Tight Neural Network Verification via Semidefinite Relaxations and Linear Reformulations*

Jianglin Lan[1][0000−0001−9057−5649], Yang Zheng[2][0000−0003−1545−231X], and Alessio Lomuscio[3][0000−0003−3420−723X]

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

Abstract. We present a novel semidefinite programming (SDP) relaxation that enables tight and efficient verification of neural networks. The tightness is achieved by combining SDP relaxations with valid linear cuts, constructed by using the reformulation-linearisation technique (RLT). The computational efficiency results from a layerwise SDP formulation and an iterative algorithm for incrementally adding RLT-generated linear cuts to the verification formulation. The layer RLT-SDP relaxation here presented is shown to produce the tightest SDP relaxation for ReLU neural networks available in the literature. We report experimental results based on MNIST neural networks showing that the method outperforms the state-of-the-art methods while maintaining acceptable computational overheads. For networks of approximately 10k nodes (1k, respectively), the proposed method achieved an improvement in the ratio of certified robustness cases from 0% to 82% (from 35% to 70%, respectively).

Keywords: Neural network · Verification · Semidefinite programming.

1 Introduction

While progress in training methods for neural networks (NNs) continues, it is well-known that NNs are susceptible to adversarial attacks [16]. This is highly problematic for uses of NNs in safety-critical systems such as the aircraft domain [1, 2, 21, 23, 30] or in any application where miss-classifications need to be minimised. The area of verification of NNs [6, 26] aims to develop methods to guarantee that NNs are robust with respect to small perturbations, with particular emphasis to noise perturbations.

* A shorter version of this article appeared in the Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI22). Virtual conference. AAAI Press.

Existing NN verification methods can be divided into two categories [26]: *complete* and *incomplete* approaches. Complete approaches guarantee to resolve any verification query, but may incur high computational cost. Incomplete approaches normally leverage forms of convex over-approximations of NNs to enable faster verification. While incomplete approaches tend to scale to larger networks, the looser their approximation is, the more likely it is that the approach may be unable to verify the problem instance. Thus, one central objective in incomplete approaches is to develop tighter convex approximations while retaining computation efficiency [34].

In this paper we provide a novel relaxation method which combines semidefinite programming (SDP) [40] with the reformulation-linearisation technique (RLT) [5] to verify NNs. The new SDP relaxation is provably tighter than existing SDP approaches, whilst enjoying a competitive efficiency.

Related Work. Complete approaches to the verification of NNs are typically based on mixed-integer linear programming (MILP) [4, 7, 9, 28, 38], satisfiability modulo theories [12, 22] or bound propagations combined with input refinement [18–20, 42]. While these approaches can provide theoretical termination guarantees, their scalability to large NNs is often problematic. Incomplete approaches for NN verification are normally based on bound propagations [31, 35, 37, 43], combinations between linear programming (LP) and relaxations [12], or duality relaxation [10, 11, 44]. The triangle relaxation [12] gives the tightest convex approximation of a single ReLU node and has inspired several other approaches [25,34]. While these methods often achieve state-of-the-art (SoA) performance, they have limited efficacy: even optimally tuned LP-based convex relaxations may fail to obtain tight bounds on the certified robustness ratio [34].

Two lines of research have attempted to alleviate this problem. The first aims to provide tighter LP relaxations by exploring interdependencies among multiple neurons and inputs, *e.g.*, `DeepPoly` [35], `kPoly` [36], `OptC2V` [37], and `PRIMA` [31]. The second seeks alternative, stronger relaxations beyond LPs. The most promising relaxation combining tightness with efficiency is presently based on SDPs [8, 13, 33].

It has been empirically observed that the relaxations generated with the SDP method in [33] are considerably tighter than standard LP relaxations. Using geometric techniques, it has been shown that the SDP relaxation for a variant of the NN verification problem is exact over a single hidden layer under mild assumptions, but becomes loose for several hidden layers [46]. To obtain tighter SDP relaxations, effective linear cuts were identified in [8]; non-convex cuts were investigated in [29].

SDPs are harder to solve than LPs. To overcome this, a dual SDP relaxation was formulated and subsequently solved using a subgradient algorithm in [10]. A layer SDP relaxation has been recently proposed in [8] by exploiting cascading network structures based on graph decomposition [48]. To the best of our knowledge, layer SDP provides the tightest relaxations that have so far been achieved by combining SDP relaxations with triangle relaxations [12]. Even so,

the relaxation gap is still considerable in large NNs, as observed in [8]. This leads to an increasing number of verification queries that cannot be resolved as the model size increases, thereby limiting the applicability of the approach.

Contributions. In this paper, we advance the SoA SDP relaxations for NN verification by using the RLT [5] in this context. Specifically, we propose a new layer RLT-SDP relaxation with valid linear cuts obtained from RLT that offers provably tighter relaxations. The linear cuts capture both *inter-layer dependencies* and *intra-layer interactions* of the network, which are presently not exploited in the existing relaxation methods. Due to the computational cost of using large numbers of linear cuts, we refine this method by introducing an iterative algorithm to integrate the RLT-generated linear cuts and the SDP relaxation. At each iteration only a portion of linear cuts are added, with their priorities being determined by the network weights. Our theoretical analysis shows that the relaxations here obtained are provably tighter than any other approach previously considered. In the experiments we report, the method obtained considerably tighter relaxations than the present SoA leading to several more queries being answerable.

2 Problem Statement and Preliminaries

Notation. We use the symbol $\mathbb{R}^n$ to denote the n-dimensional Euclidean space. We use $\text{diag}(X)$ to stack the main diagonals of matrix X as a column, and $|X|$ to get its absolute value element-wise. We use $\odot$ to denote the element-wise product, and $\mathbb{I}_b$ with $b > 0$ to denote a sequence of nonzero integers from 0 to b. We use $\|\cdot\|_\infty$ to refer to the standard ℓ_∞ norm.

NN Verification Problem. We focus on feed-forward fully-connected NNs with ReLU activations. A NN $f(x_0) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$ with L hidden layers, n_0 inputs and n_{L+1} outputs is defined as follows: x_0 is the input, $f(x_0)$ is the output, and $\hat{x}_i, x_i \in \mathbb{R}^{n_i}$, $i = 1, 2, \ldots, L$, are the pre-activation and activation vectors of the i-th layer, respectively. The NN output is $f(x_0) = W_L x_L + b_L$ 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 $W_i \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b_i \in \mathbb{R}^{n_{i+1}}$ are the weights and biases, respectively. For a vector $z \in \mathbb{R}^n$, the ReLU function is defined as $\text{ReLU}(z) = [\max(z_1, 0), \ \cdots, \ \max(z_n, 0)]^\mathsf{T}$. We focus on classification networks whereby an input x_0 is assigned to the class associated with the NN output with the highest value: $j^\star = \arg\max_{j=1,2,\cdots,n_{L+1}} f(x_0)_j$.

We now present the local robustness verification problem.

Definition 1. *Given a NN $f : \mathbb{R}^{n_0} \to \mathbb{R}^{n_{L+1}}$, an input $\bar{x}$ and a perturbation radius $\epsilon \in \mathbb{R}$, f is robust on $\bar{x}$ if $f(\bar{x})_{j^\star} - f(x_0)_j > 0$ when $j \neq j^\star$, for all x_0 s.t. $\|x_0 - \bar{x}\|_\infty \leq \epsilon$.*

Given a NN f, an input $\bar{x}$ and a perturbation ϵ, the local robustness problem concerns determining whether it is the case that f meets Definition 1 for $\bar{x}$ and

ϵ. This problem can be formulated and solved as an optimisation problem:

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

where $c^\mathsf{T} = W_L(j^\star, :) - W_L(j, :)$ and $c_0 = b_L(j^\star) - b_L(j)$. l_{i+1} and u_{i+1} are the lower and upper bounds of the activation vectors, which can be computed using bound propagation methods, see *e.g.*, [41, 44]. The optimisation problem (1) is solved for every potential adversarial target $j \neq j^\star$. If $\gamma^* > 0$ in all cases, then the network is robust on $\bar{x}$ against the adversarial input x_0.

Due to the nonlinear ReLU constraint (1a), the problem (1) is non-convex and thus hard to solve generally. In the literature, two SDP relaxations have been proposed: a global SDP relaxation [33] and a layer SDP relaxation [8].

