformulated from the last inequality in (2) with $A_i = k_i \odot W_i$ and $B_i = k_i \odot (b_i - \hat{l}_{i+1}) + \mathrm{ReLU}(\hat{l}_{i+1})$.

As shown in (Batten et al. 2021), the layer SDP relaxation (8) is considerably easier to solve than the global SDP relaxation (Raghunathan, Steinhardt, and Liang 2018) and is also tighter, *i.e.*, $\gamma^*_{\text{GlobalSDP}} \leq \gamma^*_{\text{LayerSDP}} \leq \gamma^*$, by introducing the linear cut (8c). However, due to dropping the rank-one constraint, the relaxation gap of layer SDP is still considerable in large NNs, thereby limiting the scalability and applicability of the approach.

Non-convex Layer-based SDP Relaxation. A promising way to reduce the SDP relaxation gap is introducing constraints (or cuts) to enforce the rank conditions in (7f). Non-convex cuts in the form of $\phi_i^\mathsf{T}(\tilde{X} - \tilde{x}\tilde{x}^\mathsf{T})\phi_i \leq 0$ with the designed constant vectors ϕ_i, $i = 1, \ldots, \sum_{i=0}^{L} n_i$, where $\tilde{x} = [x_0^\mathsf{T}, \cdots, x_L^\mathsf{T}]^\mathsf{T}$ and $\tilde{X} = P[\tilde{x}\tilde{x}^\mathsf{T}]$, are introduced in (Ma and Sojoudi 2020) to tighten the global SDP relaxation (Raghunathan, Steinhardt, and Liang 2018). The resulting non-convex SDP is solved via successively generating large numbers of linear cuts to approximate the non-convex cuts. This method has limited scalability due to the underlying computationally demanding global SDP and the additional large number of linear cuts.

To alleviate the computational challenge, we take inspirations from recent advances on SDP relaxation for non-convex QCQP (quadratically constrained quadratic program). In particular, a single nonlinear constraint in the form of $c^\mathsf{T}(P - \mathbf{x}\mathbf{x}^\mathsf{T})c = 0$ is used for generic SDP relaxations

in (Luo, Bai, and Peng 2019). This constraint must use the same vector c as the objective function, which limits its capability in reducing the relaxation gap (Ma and Sojoudi 2020). The method is thus undesirable for NN verification, because the vector c only has very few non-zero elements associated with the last hidden layer. Regarding the layer SDP relaxation (8), the vector c is only associated with the last PSD matrix P_{L-1}. A direct adoption of the nonlinear constraint in (Luo, Bai, and Peng 2019) to (8) will then only be able to enforce the rank condition of P_{L-1}.

We propose a new type of nonlinear constraints that allows using a set of vectors to enforce the rank conditions of all PSD matrices P_i, $i \in \mathbb{I}_{L-1}$, by exploiting the NN activation pattern, which will be explained in detail in the next section. In particular, we introduce a nonlinear constraint to each P_i in (8) and obtain the novel layer SDP relaxation:

$$\begin{aligned} \gamma^*_{\text{ncvxSDP}} := \quad & \min_{\{P_i\}_{i=0}^{L-1}} c^\mathsf{T} P_{L-1}[x_L] + c_0 \\ \text{subject to} \quad & (8a), (8b), (8c), && \text{(9a)} \\ & v_i^\mathsf{T}(P_i - \mathbf{x}_i\mathbf{x}_i^\mathsf{T})v_i = 0,\ i \in \mathbb{I}_{L-1}, && \text{(9b)} \end{aligned}$$

where $v_i \in \mathbb{R}^{\bar{n}_i}$, $i \in \mathbb{I}_{L-1}$, are verification instance based constant vectors to be constructed (see Algorithm 1). Hereby we explicitly write the vector $\mathbf{x}_i$ (referring to the elements in the first column of P_i, *i.e.*, $P_i[\mathbf{x}_i]$) to ease the analysis.

The introduction of the constraints in (9b) results in a tighter SDP relaxation, as shown in Theorem 1.

Theorem 1. *Given a feasible verification instance, we have*

$$\gamma^*_{\text{LayerSDP}} \leq \gamma^*_{\text{ncvxSDP}} \leq \gamma^*.$$

Intuitively, the relations in Theorem 1 hold because we add the extra valid constraints in (9b) into the layer SDP relaxation (8). Due to the nonlinear constraint (9b), the new layer SDP relaxation (9) is non-convex and harder to solve than the original layer SDP relaxation (8). Our main technical contribution is to circumvent the non-convexity issue by developing an iterative algorithm. This iterative algorithm, detailed in the next section, recursively solves a convex SDP problem of around the same size as (8). The algorithm is able to generate an objective value that initialises from $\gamma^*_{\text{LayerSDP}}$ and converges to $\gamma^*_{\text{ncvxSDP}}$ for v_i constructed in Algorithm 1.

An Iterative Algorithm for Solving the Non-convex Layer SDP Relaxation

This section describes an iterative algorithm to compute the optimal solution of the non-convex layer SDP relaxation (9) at moderate computational cost. We first formulate an auxiliary convex layer SDP, then use it to develop an iterative algorithm and apply it to the NN verification problem.