Global SDP Relaxation. The ReLU constraint (1a) is equivalent to a set of linear and quadratic constraints:

$$\begin{aligned}
& x_{i+1} \geq 0,\ x_{i+1} \geq W_i x_i + b_i,\ i \in \mathbb{I}_{L-1}, \\
& x_{i+1} \odot (x_{i+1} - W_i x_i - b_i) = 0,\ i \in \mathbb{I}_{L-1}.
\end{aligned} \tag{2}$$

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

$$x_i \odot x_i - (l_i + u_i) \odot x_i + l_i \odot u_i \leq 0,\ i \in \mathbb{I}_L, \tag{3}$$

where $l_0 = \bar{x} - \epsilon\mathbf{1}$, $u_0 = \bar{x} + \epsilon\mathbf{1}$ and $\mathbf{1}$ denotes column of ones. Replacing (1a)-(1c) with (2)-(3) yields an equivalent quadratically constrained quadratic programming (QCQP):

$$\gamma^* := \left\{ \min_{\{x_i\}_{i=0}^L} \quad c^\mathsf{T} x_L + c_0 \mid (2), (3) \right\}. \tag{4}$$

This QCQP problem is still non-convex due to the quadratic constraints. The techniques of polynomial lifting [24,32] can be used to reformulate them as linear constraints. The lifting matrix P is defined as

$$P = \mathbf{x}\mathbf{x}^\mathsf{T},\ \text{with}\ \mathbf{x} = [1,\ x_0^\mathsf{T},\ x_1^\mathsf{T},\ \cdots,\ x_L^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{\bar{n}}, \tag{5}$$

where $\bar{n} = 1 + \sum_{i=0}^L n_i$. The above can be reformulated as:

$$P \succeq 0,\ P[1] = 1,\ \mathrm{rank}(P) = 1. \tag{6}$$

By using (5) and (6) and dropping $\text{rank}(P) = 1$, the QCQP (4) is relaxed into a global SDP [33]:

$$\begin{aligned}
&\gamma^*_{\text{GlobalSDP}} := \min_{P} \ c^\mathsf{T} P[x_L] + c_0 \\
\text{s.t. } & P[x_{i+1}] \geq 0, \ P[x_{i+1}] \geq W_i P[x_i] + b_i, i \in \mathbb{I}_{L-1}, && \text{(7a)} \\
& \text{diag}(P[x_{i+1}x_{i+1}^\mathsf{T}] - W_i P[x_i x_{i+1}^\mathsf{T}]) - b_i \odot P[x_{i+1}] = 0, \ i \in \mathbb{I}_{L-1}, && \text{(7b)} \\
& \text{diag}(P[x_i x_i^\mathsf{T}]) - (l_i + u_i) \odot P[x_i] + l_i \odot u_i \leq 0, \ i \in \mathbb{I}_L, && \text{(7c)} \\
& P[1] = 1, P \succeq 0, && \text{(7d)}
\end{aligned}$$

where the symbolic indexing $P[\cdot]$ is used to index the elements of P. Since $\text{rank}(P) = 1$ is dropped, problem (7) gives a relaxed solution to the QCQP, *i.e.*, $\gamma^*_{\text{GlobalSDP}} \leq \gamma^*$.

Layer SDP Relaxation. The layer SDP relaxation exploits the deep structure of the layers of a NN, where the activation vector of a layer depends only on its immediate preceding layer. This structure is exploited in [8] to develop a layer-based SDP formulation of (7), where the dimension of the matrix constraint is reduced, thus improving the computational efficiency. In this work, instead of using a single large lifting matrix P for the entire network, each hidden layer i is assigned a smaller matrix P_i defined as

$$P_i = \mathbf{x}_i \mathbf{x}_i^\mathsf{T}, \text{ with } \mathbf{x}_i = [1, \ x_i^\mathsf{T}, \ x_{i+1}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{\bar{n}_i}, \tag{8}$$

where $\bar{n}_i = 1 + n_i + n_{i+1}$. Similar to (6), the constraint $P_i = \mathbf{x}_i \mathbf{x}_i^\mathsf{T}$ is equivalent to $P_i \succeq 0, \ P_i[1] = 1, \ \text{rank}(P_i) = 1$. By using (8), the layer SDP relaxation is formulated as

$$\begin{aligned}
&\gamma^*_{\text{LayerSDP}} := \min_{\{P_i\}_{i=0}^{L-1}} \ c^\mathsf{T} P_{L-1}[x_L] + c_0 \\
\text{s.t. } & P_i[x_{i+1}] \geq 0, \ i \in \mathbb{I}_{L-1}, && \text{(9a)} \\
& P_i[x_{i+1}] \geq W_i P_i[x_i] + b_i, \ i \in \mathbb{I}_{L-1}, && \text{(9b)} \\
& \text{diag}(P_i[x_{i+1}x_{i+1}^\mathsf{T}] - W_i P_i[x_i x_{i+1}^\mathsf{T}]) - b_i \odot P_i[x_{i+1}] = 0, \ i \in \mathbb{I}_{L-1}, && \text{(9c)} \\
& \text{diag}(P_i[x_i x_i^\mathsf{T}]) - (l_i + u_i) \odot P_i[x_i] + l_i \odot u_i \leq 0, \ i \in \mathbb{I}_{L-1}, && \text{(9d)} \\
& \text{diag}(P_{L-1}[x_L x_L^\mathsf{T}]) - (l_L + u_L) \odot P_{L-1}[x_L] + l_L \odot u_L \leq 0, && \text{(9e)} \\
& P_i[1] = 1, \ P_i \succeq 0, \ i \in \mathbb{I}_{L-1}, && \text{(9f)} \\
& P_i[\bar{x}_{i+1}\bar{x}_{i+1}^\mathsf{T}] = P_{i+1}[\bar{x}_{i+1}\bar{x}_{i+1}^\mathsf{T}], \ i \in \mathbb{I}_{L-2}, && \text{(9g)} \\
& P_i[x_{i+1}] \leq A_i P_i[x_i] + B_i, \ i \in \mathbb{I}_{L-1}, && \text{(9h)}
\end{aligned}$$

where $\bar{x}_{i+1} = [1, \ x_{i+1}^\mathsf{T}]^\mathsf{T}$, $A_i = k_i \odot W_i$, $B_i = k_i \odot (b_i - \hat{l}_{i+1}) + \text{ReLU}(\hat{l}_{i+1})$, $k_i = (\text{ReLU}(\hat{u}_{i+1}) - \text{ReLU}(\hat{l}_{i+1}))/(\hat{u}_{i+1} - \hat{l}_{i+1})$, with $\hat{u}_{i+1}, \hat{l}_{i+1}$ being upper and lower bounds of the pre-activation vector $\hat{x}_{i+1}$ [41, 44]. Note that the constraint (9g) appears to ensure input-output consistency between layers. Compared to (7), the new constraint (9h) is obtained from the triangle relaxation. It is shown in [8] that without (9h), the layer SDP relaxation (9) is equivalent to

the global SDP relaxation (7) based on graph decomposition [39, 47]. Also, this layer SDP relaxation achieves faster verification than the global SDP relaxation by dealing with lower dimensional constraints, and obtains a tighter relaxation by introducing (9h), *i.e.*, $\gamma^*_{\text{GlobalSDP}} \leq \gamma^*_{\text{LayerSDP}} \leq \gamma^*$.

Source of SDP Relaxation Gap. Let $\tilde{x} = [x_0^\mathsf{T},\ \cdots,\ x_L^\mathsf{T}]^\mathsf{T}$ and $P[\tilde{x}\tilde{x}^\mathsf{T}] = \tilde{x}\tilde{x}^\mathsf{T}$. It follows from (6) that $P = \mathbf{x}\mathbf{x}^\mathsf{T}$ is equivalent to $P[1] = 1,\ P[\tilde{x}\tilde{x}^\mathsf{T}] = \tilde{x}\tilde{x}^\mathsf{T}$. Observe that the global SDP relaxation (7) only includes (Eq. (7d)) the relaxed constraints $P[1] = 1$ and $P[\tilde{x}\tilde{x}^\mathsf{T}] \succeq \tilde{x}\tilde{x}^\mathsf{T}$, reformulated as $P \succeq 0$ via Schur complement. A consequence of this is that P may not be sufficiently bounded, thereby resulting in loose SDP solutions. The same argument applies for the layer SDP relaxation (9).

To bridge the gap of the global SDP relaxation, the non-convex constraint $P[\tilde{x}\tilde{x}^\mathsf{T}] \preceq \tilde{x}\tilde{x}^\mathsf{T}$ is imposed in [29] via secant approximation. Valid linear cuts are generated by an iterative algorithm, where the global SDP relaxation together with an LP need to be solved at each iteration. Unfortunately, it is known that global SDP relaxation itself is already computationally expensive. Therefore, the tightening in [29] is unlikely to lead to scalable NN verification.