Auxiliary SDP Problem. Although the non-convex layer SDP relaxation (9) is generally hard to solve, its optimal objective value $\gamma^*_{\text{ncvxSDP}}$ is bounded below by $\gamma^*_{\text{LayerSDP}}$, as shown in Theorem 1. This lower bound can be efficiently solved from the convex layer SDP relaxation (8). Hence, we can use $\gamma^*_{\text{LayerSDP}}$ as a start point to search for the value of $\gamma^*_{\text{ncvxSDP}}$. This inspires us to generate an objective value sequence $\{\gamma^{(k)}_{\text{iter}}\}$ by solving an auxiliary convex SDP problem recursively. Our aim is to generate the objective value sequence that is bounded by $\gamma^*_{\text{LayerSDP}} \leq \gamma^{(k)}_{\text{iter}} \leq \gamma^*_{\text{ncvxSDP}}$ and can converge to $\gamma^*_{\text{ncvxSDP}}$. We want to ensure these bounds so that $\gamma^{(k)}_{\text{iter}}$ is always tighter than $\gamma^*_{\text{LayerSDP}}$ and remains as a valid lower bound to γ^*. By having these properties, the sequence $\{\gamma^{(k)}_{\text{iter}}\}$ can then be used for NN verification.

The above analysis motivates us to consider the auxiliary SDP problem in the form of:

$$\begin{aligned} \gamma^*_{\text{auxSDP}} := \quad & \min_{\{P_i\}_{i=0}^{L-1}} J_{L-1} + \alpha \sum_{i=0}^{L-2} J_i \\ \text{subject to} \quad & (8a), (8b), (8c), && \text{(10a)} \\ & v_i^\mathsf{T}\mathbf{x}_i = \delta_i,\ i \in \mathbb{I}_{L-2}, && \text{(10b)} \\ & v_{L-1}^\mathsf{T}\mathbf{x}_{L-1} \geq \delta_{L-1}, && \text{(10c)} \end{aligned}$$

where $J_i = v_i^\mathsf{T} P_i v_i - 2\delta_i v_i^\mathsf{T}\mathbf{x}_i + \delta_i^2$, $i \in \mathbb{I}_{L-1}$. The weight α is a user-specified positive constant. Its value is set as $\alpha > 1$ to penalise more on the SDP relaxations of the first $L-1$ layers. This is useful to obtain a tighter NN output, as it is influenced by the SDP relaxations of the first $L-1$ layers. δ_i, $i \in \mathbb{I}_{L-2}$, are user-given scalars. v_i, $i \in \mathbb{I}_{L-1}$, are constant vectors to be constructed in Algorithm 1 and δ_{L-1} is a constant scalar to be constructed in Algorithm 2. They are constructed such that the constraints in (10b) and (10c) are always satisfied for any feasible solution to the non-convex layer SDP problem (9). It thus enables us to solve the auxiliary SDP problem to obtain a valid lower bound of $\gamma^*_{\text{ncvxSDP}}$.

We construct the vector set $\{v_i\}_{i=0}^{L-1}$ from Algorithm 1 by using the *strictly active neurons* of the NN (whose input and output are equal), which generally exist for each test input. By doing so, we know (10b) and $v_{L-1}^\mathsf{T}\mathbf{x}_{L-1} = c^\mathsf{T} P_{L-1}[x_L]$ are always satisfied. Hence, we can solve the auxiliary SDP (10) to obtain $v_{L-1}^\mathsf{T}\mathbf{x}_{L-1}$ and thus the objective value $\gamma_{\text{iter}} = v_{L-1}^\mathsf{T}\mathbf{x}_{L-1} + c_0 = c^\mathsf{T} P_{L-1}[x_L] + c_0$. We further show that solving this auxiliary SDP can obtain $\gamma_{\text{iter}} = \gamma^*_{\text{ncvxSDP}}$, based on Proposition 2.

Proposition 2. *By constructing the set $\{v_i\}_{i=0}^{L-1}$ from Algorithm 1, the optimal objective value of the auxiliary SDP problem* (10) *has the properties:*

1) *$\gamma^*_{auxSDP} \geq 0$ for any constructed δ_{L-1} that satisfies* (10c).
2) *$\gamma^*_{auxSDP} = 0$ if and only if the optimal solution satisfies $v_i^\mathsf{T}(P_i - \mathbf{x}_i\mathbf{x}_i^\mathsf{T})v_i = 0$, $i \in \mathbb{I}_{L-1}$.*
3) *When $\gamma^*_{auxSDP} = 0$, $\delta_{L-1} + c_0 \geq \gamma^*_{ncvxSDP}$.*
4) *If δ_{L-1} is constructed to satisfy $\delta_{L-1} + c_0 = \gamma^*_{ncvxSDP}$, then $c^\mathsf{T} P_{L-1}[x_L] + c_0 = \gamma^*_{ncvxSDP}$.*

According to Proposition 2, if constructing the scalar δ_{L-1} such that $\delta_{L-1} + c_0 = \gamma^*_{\text{ncvxSDP}}$, then solving the problem (10) gives the objective value $\gamma_{\text{iter}} = c^\mathsf{T} P_{L-1}[x_L] + c_0 = \gamma^*_{\text{ncvxSDP}}$. To achieve this, we propose an algorithm to iteratively update the value of δ_{L-1} and generate the objective value sequence $\{\gamma^{(k)}_{\text{iter}}\}$ that converges to $\gamma^*_{\text{ncvxSDP}}$.