3 Tightening Layer SDP Relaxation via RLT

Having highlighted the existing relaxation gap in the SoA, we now present an approach for tightening the SDP relaxation for NN verification while retaining an acceptable computational overhead. Our method combines layer SDP relaxations (9) with the RLT [5].

Motivation. We first denote a few terms for each layer of a NN: $\tilde{x}_{i+1} = [x_i^\mathsf{T},\ x_{i+1}^\mathsf{T}]^\mathsf{T}$, $\tilde{l}_{i+1} = [l_i^\mathsf{T},\ l_{i+1}^\mathsf{T}]^\mathsf{T}$ and $\tilde{u}_{i+1} = [u_i^\mathsf{T},\ u_{i+1}^\mathsf{T}]^\mathsf{T}$. Since $0 \leq \tilde{l}_i \leq \tilde{x}_{i+1} \leq \tilde{u}_i$, we have $\tilde{x}_{i+1}\tilde{l}_{i+1}^\mathsf{T} \leq \tilde{x}_{i+1}\tilde{x}_{i+1}^\mathsf{T} \leq \tilde{x}_{i+1}\tilde{u}_{i+1}^\mathsf{T}$. These nonlinear constraints can be reformulated as linear constraints on the elements of P_i:

$$P_i[\tilde{x}_{i+1}]\tilde{l}_{i+1}^\mathsf{T} \leq P_i[\tilde{x}_{i+1}\tilde{x}_{i+1}^\mathsf{T}] \leq P_i[\tilde{x}_{i+1}]\tilde{u}_{i+1}^\mathsf{T}. \tag{10}$$

The method aims to bound $P_i[\tilde{x}_{i+1}\tilde{x}_{i+1}^\mathsf{T}]$ within the region given in (10). The constraints in (10) are linear and could be directly added to (9). However, they introduce $2(n_i + n_{i+1})^2$ new inequalities, thereby increasing the computational effort required to solve the verification problem. Therefore, it is desirable to develop efficient strategies for imposing the constraints in (10). In the following we: (i) use RLT to construct valid linear cuts that are provably stronger than (10), and (ii) provide a computationally-efficient strategy for integrating the linear cuts with the layer SDP relaxation (9).

Construction of Valid Linear Cuts Using RLT. RLT involves the construction of valid linear cuts on the lifting matrices $\{P_i\}_{i=0}^{L-1}$ by using products of the existing linear constraints in (9) on the original variables $\{x_i\}_{i=0}^{L}$. Under the constraints (9a) and (9d), the variables x_i and x_{i+1} satisfy: $x_i \geq 0$, $x_i - l_i \geq 0$, $x_i - u_i \leq 0$, $x_{i+1} - l_{i+1} \geq 0$, $x_{i+1} - u_{i+1} \leq 0$. These can be used to construct the constraints: $x_i l_i^\mathsf{T} \leq x_i x_i^\mathsf{T} \leq x_i u_i^\mathsf{T}$, $(x_{i+1} - l_{i+1})(x_i - l_i)^\mathsf{T} \geq 0$,

$(x_{i+1} - l_{i+1})(x_i - u_i)^\mathsf{T} \leq 0$, $(x_i - l_i)(x_{i+1} - u_{i+1})^\mathsf{T} \leq 0$. By using (8), these nonlinear constraints are linearised as

$$P_i[x_i]l_i^\mathsf{T} \leq P_i[x_i x_i^\mathsf{T}] \leq P_i[x_i]u_i^\mathsf{T}, \tag{11a}$$
$$P_i[x_{i+1}x_i^\mathsf{T}] \geq P_i[x_{i+1}]l_i^\mathsf{T} + l_{i+1}(P_i[x_i^\mathsf{T}] - l_i^\mathsf{T}), \tag{11b}$$
$$P_i[x_{i+1}x_i^\mathsf{T}] \leq P_i[x_{i+1}]u_i^\mathsf{T} + l_{i+1}(P_i[x_i^\mathsf{T}] - u_i^\mathsf{T}), \tag{11c}$$
$$P_i[x_i x_{i+1}^\mathsf{T}] \leq P_i[x_i]u_{i+1}^\mathsf{T} + l_i(P_i[x_{i+1}^\mathsf{T}] - u_{i+1}^\mathsf{T}). \tag{11d}$$

We now make the following remarks.

Observation 1 *The linear cuts (11a) - (11d) imply (10).*

Observation 2 *The existing constraints (9d) and (9f) are stronger than the first part of (11a); while (9d) is stronger than the diagonal components of the second part of (11a).*

The proof of Observations 1 and 2 are given in the Appendix. These observations show that the targeted bounding in (10) can be realised by adding to the layer SDP relaxation (9) the following linear cuts for each P_i, $i \in \mathbb{I}_{L-1}$:

$$P_i[x_i x_i^\mathsf{T}] \leq P_i[x_i]u_i^\mathsf{T}, \tag{12a}$$
$$P_i[x_{i+1}x_i^\mathsf{T}] \geq P_i[x_{i+1}]l_i^\mathsf{T} + l_{i+1}(P_i[x_i^\mathsf{T}] - l_i^\mathsf{T}), \tag{12b}$$
$$\begin{aligned} P_i[x_{i+1}x_i^\mathsf{T}] \leq \min\{&P_i[x_{i+1}]u_i^\mathsf{T} + l_{i+1}(P_i[x_i^\mathsf{T}] - u_i^\mathsf{T}), \\ &u_{i+1}P_i[x_i^\mathsf{T}] + (P_i[x_{i+1}] - u_{i+1})l_i^\mathsf{T}\}, \end{aligned} \tag{12c}$$

where the diagonal components of (12a) are redundant.

It has been shown above that adding the linear cuts in (12) to the layer SDP relaxation (9) is efficient to bound $P_i[\tilde{x}_{i+1}\tilde{x}_{i+1}^\mathsf{T}]$ and subsequently the matrix P_i. Problem (9) also has other existing linear constraints (9a), (9b) and (9h) that can be used to construct the new constraints:

$$(x_{i+1} - W_i x_i - b_i)\,(x_{i+1} - W_i x_i - b_i)^\mathsf{T} \geq 0, \tag{13a}$$
$$(x_{i+1} - A_i x_i - B_i)\,x_{i+1}^\mathsf{T} \leq 0, \tag{13b}$$
$$(x_{i+1} - A_i x_i - B_i)\,(x_{i+1} - A_i x_i - B_i)^\mathsf{T} \geq 0. \tag{13c}$$

Observation 3 *Linear cut (13a) is weaker than (9f); while (13c) is weaker than the conjunction of (9a) - (9c) and (9h).*

Observation 4 *Adding the linear cut (13b) can tighten the layer SDP relaxation, but only its off-diagonals cut the feasible region, while the diagonals are implied by (9c).*

The proof of Observations 3 and 4 are given in the Appendix. These observations reveal that the constraint (13b) has not been included in the layer SDP relaxation (9) and it can narrow the relaxation gap. By defining $P_i[x_{i+1}x_{i+1}^\mathsf{T}] = x_{i+1}x_{i+1}^\mathsf{T}$

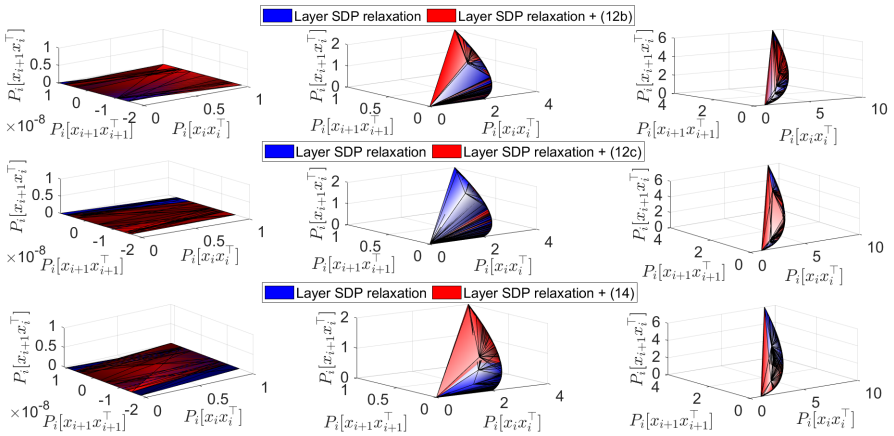

Fig. 1: Feasible region of the triple $(P_i[x_i x_i^\mathsf{T}], P_i[x_{i+1} x_{i+1}^\mathsf{T}], P_i[x_{i+1} x_i^\mathsf{T}])$ by adding linear cuts (12b), (12c) or (14), with $x_{i+1} = \mathrm{ReLU}(x_i - 1)$ and $\hat{l}_{i+1} \le x_i - 1 \le \hat{u}_{i+1}$. Left to right columns: 1) *inactive* neuron $\hat{l}_{i+1} = -1$, $\hat{u}_{i+1} = 0$; 2) *unstable* neuron $\hat{l}_{i+1} = -1$, $\hat{u}_{i+1} = 1$; 3) *strictly active* neuron $\hat{l}_{i+1} = 0$, $\hat{u}_{i+1} = 2$. For all the cases, adding each linear cut removes a portion of the relaxation region.