Iterative Algorithm and Application to NN Verification. The iterative algorithm is based on solving the problem (10) at each iteration with the scalar δ_{L-1} that is changed with the

Algorithm 1: Constructing the set of vectors $\{v_i\}_{i=0}^{L-1}$

Input: $\{W_i\}_{i=0}^{L-1}$, $\{b_i\}_{i=0}^{L-1}$, $\{\hat{l}_{i+1}\}_{i=0}^{L-1}$, c, $\{\delta_i\}_{i=0}^{L-2}$.
for $i = 0, 1, \ldots, L-1$ **do**
  $w_i^{\text{in}} \leftarrow \mathbf{0}_{1\times n_i}$, $b_i^{\text{in}} \leftarrow 0$, $w_i^{\text{out}} \leftarrow \mathbf{0}_{1\times n_{i+1}}$.
  for $j = 1, 2, \ldots, n_{i+1}$ **do**
    if $\hat{l}_{i+1}(j) \geq 0$ **then**
      $w_i^{\text{in}} \leftarrow w_i^{\text{in}} + W_i(j,:)$, $b_i^{\text{in}} \leftarrow b_i^{\text{in}} + b_i(j)$, $w_i^{\text{out}}(j) \leftarrow 1$.
    end if
  end for
  if $i \leq L-2$ **then**
    $v_i \leftarrow [\delta_i + b_i^{\text{in}},\ w_i^{\text{in}},\ -w_i^{\text{out}}]^{\mathsf{T}}$.
  else
    $v_i \leftarrow [b_i^{\text{in}},\ w_i^{\text{in}},\ c^{\mathsf{T}} - w_i^{\text{out}}]^{\mathsf{T}}$.
  end if
end for
Output: $\{v_i\}_{i=0}^{L-1}$.

iterations. The proposed iterative algorithm is summarised in Algorithm 2. The initial value of δ_{L-1} is set as $\delta_{L-1}^{(1)} = \gamma^*_{\text{LayerSDP}} - c_0$, where $\gamma^*_{\text{LayerSDP}}$ is the optimal objective value of the layer SDP relaxation (8). For each given $\delta_{L-1}^{(k)}$, the auxiliary SDP problem (10) is solved to obtain the objective value $\gamma_{\text{iter}}^{(k)}$. At each iteration, the obtained optimal objective $\gamma^{*(k)}_{\text{auxSDP}}$ of problem (10) is used to update the value of δ_{L-1}. The iteration is terminated when $\gamma^{*(k)}_{\text{auxSDP}}$ is smaller than a prescribed tolerance $\varepsilon \geq 0$. Algorithm 2 outputs the objective value $\gamma_{\text{iter}}^{(k_0)}$ for robustness verification of NNs.

To apply Algorithm 2 to the problem of robustness verification, we first need to ensure that:

1) $\gamma_{\text{iter}}^{(k)}$ is always a valid lower bound to γ^*.

2) The sequence $\{\gamma_{\text{iter}}^{(k)}\}$ converges to $\gamma^*_{\text{ncvxSDP}}$.

The first item is concerned with *soundness* of the algorithm, *i.e.*, whether $\gamma_{\text{iter}}^{(k)} > 0$ implies that $\gamma^* > 0$ and the input is robust. The second item concerns the *tightness* of the algorithm, *i.e.,* whether the sequence $\{\gamma_{\text{iter}}^{(k)}\}$ can achieve the tightness of $\gamma^*_{\text{ncvxSDP}}$ as in Theorem 1.

We now show the *soundness* and *tightness* of Algorithm 2 by analysing properties of the sequence $\{\gamma_{\text{iter}}^{(k)}\}$, which uses the properties of the sequence $\{\delta_{L-1}^{(k)}\}$ stated in Lemma 3.

Lemma 3. *The sequence $\{\delta_{L-1}^{(k)}\}$ generated by Algorithm 2 satisfies:*

$$\gamma^*_{LayerSDP} \leq \delta_{L-1}^{(k)} + c_0 \leq \gamma^*_{ncvxSDP},\ \forall k \geq 1.$$

By setting $\varepsilon = 0$, the sequence converges to $\gamma^*_{ncvxSDP} - c_0$ when $\gamma^{*(k)}_{auxSDP} = 0$.

Lemma 3 shows that $\gamma^{*(k)}_{\text{auxSDP}}$ can be used to check when the sequence $\{\delta_{L-1}^{(k)}\}$ converges. This confirms the appropriateness of using $\gamma^{*(k)}_{\text{auxSDP}} \leq \varepsilon$ as the stopping criterion in Algorithm 2. We now proceed to analyse the properties of

Algorithm 2: Proposed iterative algorithm

Input: NN parameters, $\{\delta_i\}_{i=0}^{L-2}$, α and ε.
Initialise: Construct $\{v_i\}_{i=0}^{L-1}$ using Algorithm 1. Solve $\gamma^*_{\text{LayerSDP}}$ from (8). Set $\delta_{L-1}^{(1)} = \gamma^*_{\text{LayerSDP}} - c_0$ and $k = 0$.
repeat
  Set $k \leftarrow k+1$. Solve $\gamma^{*(k)}_{\text{auxSDP}}$ and $\mathbf{x}_{L-1}^{(k)}$ from (10) with $\delta_{L-1}^{(k)}$.
  $\gamma_{\text{iter}}^{(k)} \leftarrow v_{L-1}^{\top}\mathbf{x}_{L-1}^{(k)} + c_0$, $\delta_{L-1}^{(k+1)} \leftarrow \delta_{L-1}^{(k)} + \sqrt{\gamma^{*(k)}_{\text{auxSDP}}}$.
until $\gamma^{*(k)}_{\text{auxSDP}} \leq \varepsilon$
Output: $\gamma_{\text{iter}}^{(k_0)}$, where k_0 is the final iteration.

the objective value sequence $\{\gamma_{\text{iter}}^{(k)}\}$. We start by analysing the relation between $\delta_{L-1}^{(k)}$ and $\gamma_{\text{iter}}^{(k)}$ as in Proposition 4.