and $P_i[x_i x_{i+1}^\mathsf{T}] = x_i x_{i+1}^\mathsf{T}$ and recalling that $P_i[x_{i+1}] = P_{i+1}[x_{i+1}]$ under (9g), the constraints (12a) and (13b) are merged as a linear cut for each P_i, $i \in \mathbb{I}_{L-1}$:

$$P_i[x_{i+1} x_{i+1}^\mathsf{T}] \le \min\{P_i[x_{i+1}] u_{i+1}^\mathsf{T},\ A_i P_i[x_i x_{i+1}^\mathsf{T}] + B_i P_i[x_{i+1}^\mathsf{T}]\}. \tag{14}$$

When $i = 0$, $P_1[x_0 x_0^\mathsf{T}] \le P_1[x_0] u_0^\mathsf{T}$ is also needed.

Integration of Linear Cuts with Layer SDP Relaxation. The above analysis identifies the valid linear cuts (12b), (12c) and (14) for each matrix P_i, $i \in \mathbb{I}_{L-1}$. Adding them to (9) yields the layer RLT-SDP relaxation:

$$\begin{aligned} \gamma^*_{\text{RLT-SDP}} := \min_{\{P_i\}_{i=0}^{L-1}}\ & c^\mathsf{T} P_{L-1}[x_L] + c_0 \\ \text{s.t.}\ \ & (9a) - (9h), (12b), (12c), (14). \end{aligned} \tag{15}$$

Simple numerical examples in Fig. 1 show that adding each of linear cuts (12b), (12c) and (14) shrinks the relaxation region of $(P_i[x_i x_i^\mathsf{T}], P_i[x_{i+1} x_{i+1}^\mathsf{T}], P_i[x_{i+1} x_i^\mathsf{T}])$ and thus tightens the layer SDP relaxation. It follows that the layer RLT-SDP relaxation (15) offers a tighter bound than the layer SDP relaxation (9), or formally:

Theorem 1. $\gamma^*_{GlobalSDP} \le \gamma^*_{LayerSDP} \le \gamma^*_{RLT\text{-}SDP} \le \gamma^*$.

Proof. Following the principle of RLT, the added linear cuts (12b), (12c) and (14) can always be deduced from the original linear constraints in the layer SDP relaxation (9). Hence, the optimal objective value of the layer RLT-SDP

Algorithm 1 Implementation of layer RLT-SDP relaxation

1: **Input:** NN parameters, $\{p_s\}_{s=1}^r$, $k_{\max}$.
2: **Initialise:** Ordering the linear cuts via matrices $\mathcal{O}_i$, $i \in \mathbb{I}_{L-1}$. Set $k = 1$.
3: **while** $\gamma^*_{\text{RLT-SDP}} < 0$ and $k \leq k_{\max}$ **do**
4: Set $\bar{p}_i$ as the integer part of the product $p_k n_i$, $i \in \mathbb{I}_{L-1}$.
5: Solve (15), where for each P_i, $i \in \mathbb{I}_{L-1}$, adding only $\bar{p}_i$ elements (with corresponding indexes in $\mathcal{O}_i$) of each row in (12b) and (12c).
6: Set $k = k + 1$.
7: **end while**
8: **Output:** $\gamma^*_{\text{RLT-SDP}}$

relaxation (15) still serves as a lower bound to that of the QCQP and the original verification problem (1). Moreover, adding these valid linear cuts can shrink the feasible region, as shown in Observations 1 and 4. This means that every solution to (15) is feasible to (9). Therefore, the layer RLT-SDP relaxation is at least as tight as the original layer SDP relaxation. □

Efficient Implementation. The number of linear inequalities introduced by (12b), (12c) and (14) for each P_i, $i \in \mathbb{I}_{L-1}$, are $n_i n_{i+1}$, $2n_i n_{i+1}$ and $2n_{i+1}(n_{i+1}-1)$ (by removing diagonals), respectively. For P_0, $n_0(n_0-1)$ extra linear inequalities are needed. The total number of inequalities for each P_i, $i = 1, 2, \ldots, L-1$, is $(2n_{i+1}^2 + 3n_i n_{i+1} - 2n_{i+1})$, and for P_0 is $(2n_1^2 + 3n_0 n_1 - 2n_1 + n_0^2 - n_0)$. Compared to directly imposing the constraints in (10) (which introduces $2(n_i+n_{i+1})^2$ inequalities), adding (12b), (12c) and (14) has a lower computational burden, especially for large NNs. However, adding all the inequalities in (12b), (12c) and (14) is still computationally expensive. An efficient strategy for integrating them with the layer SDP relaxation is thus necessary. The strategy we deploy is based on two observations:

- The linear cuts (12b) and (12c) capture *inter-layer dependencies* (*i.e.*, terms $x_{i+1}x_i^\mathsf{T}$). Since $x_{i+1} = \text{ReLU}(W_i x_i + b_i)$, the dependencies are also reflected in the weighting matrix W_i. Hence, the structure of W_i can be exploited to efficiently adding (12b) and (12c).
- The linear cut (14) captures the *intra-layer interactions* (*i.e.*, terms $x_i x_i^\mathsf{T}$), which cannot be clearly indicated by NN parameters (weights or biases).

Given these observations, in the following we use the linear cuts (12b) and (12c). Moreover, it is straightforward to show that Theorem 1 holds even by adding a portion of (12b) and (12c). Hence, the computational cost of the problem can be reduced by adding only a portion of them.

Algorithm 1 describes an efficient implementation of the layer RLT-SDP relaxation. The fraction of linear cuts added at each iteration are set by choosing the sequence $\{p_s\}_{s=1}^r$, where $0 \leq p_1 < \cdots < p_r \leq 1$. In practice, the sequence $\{p_s\}_{s=1}^r$ and the maximum iteration $k_{\max}$, which satisfies $1 \leq k_{\max} \leq r$, can be

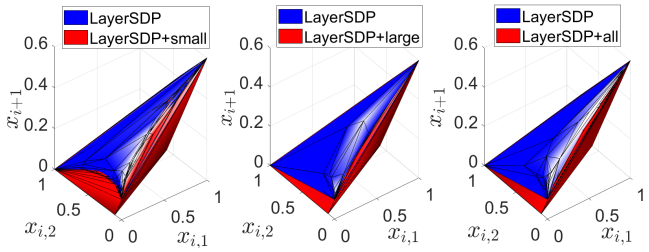

Fig. 2: Feasible region of $x_{i+1} = \text{ReLU}(W_i x_i + 0.1)$, with $W_i = [0.5 \ \ -1]$, $x_i = [x_{i,1},\ x_{i,2}]^\mathsf{T}$ and $0 \le x_{i,1}, x_{i,2} \le 1$, by adding a part or all of linear cuts (12b) and (12c). Adding only the linear cuts about $x_{i+1}x_{i,2}$ (*i.e.*, the larger element of $|W_i|$) yields a feasible region close to the one with full constraints on $x_{i+1}x_i^\mathsf{T}$.

adapted to the computational power available. Note that, in principle, a different sequence $\{p_s\}_{s=1}^r$ can be chosen for each individual layer; for simplicity we consider these to be constant. The matrix $\mathcal{O}_i$ stores the ordering (in descending order) of the elements in each row of $|W_i|$. The purpose of creating the ordering is to ensure that the part of linear cuts with larger influences on shrinking the feasible region are added first. This is based on the consideration as follows: For the neuron m at layer $i+1$, its pre-activation is $\hat{x}_{i+1,m} = W_i(m,:)x_i + b_i(m)$, where $W_i(m,:)$ is a row vector. Let w_1 and w_2 be any two elements of $W_i(m,:)$ and their corresponding inputs are $x_{i,1}$ and $x_{i,2}$, respectively. If $|w_1| > |w_2|$, then compared to those linear cuts about $x_{i+1}x_{i,2}$, the linear cuts about $x_{i+1}x_{i,1}$ has bigger influence on the feasible region of $x_{i+1,m}$. Fig. 2 provides an example for this, where it is seen that the linear cuts about $x_{i+1}x_{i,2}$ contributes more than $x_{i+1}x_{i,1}$ in shrinking the feasible region of x_{i+1}.