Proposition 4. *Let $\gamma^*_{ncvxSDP} - c_0 \in [\underline{\gamma}, \overline{\gamma}]$ and $\varphi(\delta_{L-1})$ be the value of $v_{L-1}^{\mathsf{T}}\mathbf{x}_{L-1}$ under the parameter δ_{L-1}. Consider Algorithm 2, we know that:*

1) $\varphi(\delta_{L-1})$ increases with δ_{L-1} in the sense that $(\delta_{L-1}^1 - \delta_{L-1}^2)\left[\varphi(\delta_{L-1}^1) - \varphi(\delta_{L-1}^2)\right] \geq 0$, for any $\delta_{L-1}^1, \delta_{L-1}^2 \in [\underline{\gamma}, \overline{\gamma}]$, and $\delta_{L-1}^1 \neq \delta_{L-1}^2$.

2) If $\delta_{L-1}^{(k)} + c_0 < \gamma^*_{ncvxSDP}$, then $\gamma_{iter}^{(k)} \leq \gamma^*_{ncvxSDP}$.

3) If $\delta_{L-1}^{(k)} + c_0 = \gamma^*_{ncvxSDP}$, then $\gamma_{iter}^{(k)} = \gamma^*_{ncvxSDP}$.

Given the above, we can now state the *soundness* and *tightness* of Algorithm 2 in Theorem 5.

Theorem 5. *Given a feasible verification instance, the objective value sequence $\{\gamma_{iter}^{(k)}\}$ generated by Algorithm 2 satisfies $\gamma^*_{LayerSDP} \leq \gamma_{iter}^{(k)} \leq \gamma^*_{ncvxSDP}$ and converges to $\gamma^*_{ncvxSDP}$ by setting $\varepsilon = 0$.*

Proof. By using Lemma 3 and the first property of Proposition 4, we know that the sequence $\{\gamma_{\text{iter}}^{(k)}\}$ is monotonically increasing. By further using the second property of Proposition 4, we know that the sequence $\{\gamma_{\text{iter}}^{(k)}\}$ is bounded from above as $\gamma_{\text{iter}}^{(k)} \leq \gamma^*_{\text{ncvxSDP}}$. By construction and using (10c), we also know that $\{\gamma_{\text{iter}}^{(k)}\}$ is bounded from below as $\gamma_{\text{iter}}^{(k)} = v_{L-1}^{\top}\mathbf{x}_{L-1}^{(k)} + c_0 \geq \delta_{L-1}^{(k)} + c_0 \geq \gamma^*_{\text{LayerSDP}}$. Hence, we conclude that $\gamma^*_{\text{LayerSDP}} \leq \gamma_{\text{iter}}^{(k)} \leq \gamma_{\text{iter}}^{(k+1)} \leq \gamma^*_{\text{ncvxSDP}}$. The convergence proof follows directly from that of Lemma 3 by replacing $\delta_{L-1}^{(k)} + c_0$ with $\gamma_{\text{iter}}^{(k)}$. □

Theorem 5 shows that Algorithm 2 achieves an objective value sequence $\{\gamma_{\text{iter}}^{(k)}\}$ that is always a valid lower bound to $\gamma^*_{\text{ncvxSDP}}$ and γ^*. Also, the objective values at all iterations are not worse than $\gamma^*_{\text{LayerSDP}}$ and converge to $\gamma^*_{\text{ncvxSDP}}$ in finite iterations. The finite-time convergence has been empirically observed, but how to derive an analytic upper bound for the maximal iterations is still an open question. The above analysis shows that the proposed iterative algorithm is efficient to solve the non-convex layer SDP problem (9), which would otherwise be hard to solve directly.

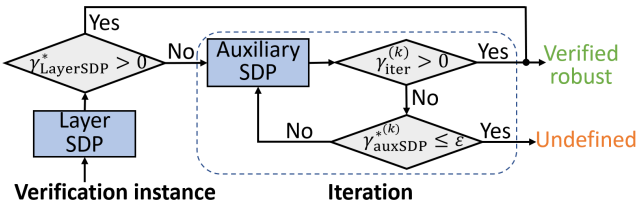

Figure 1: Flowchart of the proposed verification method.

We now show the application of Algorithm 2 to NN verification (see Figure 1). In the picture the output "Verified robust" means that the instance studied is robust against the adversarial input, while the output "Undefined" represents the situation in which robustness cannot be determined. The iteration is executed only when $\gamma^*_{\text{LayerSDP}} \leq 0$, *i.e.*, the layer SDP is unable to resolve the robustness query. As we show later, this approach reduces the computational cost, as the layer SDP approach can already verify a considerable number of instances. Also, it is clear that the use of our iterative algorithm can increase the number of instances that can be verified. In practice, the iteration is terminated whenever $\gamma^{(k)}_{\text{iter}} > 0$, even though $\gamma^{*(k)}_{\text{auxSDP}} \leq \varepsilon$ is not reached yet.

Computational Complexity. The computational cost in the method stems from solving the auxiliary SDP (10). This auxiliary SDP and the existing layer SDP (Batten et al. 2021) and global SDP (Raghunathan, Steinhardt, and Liang 2018) are all solved using the solver MOSEK (Andersen and Andersen 2000) which implements an interior-point algorithm. Before calling MOSEK, the SDP problem is reformulated as the widely-used conic optimisation form where inequality constraints are converted into equalities (Sturm 1999). As shown in (Nesterov 2003), solving a standard SDP at each iteration of the interior-point algorithm requires $\mathcal{O}(n^3m + n^2m^2 + m^3)$ time and $\mathcal{O}(n^2 + m^2)$ memory, where n is the size of the largest semidefinite constraint and m is the number of equality constraints. For the global SDP, $n = 1 + \sum_{i=0}^{L-1} n_i$ which is the sum of all layer dimensions. For the layer SDP and the auxiliary SDP (10), $n = 1 + \max_{i=0,\ldots,L-1}(n_i + n_{i+1})$ which is the largest size of P_i and it can be significantly smaller than that in the global SDP. As demonstrated in (Batten et al. 2021, Proposition 2), it is always beneficial to remove the dead neurons in any verification instance before formulating the SDP. The layer SDP and the auxiliary SDP can increase the number of equality constraints and may offset the benefits of smaller semidefinite constraints; see (Batten et al. 2021, Remark 1).

Experimental Evaluation

Settings. We conducted experiments on a Linux machine with an AMD Ryzen Threadripper 3970X 32-Core CPU, 256 GB RAM and a RTX 3090 GPU. The optimisation problems were modelled using the toolbox YALMIP (Lofberg 2004) and solved using the SDP solver MOSEK. We used $\delta_i = 1, i \in \mathbb{I}_{L-2}$, $\alpha = 1.0\text{e}5$ and $\varepsilon = 0$ to run Algorithm 2.

Networks. We evaluated on several feed-forward ReLU NNs trained on the MNIST dataset (LeCun 1998) and CIFAR10 dataset (Krizhevsky, Nair, and Hinton 2014) (where "MLP" refers to MNIST, "CF" refers to CIFAR10, and "$m \times n$" means a NN with $m-1$ hidden layers each having n neurons):

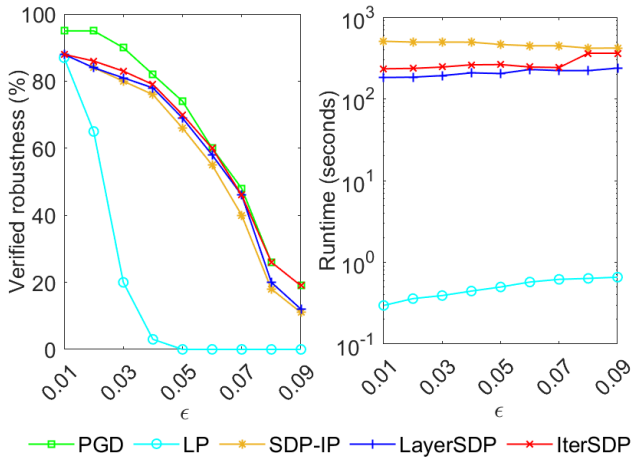

Figure 2: Verified robustness and runtime per image of different methods and perturbation ϵ for a MLP-3×50 network.

1) One MLP-3×50 network self-trained with no adversarial training, tested with perturbations ϵ from 0.01 to 0.09.
2) Three small networks MLP-Adv, MLP-LP, and MLP-SDP from (Raghunathan, Steinhardt, and Liang 2018). They are tested under the same $\epsilon = 0.1$ as in (Raghunathan, Steinhardt, and Liang 2018; Batten et al. 2021).
3) Three medium networks MLP-6×100, MLP-9×100 and MLP-6×200 from (Singh et al. 2019a). We use the same $\epsilon = 0.026, 0.026, 0.015$, respectively, as in (Batten et al. 2021; Singh et al. 2019a; Müller et al. 2021).
4) Two large networks MLP-8 × 1024-0.1 and MLP-8 × 1024-0.3 from (Li et al. 2020) that were trained using CROWN-IBP (Zhang et al. 2019) with adversarial attacks $\epsilon = 0.1, 0.3$, respectively. As in (Li et al. 2020), the test perturbations are $\epsilon = 0.1, 0.3$, respectively.
5) One CIFAR10 network CF-3 × 20 from (Li et al. 2020) and evaluated under the perturbation $\epsilon = 2/255$.

Baselines. We compared the proposed iterative method (referred to as `IterSDP`) against several SoA incomplete methods for NN robustness verification:

1) Linear relaxations: the standard linear program relaxation `LP` (Ehlers 2017) and its variants including β-`CROWN` (Wang et al. 2021), `IBP` (Gowal et al. 2019), and `PRIMA` (Müller et al. 2021).
2) SDP relaxations: `LayerSDP` (Batten et al. 2021), `SDP-IP` (*i.e.*, the global SDP relaxation (Raghunathan, Steinhardt, and Liang 2018)), and `SDP-FO` (Dathathri et al. 2020). We did not include the symbolic bound propagation methods such as `VeriNet` (Henriksen and Lomuscio 2020), because they were less effective than β-`CROWN`, as shown in the VNN-COMP 2021 (Bak, Liu, and Johnson 2021).