We show the property of Algorithm 1 in Theorem 2.

Theorem 2. *The relation $\gamma^*_{LayerSDP} \le \gamma^*_{RLT\text{-}SDP} \le \gamma^*$ holds under any choice of $\{p_s\}_{s=1}^r$. At any given iteration k of Algorithm 1, we have that $\gamma^*_{RLT\text{-}SDP_k} \le \gamma^*_{RLT\text{-}SDP_{k+1}} \le \gamma^*$.*

Proof. It is straightforward to prove the first part following Theorem 1. At each iteration k, the proportion of linear cuts added is $\bar{p}_i$, the integer part of $p_k n_i$. Since $p_{k+1} > p_k$, the proportion added at iteration $k+1$ is larger than that at iteration k and contains it as a subset. Hence, for any $k \ge 1$, every feasible solution to the optimisation problem solved at iteration $k+1$ is also a solution to the problem solved at iteration k, *i.e.*, $\gamma^*_{\text{RLT-SDP}_k} \le \gamma^*_{\text{RLT-SDP}_{k+1}}$. □

At each iteration, the layer RLT-SDP relaxation (15) is solved with a total number of $\sum_{i=1}^{L-1} 3\bar{p}_i n_{i+1}$ linear cuts. This is computationally lighter than the problem obtained by adding all the inequalities in (12b) and (12c). Furthermore, before running the algorithm, we can also remove the inactive neurons and simplify the constraints of stable neurons to reduce the sizes of the constraints

$P_i \succeq 0$, $i \in \mathbb{I}_{L-1}$. This can be realised by examining the activation pattern of the NN under a given verification query and will not relax the solution. This is a strategy used in [8].

4 Experimental Evaluation

Two sets of experiments were carried out to evaluate the precision and scalability of relaxation proposed as well as Algorithm 1. The experiments were run on a Linux machine with an Intel i9-10920X 3.5 GHz 12-core CPU with 128 GB RAM. The optimisation problems were modelled by using YALMIP [27] and solved using MOSEK [3]. We compared the results obtained against presently available SoA methods and tools.

Networks. In Experiment 1, we considered two groups of two-input, two-output, fully-connected random ReLU NNs generated by using the method in [13]. *Group 1* had four models with $L = 4$, 6, 8, 10 hidden layers, respectively, and 15 neurons for each hidden layer. *Group 2* had four three-layer models, with $n_i = 10$, 15, 50, 100 neurons per hidden layer, respectively.

In Experiment 2, we considered three groups of fully-connected ReLU NNs trained on the MNIST dataset. These are widely used in all the SDP benchmarks (By "$m \times n$" we mean a NN with $m - 1$ hidden layers each having n neurons, which is consistent with [8].):

- *Small NNs*: MLP-Adv, MLP-LP and MLP-SDP from [33] and tested under the same perturbation $\epsilon = 0.1$ as in [8, 33].
- *Medium NNs*: Models 6×100 and 9×100 from [35] and evaluated under the same $\epsilon = 0.026$ and $\epsilon = 0.015$ as in [8, 31, 35, 37]
- *Large NNs*: Models 8×1024-0.1 and 8×1024-0.3 from [25], which were trained using CROWN-IBP [45] with adversarial attack $\epsilon = 0.1$, 0.3, respectively. As in [25], they were tested under the perturbations $\epsilon = 0.1$, 0.3, respectively.

Baseline Methods. In Experiment 2, we compared the proposed layer RLT-SDP relaxation (referred to as `RLT-SDP`) against the SoA methods for verification below:

- *Complete methods*: `MILP` [38], `AI`^{2} [15], and β-`CROWN` [42].
- *Linear relaxations*: the standard linear programming relaxation `LP` [12] and its variants including `IBP` [17], `OptC2V` [37], and `PRIMA` [31]. We did not consider `kPoly` [36] and `DeepPoly` [35] , as they were shown in [31] to be weaker than `PRIMA`.
- *SDP relaxations*: `LayerSDP` [8], `SDP-IP` (*i.e.*, the global SDP relaxation (7)) [33], and `SDP-FO` [10].

Experiment 1: Efficacy of The Proposed Strategy. We investigated both network depth and width by using `RLT-SDP` to obtain an over-approximated feasible output region of the NN for a given input set. The test inputs were random values within $[0, 1]$ and the heuristic method in [13] was adopted to compute

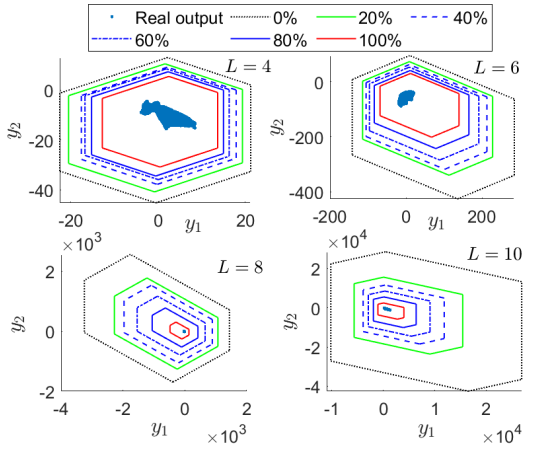

Fig. 3: Over-approximated output region by `RLT-SDP` with different percentages of linear cuts for networks of different hidden layers L. The 0% case is `LayerSDP`.

the over-approximations. Algorithm 1 was run with $\{p_s\}_{s=1}^{11} = \{0, 0.1, \cdots, 1\}$ and $k_{\max} = 11$. Without linear cuts ($p_1 = 0$), `RLT-SDP` is equivalent to `LayerSDP`.

We first studied the impact of network depth on the verification method here proposed by using the models in *Group 1*. Fig. 3 shows that for all the four models considered, adding a larger percentage of linear cuts yields a tighter over-approximation. As the number of hidden layers L increases, `LayerSDP` becomes looser and the effects of adding linear cuts becomes more significant. The figures show that across all models, even using just 20% of the linear cuts considerably reduces the over-approximation. To further analyse the gain in the approximation versus the corresponding increase in computational complexity, we considered two metrics: the improvement in approximation (or tightness) and the runtime increase. The former is the relative reduction in the feasible output regions obtained by `RLT-SDP` and `LayerSDP`; the latter is the relative increase in their runtime. As expected, it is shown in Fig. 4 that adding a larger proportion of linear cuts yields a tighter over-approximation, along with an increase in runtime. Adding the same percentage of linear cuts leads to a more significant tightness improvement on larger networks (with larger L) than on smaller ones. For each network, as the percentage of linear cuts increases, the tightness improvement becomes less significant, but the runtime increase becomes more significant. Particularly, experimentally we found that the first 20% of linear cuts contributes most significantly to the improvement in overall tightness of the method. We evaluated the impact of network width by using the models in *Group 2* and reported the results in Appendix, from which we observed very similar behaviour of the method.

These results clearly confirm Theorem 2 and demonstrate the efficiency of Algorithm 1. They also indicate that a trade-off needs to be balanced between the tightness improvement and runtime increase. Specifically, the addition of

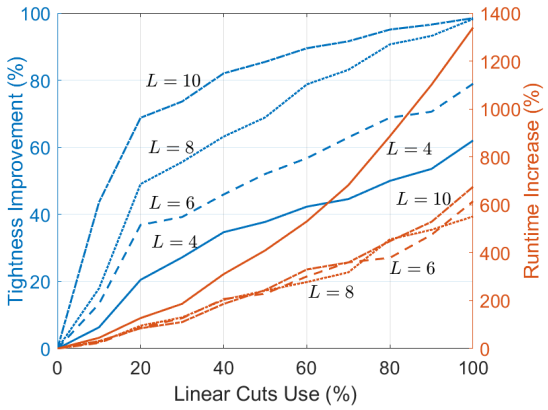

Fig. 4: Tightness improvement and runtime increase obtained by solving `RLT-SDP` with different percentages of linear cuts for networks of different hidden layers L. The 0% case is `LayerSDP`.

20% of linear cuts could be sufficient to improve considerably the precision of the SDP approach without incurring the higher computational costs associated with larger problems.