		IterSDP					LayerSDP		SDP-IP		SDP-FO	LP		IBP		PRIMA	β-CROWN	
Models	PGD	ver.	t	n_{iter}	t_{iter}	v_{iter}	ver.	t	ver.	t	ver.*	ver.	t	ver.	t	ver*.	ver.	t
MLP-Adv	94	**92**	2471	0.1	319	51	91	1866	82	12079	84	7	2.4	0	1.2	–	88	8
MLP-LP	80	**80**	203	0.3	37	20	**80**	145	**80**	50733	78	78	0.5	17	0.9	–	**80**	0.1
MLP-SDP	84	**84**	10612	0.2	1547	21	**84**	6643	80	43156	64	6	6.9	0	1.3	–	76	65
MLP-6×100	91	**83**	3749	0.5	276	48	75	1414	52	6760	◇	0	2.8	0	1.1	51	78	238
MLP-9×100	86	47	625	2.8	167	45	35	446	22	2148	◇	1	4.4	0	1.0	42.8	**69**	461
MLP-6×200	96	**94**	5597	0.5	4776	31	92	3180	76	25850	◇	3	6.3	0	0.9	69	84	190
MLP-8×1024-0.1	89	**86**	2509	0.1	636	53	83	1883	◇	◇	◇	0	32	10	2.2	–	–	–
MLP-8×1024-0.3	34	**33**	817	0.2	106	20	28	404	◇	◇	◇	0	38	0	2.8	–	–	–
CF-3×20	20	**18**	4264	0.7	241	15	17	3663	16	7804	◇	0	0.3	0	1.7	–	17	0.2

Table 1: Verified robustness (ver., %), runtime per image (t, seconds) and time cost breakdown of IterSDP for a set of benchmarks. n_{iter} is the average number of iterations per sample, t_{iter} is the average time per iteration (in seconds) and v_{iter} is the relative improvement of the objective value γ_{iter} per iteration (in %). The results of LayerSDP, SDP-IP, LP, IBP and β-CROWN are obtained by re-implementation on the same 100 images as our method. Star (*): The numbers are from the literature: SDP-FO from (Batten et al. 2021) for 100 images, and PRIMA from (Müller et al. 2021) for 1000 images. Dash (–): previously reported or re-implementation numbers are unavailable. Diamond (◇): the methods fail to verify any instance.

Results. All experiments were run on the first 100 images of the datasets. We received the permission to run VeriNet (Henriksen and Lomuscio 2020) for obtaining the activation bounds of all benchmarks. We report the verified robustness (percentage of images verified to be robust) and runtime (average solver time for verifying an image) for each method.

Figure 2 shows the results of MLP-3×50 under various ϵ and methods IterSDP, LP, SDP-IP and LayerSDP. Our method IterSDP outperforms the baselines across all ϵ, confirming the statement in Theorem 5. Notably, IterSDP improves the verified robustness up to the PGD bounds for several ϵ. IterSDP requires more runtime (about twice) when compared to LayerSDP, but it is still faster than SDP-IP. This is expected since Algorithm 2 uses LayerSDP to initialise and solves the auxiliary SDP (10) whose size is similar to LayerSDP.

Table 1 shows the comparative results for the standard benchmarks. The breakdown of the time cost of our method IterSDP is also reported. IterSDP is more precise than the baselines for all the networks, except MLP-LP, MLP-SDP and MLP-9×100. For MLP-LP and MLP-SDP, both IterSDP and LayerSDP reach the PGD upper bounds, while for MLP-9×100, IterSDP is only less precise than β-CROWN. Remarkably, for most the networks, IterSDP achieved the verified accuracy that is close to or same as the PGD upper bound. It is also shown in (Dathathri et al. 2020; Li et al. 2020) that the complete method MILP verifies 69% robust cases for MLP-SDP, 67% for MLP-8×1024-0.1 and 7% for MLP-8×1024-0.3, lower than 84%, 86% and 33% by our method respectively.

As expected, IterSDP needs more runtime than LayerSDP across all the networks, but is faster than SDP-IP. Neither SDP-IP nor SDP-FO could verify the large models MLP-8×1024-0.1 and MLP-8×1024-0.3. The results reported in (Batten et al. 2021) showed that compared to LayerSDP, SDP-FO has a runtime that is much larger for MLP-Adv and MLP-LP, but smaller for MLP-SDP. Also, SDP-FO fails to verify MLP-6×100, MLP-9×100 and MLP-6×200. These results confirm that our proposed IterSDP improves the verification precision, whilst retaining a competitive computational efficiency.

Conclusions

Relaxation methods that provide tight approximations are required to be able to verify large NNs used in applications. SDP-based methods have been proposed in the literature to accomplish this goal; yet, the relaxation gap in the present SoA is still considerable when applying them to deep and large networks, resulting in SoA tools not being able to establish the result for many verification queries. Here we proposed a novel SDP relaxation to narrow the relaxation gaps and enable efficient NN verification. The method improves the tightness by integrating a nonlinear constraint to the present SoA layer SDP method. Our approach is computationally effective because, unlike previous approaches, the resulting non-convex SDP problem is solved by computing the solutions of auxiliary convex layer SDP problems in an iterative manner. As shown, the method always yields tighter relaxations than the layer SDP and the solution sequence iteratively converges to the optimal solution of the non-convex SDP relaxation. The experiments showed that the method results in SoA performance on all benchmarks commonly used in the area.

Acknowledgements

Jianglin Lan is supported by a Leverhulme Trust Early Career Fellowship under Award ECF-2021-517. Yang Zheng is supported by the National Science Foundation under Grant No. ECCS-2154650. Alessio Lomuscio is supported by a Royal Academy of Engineering Chair in Emerging Technologies.

References

Andersen, E. D.; and Andersen, K. D. 2000. The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High performance optimization*, 197–232. Springer.

Anderson, R.; Huchette, J.; Ma, W.; Tjandraatmadja, C.; and Vielma, J. 2020. Strong mixed-integer programming formulations for trained neural networks. *Mathematical Programming*, 1–37.

Bak, S.; Liu, C.; and Johnson, T. 2021. The Second International Verification of Neural Networks Competition (VNN-COMP 2021): Summary and Results. *arXiv preprint arXiv:2103.06624*.

Bastani, O.; Ioannou, Y.; Lampropoulos, L.; Vytiniotis, D.; Nori, A.; and Criminisi, A. 2016. Measuring neural net robustness with constraints. In *Advances in Neural Information Processing Systems (NeurIPS16)*, 2613–2621.

Batten, B.; Kouvaros, P.; Lomuscio, A.; and Zheng, Y. 2021. Efficient Neural Network Verification via Layer-based Semidefinite Relaxations and Linear Cuts. In *International Joint Conference on Artificial Intelligence (IJCAI21)*, 2184–2190.

Botoeva, E.; Kouvaros, P.; Kronqvist, J.; Lomuscio, A.; and Misener, R. 2020. Efficient verification of neural networks via dependency analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI20)*, 3291–3299.

Chen, S.; Wong, E.; Kolter, J. Z.; and Fazlyab, M. 2021. DeepSplit: Scalable Verification of Deep Neural Networks via Operator Splitting. *arXiv preprint arXiv:2106.09117*.

Dathathri, S.; Dvijotham, K.; Kurakin, A.; Raghunathan, A.; Uesato, J.; Bunel, R.; Shankar, S.; Steinhardt, J.; Goodfellow, I.; Liang, P.; and Pushmeet, K. 2020. Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In *Advances in Neural Information Processing Systems (NeurIPS20)*, 1–14.

Dvijotham, K.; Stanforth, R.; Gowal, S.; Mann, T.; and Kohli, P. 2018. A dual approach to scalable verification of deep networks. *arXiv preprint arXiv:1803.06567*.

Ehlers, R. 2017. Formal verification of piece-wise linear feed-forward neural networks. In *International Symposium on Automated Technology for Verification and Analysis (ATVA17)*, 269–286.

Fazlyab, M.; Morari, M.; and Pappas, G. J. 2022. Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. *IEEE Transactions on Automatatic Control*, 67(1): 1–15.

Goodfellow, I.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Gowal, S.; Dvijotham, K. D.; Stanforth, R.; Bunel, R.; Qin, C.; Uesato, J.; Arandjelovic, R.; Mann, T.; and Kohli, P. 2019. Scalable verified training for provably robust image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (IEEE/CVF19)*, 4842–4851.

Hashemi, V.; Kouvaros, P.; and Lomuscio, A. 2021. OSIP: Tightened Bound Propagation for the Verification of ReLU Neural Networks. In *International Conference on Software Engineering and Formal Methods (SEFM21)*, 463–480. Springer.

Henriksen, P.; and Lomuscio, A. 2020. Efficient neural network verification via adaptive refinement and adversarial search. In *Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI20)*, 2513–2520. IOS Press.

Henriksen, P.; and Lomuscio, A. 2021. DEEPSPLIT: An Efficient Splitting Method for Neural Network Verification via Indirect Effect Analysis. In *International Joint Conference on Artificial Intelligence (IJCAI21)*, 2549–2555.

Horn, R. A.; and Johnson, C. R. 2012. *Matrix analysis*. Cambridge University Press.

Julian, K. D.; and Kochenderfer, M. J. 2021. Reachability Analysis for Neural Network Aircraft Collision Avoidance Systems. *Journal of Guidance, Control, and Dynamics*, 44(6): 1132–1142.

Katz, G.; Barrett, C.; Dill, D. L.; Julian, K.; and Kochenderfer, M. J. 2021. Reluplex: a calculus for reasoning about deep neural networks. *Formal Methods in System Design*, 1–30.

Kouvaros, P.; Kyono, T.; Leofante, F.; Lomuscio, A.; Margineantu, D.; Osipychev, D.; and Zheng, Y. 2021. Formal Analysis of Neural Network-Based Systems in the Aircraft Domain. In *International Symposium on Formal Methods (FM21)*, 730–740. Springer.

Krizhevsky, A.; Nair, V.; and Hinton, G. 2014. The CIFAR-10 dataset. http://www.cs.toronto.edu/kriz/cifar.html. Accessed: 2021-06-23.

Lan, J.; Zheng, Y.; and Lomuscio, A. 2022. Tight neural network verification via semidefinite relaxations and linear reformulations. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7): 7272–7280.