Experiment 2: Comparison with SoA Methods. We benchmarked the technique on the NNs built on the MNIST dataset described above. All experiments were run on the first 100 images of the dataset that were also used by `LayerSDP`, `SDP-IP`, `SDP-FO` and `LP`. The results obtained are reported in Table 1, where the runtime is the solver time. The PGD upper bounds of MLP-Adv, MLP-LP, MLP-SDP, 6×100 and 9×100 are taken from [8], while those of 8×1024-0.1 and 8×1024-0.3 are from [25]. We ran Algorithm 1 with the sequence $\{0.1, 0.2\}$ and $k_{\max} = 2$. As in `LayerSDP`, we further optimised `RLT-SDP` by removing inactive neurons in the first step.

Our results show that `RLT-SDP` based on the interval arithmetic bounds is more precise than `LayerSDP` under the same bounds and all other baseline methods for all the networks. One exception is the 9×100 network, for which β`-CROWN` achieves the highest precision. By using the tighter symbolic bound propagation [9], `RLT-SDP` significantly outperformed all the incomplete/complete baseline methods.

As expected we found `RLT-SDP` to be significantly more computationally demanding than `LayerSDP` across all the networks. However, it was still faster than `SDP-IP` for the small and median networks. Neither `SDP-IP` nor `SDP-FO` could verify the two large networks. Also, it is shown in [8] that compared to `LayerSDP`, `SDP-FO` has a runtime that is much larger for MLP-Adv and MLP-LP, but smaller for MLP-SDP. `SDP-FO` fails to verify 6×100 and 9×100. These results confirm that `RLT-SDP` remains competitive in terms of computational efficiency. We note that the runtime of `LayerSDP` in Table 1 is larger than that reported

Table 1: Vertified robustness (ver., in percentage) and runtime per image (t, in seconds) for a set of benchmarks with various sizes. The models MLP-Adv, MLP-LP, MLP-SDP, 6×100, 9×100, 8×1024-0.1, and 8×1024-0.3 are referred to as S1, S2, S3, M1, M2, L1, and L2, respectively.

		RLT-SDP		LayerSDP		SDP-IP		SDP-FO	LP	OptC2V	PRIMA	β-CROWN	MILP	AI^2	IBP
NNs	PGD	ver.	t^*	ver.†	t^*	ver.†	t^*	ver.†	ver.†	ver.†	ver.†	ver.†	ver.†	ver.†	ver.†
S1	94	**88**|**94**	3622	80|91	2164	82	12079	84	65	–	–	–	–	–	–
S2	80	**80**|**80**	159	80|80	145	80	50733	78	79	–	–	–	–	–	–
S3	84	**84**|**84**	11141	80|84	4373	80	43156	64	35	–	–	–	–	–	–
M1	91	**82**|**90**	3297	60|75	1900	–	6760	◇	21	42.9	51.0	69.9	–	–	–
M2	86	**56**|**70**	10800	22|35	7119	–	11899	◇	18	38.4	42.8	62.0	–	–	–
L1	89	**82**|	5883	–	2932	◇	◇	◇	0	–	–	–	67	52	80
L2	26	**26**|	761	–	469	◇	◇	◇	0	–	–	–	7	16	22

Dagger (†): these results are directly taken from the literature: `LayerSDP` and `SDP-FO` from [8], `SDP-IP` from [33], `OptC2V` from [37], `PRIMA` from [31], β-`CROWN` from [42], while `MILP`, `AI`^{2} and `IBP` from [25]; The first four numbers of `LP` are from [8], and the last two are obtained by implementation with interval arithmetic bounds. The results of `OptC2V`, `PRIMA` and β-`CROWN` were obtained using 1000 images while the results of 100 images were not available in the literature. Dash (–): previously reported numbers are unavailable. Diamond (◇): the methods fail to verify any instance. Star ($*$): the runtime is estimated by running over five images using the same interval arithmetic bounds. Vertical line (|): the certified robustness on the left and right are obtained using interval arithmetic bounds and symbolic interval propagation, respectively.

in [8]. This is because we directly solved the layer SDP relaxation (9), without implementing SparseColO [14] or the automatic model transformation as in [8]. Their work shows that using subroutine from SparseColO can balance the size of semidefinite constraints and equality constraints, and using automatic model transformation can reduce YALMIP overhead time, both of which significantly improve the efficiency of `LayerSDP`. Note, however, that these techniques are also directly applicable to `RLT-SDP`. Hence, the results presented here provides a like-for-like comparison between `LayerSDP` and `RLT-SDP`.

5 Conclusions

While SDP-based algorithms have shown their promise as a next generation method to verify NN, their resulting over-approximations are still too coarse to verify deep and large NNs. In this paper we put forward a novel SDP relaxation for achieving tight and efficient neural network robustness verification. We did so by combining the layerwise SDP relaxation with RLT. We showed that the method always yields tighter bounds than the present SoA. We also illustrated how a careful choice of linear cuts can mitigate the additional computational cost, thereby resulting in an overall tight and computationally balanced technique. The experiments reported demonstrated that the method achieves SoA on all benchmarks commonly used in the area.

Acknowledgements This work is partly funded by DARPA under the Assured Autonomy programme (FA8750-18-C-0095). Alessio Lomuscio is supported by a

Royal Academy of Engineering Chair in Emerging Technologies. Jianglin Lan is supported by a Leverhulme Trust Early Career Fellowship.

References

1. Akintunde, M.E., Botoeva, E., Kouvaros, P., Lomuscio, A.: Formal verification of neural agents in non-deterministic environments. In: Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS20). pp. 25–33 (2020)
2. Akintunde, M.E., Botoeva, E., Kouvaros, P., Lomuscio, A.: Verifying strategic abilities of neural-symbolic multi-agent systems. In: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR20). vol. 17, pp. 22–32 (2020)
3. Andersen, E.D., Andersen, K.D.: The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In: High performance optimization, pp. 197–232. Springer (2000)
4. Anderson, R., Huchette, J., Ma, W., Tjandraatmadja, C., Vielma, J.: Strong mixed-integer programming formulations for trained neural networks. Mathematical Programming pp. 1–37 (2020)
5. Anstreicher, K.M.: Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. Journal of Global Optimization **43**(2-3), 471–484 (2009)
6. Bak, S., Liu, C., Johnson, T.: The second international verification of neural networks competition (vnn-comp 2021): Summary and results. arXiv preprint arXiv:2103.06624 (2021)
7. Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., Criminisi, A.: Measuring neural net robustness with constraints. In: Advances in Neural Information Processing Systems (NeurIPS16). pp. 2613–2621 (2016)
8. Batten, B., Kouvaros, P., Lomuscio, A., Zheng, Y.: Efficient neural network verification via layer-based semidefinite relaxations and linear cuts. In: International Joint Conference on Artificial Intelligence (IJCAI21). pp. 2184–2190 (2021)
9. Botoeva, E., Kouvaros, P., Kronqvist, J., Lomuscio, A., Misener, R.: Efficient verification of neural networks via dependency analysis. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI20). pp. 3291–3299 (2020)
10. Dathathri, S., Dvijotham, K., Kurakin, A., Raghunathan, A., Uesato, J., Bunel, R., Shankar, S., Steinhardt, J., Goodfellow, I., Liang, P., Pushmeet, K.: Enabling certification of verification-agnostic networks via memory-efficient semidefinite programming. In: Advances in Neural Information Processing Systems (NeurIPS20). pp. 1–14 (2020)
11. Dvijotham, K., Stanforth, R., Gowal, S., Mann, T., Kohli, P.: A dual approach to scalable verification of deep networks. arXiv preprint arXiv:1803.06567 (2018)
12. Ehlers, R.: Formal verification of piece-wise linear feed-forward neural networks. In: International Symposium on Automated Technology for Verification and Analysis (ATVA17). pp. 269–286 (2017)
13. Fazlyab, M., Morari, M., Pappas, G.J.: Safety verification and robustness analysis of neural networks via quadratic constraints and semidefinite programming. IEEE Transactions on Automatatic Control **67**(1), 1–15 (2022)
14. Fujisawa, K., et al.: User's manual for sparsecolo: Conversion methods for sparse conic-form linear optimization problems. Dept. of Math. and Comp. Sci. Japan, Tech. Rep. pp. 152–8552 (2009)

15. Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., Vechev, M.: AI^2: Safety and robustness certification of neural networks with abstract interpretation. In: IEEE Symposium on Security and Privacy (SP18). pp. 3–18. IEEE (2018)
16. Goodfellow, I., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)
17. Gowal, S., Dvijotham, K.D., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Arandjelovic, R., Mann, T., Kohli, P.: Scalable verified training for provably robust image classification. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (IEEE/CVF19). pp. 4842–4851 (2019)
18. Hashemi, V., Kouvaros, P., Lomuscio, A.: Osip: Tightened bound propagation for the verification of relu neural networks. In: International Conference on Software Engineering and Formal Methods (SEFM21). pp. 463–480. Springer (2021)
19. Henriksen, P., Lomuscio, A.: Efficient neural network verification via adaptive refinement and adversarial search. In: Proceedings of the Twenty-fourth European Conference on Artificial Intelligence (ECAI20), pp. 2513–2520. IOS Press (2020)
20. Henriksen, P., Lomuscio, A.: Deepsplit: An efficient splitting method for neural network verification via indirect effect analysis. In: International Joint Conference on Artificial Intelligence (IJCAI21). pp. 2549–2555 (2021)
21. Julian, K.D., Kochenderfer, M.J.: Reachability analysis for neural network aircraft collision avoidance systems. Journal of Guidance, Control, and Dynamics **44**(6), 1132–1142 (2021)
22. Katz, G., Huang, D., Ibeling, D., Julian, K., Lazarus, C., Lim, R., Shah, P., Thakoor, S., Wu, H., Zeljic, A., Dill, D., Kochenderfer, M., Barrett, C.: The marabou framework for verification and analysis of deep neural networks. In: International Conference on Computer Aided Verification (CAV19). pp. 443–452 (2019)
23. Kouvaros, P., Kyono, T., Leofante, F., Lomuscio, A., Margineantu, D., Osipychev, D., Zheng, Y.: Formal analysis of neural network-based systems in the aircraft domain. In: International Symposium on Formal Methods (FM21). pp. 730–740. Springer (2021)
24. Lasserre, J.B.: Moments, positive polynomials and their applications, vol. 1. World Scientific (2009)
25. Li, L., Qi, X., Xie, T., Li, B.: Sok: Certified robustness for deep neural networks. arXiv preprint arXiv:2009.04131 (2020)
26. Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., Kochenderfer, M.J., et al.: Algorithms for verifying deep neural networks. Foundations and Trends® in Optimization **3-4**, 244–404 (2020)
27. Lofberg, J.: Yalmip: A toolbox for modeling and optimization in matlab. In: IEEE International Conference on Robotics and Automation (ICRA04). pp. 284–289. IEEE (2004)
28. Lomuscio, A., Maganti, L.: An approach to reachability analysis for feed-forward relu neural networks. CoRR **abs/1706.07351** (2017)
29. Ma, Z., Sojoudi, S.: Strengthened sdp verification of neural network robustness via non-convex cuts. arXiv preprint arXiv:2010.08603 (2020)
30. Manzanas Lopez, D., Johnson, T., Tran, H.D., Bak, S., Chen, X., Hobbs, K.L.: Verification of neural network compression of acas xu lookup tables with star set reachability. In: AIAA Scitech 2021 Forum. p. 0995 (2021)
31. Müller, M.N., Makarchuk, G., Singh, G., Püschel, M., Vechev, M.: PRIMA: Precise and general neural network certification via multi-neuron convex relaxations. arXiv preprint arXiv:2103.03638 (2021)

32. Parrilo, P.A.: Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization. California Institute of Technology (2000)
33. Raghunathan, A., Steinhardt, J., Liang, P.: Semidefinite relaxations for certifying robustness to adversarial examples. In: Advances in Neural Information Processing Systems (NeurIPS18). pp. 10877–10887 (2018)
34. Salman, H., Yang, G., Zhang, H., Hsieh, C., Zhang, P.: A convex relaxation barrier to tight robustness verification of neural networks. In: Advances in Neural Information Processing Systems (NeurIPS19). pp. 9835–9846 (2019)
35. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proceedings of the ACM on Programming Languages **3**(POPL), 41:1–41:30 (2019)
36. Singh, G., R. Ganvir, R., Püschel, M., Vechev, M.: Beyond the single neuron convex barrier for neural network certification. In: Advances in Neural Information Processing Systems (NeurIPS19). pp. 15098–15109 (2019)
37. Tjandraatmadja, C., Anderson, R., Huchette, J., Ma, W., Patel, K., Vielma, J.P.: The convex relaxation barrier, revisited: Tightened single-neuron relaxations for neural network verification. In: Advances in Neural Information Processing Systems (NeurIPS20). pp. 1–12 (2020)
38. Tjeng, V., Xiao, K., Tedrake, R.: Evaluating robustness of neural networks with mixed integer programming. In: International Conference on Learning Representations (ICLR19). pp. 1–21 (2019)
39. Vandenberghe, L., Andersen, M.S.: Chordal graphs and semidefinite optimization. Foundations and Trends in Optimization **1**(4), 241–433 (2015)
40. Vandenberghe, L., Boyd, S.: Semidefinite programming. SIAM Review **38**(1), 49–95 (1996)
41. Wang, S., Pei, K., Whitehouse, J., Yang, J., Jana, S.: Efficient formal safety analysis of neural networks. In: Advances in Neural Information Processing Systems (NeurIPS18). pp. 6367–6377 (2018)
42. Wang, S., Zhang, H., Xu, K., Lin, X., Jana, S., Hsieh, C.J., Kolter, J.Z.: Beta-crown: Efficient bound propagation with per-neuron split constraints for complete and incomplete neural network verification. arXiv preprint arXiv:2103.06624 (2021)
43. Weng, T., Zhang, H., Chen, H., Song, Z., Hsieh, C., Boning, D., Dhillon, I., Daniel, L.: Towards fast computation of certified robustness for relu networks. In: International Conference on Machine Learning (ICML18). pp. 5276–5285 (2018)
44. Wong, E., Kolter, Z.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: International Conference on Machine Learning (ICML18). pp. 5286–5295 (2018)
45. Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D., Hsieh, C.J.: Towards stable and efficient training of verifiably robust neural networks. arXiv preprint arXiv:1906.06316 (2019)
46. Zhang, R.Y.: On the tightness of semidefinite relaxations for certifying robustness to adversarial examples. arXiv preprint arXiv:2006.06759 (2020)
47. Zheng, Y.: Chordal sparsity in control and optimization of large-scale systems. Ph.D. thesis, University of Oxford (2019)
48. Zheng, Y., Fantuzzi, G., Papachristodoulou, A.: Chordal and factor-width decompositions for scalable semidefinite and polynomial optimization. Annual Reviews in Control **52**, 243–279 (2021). https://doi.org/10.1016/j.arcontrol.2021.09.001

Appendix

Proof of Observation 1. The constraints (10) are equivalently expanded as

$$\begin{bmatrix} P_i[x_i]l_i^\top & P_i[x_i]l_{i+1}^\top \\ P_i[x_{i+1}]l_i^\top & P_i[x_{i+1}]l_{i+1}^\top \end{bmatrix} \leq \begin{bmatrix} P_i[x_i x_i^\top] & P_i[x_i x_{i+1}^\top] \\ P_i[x_{i+1}x_i^\top] & P_i[x_{i+1}x_{i+1}^\top] \end{bmatrix} \leq \begin{bmatrix} P_i[x_i]u_i^\top & P_i[x_i]u_{i+1}^\top \\ P_i[x_{i+1}]u_i^\top & P_i[x_{i+1}]u_{i+1}^\top \end{bmatrix}.$$

The constraints on the diagonals are satisfied by imposing (11a) for each P_i and recalling that $P_i[x_{i+1}x_{i+1}^\top] = P_{i+1}[x_{i+1}x_{i+1}^\top]$ due to the input-output consistency guaranteed by (9g). The off-diagonals contain the constraints:

$$P_i[x_{i+1}x_i^\top] \geq \max\{P_i[x_{i+1}]l_i^\top,\ l_{i+1}P_i[x_i^\top]\} \tag{16a}$$

$$P_i[x_{i+1}x_i^\top] \leq \min\{P_i[x_{i+1}]u_i^\top,\ u_{i+1}P_i[x_i^\top]\}. \tag{16b}$$

Since $P_i[x_i^\top]-l_i^\top \geq 0$, $P_i[x_{i+1}^\top]-l_{i+1}^\top \geq 0$, $P_i[x_i^\top]-u_i^\top \leq 0$ and $P_i[x_{i+1}^\top]-u_{i+1}^\top \leq 0$, (11b) always implies (16a), while the conjunction of (11c) and (11d) imply (16b). This confirms that the constraints in (10) can be implied by the linear cuts (11a) - (11d).

Proof of Observation 2. According to (8), the constraint $P_i \succeq 0$ in (9f) implies $P_i[x_i x_i^\top] \geq x_i x_i^\top$ (which can be derived via Schur complement). Further considering the constraint (9d), it is true that $P_i[x_i x_i^\top] \geq P_i[x_i]l_i^\top$. To prove the second statement, re-organising (9d) as

$$\mathrm{diag}(P_i[x_i x_i^\top]) \leq u_i \odot P_i[x_i] + l_i \odot (P_i[x_i] - u_i).$$

This inequality implies $\mathrm{diag}(P_i[x_i x_i^\top]) \leq u_i \odot P_i[x_i]$ because $P_i[x_i] - u_i \leq 0$. It then yields the diagonals of the second part of (11a).

Proof of Observation 3. The proof of the first statement is similar to that of Proposition 1 in [29], and is thus omitted here. To prove the second statement, (13c) is rewritten as the joint of two constraints:

$$x_{i+1} - A_i x_i - B_i \leq 0 \vee x_{i+1} - A_i x_i - B_i \geq 0. \tag{17}$$

The first joint of (17) is exactly the existing constraint (9h). The second joint is analysed by reformulating

$$x_{i+1} - A_i x_i - B_i = k_i \odot (P_i[x_{i+1}] - W_i P_i[x_i] - b_i) + \Delta_i, \tag{18}$$

where $\Delta_i = (1 - k_i) \odot P_i[x_{i+1}] + k_i \odot \hat{l}_{i+1} - \mathrm{ReLU}(\hat{l}_{i+1})$. Under the existing constraints (9a) - (9c), it is true that (i) $P_i[x_{i+1}] = 0$ or (ii) $P_i[x_{i+1}]-W_i P_i[x_i]-b_i = 0$.

Without loss of generality, assume that x_{i+1} is of dimension one. For case (i), the node $i+1$ is *inactive*, meaning that $\hat{l}_{i+1} \leq \hat{u}_{i+1} \leq 0$. This leads to $k_i = 0$. Substituting $P_i[x_{i+1}] = 0$, $k_i = 0$ and $\mathrm{ReLU}(\hat{l}_{i+1}) = 0$ into (18) gives $P_i[x_{i+1}] - A_i P_i[x_i] - B_i = 0$. Hence, the second joint of (17) holds.

For case (ii), the discussion is further divided into two parts: (a) The node $i+1$ is *strictly active*, *i.e.*, $0 \leq \hat{l}_{i+1} \leq \hat{u}_{i+1}$. Then, $\Delta_i = (1-k_i) \odot (P_i[x_{i+1}] - \hat{l}_{i+1}) \geq 0$ because by definition k_i always satisfies $0 \leq k_i \leq 1$. Substituting $P_i[x_{i+1}] - W_i P_i[x_i] - b_i = 0$ and $\Delta_i \geq 0$ into (18) gives $P_i[x_{i+1}] - A_i P_i[x_i] - B_i \geq 0$. (b) The node $i+1$ is *unstable*, *i.e.*, $\hat{l}_{i+1} < 0 < \hat{u}_{i+1}$. Then $k_i = \frac{\hat{u}_{i+1}}{\hat{u}_{i+1} + |\hat{l}_{i+1}|}$ and $0 \leq P_i[x_{i+1}] \leq \hat{u}_{i+1}$. This leads to

$$\begin{aligned}\Delta_i =& (1-k_i) \odot P_i[x_{i+1}] + k_i \odot \hat{l}_{i+1} \\ =& \frac{|\hat{l}_{i+1}|}{\hat{u}_{i+1} + |\hat{l}_{i+1}|} \odot P_i[x_{i+1}] - \frac{\hat{u}_{i+1}}{\hat{u}_{i+1} + |\hat{l}_{i+1}|} \odot |\hat{l}_{i+1}| \\ \leq& 0.\end{aligned}$$

Hence, the second joint of (17) is automatically cut out.

In summary, under the existing constraints (9a) - (9c), the linear cut (13c) automatically holds when the node $i+1$ is *inactive* or *strictly active*. When the node $i+1$ is *unstable*, the part $P_i[x_{i+1}] - A_i P_i[x_i] - B_i \geq 0$ is automatically cut out and only the part $P_i[x_{i+1}] - A_i P_i[x_i] - B_i \leq 0$ remains. Furthermore, it is easy to see that the remaining part is readily implied by (9h). The above analysis applies to x_{i+1} of any dimension. This concludes that (13c) is already implied by the combination of existing constraints (9a) - (9c) and (9h).

Proof of Observation 4. The proof follows from that of Observation 3 and is only sketched here. Under the existing constraints (9a) - (9c), the term $P_i[x_{i+1}] - A_i P_i[x_i] - B_i$ equals to 0 when the node $i+1$ is *inactive*, equals to $(1-k_i) \odot (P_i[x_{i+1}] - \hat{l}_{i+1}) \geq 0$ when *strictly active*, and equals to $\frac{|\hat{l}_{i+1}|}{\hat{u}_{i+1} + |\hat{l}_{i+1}|} \odot P_i[x_{i+1}] - \frac{\hat{u}_{i+1}}{\hat{u}_{i+1} + |\hat{l}_{i+1}|} \odot |\hat{l}_{i+1}|$ when *unstable*. In the first case, the constraint (13b) automatically holds. In the last two cases, adding (13b) reduces the searching space and tightens the SDP relaxation. It is easy to see that the diagonals (13b) will not cut the feasible region, as they are implied by the existing constraint (9c).

MOSEK Settings. The solver MOSEK adopts the interior-point method to solve conic optimisation problems (*e.g.*, SDP). It normally takes many iterations to obtain an optimal objective value that meets the default accuracy. In practice, the default accuracy is not necessary for neural network verification, where a relatively less accurate solution (lower bound) is sufficient. Hence, we change the default settings of MOSEK to get a relatively less accurate with fewer iterations needed, to improve efficiency of the proposed algorithm. Particularly, we use 1.0e−6 to replace all the following default values:
MSK_DPAR_INTPNT_ CO_TOL_MU_RED = 1.0e−8,

MSK_DPAR_INTPNT_ CO_TOL_REL_GAP = 1.0e−8,
MSK_DPAR_INTPNT_ CO_TOL_INFEAS = 1.0e−12,
MSK_DPAR_INTPNT_ CO_TOL_FEAS = 1.0e−8,
MSK_DPAR_INTPNT_ CO_TOL_PFEAS = 1.0e−8.

Additional Results for Experiment 1. The results of the network width case in Experiment 1 are shown in Fig. 5 and Fig. 6.

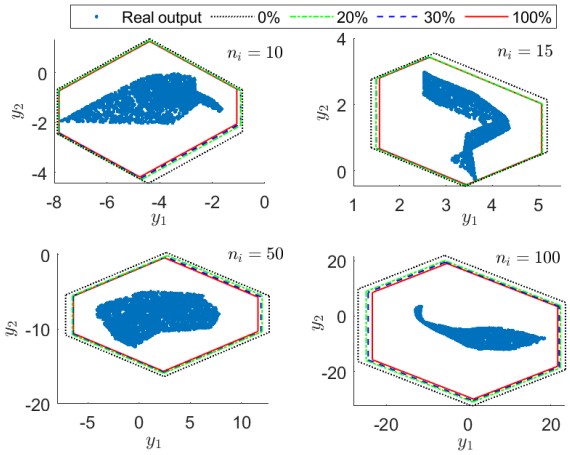

Fig. 5: Over-approximations of feasible output region by solving `RLT-SDP` with different percentages of linear cuts for networks having different numbers of neurons n_i per hidden layer. The 0% case is `LayerSDP`.

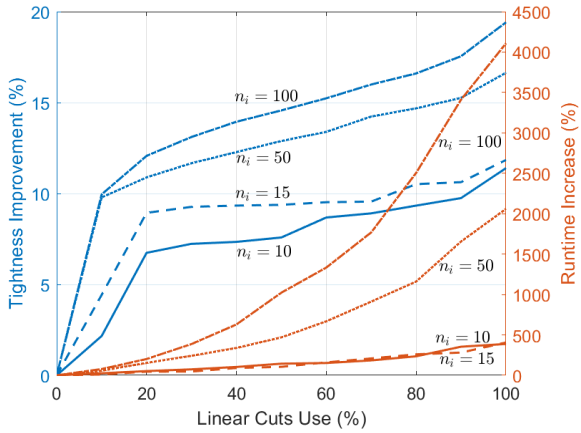

Fig. 6: Tightness improvement and runtime increase obtained by solving `RLT-SDP` with different percentages of linear cuts for networks having different numbers of neurons n_i per hidden layer. The 0% case is `LayerSDP`.