Lasserre, J. B. 2009. *Moments, positive polynomials and their applications*, volume 1. World Scientific.

LeCun, Y. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Li, L.; Qi, X.; Xie, T.; and Li, B. 2020. SoK: Certified robustness for deep neural networks. *arXiv preprint arXiv:2009.04131*.

Liu, C.; Arnon, T.; Lazarus, C.; Strong, C.; Barrett, C.; Kochenderfer, M. J.; et al. 2020. Algorithms for Verifying Deep Neural Networks. *Foundations and Trends® in Optimization*, 3-4: 244–404.

Lofberg, J. 2004. YALMIP: A toolbox for modeling and optimization in MATLAB. In *IEEE International Conference on Robotics and Automation (ICRA04)*, 284–289. IEEE.

Lomuscio, A.; and Maganti, L. 2017. An approach to reachability analysis for feed-forward ReLU neural networks. *CoRR*, abs/1706.07351.

Luo, H.; Bai, X.; and Peng, J. 2019. Enhancing semidefinite relaxation for quadratically constrained quadratic programming via penalty methods. *Journal of Optimization Theory and Applications*, 180(3): 964–992.

Ma, Z.; and Sojoudi, S. 2020. Strengthened SDP verification of neural network robustness via non-convex cuts. *arXiv preprint arXiv:2010.08603*.

Manzanas Lopez, D.; Johnson, T.; Tran, H.-D.; Bak, S.; Chen, X.; and Hobbs, K. L. 2021. Verification of Neural Network Compression of ACAS Xu Lookup Tables with Star Set Reachability. In *AIAA Scitech 2021 Forum*, 0995.

Müller, M. N.; Makarchuk, G.; Singh, G.; Püschel, M.; and Vechev, M. 2021. PRIMA: Precise and general neural network certification via multi-neuron convex relaxations. *arXiv preprint arXiv:2103.03638*.

Nesterov, Y. 2003. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media.

Newton, M.; and Papachristodoulou, A. 2021. Exploiting Sparsity for Neural Network Verification. In *Learning for Dynamics and Control*, 715–727. PMLR.

Parrilo, P. A. 2000. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. California Institute of Technology.

Raghunathan, A.; Steinhardt, J.; and Liang, P. 2018. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems (NeurIPS18)*, 10877–10887.

Salman, H.; Yang, G.; Zhang, H.; Hsieh, C.; and Zhang, P. 2019. A convex relaxation barrier to tight robustness verification of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS19)*, 9835–9846.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019a. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 41:1–41:30.

Singh, G.; R. Ganvir, R.; Püschel, M.; and Vechev, M. 2019b. Beyond the single neuron convex barrier for neural network certification. In *Advances in Neural Information Processing Systems (NeurIPS19)*, 15098–15109.

Sturm, J. F. 1999. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4): 625–653.

Tjandraatmadja, C.; Anderson, R.; Huchette, J.; Ma, W.; Patel, K.; and Vielma, J. P. 2020. The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In *Advances in Neural Information Processing Systems (NeurIPS20)*, 1–12.

Tjeng, V.; Xiao, K.; and Tedrake, R. 2019. Evaluating robustness of neural networks with mixed integer programming. In *International Conference on Learning Representations (ICLR19)*, 1–21.

Tran, H.-D.; Yang, X.; Lopez, D. M.; Musau, P.; Nguyen, L. V.; Xiang, W.; Bak, S.; and Johnson, T. T. 2020. NNV: The neural network verification tool for deep neural networks and learning-enabled cyber-physical systems. In *International Conference on Computer Aided Verification*, 3–17. Springer.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018a. Efficient formal safety analysis of neural networks. *Advances in Neural Information Processing Systems*, 31.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018b. Efficient formal safety analysis of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS18)*, 6367–6377.

Wang, S.; Pei, K.; Whitehouse, J.; Yang, J.; and Jana, S. 2018c. Formal security analysis of neural networks using symbolic intervals. In *27th USENIX Security Symposium (USENIX Security 18)*, 1599–1614.

Wang, S.; Zhang, H.; Xu, K.; Lin, X.; Jana, S.; Hsieh, C.-J.; and Kolter, J. Z. 2021. Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. *arXiv preprint arXiv:2103.06624*.

Weng, T.; Zhang, H.; Chen, H.; Song, Z.; Hsieh, C.; Boning, D.; Dhillon, I.; and Daniel, L. 2018. Towards fast computation of certified robustness for ReLU networks. In *International Conference on Machine Learning (ICML18)*, 5276–5285.

Wong, E.; and Kolter, Z. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning (ICML18)*, 5286–5295.

Xu, K.; Zhang, H.; Wang, S.; Wang, Y.; Jana, S.; Lin, X.; and Hsieh, C.-J. 2020. Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers. *arXiv preprint arXiv:2011.13824*.

Zhang, H.; Chen, H.; Xiao, C.; Gowal, S.; Stanforth, R.; Li, B.; Boning, D.; and Hsieh, C.-J. 2019. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*.

Zhang, R. Y. 2020. On the tightness of semidefinite relaxations for certifying robustness to adversarial examples. *arXiv preprint arXiv:2006.06759*.

Zheng, Y.; Fantuzzi, G.; and Papachristodoulou, A. 2021. Chordal and factor-width decompositions for scalable semidefinite and polynomial optimization. *Annual Reviews in Control*, 52: 243–279.